UM ALGORÍTMO HEURÍSTICO PARA RESOLVER UM PROBLEMA DE ALOCAÇÃO DE CANAIS DE COMUNICAÇÃO.

Marcelo Galvão Fonseca

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS (M.Sc.)

APROVADA POR:

Antonio A. F. Oliveira
Antônio A. F. Oliveira
Presidente

Paulo O Boaventura

Arlindo G. Alvarenga
Arlindo G. Alvarenga

Rio de Janeiro, RJ - BRASIL
Janeiro de 1983.

Fonseca, Marcelo Galvão

Um algorítmo heurístico para resolver um problema de alocação de canais de comunicação
(Rio de Janeiro, 1983).

vi, 88p. 29,7cm (COPPE - UFRJ, M.Sc., Engenharia de Sistemas e Computação)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Alocação de Canais, 2. Multicoloração

I. COPPE/UFRJ II. Título(série).

A Janinha minha espôsa e Andrē meu filho, pelo apoio e motivação para a realização deste trabalho.

AGRADECIMENTOS

Este trabalho carrega uma grande contribuição dos professores Antônio A. F de Oliveira e Paulo Boaventura a quem sinceramente agradeço.

Dirijo também meus agradecimentos aos funcionários da ELECTRA e PTEL, na pessoa de seus presidentes, Dr Raimundo Santiago e Dr Nédio Lopes Marques, pela valiosa quantidade de informações a respeito do assunto deste trabalho.

Lembro também de minha cunhada Ana Maria, pela ajuda na revisão, e de meu gerente Alfredo Lucena, além do meu coordenador Juan Carlos Tonsic pelo apoio e compreenção no uso dos equipamentos C-530 nas instalações da COBRA.

SUMÁRIO

Neste trabalho consideramos o problema de alocar novos canais às linhas de um sistema de comunicações "carrier" , atendendo a condição de que um mesmo canal não seja usado em duas linhas que sejam, ou adjacentes ou que possam ser ligadas por alguma outra linha.

Inicialmente esse problema é transformado num de obter uma multicoloração para o conjunto de nós de um grafo. Da análise desse novo problema alguns procedimentos heurísticos podem ser intuidos e se propõe um algorítmo de resolução - que os utilize.

Para mostrar o desempenho desse método, alguns exemplos de aplicação são também apresentados.

ABSTRACT

We consder here the problem of allocating new channels to the lines of a carrier system of communications subject to the constraint that the same channel cannot be used in two lines which are adjacent or have another line conecting them.

First we transform this problem in one of obtainning a multicoloration of the node set of a graph. From the analysis of this new problem some heuristc procedures can be derived and an algorithm using them is constructed. Some examplens testing the performance of this algorthm are also presented.

INDICE

1. MOTIVAÇÃO:

Este trabalho foi motivado pela tentativa de resolução de um problema de alocação de canais de telecomunicações carrier em linhas de alta tensão. Ou seja; em uma rede de transmissão de energia elétrica, as empresas que operam nesta rede, utilizam-na como veículo de suas comunicações.

A rede de transmissão de energia elétrica é composta por barramentos , (subestações de usinas geradoras ou de empresas de distribuição ou de energia), e por linhas de transmissão, que interligam estas subestações.

As comunicações são feitas através de ondas eletromagnéticas, ditas ONDAS PORTADORAS, em frequências pré-estabelecidas. Para cada frequência de comunicação, é associada uma banda de segurança para evitar interferências ou ruídos das comunicações feitas nas frequências vizinhas. Assim uma comunicação feita numa frequência de X Hz tem uma margem de segurança de +-Y Hz no qual nenhuma outra comunicação é feita. Esta banda total, de X-Y Hz a X+Y Hz é chamada canal de comunicação.

A gama total de frequências destinada a este tipo de comunicação, está fixada por uma norma internacional de utilização de frequências de comunicação. Disto resulta que o número de canais existentes é finito e facilmente enumerável.

A principal restrição do uso deste canais de telecomunicações nestas linhas, é quanto a interferência que ocorre se um mesmo canal é utilizado em duas linhas adjacentes. Existem fórmulas que conseguem expressar a atenuação desta interferência ao longo de uma linha vizinha de acordo com as características físicas desta linha; tais como sua extensão, tipo e espessura de cabo usado e etc. Esta interferência chega a ser forte o suficiente para viajar por mais de uma linha, e provocar efeitos nas linhas subsequentes.

Entretanto, após análises feitas sobre estas fórmulas, notou-se que o mais relevante é o número de subestações por onde o sinal atravessa. Isto se deve ao fato de que estas subestações são um filtro natural para a atenuação deste sinal. Assim, desta análise, consegue-se afirmar que um determinado canal utilizado em uma determinada linha de uma dada rede, só poderá ser utilizado nesta mesma rede em linhas que tenham pelo menos duas linhas que a intercale da primeira. Sendo esta a primeira restrição do problema original proposto.

Como já existe na realidade uma situação inicial de utilização dos canais de comunicações, o problema é na verdade o de expansão da comunicação.

Para prosseguirmos, é apropriado rever um pouco da teoria de grafos, uma vez que, como veremos no capítulo 3, esta rede apresentada, se representada por grafos facilita a formulação algorítmica do problema. Além disto, notemos também que a teoria de grafos já data de longa era, existindo hoje em dia uma vasta literatura sobre o assunto. Entretanto não existe até agora um consenso comum em tôrno dos termos adotados. Portanto é comum encontrarmos, em autores diferentes, nomes diferentes para um mesmo elemento de um grafo.

2. TEORIA DE GRAFOS:

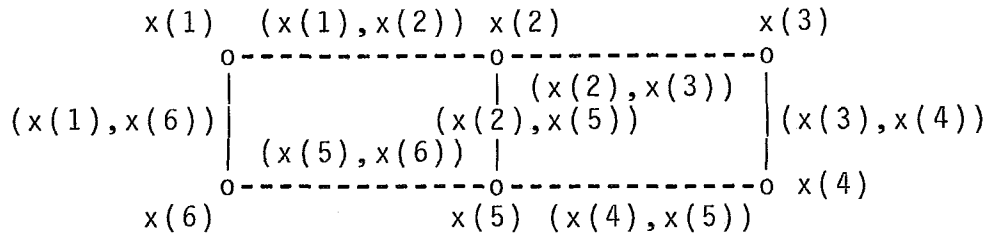
2.1. O que é um grafo?

Um grafo G=(X,W), consiste-se de um conjunto de objetos dados por X = (x(1), x(2),...x(n)), chamados pelas literaturas mais conhecidas por pontos, nós ou vértices. E por um outro conjunto, W = (w(1),w(2),...w(n)), normalmente chamados de linhas, arcos ou arestas.

O conjunto X, em geral representa um conjunto de estados possíveis de um processo qualquer. E W as relações ou transições possíveis entre estes estados. Portanto os elementos de W fazem uma ligação entre os elementos de X. Assim os elementos de W também são representados por (x(i),x(j)), onde x(i),x(j) E X; o que significa uma relação entre os elementos x(i) e x(j) de X. Daí mais um nome para os elementos de W, "ligação".

A representação mais simples de um grafo, é a representação gráfica. Que para grafos suficientemente pequenos permite uma visualização mais rápida dele como um todo.

Fig. 1.



um grafo G.

Para grafos de dimensões maiores, usamos a representação dada na definição. Que tem a vantagem de ser armazenada e trabalhada em computadores, sob as várias técnicas de armazenamento de dados.

2.2. Aplicações:

Grafo se aplica a uma infinidade de problemas. E hoje em dia, vários destes problemas só são resolvidos graças ao desenvolvimento da teoria de grafos. É na verdade, uma ferramenta que ajuda na formalização do problema e guia os mecanismos de resolução para suas soluções.

2.3. Definições.

2.3.1. Incidência:

Como vimos, um arco em G interliga dois de seus nós. Seja (x(i),x(j)) um arco de G. Diz-se que este arco incide em x(i) e em x(j).

2.3.2. Grau de incidência: Diz-se grau de incidência de um nó de um grafo, o número de arcos deste grafo que incidem neste nó.

2.3.3. Adjacência: Dois nós de um grafo G, são chamados de adjacentes se e só se existir um arco em G que os interligue.

2.3.4. Matriz de adjacências: É uma matriz A(n x n), onde n é o n número de nós de G, onde cada elemento desta matriz é dado por:

a(i,j)= 1 se o nó i é adjacente ao nó j em G
0 no caso contrário.

Esta matriz é uma das formas de representar o conjunto de arcos de G.

2.3.5. Caminho: Define-se por Caminho em um grafo G, como sendo uma sequência de nós e de arcos de G, de forma que esta sequência começa e termina com um nó. Um arco é sempre precedido de um nó em que ele incide. Um nó, (a menos do primeiro), é sempre precedido de um arco que incida nele.

2.3.6. Circuito: É um caminho, onde o primeiro nó é igual ao último.

2.4. Tipos de grafos:

2.4.1. Nulos: G=(X,W), onde X = W = vazio.

2.4.2. Conexos: G=(X,W), onde para quaisquer dois nós de G, existe ao menos um caminho em G onde estes dois nós aparecem simultâneamente.

2.4.3. Desconexos: G=(X,W), se um grafo é não conexo, ele é dito desconexo.

2.4.4. Subgrafo: S=(Y,Z) é um subgrafo do grafo G=(X,W) se e só se Y contido ou igual a X; Z contido ou igual W tal que, todos os caminhos que ocorrem em G passando apenas por nós de Y, também estão em S.

2.4.5. Componente conexa: Uma componente conexa de um grafo G é um subgrafo conexo de G, tal que não existe outro subgrafo conexo de G que o contenha.

Fig. 2

2.5.Árvores:

É um grafo conexo onde não existe nenhum circuito. Aqui valem as definições de árvore nula, componente e subárvore, de modo análogo as definições anteriores.

Vejamos algumas propriedades das árvores:

2.5.1. Teorema 1: Existe um e só um caminho entre quaisquer dois nós de uma árvore.

2.5.2. Teorema 2: Um grafo G conexo, é uma árvore se e só se para quaisquer dois nós de G, existe um e só um caminho que os interliga.

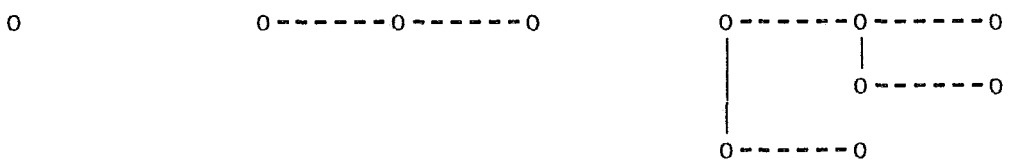
2.5.3. Teorema 3: Uma árvore com n nós tem n-1 arcos.

2.5.4. Teorema 4: Qualquer grafo conexo com n nós e n-1 arcos é uma árvore.

Estes teoremas, suas demonstrações, assim como as definições apresentadas neste capítulo, podem ser consultadas no livro de Deo(07).

Fig. 3

Árvores:



2.6.Nota:

É importante frisar que foi proposital a ausência do conceito de grafo direcionado. Para o problema em questão, este conceito é irrelevante posto que neste caso em nenhum dos arcos existe a idéia de direção.

3. O problema de expansão da utilização dos canais de telecomunicações cárrier em uma rede de alta tensão é apresentado como:

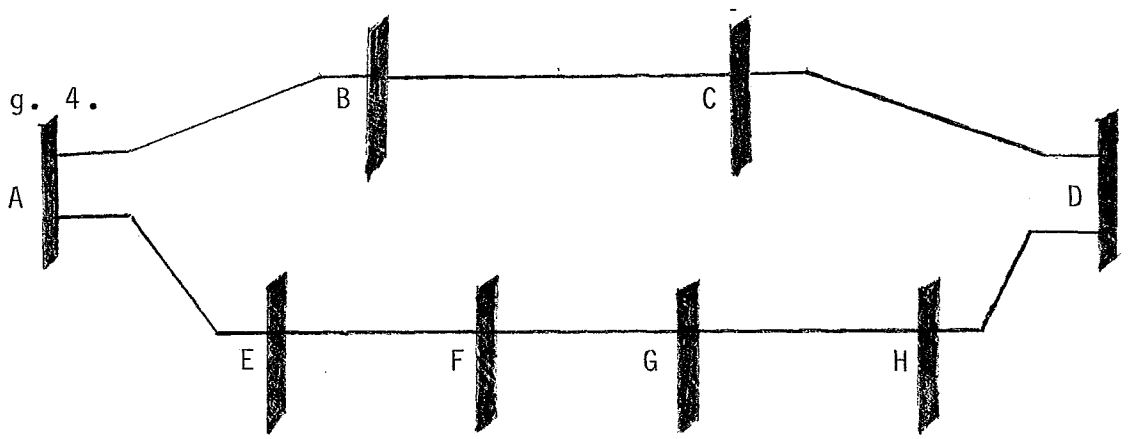
Dada uma rede de transmissão de energia elétrica, sobre a qual se tem alocada a cada linha um conjunto de canais de telecomunicações, deseja-se ampliar estes conjuntos de canais de forma que:

- Os canais possíveis pertençam a um universo finito de canais;

- um canal alocado a uma linha não pode estar alocado em linhas que tenham uma distância menor que duas linhas da linha atual.

A figura abaixo apresenta um trecho típico de uma rede de transmissão de energia elétrica, onde as barras "cheias", (A, B, C, D, E, F, G e H), são as subestações e a linhas que as interligam as linhas de transmissão.

Fig. 4.

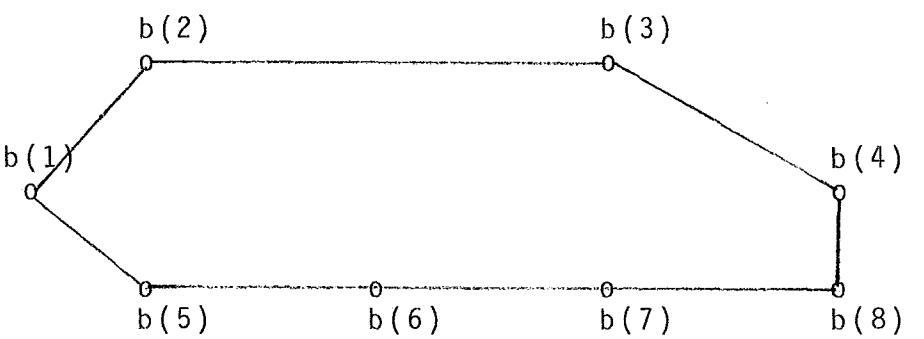


Vamos seguir tentando partir de uma situação simplificada, obter uma apresentação formal do problema, até alcançar uma apresentação formal satisfatória.

3.1. Transformações do problema de expansão da utilização dos canais de telecomunicações cârrier em uma rede de alta tensão.

A rede apresentada é nitidamente um grafo G=(X,V), onde X sendo o conjunto de nós de G, corresponde ao conjunto das subestações da rede; e V sendo os arcos de G, corresponde as linhas da rede.

Fig. 5.



Comecemos com o problema P1, uma simplificação do problema original

PROBLEMA P1.

Suponhamos que na rede ainda não existe nenhuma comunicação estabelecida. Suponhamos também que se deseja alocar apenas 1 canal por linha; obteremos então o problema P1 como abaixo:

Seja o grafo G=(X,V). Deseja-se "colorir" seus arcos de forma que uma cor não se repita a uma distância menor que dois arcos.

Como é fácil observar no nosso problema, as informações, (canais a serem alocados), se referem aos arcos de G, e não a seus nós. Portanto o mais apropriado, é utilizar um grafo auxiliar, dito grafo adjunto, que é uma transformação do grafo original, de forma a preservar suas características e que, as informações passem agora se referir aos nós deste. Tal grafo é expresso da forma L(G)=(V,W), onde V (o conjunto dos nós de L(G)), é o conjunto dos arcos de G, e W (o conjunto de arcos de L(G) é dado por

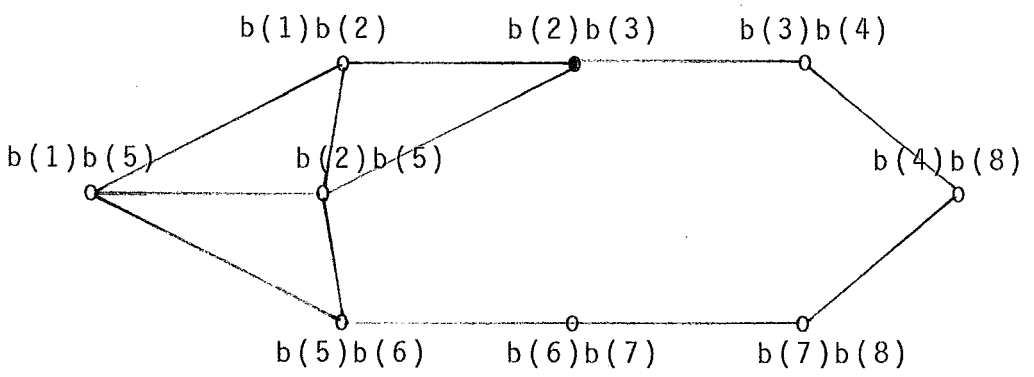
$$W=\left\{(v(i),v(j)) \;/\; v(i),\ v(j)\ E\ V,\ v(i)\ adj\ v(j)\right\}.$$

A matriz de adjacências de L(G), AL, pode ser dado por a(i,j), (elemento de AL da linha i coluna j), onde

$$a(i,j)=\begin{cases}1 & \text{se } v(i) \text{ adjacente } v(j) \text{ em } G\\ 0 & \text{no caso contrário.}\\ & \text{para todo } 1 <= i,j <= |\ V\ |\end{cases}$$

O grafo L(G) como definido acima é o indicativo das adjacências de arestas em G. Ou seja se v(i),v(j) E V e v(i) adj v(j) em L(G), significa que os arcos vi e vj incidem sobre um mesmo nó em G.

Fig. 6.



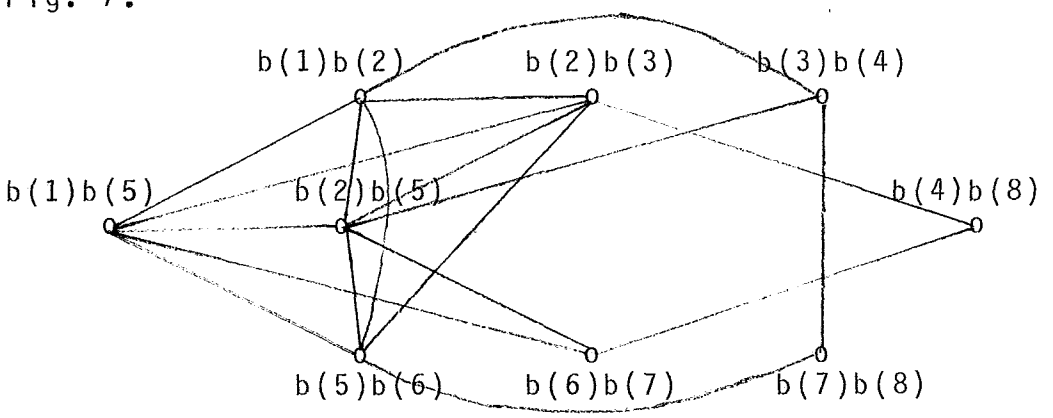
A figura 6 é uma evolução do exemplo iniciado nas figuras 4 e 5, no desenrolar do problema; tendo a matriz de adjacências AL como abaixo:

AL	b(1) b(2)	b(1) b(5)	b(2) b(3)	b(2) b(5)	b(3) b(4)	b(4) b(8)	b(5) b(6)	b(6) b(7)	b(7) b(8)
b(1)b(2)	0	1	1	1	0	0	0	0	0
b(1)b(5)	1	0	0	1	0	0	1	0	0
b(2)b(3)	1	0	0	1	1	0	0	0	0
b(2)b(5)	1	1	1	0	0	0	1	0	0
b(3)b(4)	0	0	1	0	0	1	0	0	0
b(4)b(8)	0	0	0	0	1	0	0	0	1
b(5)b(6)	0	1	0	1	0	0	0	1	0
b(6)b(7)	0	0	0	0	0	0	1	0	1
b(7)b(8)	0	0	0	0	0	1	0	1	0

O próximo passo é introduzir no nosso grafo auxiliar, algumas ferramentas que diminuam o esforço para observar os nós a uma distância de 2 em L(G), afim de avaliar a disponibilidade de cores do nó atual. Para tanto constrói-se AL2 como o quadrado booleano de AL. Zerando sua diagonal principal, esta matriz corresponde ao grafo L2(G)=(V,Z), indicativo das adjacências de arestas com uma aresta intercalada em G; onde V é o mesmo de L(G) e Z, (o conjunto dos arcos de L2(G)) é dado por

$$Z = \{(v(i),v(j)) \ / \ v(i),v(j) \in V,\ a(i,j) = 1,\ a(i,j) \in AL2\}$$

Fig. 7.



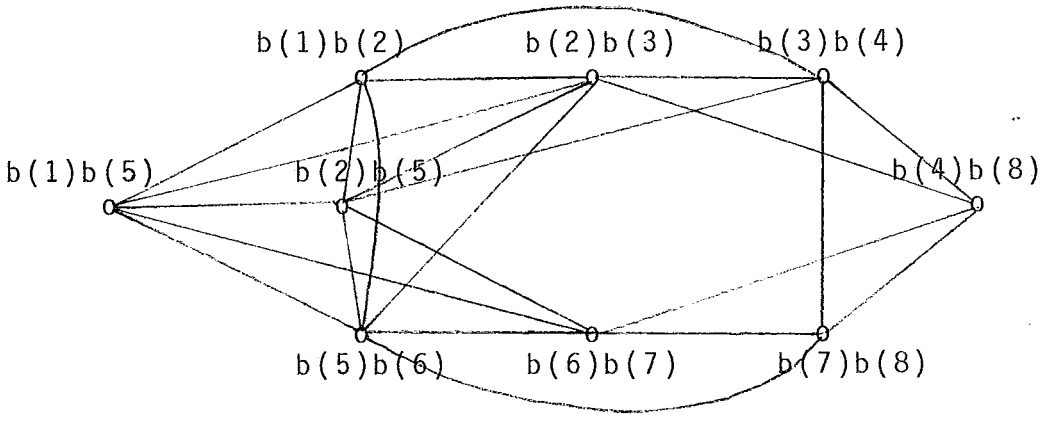
E sua matriz de adjacências, AL2, para o exemplo apresentado fica:

AL2	b(1) b(2)	b(1) b(5)	b(2) b(3)	b(2) b(5)	b(3) b(4)	b(4) b(8)	b(5) b(6)	b(6) b(7)	b(7) b(8)
b(1)b(2)	0	1	1	1	1	0	1	0	0
b(1)b(5)	1	0	1	1	0	0	1	1	0
b(2)b(3)	1	1	0	1	0	1	1	0	0
b(2)b(5)	1	1	1	0	1	0	1	1	0
b(3)b(4)	1	0	0	1	0	0	0	0	1
b(4)b(8)	0	0	1	0	0	0	0	1	0
b(5)b(6)	1	1	1	1	0	0	0	0	1
b(6)b(7)	0	1	0	1	0	1	0	0	0
b(7)b(8)	0	0	0	0	1	0	1	0	0

Observemos no entanto que L2(G) = (V,Z) perde as relações entre os nós V de L(G), o que significa que em L2(G), perde-se a noção de adjacência entre os nós de L(G). Como nos intereça manter estas relações, e as relações entre dois nós com um que os intercale, constrói-se a

matriz ALL como sendo a soma booleana das matrizes AL2 e AL. Esta matriz corresponderá a um grafo LL(G) = (V, Z U W), que é o indicativo das adjacências de arestas com até uma aresta que as intercale em G.

Fig. 8.



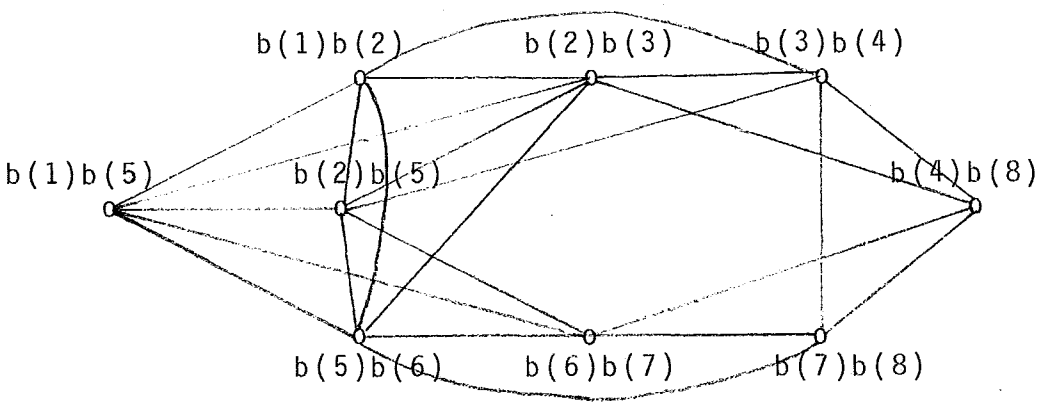
No nosso exemplo, a matriz de adjacências ALL fica...

ALL	b(1) b(2)	b(1) b(5)	b(2) b(3)	b(2) b(5)	b(3) b(4)	b(4) b(8)	b(5) b(6)	b(6) b(7)	b(7) b(8)
b(1)b(2)	0	1	1	1	1	0	1	0	0
b(1)b(5)	1	0	1	1	0	0	1	1	0
b(2)b(3)	1	1	0	1	1	1	1	0	0
b(2)b(5)	1	1	1	0	1	0	1	1	0
b(3)b(4)	1	0	1	1	0	1	0	0	1
b(4)b(8)	0	0	1	0	1	0	0	1	1
b(5)b(6)	1	1	1	1	0	0	0	1	1
b(6)b(7)	0	1	0	1	0	1	1	0	1
b(7)b(8)	0	0	0	0	1	1	1	1	0

E agora, aproveitando a evolução dos dados obtidos, o problema P1 se transforma em:

Colorir LL(G) de forma que dois nós adjacentes tenham cores diferentes. (Problema de coloração já bastante conhecido e estudado).

Fig. 9.



O objetivo aqui é fazer evoluir o problema P1 até que se consiga formular o problema original de modo a ser implementável, aproveitando técnicas já conhecidas na área de coloração, e até mesmo podermos apresentar outra. Desta forma, vejamos...

O PROBLEMA P2:

Seja o grafo LLG = (V,ZW), deseja-se associar a cada nó v(i) E V um conjunto de cores S(v(i)) com NS(v(i)) cores de tal forma que:

Para todo v(i),v(j) E V se v(i) adj v(j) então S(v(i)) n S(v(j)) = vazio.

Este problema é uma generalização de P1, onde S(v(i)) era um conjunto composto de uma só cor.

Resolver P2 seria resolver em G, a multi-coloração de seus arcos, onde cada cor não se repetiria a uma distância menor que 2 arcos.

Temos desta forma conseguido introduzir ao nosso grafo auxiliar, o conceito da multicoloração. Só nos resta agora implementar a existência de uma situação inicial, ou seja, o grafo já estando parcialmente colorido, e desejando uma expansão destas cores. Segue então a evolução do problema P2 para ...

O PROBLEMA P3:

Seja o grafo LLG = (V,ZW), onde a cada nó v(i) E V, existe um conjunto de cores S(v(i)) já associado, com NS(v(i)) cores.

Deseja-se ampliar S(v(i)) com mais um conjunto de cores R(v(i)) com NR(v(i)) cores de tal forma que:

(i) S(v(i)) n R(v(i)) = vazio para todo v(i) E V;
(ii) Para todo v(i), v(j) E V, se v(i) adj v(j) então
(S(v(i)) U R(v(i))) n (S(v(j)) U R(v(j))) = vazio.

Sendo conhecido V; ZW; S(v(i)), NS(v(i)) e NR(v(i)) para todo v(i) E V pede-se encontrar R(v(i)) para todo v(i) E V.

Resolver P3, seria resolver em G o problema da ampliação das cores de seus arcos, de tal forma que uma cor não se repita a uma distância menor que dois arcos.

A restrição (ii) de P3 pode ser desenvolvida em...

Para todo v(i), v(j) E V, se v(i) adj v(j) então
(S(v(i)) U R(v(i))) n (S(v(j)) U R(v(j))) = vazio.

Para que esta restrição seja atendida, as quatro restrições abaixo também terão que ser atendidas:

Para todo v(i), v(j) E V, se v(i) adj v(j) então
(iii) - S(v(i)) n S(v(j)) = vazio
(iv) - S(v(i)) n R(v(j)) = vazio
(v) - R(v(i)) n S(v(j)) = vazio
(vi) - R(v(i)) n R(v(j)) = vazio

Para simplificar, primeiramente podemos observar que as restrições (iv) e (v) são redundantes. Pois se elas tem que ser válidas para todos os pares de nós adjacentes, quando formos observar o nó v(b) adjacente a v(c) e mais tarde observar o nó v(c), estaremos em redundância nestas restrições. Portanto se suprimirmos a restrição (iv) o problema manterá suas características.

Observemos agora a restrição (iii). Ela exige que as cores já alocadas a dois nós adjacentes não coincidam. Ou seja esta restrição apenas procura assegurar que a solução já existente seja compatível com o problema proposto. Se soubermos de ante-mão que a solução atual é compatível, então também podemos desprezar esta restrição. Na realidade na maioria dos casos é fácil saber que a solução atual já obedece a esta retrição.

E agora podemos re-escrever P3 como abaixo...

Seja o grafo LLG = (V,ZW), e a cada nó v(i) de V, existe um conjunto S(v(i)) com NS(v(i)) cores associadas, de forma que para todo v(i) e v(j) E V, se v(i) adj v(j) então S(v(i)) n S(v(j)) = vazio.

Deseja-se ampliar S(v(i)) com mais um conjunto R(v(i)) com NR(v(i)) cores, de tal forma que:

Para v(i) E V,
S(v(i)) n R(v(i)) = vazio.....(i)
Para todo v(j) E V se v(j) adj v(i) então
R(v(i)) n S(v(j)) = vazio.....(v)
R(v(i)) n R(v(j)) = vazio.....(vi)

Conseguindo assim uma formulação do nosso problema proposto, e ainda identificando simplificações que conseguiram evitar algumas possíveis duplicidade de esforsos. Chamemos esta formulação de "Formulação via Multicoloração", ou "FM". O objetivo mais adiante é apresentar um algorítmo que trabalhe sobre a formulação apresentada para obter uma ou mais soluções para o problema.

Antes porém, vejamos o mesmo problema sob um outro ponto de vista.

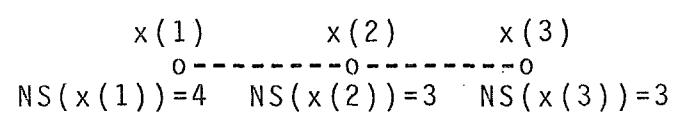
4. OUTRO ENFOQUE PARA O PROBLEMA PROPOSTO:

Vale a pena apresentarmos agora, um outro enfoque para o nosso problema. Este capítulo mostra que qualquer problema de multi-coloração pode ser transformado num outro de coloração simples, e daí tentar utilizar as técnicas já conhecidas do assunto. Porém, mostraremos também que para o caso de expansão, estas técnicas não são válidas apesar da representação "monocromática".

Para a transição de um problema de multi-coloração sob uma formulação "policromática", representada por um grafo G, para outra "monocromática"; devemos construir um grafo B, que consiga trazer todas as características do problema original para a nova representação. Este grafo pode ser construido seguindo a linha de raciocínio apresentada a baixo, acompanhada das figuras exemplo.

Fig. 10.

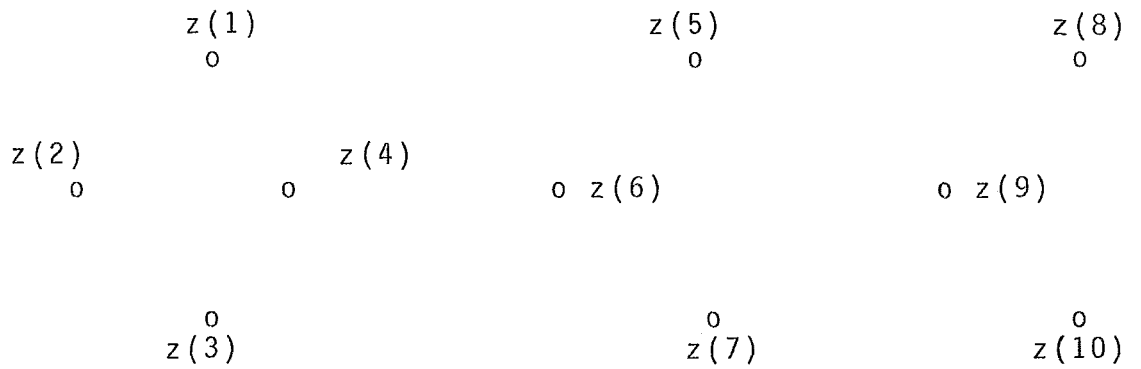
Grafo G



Seja G=(X,Y) um grafo qualquer. Deseja-se fazer uma multi-coloração de seus nós com NS(x(i)) cores para cada nó.

Para cairmos num problema de coloração simples, teremos que gerar em B para cada nó de G tantos nós quantas cores se deseja aplicar a este nó de G.

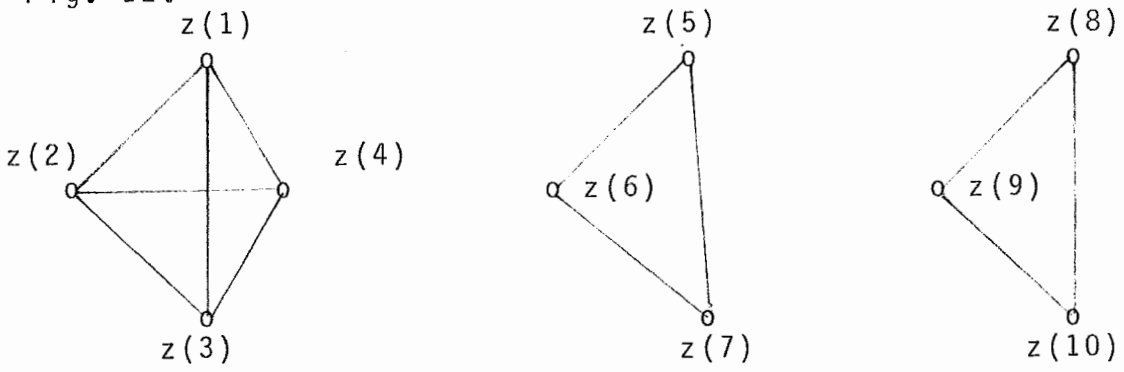
Fig. 11.



Seja B=(M,T). Assim é fácil estabelecer que M será um conjunto composto de tantos nós quanto o somatório em X, (para todo x(i) E X) do número de cores a serem aplicadas em x(i). Ou seja, somatório de NS em X.

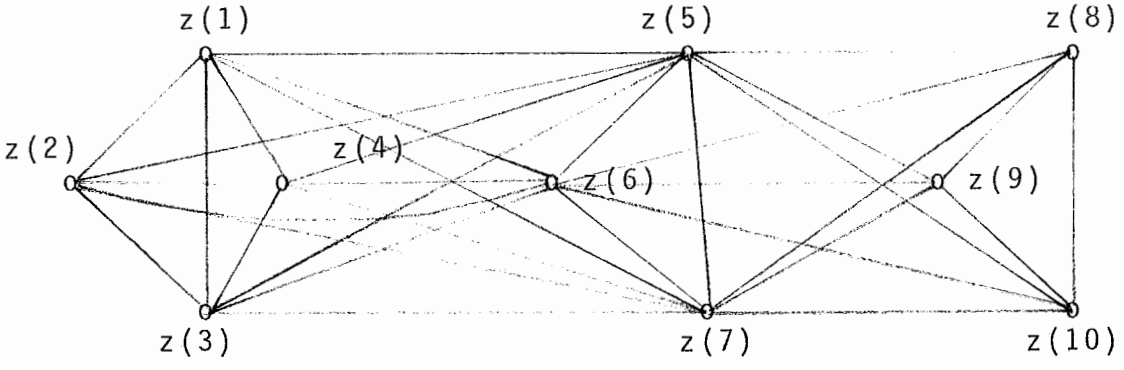
Para garantir em G que as cores de cada nó sejam diferentes entre si, teremos que em B fazer com que cada nó gerado de um mesmo nó em G, (irmãos), sejam ligados entre sí. Cada conjunto de irmãos em B corresponderá a um subgrafo completo. Então já temos T compondo-se de tantos arcos quanto o somatório em X do número de combinações dos NS(x(i)) elementos de S(x(i)) grupados em 2 a 2.

Fig. 12.



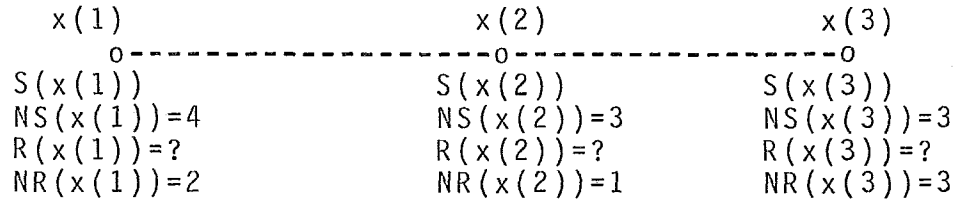
Finalmente para garantir que dois nós adjacentes em G não tenham cores comuns entre si, teremos que colocar um arco em B interligando cada "irmão" de uma "família" com todos os "irmãos" de todas as "famílias" geradas por nós adjacentes em G.

Fig. 13.



Quando quizermos neste caso introduzir o conceito da expansão da multicoloração, voltamos à figura 10 e acrescentamos este conceito como abaixo:

Fig. 14



Onde S(x(i)) é o conjunto de cores já alocadas ao nó x(i), com NS(x(i)) cores alocadas; R(x(i)) o conjunto de cores com NR(x(i)) cores a serem ampliadas ao nó x(i).

O grafo B pode ser obtido pela geração de tantos nós quanto a soma NS(x(i))+NR(x(i)) para cada nó x(i) de G, e a este grupo de nós chamemos de "filhos" de x(i). Daí em diante continua-se a desenvolver B como já visto, ou seja, cada grupo de "irmãos" formado por um subgrafo completo em B, e todos os "primos" em B interligados entre si. Lembrando somente que a cada grupo de irmãos em B, teremos alguns nós já coloridos e outros não, na verdade são precisamente NS(x(i)) e NR(x(i)) respectivamente.

Podemos agora apresentar o problema P4 como sendo:

Seja o grafo B=(N,T), onde alguns elementos de N estão coloridos e outros não; colorir todo nó n(i) não colorido de N de tal forma que:

cor de n(i) seja diferente a cor de n(j) para todo n(i) adj n(j) em N.

É fácil demonstrar que a formulação P4 é equivalente à de P3.

A seguir veremos alguns algorítmos de coloração. Estes algoritmos foram selecionados dentre os apresentados pela tese de Santos(1). Para que estes algorítmos possam realizar em G uma expansão das cores já existentes em seus nós, basta apenas fazer uma pequena alteração em cada algorítmo, mas conforme veremos ao final da apresentação deste capítulo, tais alterações não podem ser aplicadas para resolver o problema proposto.

Em seguida apresentaremos um novo algorítmo de multi-coloração, desenvolvido especificamente para resolver o problema proposto.

4.1. Algorítmo de coloração sequencial.

Selecionamos de Santos(1) 3 algorítmos por aproximação.

O primeiro deles é um que obtém um colorido sequencial, que é o procedimento mais intuitivo para se colorir um grafo. Este algorítmo consiste em tomar os pontos um a um e tentar colorí-los com uma cor já utilizada até agora.

A definição de colorido sequencial foi dada por Matula, Marble e Isaacson(2).

Definição:

Um colorido sequencial de G é, dada uma ordenação v(1),v(2),...v(n) de seus nós, um k-colorido de G no qual as cores 1,2,...,k são atribuidas da seguinte forma:

i) A cor 1 é atribuida a v(1), com isto (v(1)) recebe o 1-colorido.

ii) (v(1),v(2),...,v(i-1)) recebe um j-colorido; Atribui-se as mesmas cores a v(1),v(2),...,v(i-1) em (v(1),v(2),...,v(i)) e a v(i) atribui-se a cor r, onde r = j+1 é o menor inteiro positivo para o qual a cor r não ocorre na adjacência de v(i) em (v(1)...v(i)). E isto fornece um j-colorido a (v(1),...v(i)) se r = j, ou um j+1-colorido se r=j+1. Vide o algorítmo L1.

Algumas heurísticas podem ser aplicadas de forma a obter maior rapidez no algorítmo L1. A mais intuitiva é ordenar os nós de G em ordem decrescente do índice de incidência dos nós. A idéia é colorir primeiro os nós que tenham a maior vizinhança, pois se for colorido mais tarde aumentará a probabilidade de requisitar uma cor nova justamente por ter uma vizinhança maior. Esta ordenação é co-

nhecida por "largest-first".

Uma outra ordenação dos nós também é sugerida. Baseia-se no fato de poder haver mais de um nó com o mesmo grau de incidência em um grafo. Portanto a ordenação "largest-first" pode não ser a única.

Define-se então a ordenação "smallest-last" dos pontos de G através do algoritmo abaixo:

```
Procedimento escolha(i,n,V);
     comentário: n é o número de elementos de V;
     i:=n+1;
     a:=infinito;
     para i:=1 até n faça
          comentário: vj é o j-ésimo elemento de V;
          se grau(v(j)) < a  então a:=grau(v(j));
                                   i:=j;
fim;


Procedimento Smalest-last;
     k:=infinito;
     escolha(n,|V|,V);
     para j:=n-1 decrescendo até 1 faça:
          escolha(i,|V-(v(n),....v(j+1))|,
                        V-(v(n),...v(j+1)));
          se k > i então k:=i;
fim;
```

Este processo também pode ser melhorado através da técnica da TROCA BICROMÁTICA, que consiste em ao acrescentar um ponto v(i) ao subgrafo j-colorido (v(1),...v(i-1)) a j+1-ésima cor só será necessária se v(i) for adjacente a pontos com as cores 1,2,...j. Ora se algum destes pontos não fizer parte de um sub-grafo completo de G com j pontos pode ser possível trocar a cor de um deles de forma a se obter um j-colorido de (v(1),...v(i-1)) tal que o número de cores distintas aos nós adjacentes a v(i) seja no máximo j-1, liberando portanto uma cor para v(i).

De acordo com Santos(01) o algoritmo de coloração sequencial usando tanto a ordenação "largest-first" quanto a ordenação "smallest-last", tem sua complexidade de ordem O(n+m).

As alterações nos algorítmos de coloração para resolver a expansão da coloração em G, correspondem a coloração de um grafo que teve seu conjunto de nós e de arcos expandidos, e ainda não coloridos.

Ou por outro lado, o problema pode ser encarado como se o grafo B fosse visto em um determinado instante do processamento do algorítmo de coloração antes de estar completamente colorido. Cabe aquí promover as alterações necessárias para que o processamento "continue" e obtenha a coloração final.

Para o caso do algorítmo de coloração sequencial, o melhor seria primeiramente ordenar os nós do grafo de maneira que todo nó já colorido tenha uma ordem menor que qualquer nó não colorido. Entre os nós não coloridos a ordenação pode ser feita em ordem decrescente com o grau de incidência dos nós. Seguindo a heurística do algorítmo original.

Seja n1 o número de nós já coloridos. Seja k o número de cores já usadas na coloração inicial. Podemos chamar as cores já usadas por 1,2,...,k. E assim o algorítmo de coloração sequencial ficará como apresentado no algorítmo L2 .

O Algorítmo L1: Coloração sequencial

```
Begin
      c1:=1;
      k:=1;
      for i:=2 to n do begin
            r:=min(j/ A(i) n c(j) = vazio);
            comment: r = menor j tal que a adjacência de i
            comment:     não use a cor j.
            c(i):=r;
            k:=máx(k,r);
      end;
end;
```

Onde A(i) é o conjunto das cores já associadas aos nós da vizinhança do nó v(i); c(j) = cor j.

Algorítmo L2: Expansão por coloração sequencial

```
begin
     Ordenar os nós de modo que todo nó colorido tenha
     ordem menor que qualquer não colorido, e os não
     coloridos em ordem decrescente do grau de incidência
     entre si;
     for i:=n1+1 to n do
     begin
          r:=min( j / A(i) n c(j) = vazio );
          c(i):=r;
          k:=máx(k,r);
     end;
end;
```

4.2. Algorítmo de coloração por classe de cor.

O segundo algorítmo é classificado como algorítmo de coloração por classe, pois tenta aplicar uma cor a um número máximo de nós de G. Em seguida uma segunda cor será aplicada e assim por diante até que G esteja todo colorido. Vide o algorítmo L3.

Santos(01) mostra que fazendo a ordenação "largest-first", dos nós de G, este algorítmo tem sua complexidade da ordem de O(n+m).

Para o caso de expansão, podemos fazer a mesma ordenação dos nós como indicada no algorítmo de coloração sequencial para o caso de expansão. Consideremos nl o número de nós já coloridos, k o número de cores já usadas. Vide o algorítmo L4.

O algoritmo L3: Coloração por classe

```
begin
     for i:=1 to n do c(i):=-1;
     cor:=k:=1;
     while "existe nó não colorido" do
     begin
          for i:=1 to n do
          if c(i) = -1
          then if A(i) n  cor = vazio
                 then c(i):=cor;
          incr cor;
     end;
end;
```

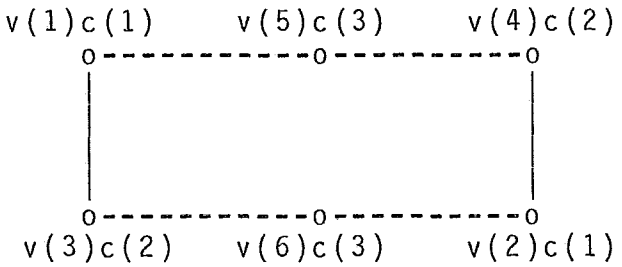
Algorítmo L4: Expansão pela coloração por classe

```
begin
     Ordenar os nós do grafo de maneira
     que todo nó já colorido tenha ordem
     menor que qualquer não colorido, e os
     não coloridos fiquem na ordem "largest-
     first" entre si, de acordo com o grau de
     incidência dos nós.
     for i:=n+1 to n do c(i):=-1;
     cor:=1;
     while "existe nó não colorido" do
     begin
          for i:=n1+1 to n do
          if c(i) = -1
          then if A(i) n  cor = vazio
               then c(i):=cor;
          incr cor;
     end;
     k:=máx( k, cor );
end;
```

4.3. Algorítmo de coloração aos pares.

Wood(3) observa que nos coloridos por classe, o número de cores pode ser desnecessariamente aumentado, dependendo da forma como os pontos estejam ordenados. Wood ilustrou este fato com um exemplo muito simples.

Fig. 15.



Como poderia ser evitado que v(1) e v(2) fossem coloridos com a mesma cor?

O problema é resolvido criando-se uma matriz de similaridades para indicar quais pares de nós devem receber a mesma cor. A matriz de similaridades entre v(i) e v(j) é medida através da matriz de adjacências do grafo assim:

s(i,j) = 0 se a(i,j)=1
somatório 1 = p = n de b(p) se a(i,j) = 0
onde b(p) = 1 se a(i,p) = a(p,j) = 1
0 no caso contrário
a(i,j) elementos da matriz de adjacências.
b(p) o indicativo da existência de um nó que intercale v(i) e v(j) no grafo.

Com isto a similaridade entre dois pontos não adjacentes v(i) e v(j) fica definida como sendo o número de outros pontos v(p) que sejam adjacentes a ambos. Vide o algorítmo L5.

E a ordem de complexidade do algorítmo de coloração aos pares é de O(mnn).

E no algorítmo de coloração aos pares, a alteração para atender o caso de expansão de cores, bastará apenas que nas linhas 7 e 8 do algorítmo, i e j variem de n1+1 até n; pois como já sabemos que de 1 a n1 os nós já estão coloridos. Vide o algorítmo L6.

Algorítmo L5: Coloração aos pares

```
begin
   construir a matriz de similaridades S;
   z = maior valor de S;
   k:=0;
   for m=z downto 0 do
   begin
      for i:=1 to n do
      for j:=1 to n do
      if S(i,j) = m
      then if v(i) e v(j) não coloridos
         then begin
            if d(i) > k or d(j) > k
            then begin
               r:=min(p/C(p) n (A(i) U A(j))=vazio);
               comment: r = a menor cor disponível a
                              v(i) e v(j);
               Colore v(i) e v(j) com a cor r;
               k:= máx( k , r );
            end;
         end else if v(i) ou v(j) não colorido
            then begin
               seja v(b) colorido e v(c) não colorido;
               r:=cor de v(b);
               if d(c) > k and (r) n A(c) = vazio
               then colore v(c) com a cor r;
            end;
   end;
end;
```

Onde A(k) = conjunto das cores dos nós vizinhos ao nó k;
d(k) =

Algorítmo L6: Expansão pela coloração aos pares

```
begin
   construir a matriz de similaridades S;
   z = maior valor de S;
   k:=0;
   for m=z downto 0 do
   begin
      for i:=n1+1 to n do
      for j:=n1+1 to n do
      if S(i,j) = m
      then if v(i) e v(j) não coloridos
         then begin
            if d(i) > k or d(j) > k
            then begin
               r:=min(p/C(p) n (A(i) U A(j))=vazio);
               comment: r = a menor cor disponível a
                              v(i) e v(j);
               colore v(i) e v(j) com a cor r;
               k:= máx( k , r );
            end;
         end else if v(i) ou v(j) não colorido
            then begin
               seja v(b) colorido e v(c) não colorido;
               r:=cor de v(b);
               if d(c) > k and (r) n A(c) = vazio
               then colore v(c) com a cor r;
            end;
   end;
end;
```

Nos três métodos de coloração apresentados, nas transformações para os casos de expansão, a complexidade tem como um limite superior a ordem da complexidade dos algorítmos originais quando do caso de coloração de todo o grafo G, (colorir n nós).

4.4.

Aspectos que invalidam o uso dos algorítmos aproximativos no nosso problema original.

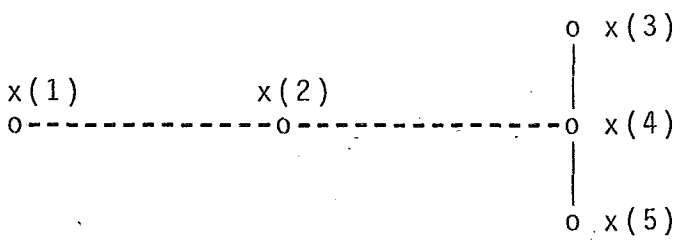
Tudo começa com o próprio fato de ser uma expansão da coloração, e não da coloração pura e simples. Seria mais fácil o uso destes algoritmos, se conseguíssemos garantir que a situação original tivesse sido implantada como resultado da aplicação de seu próprio processamento anteriormente. Se assim fosse, seria mais natural o uso do mesmo novamente para dar prosseguimento ao trabalho.

Porém, a verdade não é esta. A situação inicial foi implementada por processos manuais, sujeitos a uma distância muito maior da otimalidade do qualquer destes algoritmos viessem a dar.

O problema ainda não é só este, o mais crucial está no fato de as cores possíveis pertencerem a um conjunto finito, e muitas vezes, bem restrito em alguns nós. O exemplo a seguir pode clarear bem a questão:

Supondo o grafo abaixo, onde o universo de cores é dado por P=(1,2,3). Supondo ainda que os nós x(1) e x(3) estejam coloridos com a cor 2, e nó x(5) com a cor 3. A necessidade de expanção é só a de colorir os nós x(2) e x(4) com uma cor cada. Vejamos que ao nó 2 temos a disponibilidade de colorí-lo com as cores (1 e 3), e ao nó x(4) apenas com a cor (1).

Fig. 16.



Entretanto, se por qualquer motivo, em qualquer dos algoritmos apresentados, o nó x(2) for colorido antes com a cor 1, o nó x(4) não conseguirá ser mais colorido. Tudo isto se deve ao fato de que os algoritmos aproximativos não possuem a característica de realizar um back-track e tentar corrigir uma coloração já obtida. Atingindo assim a situação de coloração impossível sem que todas as situações possíveis tenham sido examinadas.

Apenas a heurística da troca bicromática poderia corrigir este defeito. Porém tem a séria limitação de que apenas alguns dos nós deverão ser coloridos, e os outros não. E assim esta heurística pode vir a falhar também.

Desta forma, somente um algorítmo que analise todas as soluções possíveis, tem chances ao sucesso. Seguiremos apresentando agora um que tentando observar todas as combinações de cores possíveis a cada nó, encontre uma solução viável para o problema.

5. MULTICOLORAÇÃO:

Vamos agora mostrar um algoritmo que liste todas as soluções para o problema P3. Na verdade ele poderia ter sido desenvolvido sob a formulação do problema P4, porém como veremos logo adiante, em termos de espaço em memória na fase de implementação, P3 é bem mais apropriado.

5.1. Novamente o problema P3...

Seja o grafo LLG = (V,ZW), e a cada nó v(i) de V, existe um conjunto S(v(i)) com NS(v(i)) cores associadas, de forma que para todo v(i) e v(j) E V, se v(i) adjacente a v(j) então S(v(i)) n S(v(j)) = vazio.

Deseja-se ampliar S(v(i)) com mais um conjunto R(v(i)) com NR(v(i)) cores, de tal forma que:

R(v(i)) n S(v(i)) = vazio
Para todo v(j) E V, se v(j) adj v(i) então
R(v(i)) n S(v(j)) = vazio
R(v(i)) n R(v(j)) = vazio.

Chamaremos de conjunto das cores disponíveis ao nó v(i) o conjunto C(v(i)) dado por
C(v(i)) = P-S(v(i))-U(S(v(j))) para todo v(j) E V,
v(j) adj v(i); onde P = universo das cores;

Onde
U(S(.)) conjunto união dos conjuntos S(.);
e NC(v(i)) o número de cores do conjunto C(v(i)).

É imediato que se R(v(i)) contido em C(v(i)) então

R(v(i)) n S(v(i)) = vazio e
para todo v(j) E V, se v(j) adj v(i)

então R(v(i)) n S(v(j)) = vazio.

E novamente vamos re-escrever o problema P3.

Seja o grafo LLG = (V,ZW), e a cada nó de v(i) E V, existe um conjunto S(v(i)) com NS(v(i)) cores associadas, de forma que para todo v(i), v(j) E V, se v(i) adj v(j) então S(v(i)) n S(v(j)) = vazio.

Deseja-se ampliar S(v(i)) com mais um conjunto R(v(i)) com NR(v(i)) cores,
de tal forma que:
Para todo v(i) E V,
R(v(i)) contido em C(v(i))
para todo v(j) E V, se v(j) adj v(i)
então R(v(i)) n R(v(j)) = vazio.

É a criação dos conjuntos C(.) que justifica o uso da formulação P3 do problema e não a P4 na implementação do algorímo; porque senão ele será duplicado várias vezes na memória. Basta lembrar que a cada nó v(i) de G gera NS(v(i))+NR(v(i)) nós em B, e para cada um deles o conjunto C seria o mesmo C(v(i)), o que significa uma repetição de C várias vezes na memória.

Como para cada nó v(i) de V, temos NC(v(i)) cores disponíveis e uma necessidade de alocar NR(v(i)) cores a este nó. Se NR(v(i)) NC(v(i)) para algum v(i) de V, então o problema não tem solução; senão para cada nó v(i) de V teremos tantas soluções diferentes possíveis quanto dado pelo número de combinações dos NC(v(i)) elementos de C(v(i)) grupados em NR(v(i)) a NR(v(i)).

Suponhamos um algorítmo ALFA que liste todas as combinações de cores possíveis com NR(v(i)) cores dentre as cores do conjunto C(v(i)) com NC(v(i)) cores, em uma ordem qualquer. Passemos a reconhecer agora cada uma destas combinações pelo índice de ordem em que ela aparece na lista gerada pelo algorítmo. Uma solução para o problema P3 pode ser reconhecida agora pelo vetor W=(R(1),R(2),...,R(n)), onde R(i) é o índice que fornece a ordem de aparição na lista de combinações geradas pelo algorítmo ALFA para o nó v(i).

O algorítmo proposto adiante trabalha no espaço de soluções inviáveis, até que uma solução viável seja encontrada. Podendo parar mostrando a solução encontrada ou continuar a procurar outra solução.

Notemos no entanto que a função objetivo do problema P3 é apenas encontrar uma solução viável. Mais tarde apresentaremos alguns argumentos que podem nos guiar para uma boa solução.

5.2. O algoritmo de multicoloração

A idéia do algoritmo para resolver P3, é basicamente escolher uma solução para o primeiro nó. Tendo-a escolhido, tenta obter uma solução para o segundo, de forma a atender a restrição:

se v(1) adj v(2), então R(v(1)) n R(v(2)) = vazio.

Se conseguir uma solução para v(2), então tentar obter uma solução para v(3). Da mesma forma, se tiver sucesso seguir para v(4),
senão volta parar v(2) e procurar uma próxima solução para v(2).

O algoritmo pode ser pensado sobre uma árvore hierárquica onde "nó" seria a solução corrente de todos os nós até o nó atual; "pai" seria a solução corrente de todos os nós até o nó anterior ao atual; "filho", a primeira solução encontrada do próximo nó, incluindo a solução apresentada por "nó", e "irmão" seria a mesma solução representada por "nó", a menos da solução corrente do nó atual. "raiz" será o elemento de partida e o de chegada do algoritmo. Pode-se usar o artifício de fazer "raiz" como sendo o pai do primeiro nó.

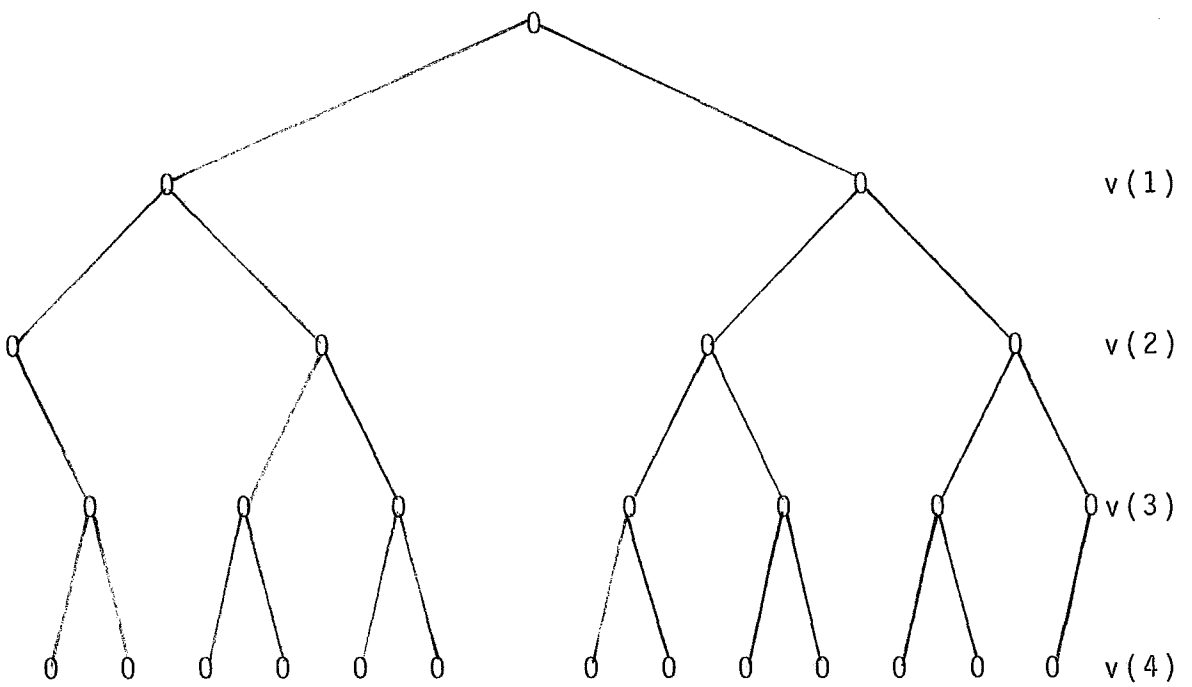
Notar que o termo solução a um nó v(i), aqui está se referindo apenas a uma combinação de NR(v(i)) cores do conjunto C(v(i)).

Esta árvore estará disposta da seguinte maneira. Cada nível se refere a um nó de LLG, a menos da raiz, que é um elemento artificial. Partindo da raiz, ela terá tantos filhos quantas soluções houverem para o primeiro nó. Ou seja no primeiro nível ocorrerão tantos elementos quanto o número de combinações dos NC(v(1)) elementos de C(v(1)) grupados em NR(v(1)) a NR(v(1)). Para cada nó do primeiro nível serão gerados tantos filhos quanto o número de combinações dos NC(v(2)) elementos de C(v(2)), grupados em NR(v(2)) a NR(v(2)). E assim por diante.

O algoritmo executa uma busca em profundidade tentando descer nos nós desta árvore até que consiga chegar ao nível mais baixo, o que corresponderá a uma solução se só for permitido descer de um nível para o seguinte se para a solução encontrada para "nó" for compatível com as soluções de seus ancestrais. Ou seja, atende à retrição:

para j:=1 até i-1
v(j) adj v(i) então R(v(i)) n R(v(j)) = vazio;

Fig. 17.



Veremos mais tarde que não será necessário construir esta árvore. O que seria muito caro em tempo e espaço.

Na definição de árvore, chamamos de "folhas", os nós de uma árvore que não se ligam a nenhum nó de nível inferior. Pela construção de nossa árvore, sabemos que todas as folhas estão no seu nível mais inferior.

O algorítmo apresentado abaixo, parte da raiz e percorre todos os caminhos desta árvore, chegando a todas as suas folhas. Como podemos verificar se trata nada mais nada menos que o algorítmo de varredura dos nós de uma árvore, seguindo a pré-ordem, ou ordem prefixa.

```
begin
     no:=raiz;
     k:=0;
     while verdade = verdade do
     begin
          while filho(nó) diferente de vazio do
          begin
               no:=filho(nó);
               incr k;
               W(k):=nó;
          end;
          listar W;
          while irmaõ(nó) = vazio do
          begin
               if pai(nó) = raiz
               then pare
               else begin
                    decr k;
                    nó:=pai(nó);
               end;
          end;
          nó:=irmão(nó);
          W(k):=nó;
     end;
end;
```

Este algorítmo é apresentado em quase todas as referências a árvores binárias, principalmente nas apresentações das gerações de códigos de compiladores de linguagens de computadores que tenham expressões lógicas e/ou matemáticas. Berztiss(5) cap 5.e também o apresenta.

Podemos inserir neste algorítmo a crítica,

```
for j:=1 to k-1 do
if R(k) n R(j) = vazio
then "solução atual":=falso;
```

onde k = nível atual na árvore. Esta crítica fornecerá básicamente a todo instante a informação de que a solução que está sendo encontrada é incompatível ou não com as soluções de seus ancestrais. Ou seja, em LLG estamos tentando multi-colorir um nó com cores já usadas em nós adjacentes nas suas soluções.

Este teste se corretamente implementado no algorítmo de varredura da árvore na pré-ordem, pode provocar "podas" nos galhos da árvore. Isto é válido, pois toda ramificação ou subárvore abaixo do nó com a violação mencionada não corrige a incompatibilidade da solução atual com a de algum de seus ancestrais.

Para reforçar a conveniência da poda, convêm notar que a dimensão da árvore em que estamos trabalhando é diretamente proporcional ao tempo de execução do algorítmo apresentado. E como é fácil observar, esta árvore cresce exponencialmente com o número de nós de LLG.

Voltando às podas; se conseguirmos provocar as podas nos níveis mais altos da árvore, estaremos cada vez podando subárvores maiores, e com isto diminuindo o esforço desnecessário. Em vista disto podemos concluir que será dependente da ordenação dada aos nós.

5.3. Primeira heurística:

Uma primeira heurística visando melhorar o rendimento do algorítmo seria então colocar no níveis mais próximos da raiz da árvore, aqueles nós de LLG que estão mais propensos a gerar podas. Estes nós são os que tem a maior demanda de cores e a menor disponibilidade. Isto é caracterizado pelo número de combinações possíveis de cores viáveis para solucionar o nó. Ou seja, se ordenarmos em ordem crescente os nós de acordo com o número de combinações possíveis para resolver o nó, que é dado pelo número de

combinações dos NC(v(i)) elementos de C(v(i)) grupados em NR(v(i)) a NR(v(i)); estaremos colorindo primeiro os nós críticos, os que deverão gerar as maiores podas na árvore.

A escolha do peso aos nós ser dada pelo número de combinações dos NC(v(i)) elementos de C(v(i)), grupados em NR(v(i)) a NR(v(i)), e não outros valores tais como a pura demanda, a simples disponibilidade ou até mesmo a razão entre eles merece ser visto com cuidado.

Primeiramente se fosse escolhido puramente o número de cores disponíveis (NC(v(i))) como peso ao nó v(i) estaríamos desprezando a demanda dele. Ou seja esta informação poderia não espelhar a tentência de poda deste nó corretamente. Basta pensar que uma disponibilidade de 50 a um nó que exige uma demanda de também 50 irá ter peso idêntico a um nó que tenha uma disponibilidade também de 50 cores, mas uma demanda de 10 por exemplo.

Se escolhessemos a demanda como pêso, em raciocínio semelhante ao anterior poderemos apresentar exemplos que demonstrem que não é esta uma escolha apropriada.

Finalmente, para a escolha da razão entre estes valores (NC(v(i))/NR(v(i))) podemos tomar o seguinte exemplo com 3 nós:

NC(v(i)) = 50	NR(v(i)) = 10
NC(v(j)) = 11	NR(v(j)) = 2
NC(v(k)) = 6	NR(v(k)) = 1

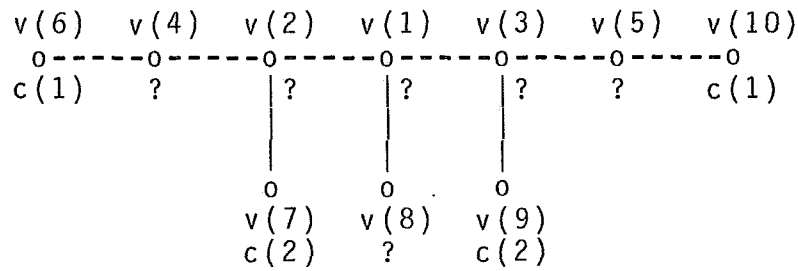
Notemos que a razão da disponibilidade pela demanda em todos os três é 5 para 1 para o nó v(i), 5.5 para 1 para o nó v(j) e 6 para 1 para v(k). Ou seja, nestes têrmos, v(i) está mais propenso que v(j) e este mais ainda que v(k) para gerar uma poda. Suponha então que a vizinhança de cada um deles lhes tome duas cores de sua disponibilidade. Refazendo as contas notamos que a ordem exatamente se inverteu.

Nenhum destes contra-exemplos são aplicáveis se o peso for o sugerido.

No capítulo 7, apresentaremos as comparações de tempo de execução do algorítmo, trabalhando com a ordenação dos nós indicada, com a ordenação inversa, e sem nenhuma ordenação heurística.

Vejamos agora o grafo G como desenhado abaixo, com 10 nós, onde o universo de cores é (c(1), c(2)).

Fig. 18.



Desejamos colorir os nós v(1),v(2),v(3),v(4),v(5) e v(8) com uma cor cada. Teremos as cores disponíveis a estes nós dadas por

C(v(1)) := (c(1), c(2))

C(v(2)) := (c(1))

C(v(3)) := (c(1))

C(v(4)) := (c(2))

C(v(5)) := (c(2))

C(v(8)) := (c(1), c(2))

Podemos notar que se v(1) receber a cor c(1) os nós v(2) e v(3) não conseguirão ser coloridos. Porque a escolha da cor c(1) para o nó v(1) esgotará todas a possibilidades de colorir alguns de seus nós vizinhos.

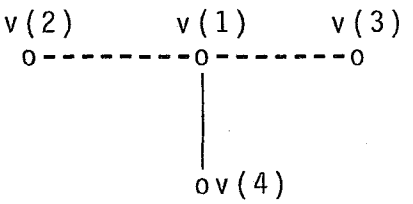
Dando sequência ao raciocínio que nos levou a adotar a heurística de ordenar os nós de LLG, que é tentar reconhecer o mais cedo possível uma incompatibilidade da solução que está sendo estudada. Justifica uma heurística que indicasse qual o caminho que mais provavelmente nos levaria a uma solução viável.

5.4. A segunda heurística:

Primeiramente atribui-se um peso a cada cor do conjunto de cores disponíveis a este nó. Este peso pode ser o número de nós vizinhos ao nó atual, onde a cor em questão também esteja disponível. Ora, este peso diz quantos vizinhos ao nó atual irão perder uma cor do seu conjunto de cores disponíveis se for usada esta cor neste nó. Agora ordenamos os conjutos de cores disponíveis de cada nó, em ordem crescente com o peso calculado. Finalmente se usarmos uma rotina que gere todas as combinações de cores do conjunto de cores disponíveis em ordem lexicograficamente crescente, o algorítmo de multi-coloração proposto irá tentar colorir cada nó com um conjunto de cores que prejudicará a menor vizinhança possível.

O peso proposto para ser atribuído as cores disponíveis, pode ser um pouco mais elaborado, visando a melhorar a heurística. Vejamos o subgrafo abaixo, no momento em que queremos colorir v(1)...

Fig. 19.



e sejam os conjuntos de cores disponíveis dados por:

C(v(1)) := (c(1), c(2), c(3)) ;

C(v(2)) := (c(1)) ;

C(v(3)) := (c(2), c(3)) ;

C(v(4)) := (c(2), c(3)) ;

Notemos que os pesos das cores de C(v(1)) como apresentados anteriormente serão:

P(v(1), c(1)):=1 ;

P(v(1), c(2)):=2 ;

P(v(1), c(3)):=2.

Entretanto se colorirmos v(1) com a cor c(1), estaremos esgotando a única possibilidade de colorir v(2). Assim se redefinirmos P(v(i), c(r)) como sendo proporsional ao somatório do inverso do número de combinações de NC(v(j))-1 em NR(v(j)) a NR(v(j)); para todo v(j) E V, v(j) adjacente a v(i) e c(r) E C(v(j)) n C(v(i)).

Assume-se que se x menor que y, a combinação de x em y a y é zero.

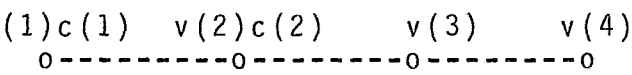
Agora os pesos das cores disponíveis são proporcionais ao número de nós vizinhos onde esta cor também está disponível, e inversamente proporcional à necessidade desta cor nesta vizinhança.

O peso atribuido desta forma, tenta evitar o uso de uma cor em um nó sempre que souber que isto esgotará a diponibilidadede algum nó de sua vizinhança.

Cabe ressaltar que os pesos não levam em consideração o fato da vizinhança já ter sido colorida ou não. A primeira idéia seria desprezar os nós vizinhos já coloridos. Porém como estamos nos propondo a resolver um problema de expansão, devemos pressupor que posteriormente outra expansão possa ser requerida nesta rede. Assim a heurística de não esgotar a disponibilidade de cores de nós já coloridos faz sentido. E na verdade poderia ser uma boa meta para a resolução do problema. Outra vantagem desta concepção é o fato de que não será preciso calcular estes pesos mais de uma vez para cada cor disponível em cada nó. O cálculo pode ser feito no início do processo e nunca mais se alterar.

Existe um outro detalhe que podemos ressaltar. Do algoritmo de coloração por pares, temos como heurística tentar colorir pares de nós com uma mesma cor, sempre que este par tem um nó que os intercale. No grafo abaixo, notemos que se colorirmos v(4) com a cor c(1),

Fig. 20.



o número cromático deste grafo se elevará a 3. Entretanto, se colorirmos v(4) com c(2) o número cromático cairá para 2. O algorítmo de coloração aos pares tenta justamente trabalhar em cima disto.

A criação dos pesos das cores disponíveis como definida anteriormente, diz que quanto menor o peso de uma cor disponível em um nó, maior é a vizinhança deste nó que não tem esta cor como disponível. Com isto, maior é a vizinhança deste nó que está intercalando-o de um outro nó que já usa esta cor. Este raciocínio cai em perfeita harmonia com a heurística do método de coloração por pares.

5.5. Novo algorítmo de multi-coloração.

```
begin
   ler LLG;
   Ler as dimensões das soluções desejadas;
   Ordenar os nós de LLG;
   Calcular os conjuntos de cores disponíveis aos nós de LLG;
   Calcular os pesos das cores disponíveis aos nós de LLG;
   Ordenar as cores dos conjuntos de cores disponíveis aos nós
   de LLG de acôrdo com os pesos obtidos;
   for número.da.solução:=0 to número.de.soluções.desejadas
   do begin
        if "próxima.solução.geral" = verdade
        then lista solução encontrada
        else if número.da.solução = 0
            then begin
               escreva "sistema sem solução";
               stop;
            end else begin
               escreva "todas soluções listadas";
               stop;
            end;
   end;
   escreva "nem todas as soluções possíveis foram listadas";
end;
```

O algorítmo acima, é responsável pelos pedidos de leitura do grafo e dos cálculos iniciais, tais como pesos das cores, ordenação das cores disponíveis e etc. E também pelo loop de trabalho do processo.

A próxima.solução.geral, terá valor verdade enquanto a partir da solução geral atual, for possível conseguir outra solução geral. É chamado de solução geral sempre que for encontrada uma configuração de cores para cada nó de LLG, tal que as restrições do problema sejam atendidas. Quando a partir de uma solução não se conseguir nenhuma outra solução geral, esta rotina retorna com valor falso.

Esta rotina como já vimos é uma variante das rotinas de varredura de uma árvore, segundo a pré-ordem.

```
lógical procedure próxima.solução.geral;
begin
      label L0,L1,L2,L3,L4;
L0:   if "próxima.solução(nó)" = falso
      then goto L3;
L1:   if no maior que número.de.nós
      then goto L4;
      if "situação.da.solução.atual.do(nó)" = falso
      then goto L0;
      if "avança" = verdade
      then goto L1;
L2:   próxima.solução.geral:=falso;
      return;
L3:   if "retrocede" = falso or NR(nó) igual 0
      then goto L2
      else goto L1;
L4:   próxima.solução.geral:=verdade;
end;
```

A situação.da.solução.atual.do(nó), retornará verdade sempre que o conjunto de cores que solucionam a ampliação das cores deste nó não conflitam com as soluções de sua vizinhança. Caso contrário retornará com o valor falso.

Esta rotina é chamada pela próxima.solução.geral, para validar ou invalidar uma configuração de cores para um determinado nó em estudo.

```
lógical procedure situação.da.solução.atual.do(word nó);
begin
      situação.da.solução.atual.do:=verdade;
      for i:=0 to nó-1 do
      if i adj nó
      then if R(i) n  R(nó) ≠ vazio
           then situação.da.solução.atual.do:=falso;
end;
```

Avança, simplesmente incrementa a variável nó, com o objetivo de tentar obter uma solução para o próximo nó de LLG. Esta rotina é chamada pelo próxima.solução.geral sempre que conseguiu obter soluções para todos os nós anteriores. Esta rotina volta sempre com valor verdade, a menos que o próximo nó não tenha nenhuma solução possível. Notar que aqui não é feita a crítica de o nó ter ultrapassado o número de nós de LLG. Esta crítica é feita no próxima.solução.geral.

```
lógical procedure avança;
begin
      incr nó;
      if nó <= número.de.nós
      then avança:=verdade
      else avança:="primeira.solução.do(nó)";
end;
```

Primeira.solução.do(nó), fornece para o nó a menor combinação de cores disponíveis neste nó, como sua primeira solução. Só irá retornar com valor falso se o número de cores disponíveis for menor que a quantidade de cores a ser aplicado no nó. O que configura uma solução impossível.

```
lógical procedure primeira.solução.do(word nó);
begin
     R(nó) := vazio;
     if NR(nó) <= NC(nó)
     then begin
          for i:=1 to NR(nó) do
              R(nó):=R(nó) U ( C(nó,i) );
          primeira.solução.do:=verdade;
     end else primeira.solução.do:=falso;
end;
```

A rotina Retrocede, retrocede o nó atual a sua ligação mais perto, (na ordem dada pela ordenação inicial), que tem solução em conflito com a solução do nó atual. Se já está no primeiro nó, e tentar retroceder retorna com valor falso. Caso contrário, tenta obter a próxima solução do nó encontrado. Enquanto não conseguir continua a retroceder. Quando conseguir, retorna verdade.

```
lógical procedure retrocede;
begin
     flag:=falso;
     while flag = falso do
     if nó maior que 0
     then begin
          j:=-1;
          for k:=1 to nó-1 do
          if k adj no and R(k) n R(nó) diferente de vazio
          then j:=k;
          comment: ao sair deste loop, j será a ligação do
                   nó, cuja solução já foi encontrada, que
                   está mais perto dele e cuja solução
                   conflita com a do nó.
          if j maior ou igual a 0
          then nó:=j
          else nó:=n1-1;
          flag:=próxima.solução(nó);
     end else begin
          retrocede:=falso;
          return;
     end;
     retrocede:=verdade;
end;
```

A próxima.solução.do(nó) é uma pequena adaptação do algoritmo de geração das combinações de n caracteres de uma string alfa, em órdem lexicograficamente crescente.

```
lógical procedure próxima.solução.do(word nó);
begin
     word (*) a;
     for i:=1 to NR(nó) do
     for j:=1 to NC(nó) do
     if R(nó,i) = C(nó,j)
     then a(i):=j;
     comment: a(.) e o vetor dos índices das cores de C que
              ocorrem em R;
     a(NR(nó)+1):=NC(nó)+1;
     j:=-1;
     for i:=1 to NR(nó) do
     if a(i)+1 maior que a(i+1)
     then j:=i;
     if j   0
     then begin
          próxima.solução:=falso;
          return;
          comment: Isto ocorrerá quando a(1)=NC(nó)-NR(nó)+1,
                   a(2) = a(1)+1,...a(nr(nó))=NC(nó), o que
                   caracteriza a maior combinação
                   lexicograficamente ordenada do conjunto
                   C(nó);
     end;
     a(j):=a(j)+1;
     for i:=j+1 to NR(nó) do
     a(i):=a(i-1)+1;
     próxima.solução:=verdade;
end;
```

Demonstração dos algoritmos propostos;

Avança:	E trivial e imediata da proposição;
Primeira.solução:	Esta rotina é uma definição, há o que ser demonstrado;
Retrocede:	Também trivial e imediato da proposição.
Situação.da.solução. .atual.do	Novamente é trivial e imediato da proposição;
Próxima.solução.geral:	Já demonstrado;
Próxima.solução:	Vide apresentação e demonstração do algoritmo ALFA.

Apresentação do algorîtmo ALFA.

O algorîtmo ALFA, é um algorîtmo de geração de todas as combinações lexicograficamente crescente de p elementos de um conjunto com n elementos.

Seja um conjunto C, com n elementos, do qual se deseja obter todas as combinações, sem repetição, de p elementos. Suponhamos que C esteja ordenado de uma forma arbitrária. Atribui-se então aos elementos de C, um îndice que identifica sua ordem de aparição em C, e sem perda de generalidades passamos a reconhecer um elemento de C pelo îndice que lhe foi atribuido. Ou seja, C passa a corresponde ao conjunto (1,2,3,...,n).

Passamos a representar agora uma combinação qualquer de p elementos de C com o conjunto A=(a1,a2,...ap), onde ai = o elemento de C que faz parte da combinação representada, i E (1,...p); a(i) E (1,...n).

```
00    procedure ALFA;
01    begin
02          label L0;
03          for i=1 to p do a(i):=i;
04          comment: E esta é a menor combinação lexicogra-
05                   ficamente ordenada do conjunto C;
06          a(p+1):=n+1;
07    L0:   listar A;
08          j:=-1;
09          for i:=1 to p do
10          if a(i)+1 < a(i+1)
11          then j:=i;
12          if j < 0 then stop;
13          comment: Isto irá ocorrer quando a(1)=n-p+1, a(2)=
14                a(1)+1, ...a(p)=n,a(p+1)=n+1. E esta e a maior
15                combinação lexicograficamente ordenada de p
16                elementos do conjunto C.
17          a(j):=a(j)+1;
18          for i:=j+1 to p do
19          a(i):=a(i-1)+1;
20          goto L0;
21    end;
```

Este algorítmo pode ser encontrado no livro de Even(06), capítulo 2, páginas 32 a 37.

Demonstração do algorítmo ALFA.

Hipótese: O algorítmo ALFA gera todas as combinações de p elementos do conjunto C, com n elementos, em ordem lexicograficamente crescente. Assumindo-se que C tem seus elementos ordenados em ordem crescente e que n menor ou igual a p.

Tese: 1) A primeira combinação gerada pelo algorítmo A=(1,2,...p) é obviamente a menor combinação lexicograficamente ordenada de p elementos do conjunto C. Linhas 00-07.

2) Seja agora A=(a(1),a(2),...,a(p)) uma combinação qualquer gerada pelo algorítmo. Sabemos que a(1),a(2),...,a(p). Notar que existe a(p+1)=n+1 dado na linha 06.

Nas linhas 08-11, procura-se o maior i tal que a(i)+1 maior a(i+1), i maior ou igual a p. se não

for encontrado nenhum i,

então como a(i) E (1,...n), para i E (1,...p);

segue a(p)+1 = a(p+1) = n+1, implica a(p) = n
a(p-1)+1 = a(p) = n, implica a(p-1) = n-1
.....
a(1)+1 = a(2) = n-p, implica a(1) = n-p+1.

E esta é a maior combinação de p elementos de C lexicograficamente ordenada. Neste caso o algorítmo pára quando encontrada a maior combinação. Vide linhas 12-15

3) Tendo encontrado i tal que a(i)+1 maior que a(i+1), e i menor ou igual a p;

sendo i o maior onde esta relação ocorre,

então a(i+1)=a(i+2)-1;
a(i+2)=a(i+3)-1;
.....
a(p)=a(p+1)-1=n.

Para se gerar a próxima combinação de p elementos de C em ordem lexicograficamente crescente, se incrementarmos qualquer elemento a(j) onde i maior que j, então teremos que pelo menos fazer
a(j) = a(j)+1; a(j+1) = a(j)+1; ...a(p)=a(p)+1=n+1.
Como n+1 não pertence a C, a combinação montada não existe. Portanto j maior ou igual a i. Suponha j maior que i. Incrementando a(j), a menor combinação lexicograficamente ordenada de C a partir deste incremento será

A1= (a(1),a(2),...,a(j-1),a(j)+1,...a(p)).

Entretanto se tomarmos a combinação abaixo

A2= (a(1),a(2),...a(i-1),a(i)+1,a(i+1),...a(p)),

podemos observar que para a próxima combinação em ordem lexicograficamente crescente teremos que incrementar a(i)!

Uma vez incrementando a(i), a menor combinação lexicograficamente ordenada será dada por
A3 = (a(1),a(2),...,a(i-1),a(i)+1,a(i)+2,...a(i)+n-i+1)
e é exatamente esta a combinação gerada nas linhas 16-18 do algorítmo ALFA como próxima combinação.

Como já vimos, a menor combinação é gerada pelo algorítmo como a primeira combinação. A fase de geração da próxima combinação, gera a menor combinacao lexicografica-

mente ordenada maior que a atual. Com isto todas as combinações serão geradas. Como ao reconhecer a última combinação, (a maior), o algorítmo pára, então demonstramos o que desejávamos.

5.6. Características da implementação:

A implementação do algorítmo, foi feita usando um vetor em que todas as informações são armazenadas continuamente e apontadas por ponteiros auxiliares. Por exemplo, as primeiras "nnós" palavras deste vetor contêm os ponteiros para a relação das cores já alocadas a cada nó; ou seja, a primeira palavra é o ponteiro para a relação de cores do primeiro nó, e assim por diante. Salientando apenas que as relações de cores de cada nó também são colocadas neste mesmo vetor. Da mesma forma ficam armazenadas as relações de nós vizinhos, as soluções e os conjuntos de cores disponíveis de cada nó.

Também nesta região, fica uma área que corresponde à ordenação atual dos nós de LLG. A existência desta área simplifica o esforço que seria necessário para a ordenação dos nós de LLG provocar também a troca de posições dos ponteiros mencionados acima. Desta forma, sempre que se quizer conhecer as informações sobre o n-ésimo elemento da ordenação dada, basta entrar na n-ésima posição deste vetor e daí obtemos uma correlação com o nó correspondente de LLG.

A principal vantagem no uso de um vetor de memória consecutivo, no qual se armazenam as informações como apresentada, está no fato de o programa se dimensionar automáticamente (em termos de uso de memória), de acordo com o número de nós de G. Ficando desta forma desnecessária uma compilação e posterior montagem do mesmo, sempre que o número de nós ultrapasse uma determinada grandeza. Está claro que ainda assim estamos limitados à memória máxima de usuário suportada pelo equipamento hospedeiro, (no caso um C-530).

Temos apresentado então o algorítmo proposto, e como elemento auxiliar, segue um quadro de apresentação dos conjuntos de variávies usadas pelo algorítmo:

Definição dos conjuntos e variáveis usados no desenvolvimento do problema.

LLG	Grafo sobre o qual se desenrolara o problema.
V	Conjunto dos nós de LLG.
ZW	Conjunto dos arcos de LLG.
vi	Nó de V.
(v(i),v(j))	Arco de ZW.
P	Conjunto Universo das cores.
S	Conjunto dos conjuntos cores associadas nós de LLG.
S(v(i))	Conjunto das cores associadas ao nó v(i).
S(v(i),j)	J-ésima cor associada ao nó v(i)
NS	Conjunto das quant. cores associadas aos nós de LLG.
NS(v(i))	Número de cores associadas ao nó v(i).
C	Conjunto dos conjuntos cores dispon. aos nós de LLG.
C(v(i))	Conjunto das cores disponíveis ao nó v(i)
C(v(i),j)	J-ésima cor disponível ao nó v(i).
NC	Conjunto quant. cores disponíveis aos nós de LLG.
NC(v(i))	Número de cores disponíveis ao nó v(i).
R	Conjunto dos conjuntos de cores a serem associadas aos nós de LLG.
R(v(i))	Conjunto de cores a serem associadas ao nó v(i).
R(v(i),j)	J-ésima cor a ser associada ao nó v(i).
NR	Conjunto das quantidades de cores a serem associadas aos nós de LLG.
NR(v(i))	Núm. de cores a serem associadas ao nó v(i).
Lg(v(i))	Ou Ligação(v(i)) conjunto dos nós que se ligam com v(i).
Lg(v(i),j)	Ou Ligação(v(i),j), j-ésima lig. do nó v(i).
Nlg	Ou Número.de.ligações, é conjunto dos números de ligações em LLG.
Nlg(v(i))	Ou Número.de.ligações, é número de ligações de v(i).
Peso	Conjunto dos pesos atribuidos aos nós de LLg.
Peso(v(i))	Peso atribuido ao nó v(i).
P.cores	Ou Peso.das.cores, é o conjunto dos conjuntos de pesos atribuidos as cores disponíveis aos nós de LLG.
P.cores(v(i))	Ou Peso.das.cores(v(i)), é o conjunto dos pesos atribuidos às cores disponíveis ao nó v(i).
P.cores(v(i),j)	Ou Peso.das.cores(v(i),j), é o peso atribuido à j-ésima cor disponível ao nó v(i).

6. APERFEIÇOAMENTO DAS HEURÍSTICAS.

Para acompanhar a fase de depuração, é comum ser adicinado no algorítmo, algumas rotinas que listem no vídeo as principais variáveis do programa. Fazendo estas rotinas serem executadas em pontos estratégicos, é possível visualizar bem o seu comportamento.

Porém esta técnica, após a fase de depuração, no nosso caso, permite identificar alguns elementos que podem acelerar o processo. Em geral, estes elementos são de fácil persepção se acompanharmos este algorítmo atravéz do comportamentos das variáveis em observação.

6.1. O primeiro elemento identificado, foi na rotina de geração da próxima solução de um nó, que ao gerar uma solução, esta ainda continha cores conflitantes com as de sua vizinhança. Por exemplo, um determinado nó v(j) tem o conjunto de cores disponíveis dado por C=(1,2,3,4,5,6,7,8,9,10), e a solução atual é R=(1,4,5,6). Suponha que esta solução tenha as cores 4 e 5 conflitando com sua vizinhança. Se aplicarmos o algorítmo de geração da próxima solução de um nó, como apresentado no capítulo anterior, a nova solução será R=(1,4,5,7). Ora, esta solução obviamente continuará em conflito com a vizinhança já colorida, uma vez que no passo atual só estamos tentando colorir v(j), e como já sabemos, as cores 4 e 5 não poderão pertencer à solução deste nó.

Este problema ocorre porque os conjuntos C só são determinados no início do processo, e depois não mais alterados. Se eles fossem recalculados sempre que um nó vizinho obtivesse uma solução, C´=C-R(vizinho), o problema em questão não iria ocorrer. Notemos no entanto, que o algorítmo primeiramente quase duplicaria seu espaço em memória, pois os conjuntos C (disponibilidade), são responsáveis pela maior área de dados em memória. Finalmente, o tempo de processamento poderia aumentar bastante, pois a cada solução de um nó, teria que ser recalculado toda a disponibilidade de sua vizinhança.

A solução implementada foi uma alteração na rotina Situação.da.Solução.Atual, quando averigua-se em todo nó adjacente já colorido n(j) do nó atual, se a solução indicada para n(j) é incompatível, ou seja, R(no) n R(n(j)) não for vazio, determina-se a cor que tenha o menor índice, (na ordem em que elas aparecem na solução do nó atual), e que também esteja alocada à coloração de n(j).

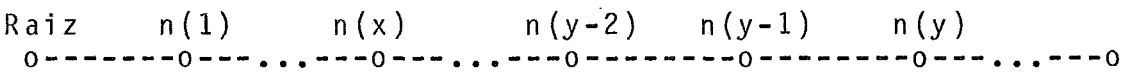
Uma vez de posse deste dado, altera-se a rotina Próxima.Solução.do.nó.Atual, de maneira que a próxima solução deste nó seja a de menor ordem lexicográfica, maior que a anterior, em que a referida cor não apareça.

Isto irá provocar uma aceleração horizontal no algorítmo que percorre a árvore da figura 17.

6.2. O segundo elemento identificado, é de concepção semelhante ao primeiro, só que aplicado aos nós e não às cores. Ou seja, ao identificar um nó n(j) que não consegue ser solucionado devido as soluções de seus antecessores, não adianta nada retroceder ao nó n(j-1) se este não for seu vizinho em LLG. Em vista disto, a solução dada foi fazer o algorítmo retroceder sempre ao nó vizinho de maior ordem, menor que a do nó atual, cuja solução é incompatível com a deste último.

Esta heurística tenta retroceder na árvore da figura 17, ao nível mais alto possível, de forma a ainda garantir não perder nenhuma solução possível. Na figura abaixo temos os níveis da árvore apresentados na horizontal,

Fig. 21



onde n(y) é o nível representando as soluções possíveis do nó atual, e n(x) o menor nível mais próximo de n(y) tal que se refere a um seu vizinho

Fica fácil notar que qualquer retrocesso do nível n(y) (nó atual), que não atinja ao menos o nível n(x) não

conseguirá desatar a incompatibiliadade de soluções do nó atual com a de seus ancestrais.

6.3. Como podemos notar no ítem anterior, a ordenação dos nós do grafo LLG, conforme implementada, não garante que 2 ou mais nós consecutivos sejam vizinhos em LLG. Portanto, o algorítmo perde tempo em procurar soluções para um nó que esteja "entre" dois outros vizinhos, e que estejam disputando cores para suas soluções. Ainda no ítem anterior, mostramos uma forma de "retroceder" com eficiência, evitando passar por nós que não iriam interferir na vizinhaça de um outro nó que não consegue ser solucionado. Ou seja, estão ocorrendo vários retrocessos de um nó para o outro antes que uma solução compatível seja encontrada para ambos.

É fácil mostrar que no pior caso, para se obter uma solução compatível para n nós, o algorítmo tem o tempo de processamento de ordem igual ao produto dos números de soluções possíveis para estes nós. Ou seja, cresce exponencialmente com o número de nós (n).

A idéia é grupar conjuntos de nós que estão tendo maior dificuldade de obter uma solução compatível, sem nenhum outro nó que os intercale na sequência de nós ordenada anteriormente. Contudo ordenar a priori os nós de forma a procurar atender este critério é impossível, uma vez que só durante a evolução do algorítmo é que se estebelece a dificuldade de se obter soluções compatáveis para dois vizinhos.

Podemos no entanto reordená-los durante o tempo em que se está determinando as soluções. Para isto foi feito o seguinte: sempre que se for executar um RETROCESSO, faz-se com que o nó atual se torne o seguinte ao nó para onde se vai retroceder, deslocando-se de uma posição todos os nós que os intercalavam na sequência da ordenação anterior. Ou seja, suponha uma sequência ordenada ABCDEFGHIJKL, que o nó atual é o L, e que o retrocesso será para o nó G; a nova sequência se tornará ABCDEFGLHIJK.

Esta heurística é semelhante à 6.2, que nos retrocessos do algorítmo evita perder tempo analisando nós que não estão no conflítuo atual, por não pertencerem a sua vizinhança. Evitando agora os vizinhos que não dão trabalho, (pouco conflitantes), e antecipando a disputa entre apenas os vizinhos mais conflitantes. Passemos a reconhecer doravante esta heurística por "Ordenação dinâmica dos nós".

Só resta mostrar que isto não afeta a garantia de que todas as soluções que ainda faltam, sejam encontradas. Isto não é difícil. Basta lembrar que primeiramente os nós menores que o vizinho para o qual se retrocedeu, não sofrem modificações na solução que lhes foi atribuida até agora. O vizinho para onde se retrocedeu vai mostrar sua próxima solução maior que a anterior. As rotinas AVANÇA e PROXIMA.SOLUÇÃO.DO.NÓ garantem que todas as soluções a partir do nó atual serão vistas. Assim nenhuma solução possível será perdida.

7. EXEMPLOS

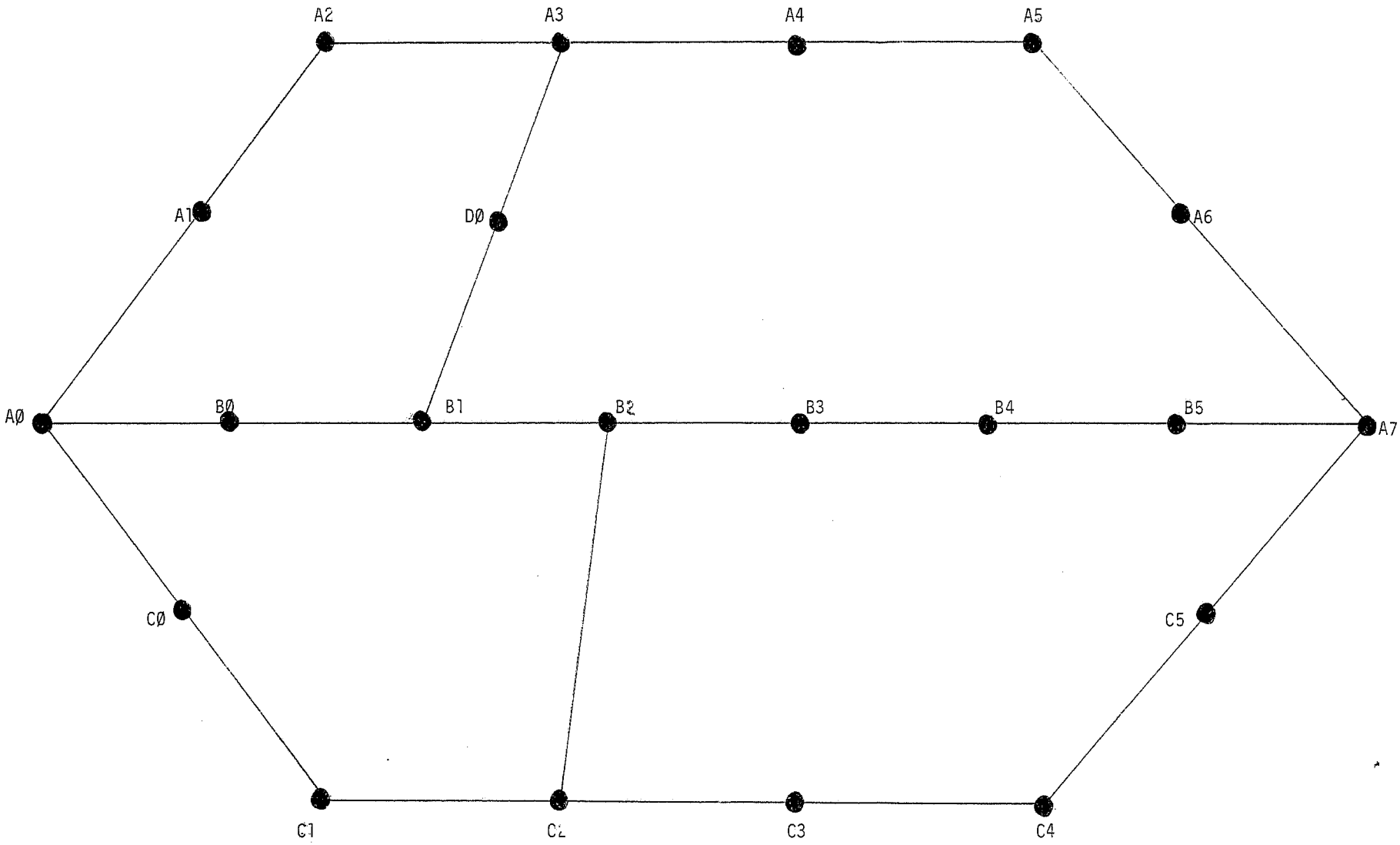
Foram rodados quatro exemplos básicos. O primeiro, segundo e terceiro foram casos hipotéticos. E o último foi baseado numa rede de transmissão de energia situada no sul do país, contendo por isto dimenções da ordem de grandeza desejadas para o problema em questão.

Estes exemplos foram repetidos em 3 casos cada. Sendo que cada caso fazia a ordenação dos nós de LLG de forma diferente. Ou seja, no caso 1, os nós foram ordenados em forma crescente de acôrdo com o peso definido pela heurística da ordenação dos nós. No segundo caso, os nós não sofreram ordenação nenhuma, e no terceiro caso, a ordenação inversa da sugerida pela heurística.

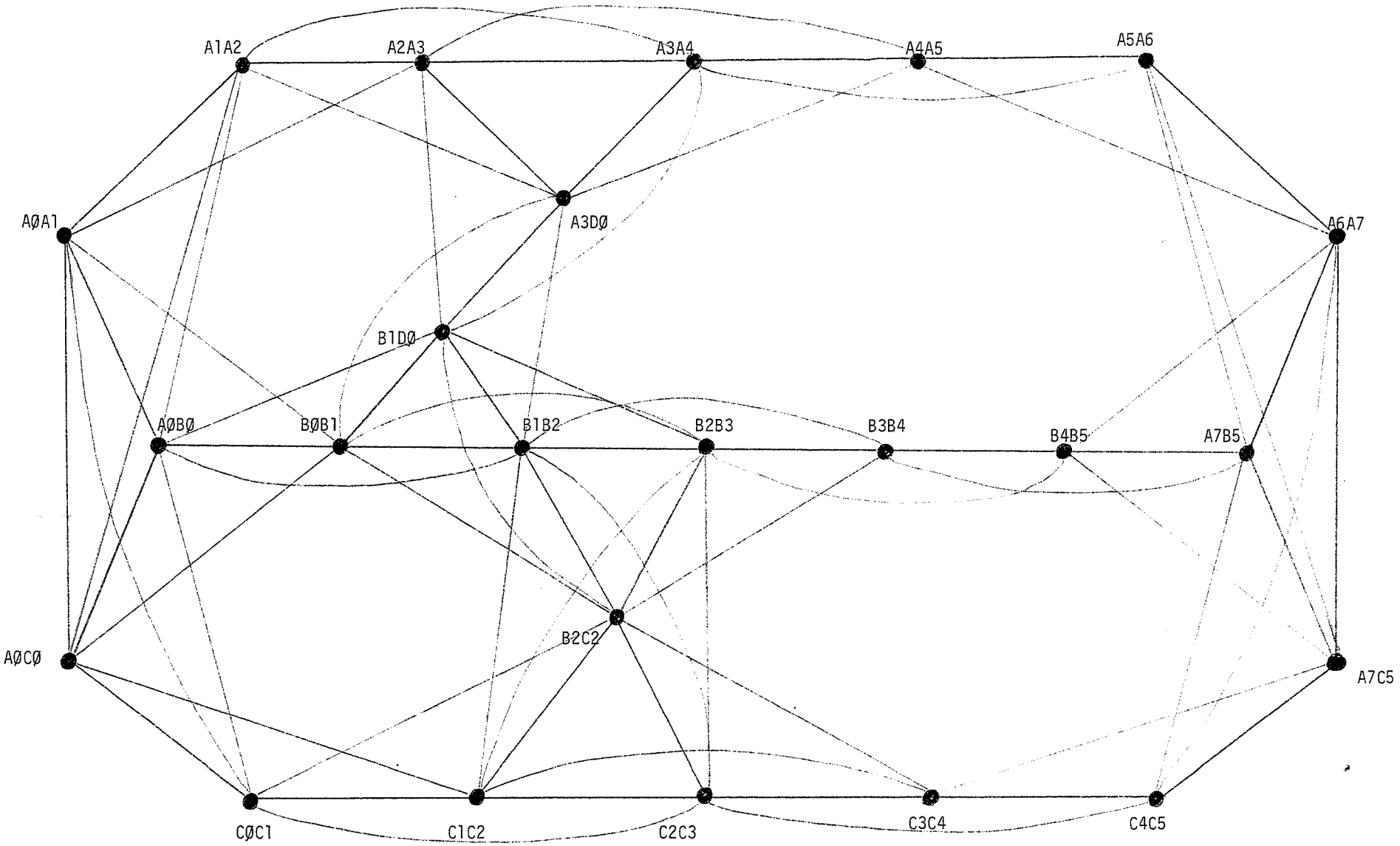
Cada caso destes, se dividiu ainda em mais tres sub-casos. Agora quanto a ordenação das cores disponíveis. No primeiro sub-caso, as cores foram ordenadas de forma crescente de acôrdo com os pesos definidos na heurística da ordenação das cores dispiníveis. O segundo sub-caso, sem ordenação nenhuma, e o terceiro com a ordenação inversa.

A seguir vem o grafo G referente aos exemplos 1, 2 e 3; e na página seguinte o grafo LLG formado a partir de G.

GRAFO G. EXEMPLOS 1, 2 e 3.



GRAFO LLG: EXEMPLOS 1, 2 e 3.



7.1.

Exemplo 1.

Tomando o grafo LLG mostrado na página anterior, todo branco, fez-se uma expansão, que no caso é uma multi-coloração, com as demandas dadas pelo vetor demanda abaixo:

D=(5,5,5,4,4,4,4,4,5,5,4,5,3,3,4,0,5,5,5,4,4,4,4,3)

Os tempos obtidos foram confome abaixo:

Caso 1.1: 4,026 segundos de cpu.

Caso 1.2: 4,025 segundos de cpu.

Caso 1.3: 4,029 segundos de cpu.

Caso 2.1: 4,222 segundos de cpu.

Caso 2.2: 4,217 segundos de cpu.

Caso 2.3: 4,218 segundos de cpu.

Caso 3.1: 4,086 segundos de cpu.

Caso 3.2: 4,085 segundos de cpu.

Caso 3.3: 4,083 segundos de cpu.

Observe que a ordenação das cores no caso de multi-coloração, (não de expansão da multi-coloração), não interferiu nos tempos de execução. E isto é bem razoável, posto que num grafo todo branco, todas as cores estarão igualmente disponíveis a todos os nós deste grafo.

Por outro lado, observa-se que a não ordenação dos nós representou uma situação pior que a ordenação inversa. Neste caso, podemos concluir que a aproximação dos nós com maior tendência ao conflito foi mais relevante para a multicoloração, (não expansão da multicoloração), do que a exigência da ordenação sugerida.

MULTI-COLORACAO DE UM GRAFO G. EXEMPLO I, CASO - 1.1 -

SOLUCAO NUMERO: 1

SOLUCOES PARA O NO A0A1:	0 >	1	2	3	4	5
SOLUCOES PARA O NO A0B0:	1 >	6	7	8	9	10
SOLUCOES PARA O NO A0C0:	2 >	11	12	13	14	15
SOLUCOES PARA O NO A1A2:	3 >	16	17	18	19	
SOLUCOES PARA O NO A2A3:	4 >	6	7	8	9	
SOLUCOES PARA O NO A3A4:	5 >	10	11	12	13	
SOLUCOES PARA O NO A3D0:	6 >	1	2	3	4	
SOLUCOES PARA O NO A4A5:	7 >	14	15	16	17	
SOLUCOES PARA O NO A5A6:	8 >	1	2	3	4	5
SOLUCOES PARA O NO A6A7:	9 >	6	7	8	9	10
SOLUCOES PARA O NO A7B5:	10 >	16	17	18	19	
SOLUCOES PARA O NO A7C5:	11 >	11	12	13	14	15
SOLUCOES PARA O NO B0B1:	12 >	18	19	20		
SOLUCOES PARA O NO B1B2:	13 >	21	22	23		
SOLUCOES PARA O NO B1D0:	14 >	14	15	16	17	
SOLUCOES PARA O NO B2C2:	16 >	1	2	3	4	5
SOLUCOES PARA O NO B3B4:	17 >	6	7	8	9	10
SOLUCOES PARA O NO B4B5:	18 >	1	2	3	4	5
SOLUCOES PARA O NO C0C1:	19 >	16	17	18	19	
SOLUCOES PARA O NO C1C2:	20 >	6	7	8	9	
SOLUCOES PARA O NO C2C3:	21 >	10	11	12	13	
SOLUCOES PARA O NO C3C4:	22 >	16	17	18	19	
SOLUCOES PARA O NO C4C5:	23 >	1	2	3		

MULTI-COLORACAO DE UM GRAFO G, EXEMPLO I, CASO - 1.2 -

SOLUCAO NUMERO: 1

SOLUCOES PARA O NO A0A1:	0 >	1	2	3	4	5
SOLUCOES PARA O NO A0B0:	1 >	6	7	8	9	10
SOLUCOES PARA O NO A0C0:	2 >	11	12	13	14	15
SOLUCOES PARA O NO A1A2:	3 >	16	17	18	19	
SOLUCOES PARA O NO A2A3:	4 >	6	7	8	9	
SOLUCOES PARA O NO A3A4:	5 >	10	11	12	13	
SOLUCOES PARA O NO A3D0:	6 >	1	2	3	4	
SOLUCOES PARA O NO A4A5:	7 >	14	15	16	17	
SOLUCOES PARA O NO A5A6:	8 >	1	2	3	4	5
SOLUCOES PARA O NO A6A7:	9 >	6	7	8	9	10
SOLUCOES PARA O NO A7B5:	10 >	16	17	18	19	
SOLUCOES PARA O NO A7C5:	11 >	11	12	13	14	15
SOLUCOES PARA O NO B0B1:	12 >	18	19	20		
SOLUCOES PARA O NO B1B2:	13 >	21	22	23		
SOLUCOES PARA O NO B1D0:	14 >	14	15	16	17	
SOLUCOES PARA O NO B2C2:	15 >	1	2	3	4	5
SOLUCOES PARA O NO B3B4:	17 >	6	7	8	9	10
SOLUCOES PARA O NO B4B5:	18 >	1	2	3	4	5
SOLUCOES PARA O NO C0C1:	19 >	16	17	18	19	
SOLUCOES PARA O NO C1C2:	20 >	6	7	8	9	
SOLUCOES PARA O NO C2C3:	21 >	10	11	12	13	
SOLUCOES PARA O NO C3C4:	22 >	16	17	18	19	
SOLUCOES PARA O NO C4C5:	23 >	1	2	3		

MULTI-COLORACAO DE UM GRAFO G. EXEMPLO I, CASO - 1.3 -

SOLUCAO NUMERO: 1

SOLUCOES PARA O NO A0A1:	0 >	1	2	3	4	5
SOLUCOES PARA O NO A0B0:	1 >	6	7	8	9	10
SOLUCOES PARA O NO A0C0:	2 >	11	12	13	14	15
SOLUCOES PARA O NO A1A2:	3 >	16	17	18	19	
SOLUCOES PARA O NO A2A3:	4 >	6	7	8	9	
SOLUCOES PARA O NO A3A4:	5 >	10	11	12	13	
SOLUCOES PARA O NO A3D0:	6 >	1	2	3	4	
SOLUCOES PARA O NO A4A5:	7 >	14	15	16	17	
SOLUCOES PARA O NO A5A6:	8 >	1	2	3	4	5
SOLUCOES PARA O NO A6A7:	9 >	6	7	8	9	10
SOLUCOES PARA O NO A7B5:	10 >	16	17	18	19	
SOLUCOES PARA O NO A7C5:	11 >	11	12	13	14	15
SOLUCOES PARA O NO B0B1:	12 >	18	19	20		
SOLUCOES PARA O NO B1B2:	13 >	21	22	23		
SOLUCOES PARA O NO B1D0:	14 >	14	15	16	17	
SOLUCOES PARA O NO B2C2:	15 >	1	2	3	4	5
SOLUCOES PARA O NO B3B4:	17 >	6	7	8	9	10
SOLUCOES PARA O NO B4B5:	18 >	1	2	3	4	5
SOLUCOES PARA O NO C0C1:	19 >	16	17	18	19	
SOLUCOES PARA O NO C1C2:	20 >	6	7	8	9	
SOLUCOES PARA O NO C2C3:	21 >	10	11	12	13	
SOLUCOES PARA O NO C3C4:	22 >	16	17	18	19	
SOLUCOES PARA O NO C4C5:	23 >	1	2	3		

MULTI-COLORACAO DE UM GRAFO G. EXEMPLO 1, CASO - 2.1 -

SOLUCAO NUMERO: 1

SOLUCOES PARA O NO A0A1:	0 >	1	2	3	4	5
SOLUCOES PARA O NO A0B0:	1 >	6	7	8	9	10
SOLUCOES PARA O NO A0C0:	2 >	11	12	13	14	15
SOLUCOES PARA O NO A1A2:	3 >	16	17	18	19	
SOLUCOES PARA O NO A2A3:	4 >	6	7	8	9	
SOLUCOES PARA O NO A3A4:	5 >	1	2	3	4	
SOLUCOES PARA O NO A3D0:	6 >	5	10	11	12	
SOLUCOES PARA O NO A4A5:	7 >	13	14	15	16	
SOLUCOES PARA O NO A5A6:	8 >	5	6	7	8	9
SOLUCOES PARA O NO A6A7:	9 >	1	2	3	4	10
SOLUCOES PARA O NO A7B5:	10 >	11	12	13	14	
SOLUCOES PARA O NO A7C5:	11 >	15	16	17	18	19
SOLUCOES PARA O NO B0B1:	12 >	16	17	18		
SOLUCOES PARA O NO B1B2:	13 >	1	2	3		
SOLUCOES PARA O NO B1D0:	14 >	13	14	15	19	
SOLUCOES PARA O NO B2C2:	16 >	4	5	6	7	8
SOLUCOES PARA O NO B3B4:	17 >	9	10	15	16	17
SOLUCOES PARA O NO B4B5:	18 >	5	6	7	8	20
SOLUCOES PARA O NO C0C1:	19 >	16	17	18	19	
SOLUCOES PARA O NO C1C2:	20 >	9	10	20	21	
SOLUCOES PARA O NO C2C3:	21 >	11	12	13	14	
SOLUCOES PARA O NO C3C4:	22 >	1	2	3	22	
SOLUCOES PARA O NO C4C5:	23 >	5	6	7		

MULTI-COLORACAO DE UM GRAFO G, EXEMPLO I, CASO - 2.2 -

SOLUCAO NUMERO: 1

SOLUCOES PARA O NO A0A1:	0 >	1	2	3	4	5
SOLUCOES PARA O NO A0B0:	1 >	6	7	8	9	10
SOLUCOES PARA O NO A0C0:	2 >	11	12	13	14	15
SOLUCOES PARA O NO A1A2:	3 >	16	17	18	19	
SOLUCOES PARA O NO A2A3:	4 >	6	7	8	9	
SOLUCOES PARA O NO A3A4:	5 >	1	2	3	4	
SOLUCOES PARA O NO A3D0:	6 >	5	10	11	12	
SOLUCOES PARA O NO A4A5:	7 >	13	14	15	16	
SOLUCOES PARA O NO A5A6:	8 >	5	6	7	8	9
SOLUCOES PARA O NO A6A7:	9 >	1	2	3	4	10
SOLUCOES PARA O NO A7B5:	10 >	11	12	13	14	
SOLUCOES PARA O NO A7C5:	11 >	15	16	17	18	19
SOLUCOES PARA O NO B0B1:	12 >	16	17	18		
SOLUCOES PARA O NO B1B2:	13 >	1	2	3		
SOLUCOES PARA O NO B1D0:	14 >	13	14	15	19	
SOLUCOES PARA O NO B2C2:	15 >	4	5	6	7	8
SOLUCOES PARA O NO B3B4:	17 >	9	10	15	16	17
SOLUCOES PARA O NO B4B5:	18 >	5	6	7	8	20
SOLUCOES PARA O NO C0C1:	19 >	16	17	18	19	
SOLUCOES PARA O NO C1C2:	20 >	9	10	20	21	
SOLUCOES PARA O NO C2C3:	21 >	11	12	13	14	
SOLUCOES PARA O NO C3C4:	22 >	1	2	3	22	
SOLUCOES PARA O NO C4C5:	23 >	5	6	7		

MULTI-COLORACAO DE UM GRAFO G. EXEMPLO I, CASO - 2.3 -

SOLUCAO NUMERO: 1

SOLUCOES PARA O NO A0A1:	0 >	1	2	3	4	5
SOLUCOES PARA O NO A0B0:	1 >	6	7	8	9	10
SOLUCOES PARA O NO A0C0:	2 >	11	12	13	14	15
SOLUCOES PARA O NO A1A2:	3 >	16	17	18	19	
SOLUCOES PARA O NO A2A3:	4 >	6	7	8	9	
SOLUCOES PARA O NO A3A4:	5 >	1	2	3	4	
SOLUCOES PARA O NO A3D0:	6 >	5	10	11	12	
SOLUCOES PARA O NO A4A5:	7 >	13	14	15	16	
SOLUCOES PARA O NO A5A6:	8 >	5	6	7	8	9
SOLUCOES PARA O NO A6A7:	9 >	1	2	3	4	10
SOLUCOES PARA O NO A7B5:	10 >	11	12	13	14	
SOLUCOES PARA O NO A7C5:	11 >	15	16	17	18	19
SOLUCOES PARA O NO B0B1:	12 >	16	17	18		
SOLUCOES PARA O NO B1B2:	13 >	1	2	3		
SOLUCOES PARA O NO B1D0:	14 >	13	14	15	19	
SOLUCOES PARA O NO B2C2:	16 >	4	5	6	7	8
SOLUCOES PARA O NO B3B4:	17 >	9	10	15	16	17
SOLUCOES PARA O NO B4B5:	18 >	5	6	7	8	20
SOLUCOES PARA O NO C0C1:	19 >	16	17	18	19	
SOLUCOES PARA O NO C1C2:	20 >	9	10	20	21	
SOLUCOES PARA O NO C2C3:	21 >	11	12	13	14	
SOLUCOES PARA O NO C3C4:	22 >	1	2	3	22	
SOLUCOES PARA O NO C4C5:	23 >	5	6	7		

MULTI-COLORACAO DE UM GRAFO G. EXEMPLO I, CASO - 3.1 -

SOLUCAO NUMERO: 1

SOLUCOES PARA O NO A0A1:	0 >	9	10	11	12	13
SOLUCOES PARA O NO A0B0:	1 >	7	8	14	15	16
SOLUCOES PARA O NO A0C0:	2 >	5	6	17	18	19
SOLUCOES PARA O NO A1A2:	3 >	1	2	3	4	
SOLUCOES PARA O NO A2A3:	4 >	5	6	7	8	
SOLUCOES PARA O NO A3A4:	5 >	9	10	11	12	
SOLUCOES PARA O NO A3D0:	6 >	13	14	15	16	
SOLUCOES PARA O NO A4A5:	7 >	1	2	3	4	
SOLUCOES PARA O NO A5A6:	8 >	8	13	14	15	16
SOLUCOES PARA O NO A6A7:	9 >	9	10	11	12	17
SOLUCOES PARA O NO A7B5:	10 >	4	5	6	7	
SOLUCOES PARA O NO A7C5:	11 >	18	19	20	21	22
SOLUCOES PARA O NO B0B1:	12 >	1	2	3		
SOLUCOES PARA O NO B1B2:	13 >	4	5	6		
SOLUCOES PARA O NO B1D0:	14 >	17	18	19	20	
SOLUCOES PARA O NO B2C2:	16 >	16	21	22	23	24
SOLUCOES PARA O NO B3B4:	17 >	1	2	3	8	9
SOLUCOES PARA O NO B4B5:	18 >	13	14	15	16	23
SOLUCOES PARA O NO C0C1:	19 >	1	2	3	4	
SOLUCOES PARA O NO C1C2:	20 >	7	8	9	10	
SOLUCOES PARA O NO C2C3:	21 >	11	12	13	14	
SOLUCOES PARA O NO C3C4:	22 >	4	5	6	15	
SOLUCOES PARA O NO C4C5:	23 >	1	2	3		

MULTI-COLORACAO DE UM GRAFO G. EXEMPLO 1, CASO - 3.2 -

SOLUCAO NUMERO: 1

SOLUCOES PARA O NO A0A1:	0	>	9	10	11	12	13
SOLUCOES PARA O NO A0B0:	1	>	7	8	14	15	16
SOLUCOES PARA O NO A0C0:	2	>	5	6	17	18	19
SOLUCOES PARA O NO A1A2:	3	>	1	2	3	4	
SOLUCOES PARA O NO A2A3:	4	>	5	6	7	8	
SOLUCOES PARA O NO A3A4:	5	>	9	10	11	12	
SOLUCOES PARA O NO A3D0:	6	>	13	14	15	16	
SOLUCOES PARA O NO A4A5:	7	>	1	2	3	4	
SOLUCOES PARA O NO A5A6:	8	>	8	13	14	15	16
SOLUCOES PARA O NO A6A7:	9	>	9	10	11	12	17
SOLUCOES PARA O NO A7B5:	10	>	4	5	6	7	
SOLUCOES PARA O NO A7C5:	11	>	18	19	20	21	22
SOLUCOES PARA O NO B0B1:	12	>	1	2	3		
SOLUCOES PARA O NO B1B2:	13	>	4	5	6		
SOLUCOES PARA O NO B1D0:	14	>	17	18	19	20	
SOLUCOES PARA O NO B2C2:	16	>	16	21	22	23	24
SOLUCOES PARA O NO B3B4:	17	>	1	2	3	8	9
SOLUCOES PARA O NO B4B5:	18	>	13	14	15	16	23
SOLUCOES PARA O NO C0C1:	19	>	1	2	3	4	
SOLUCOES PARA O NO C1C2:	20	>	7	8	9	10	
SOLUCOES PARA O NO C2C3:	21	>	11	12	13	14	
SOLUCOES PARA O NO C3C4:	22	>	4	5	6	15	
SOLUCOES PARA O NO C4C5:	23	>	1	2	3		

MULTI-COLORACAO DE UM GRAFO G. EXEMPLO I, CASO - 3.3 -

SOLUCAO NUMERO: 1

SOLUCOES PARA O NO A0A1:	0	>	9	10	11	12	13
SOLUCOES PARA O NO A0B0:	1	>	7	8	14	15	16
SOLUCOES PARA O NO A0C0:	2	>	5	6	17	18	19
SOLUCOES PARA O NO A1A2:	3	>	1	2	3	4	
SOLUCOES PARA O NO A2A3:	4	>	5	6	7	8	
SOLUCOES PARA O NO A3A4:	5	>	9	10	11	12	
SOLUCOES PARA O NO A3D0:	6	>	13	14	15	16	
SOLUCOES PARA O NO A4A5:	7	>	1	2	3	4	
SOLUCOES PARA O NO A5A6:	8	>	8	13	14	15	16
SOLUCOES PARA O NO A6A7:	9	>	9	10	11	12	17
SOLUCOES PARA O NO A7B5:	10	>	4	5	6	7	
SOLUCOES PARA O NO A7C5:	11	>	18	19	20	21	22
SOLUCOES PARA O NO B0B1:	12	>	1	2	3		
SOLUCOES PARA O NO B1B2:	13	>	4	5	6		
SOLUCOES PARA O NO B1D0:	14	>	17	18	19	20	
SOLUCOES PARA O NO B2C2:	16	>	16	21	22	23	24
SOLUCOES PARA O NO B3B4:	17	>	1	2	3	8	9
SOLUCOES PARA O NO B4B5:	18	>	13	14	15	16	23
SOLUCOES PARA O NO C0C1:	19	>	1	2	3	4	
SOLUCOES PARA O NO C1C2:	20	>	7	8	9	10	
SOLUCOES PARA O NO C2C3:	21	>	11	12	13	14	
SOLUCOES PARA O NO C3C4:	22	>	4	5	6	15	
SOLUCOES PARA O NO C4C5:	23	>	1	2	3		

7.2. Exemplo 2.

O exemplo 2, toma o mesmo grafo LLG do exemplo anterior, só que agora colorido com as cores sugeridas pela solução do caso 3.2 do exemplo 1. E a partir daí requisita uma expansão impossível da multi-coloração de LLG.

A demanda requisitada neste caso é dada pelo vetor demanda...

D=(1,2,1,1,3,1,1,4,1,2,3,2,2,1,1,2,1,1,2,1,2,3,3,1).

Os tempos obtidos são os mostrados abaixo:

Caso 1.1: 0,852 segundos de cpu.

Caso 1.2: 0,852 segundos de cpu.

Caso 1.3: 0,891 segundos de cpu.

Caso 2.1: 0,947 segundos de cpu.

Caso 2.2: 0,948 segundos de cpu.

Caso 2.3: 1,464 segundos de cpu.

Caso 3.1: 2,375 segundos de cpu.

Caso 3.2: 2,375 segundos de cpu.

Caso 3.3: 2,250 segundos de cpu.

7.3. Exemplo 3.

O terceiro exemplo é o mesmo que o segundo, só que desta vez, ampliando o universo de cores para 30 cores. Ou seja, com a mesma solução do exemplo 3 (caso 3.2), aplicada a LLG ainda branco, com uma expansão da demanda dada pelo vetor D abaixo:

D=(1,2,1,1,3,1,1,4,1,2,3,2,2,1,1,2,1,1,2,1,2,3,3,1).

No entanto com um universo de 30 cores, obteve-se os tempos abaixo:

Caso 1.1: 1,507 segundos de cpu.

Caso 1.2: 1,509 segundos de cpu.

Caso 1.3: 2,394 segundos de cpu.

Caso 2.1: 1,481 segundos de cpu.

Caso 2.2: 1,482 segundos de cpu.

Caso 2.3: 1,595 segundos de cpu.

Caso 3.1: 2,760 segundos de cpu.

Caso 3.2: 2,760 segundos de cpu.

Caso 3.3: 1,896 segundos de cpu.

Tanto no exemplo 2 como no 3, podemos observar que a ordenação crescente das cores quanto aos pesos sugeridos, ou sem ordenação não representou a menor diferença, independente da ordenação dada aos nós. A explicação deste fato pode ser reforçada com uma análise dos resultados no exemplo 4, onde a diferença nestes dois casos girou em torno de 10 por cento. Como nos dois exemplos em questão, o grafo tem dimensões reduzidas, os tempos se tornaram semelhantes, (imperseptivelmente diferentes). Lembrar que esta diferença deve crescer exponencialmente com o número

de nós de LLG.

A ordenação dos nós provocou realmente os resultados esperados, ou seja, o melhor tempo para a ordenação crescente, em seguida sem ordenação e o pior tempo para o de ordenação inversa.

Notemos que nestes dois exemplos o caso 3.3 foi melhor que o 3.1 e o 3.2. Ou seja, a ordenação inversa das cores foi melhor que a sugerida ou sem ordenação. Para justificar este fato convém levantar os seguintes pontos:

a) Os casos 3.x de todos os exemplos, tem a ordenação dos nós invertida em relação a sugerida. Ou seja, estaremos provocando podas muito pequenas, (vide a fig. 17 e as explicações da heurística da ordenação dos nós, cap 5).

b) A heurística da ordenação das cores disponíveis, visa tentar pegar primeiramente, o conjunto de cores das disponíveis, que mais provavelmente apresentará uma solução não conflitante com sua vizinhança. A inversão desta heurística (casos x.3), está justamente reconhecendo então cores que deverão se tornar incompatíveis as soluções de seus vizinhos, e assim identificando uma poda antecipadamente.

c) O grafo em questão tem dimensões reduzidas.

As observações (a) e (b), nos leva a concluir que a inversão das duas heurísticas retorna a uma situação que corresponde a idéia da primeira heurística; que é reconhecer o mais cedo possível uma poda. No entanto isto só deve ser válido para grafos de dimensões reduzidas (c), ou muito congestinados, pois senão a observação (b) será menos relevante que os efeitos da ordenação inversa dos nós. Nunca nos esquecendo também que ainda assim, os tempos nestes casos não foram melhores que os do caso 1.1, (das heurísticas sugeridas).

As soluções encontradas são mostradas a seguir.

EXPANSAO DA MULTI-COLORACAO DOS NOS DE UM GRAFO G, EXEMPLO III, CASO - 1.1 -

SOLUCAO NUMERO: 1

SOLUCOES PARA O NO A0A1:	0	>	25			
SOLUCOES PARA O NO A0B0:	1	>	20	21		
SOLUCOES PARA O NO A0C0:	2	>	22			
SOLUCOES PARA O NO A1A2:	3	>	24			
SOLUCOES PARA O NO A2A3:	4	>	21	22	23	
SOLUCOES PARA O NO A3A4:	5	>	25			
SOLUCOES PARA O NO A3D0:	6	>	27			
SOLUCOES PARA O NO A4A5:	7	>	17	18	19	20
SOLUCOES PARA O NO A5A6:	8	>	28			
SOLUCOES PARA O NO A6A7:	9	>	24	25		
SOLUCOES PARA O NO A7B5:	10	>	22	21	23	
SOLUCOES PARA O NO A7C5:	11	>	26	27		
SOLUCOES PARA O NO B0B1:	12	>	24	26		
SOLUCOES PARA O NO B1B2:	13	>	25			
SOLUCOES PARA O NO B1D0:	14	>	28			
SOLUCOES PARA O NO B2B3:	15	>	22	23		
SOLUCOES PARA O NO B2C2:	16	>	30			
SOLUCOES PARA O NO B3B4:	17	>	18			
SOLUCOES PARA O NO B4B5:	18	>	28	29		
SOLUCOES PARA O NO C0C1:	19	>	23			
SOLUCOES PARA O NO C1C2:	20	>	28	29		
SOLUCOES PARA O NO C2C3:	21	>	15	26	27	
SOLUCOES PARA O NO C3C4:	22	>	23	24	25	
SOLUCOES PARA O NO C4C5:	23	>	8			

EXPANSAO DA MULTI-COLORACAO DOS NOS DE UM GRAFO G. EXEMPLO III, CASO - 1.2 -

SOLUCAO NUMERO: 1

SOLUCOES PARA O NO A0A1:	0	>	25			
SOLUCOES PARA O NO A0B0:	1	>	20	21		
SOLUCOES PARA O NO A0C0:	2	>	22			
SOLUCOES PARA O NO A1A2:	3	>	24			
SOLUCOES PARA O NO A2A3:	4	>	21	22	23	
SOLUCOES PARA O NO A3A4:	5	>	25			
SOLUCOES PARA O NO A3D0:	6	>	27			
SOLUCOES PARA O NO A4A5:	7	>	17	18	19	20
SOLUCOES PARA O NO A5A6:	8	>	28			
SOLUCOES PARA O NO A6A7:	9	>	24	25		
SOLUCOES PARA O NO A7B5:	10	>	21	22	23	
SOLUCOES PARA O NO A7C5:	11	>	26	27		
SOLUCOES PARA O NO B0B1:	12	>	24	26		
SOLUCOES PARA O NO B1B2:	13	>	25			
SOLUCOES PARA O NO B1D0:	14	>	28			
SOLUCOES PARA O NO B2B3:	15	>	22	23		
SOLUCOES PARA O NO B2C2:	16	>	30			
SOLUCOES PARA O NO B3B4:	17	>	18			
SOLUCOES PARA O NO B4B5:	18	>	28	29		
SOLUCOES PARA O NO C0C1:	19	>	23			
SOLUCOES PARA O NO C1C2:	20	>	28	29		
SOLUCOES PARA O NO C2C3:	21	>	15	26	27	
SOLUCOES PARA O NO C3C4:	22	>	23	24	25	
SOLUCOES PARA O NO C4C5:	23	>	8			

EXPANSAO DA MULTI-COLORACAO DOS NOS DE UM GRAFO G. EXEMPLO III, CASO - 1.3 -

SOLUCAO NUMERO: 1

SOLUCOES PARA O NO A0A1:	0	>	28			
SOLUCOES PARA O NO A0B0:	1	>	22	23		
SOLUCOES PARA O NO A0C0:	2	>	24			
SOLUCOES PARA O NO A1A2:	3	>	27			
SOLUCOES PARA O NO A2A3:	4	>	20	25	26	
SOLUCOES PARA O NO A3A4:	5	>	28			
SOLUCOES PARA O NO A3D0:	6	>	29			
SOLUCOES PARA O NO A4A5:	7	>	21	22	23	24
SOLUCOES PARA O NO A5A6:	8	>	29			
SOLUCOES PARA O NO A6A7:	9	>	25	28		
SOLUCOES PARA O NO A7B5:	10	>	21	23	24	
SOLUCOES PARA O NO A7C5:	11	>	26	27		
SOLUCOES PARA O NO B0B1:	12	>	26	27		
SOLUCOES PARA O NO B1B2:	13	>	25			
SOLUCOES PARA O NO B1D0:	14	>	21			
SOLUCOES PARA O NO B2B3:	15	>	23	24		
SOLUCOES PARA O NO B2C2:	16	>	30			
SOLUCOES PARA O NO B3B4:	17	>	26			
SOLUCOES PARA O NO B4B5:	18	>	29	30		
SOLUCOES PARA O NO C0C1:	19	>	25			
SOLUCOES PARA O NO C1C2:	20	>	28	29		
SOLUCOES PARA O NO C2C3:	21	>	26	27	15	
SOLUCOES PARA O NO C3C4:	22	>	23	24	25	
SOLUCOES PARA O NO C4C5:	23	>	29			

EXPANSAO DA MULTI-COLORACAO DOS NOS DE UM GRAFO G. EXEMPLO III, CASO - 2.1 -

SOLUCAO NUMERO: 1

SOLUCOES PARA O NO A0A1:	0	>	20			
SOLUCOES PARA O NO A0B0:	1	>	21	22		
SOLUCOES PARA O NO A0C0:	2	>	23			
SOLUCOES PARA O NO A1A2:	3	>	24			
SOLUCOES PARA O NO A2A3:	4	>	21	22	23	
SOLUCOES PARA O NO A3A4:	5	>	20			
SOLUCOES PARA O NO A3D0:	6	>	25			
SOLUCOES PARA O NO A4A5:	7	>	17	18	19	24
SOLUCOES PARA O NO A5A6:	8	>	22			
SOLUCOES PARA O NO A6A7:	9	>	21	23		
SOLUCOES PARA O NO A7B5:	10	>	24	25	26	
SOLUCOES PARA O NO A7C5:	11	>	27	28		
SOLUCOES PARA O NO B0B1:	12	>	24	26		
SOLUCOES PARA O NO B1B2:	13	>	23			
SOLUCOES PARA O NO B1D0:	14	>	27			
SOLUCOES PARA O NO B2B3:	15	>	22	25		
SOLUCOES PARA O NO B2C2:	16	>	28			
SOLUCOES PARA O NO B3B4:	17	>	18			
SOLUCOES PARA O NO B4B5:	18	>	29	30		
SOLUCOES PARA O NO C0C1:	19	>	24			
SOLUCOES PARA O NO C1C2:	20	>	26	27		
SOLUCOES PARA O NO C2C3:	21	>	15	29	30	
SOLUCOES PARA O NO C3C4:	22	>	23	24	25	
SOLUCOES PARA O NO C4C5:	23	>	8			

EXPANSAO DA MULTI-COLORACAO DOS NOS DE UM GRAFO G. EXEMPLO III, CASO - 2.2 -

SOLUCAO NUMERO: 1

SOLUCOES PARA O NO A0A1:	0	>	20			
SOLUCOES PARA O NO A0B0:	1	>	21	22		
SOLUCOES PARA O NO A0C0:	2	>	23			
SOLUCOES PARA O NO A1A2:	3	>	24			
SOLUCOES PARA O NO A2A3:	4	>	21	22	23	
SOLUCOES PARA O NO A3A4:	5	>	20			
SOLUCOES PARA O NO A3D0:	6	>	25			
SOLUCOES PARA O NO A4A5:	7	>	17	18	19	24
SOLUCOES PARA O NO A5A6:	8	>	21			
SOLUCOES PARA O NO A6A7:	9	>	22	23		
SOLUCOES PARA O NO A7B5:	10	>	24	25	26	
SOLUCOES PARA O NO A7C5:	11	>	27	28		
SOLUCOES PARA O NO B0B1:	12	>	24	26		
SOLUCOES PARA O NO B1B2:	13	>	23			
SOLUCOES PARA O NO B1D0:	14	>	27			
SOLUCOES PARA O NO B2B3:	15	>	22	25		
SOLUCOES PARA O NO B2C2:	16	>	28			
SOLUCOES PARA O NO B3B4:	17	>	18			
SOLUCOES PARA O NO B4B5:	18	>	21	29		
SOLUCOES PARA O NO C0C1:	19	>	24			
SOLUCOES PARA O NO C1C2:	20	>	26	27		
SOLUCOES PARA O NO C2C3:	21	>	15	29	30	
SOLUCOES PARA O NO C3C4:	22	>	23	24	25	
SOLUCOES PARA O NO C4C5:	23	>	8			

EXPANSAO DA MULTI-COLORACAO DOS NOS DE UM GRAFO G, EXEMPLO III, CASO - 2.3 -

SOLUCAO NUMERO: 1

SOLUCOES PARA O NO A0A1:	0	>	23			
SOLUCOES PARA O NO A0B0:	1	>	22	25		
SOLUCOES PARA O NO A0C0:	2	>	26			
SOLUCOES PARA O NO A1A2:	3	>	24			
SOLUCOES PARA O NO A2A3:	4	>	20	21	22	
SOLUCOES PARA O NO A3A4:	5	>	23			
SOLUCOES PARA O NO A3D0:	6	>	25			
SOLUCOES PARA O NO A4A5:	7	>	24	26	27	28
SOLUCOES PARA O NO A5A6:	8	>	21			
SOLUCOES PARA O NO A6A7:	9	>	23	25		
SOLUCOES PARA O NO A7B5:	10	>	24	26	27	
SOLUCOES PARA O NO A7C5:	11	>	28	29		
SOLUCOES PARA O NO B0B1:	12	>	24	27		
SOLUCOES PARA O NO B1B2:	13	>	23			
SOLUCOES PARA O NO B1D0:	14	>	26			
SOLUCOES PARA O NO B2B3:	15	>	25	22		
SOLUCOES PARA O NO B2C2:	16	>	30			
SOLUCOES PARA O NO B3B4:	17	>	28			
SOLUCOES PARA O NO B4B5:	18	>	30	21		
SOLUCOES PARA O NO C0C1:	19	>	24			
SOLUCOES PARA O NO C1C2:	20	>	27	28		
SOLUCOES PARA O NO C2C3:	21	>	26	29	15	
SOLUCOES PARA O NO C3C4:	22	>	23	24	25	
SOLUCOES PARA O NO C4C5:	23	>	30			

EXPANSAO DA MULTI-COLORACAO DOS NOS DE UM GRAFO G. EXEMPLO III, CASO - 3.1 -

SOLUCAO NUMERO: 1

SOLUCOES PARA O NO A0A1:	0	>	20			
SOLUCOES PARA O NO A0B0:	1	>	24	25		
SOLUCOES PARA O NO A0C0:	2	>	22			
SOLUCOES PARA O NO A1A2:	3	>	21			
SOLUCOES PARA O NO A2A3:	4	>	23	24	25	
SOLUCOES PARA O NO A3A4:	5	>	20			
SOLUCOES PARA O NO A3D0:	6	>	22			
SOLUCOES PARA O NO A4A5:	7	>	17	18	19	26
SOLUCOES PARA O NO A5A6:	8	>	22			
SOLUCOES PARA O NO A6A7:	9	>	21	23		
SOLUCOES PARA O NO A7B5:	10	>	27	28	29	
SOLUCOES PARA O NO A7C5:	11	>	25	26		
SOLUCOES PARA O NO B0B1:	12	>	26	27		
SOLUCOES PARA O NO B1B2:	13	>	23			
SOLUCOES PARA O NO B1D0:	14	>	21			
SOLUCOES PARA O NO B2B3:	15	>	28	29		
SOLUCOES PARA O NO B2C2:	16	>	24			
SOLUCOES PARA O NO B3B4:	17	>	18			
SOLUCOES PARA O NO B4B5:	18	>	22	24		
SOLUCOES PARA O NO C0C1:	19	>	23			
SOLUCOES PARA O NO C1C2:	20	>	25	26		
SOLUCOES PARA O NO C2C3:	21	>	15	27	30	
SOLUCOES PARA O NO C3C4:	22	>	23	28	29	
SOLUCOES PARA O NO C4C5:	23	>	8			

EXPANSAO DA MULTI-COLORACAO DOS NOS DE UM GRAFO G. EXEMPLO III, CASO - 3.2 -

SOLUCAO NUMERO: 1

SOLUCOES PARA O NO A0A1:	0	>	20			
SOLUCOES PARA O NO A0B0:	1	>	24	25		
SOLUCOES PARA O NO A0C0:	2	>	22			
SOLUCOES PARA O NO A1A2:	3	>	21			
SOLUCOES PARA O NO A2A3:	4	>	23	24	25	
SOLUCOES PARA O NO A3A4:	5	>	20			
SOLUCOES PARA O NO A3D0:	6	>	22			
SOLUCOES PARA O NO A4A5:	7	>	17	18	19	26
SOLUCOES PARA O NO A5A6:	8	>	21			
SOLUCOES PARA O NO A6A7:	9	>	22	23		
SOLUCOES PARA O NO A7B5:	10	>	26	28	29	
SOLUCOES PARA O NO A7C5:	11	>	25	27		
SOLUCOES PARA O NO B0B1:	12	>	26	27		
SOLUCOES PARA O NO B1B2:	13	>	23			
SOLUCOES PARA O NO B1D0:	14	>	21			
SOLUCOES PARA O NO B2B3:	15	>	22	28		
SOLUCOES PARA O NO B2C2:	16	>	24			
SOLUCOES PARA O NO B3B4:	17	>	18			
SOLUCOES PARA O NO B4B5:	18	>	21	24		
SOLUCOES PARA O NO C0C1:	19	>	23			
SOLUCOES PARA O NO C1C2:	20	>	29	30		
SOLUCOES PARA O NO C2C3:	21	>	15	25	27	
SOLUCOES PARA O NO C3C4:	22	>	23	26	28	
SOLUCOES PARA O NO C4C5:	23	>	8			

EXPANSAO DA MULTI-COLORACAO DOS NOS DE UM GRAFO G. EXEMPLO III, CASO - 3.3 -

SOLUCAO NUMERO: 1

SOLUCOES PARA O NO A0A1:	0	>	22			
SOLUCOES PARA O NO A0B0:	1	>	28	29		
SOLUCOES PARA O NO A0C0:	2	>	24			
SOLUCOES PARA O NO A1A2:	3	>	23			
SOLUCOES PARA O NO A2A3:	4	>	21	24	25	
SOLUCOES PARA O NO A3A4:	5	>	20			
SOLUCOES PARA O NO A3D0:	6	>	22			
SOLUCOES PARA O NO A4A5:	7	>	27	28	29	30
SOLUCOES PARA O NO A5A6:	8	>	23			
SOLUCOES PARA O NO A6A7:	9	>	21	26		
SOLUCOES PARA O NO A7B5:	10	>	28	29	30	
SOLUCOES PARA O NO A7C5:	11	>	24	27		
SOLUCOES PARA O NO B0B1:	12	>	26	27		
SOLUCOES PARA O NO B1B2:	13	>	25			
SOLUCOES PARA O NO B1D0:	14	>	23			
SOLUCOES PARA O NO B2B3:	15	>	28	22		
SOLUCOES PARA O NO B2C2:	16	>	29			
SOLUCOES PARA O NO B3B4:	17	>	24			
SOLUCOES PARA O NO B4B5:	18	>	23	25		
SOLUCOES PARA O NO C0C1:	19	>	25			
SOLUCOES PARA O NO C1C2:	20	>	23	27		
SOLUCOES PARA O NO C2C3:	21	>	24	30	15	
SOLUCOES PARA O NO C3C4:	22	>	25	26	28	
SOLUCOES PARA O NO C4C5:	23	>	23			

7.4. Exemplo 4.

Este é um exemplo de dimensões reais; isto é, com 173 nós (LLG) e com universo de 100 cores.

Os tempos obtidos para cada um dos casos, estão mostrados a seguir.

Caso 1.1: 164,062 segundos de cpu.

Caso 1.2: 173,407 segundos de cpu.

Caso 1.3: 338,737 segundos de cpu.

Caso 2.1: 174,236 segundos de cpu.

Caso 2.2: maior que 1800,000 segundos de cpu, e sem obter solução.

Caso 2.3: 299,013 segundos de cpu.

Caso 3.1: 177,246 segundos de cpu.

Caso 3.2: 199,833 segundos de cpu.

Caso 3.3: 233,733 segundos de cpu.

Neste exemplo é importante lembrar que em todos os casos foram feitas as transformações de G para LLG.

8. CONCLUSÕES

Podemos observar pela formulação dada ao problema proposto, que o algorítmo apresentado soluciona problemas gerais de alocação de recursos com restrições que incluam elementos associados a vértices distintos.

Em particular, o problema da "patrulhinha", é um caso análogo, e que passa a ter mais esta ferramenta disponível para sua resolução. Este problema é basicamente a alocação de canais de telecomunicações a setores (bairros), de uma determinada região (cidade), de tal forma que dois setores vizinhos não usem o mesmo canal sob pena de interferências. Estes canais serão usados para a comunicação entre estações móveis e uma central, ou entre si. Este complexo pode pertencer a grandes empresas de transporte, (taxis, caminhões e etc), empresas de grande porte em geral, ou a setores da administração municipal, como é o caso das viaturas da polícia, as "patrulhinhas". É fácil notar que a "multicoloração", alocação de mais de um canal por setor, e a sua expansão, fazem bastante sentido neste problema.

Desta forma não estamos presos a um tratamento específico de um determinado problema, mas sim envoltos com um algorítmo tipicamente Branch-and-Bound com algumas heurísticas aplicadas.

Delas podemos destacar a ordenação dinâmica dos nós, que é a principal responsável pela melhoria do desempenho do algorítmo, e em algumas circunstâncias, a que viabilizou sua execução em termos de tempo.

Outra heurística que merece destaque é a da ordenação das cores disponíveis dos nós, conjugado com o algorítmo de obtenção das combinações em ordem lexicograficamente crescente dos elementos deste conjunto em grupos de NR cores. Observando os pesos atribuidos às cores disponíveis, veremos (conforme apresentado no capítulo 5.4) que as cores de menor ordem são as que menos esgotam as disponibilidades da vizinhança deste nó. Assim sendo, a idéia de se tentar colorir o nó em questão com esta cor, em princípio está preservando maior capacidade de colorir os vizinhos; e se estes já estão coloridos, preserva maior ca-

pacidade de uma futura expansão de sua coloração; estabelecendo desta forma um sentido de orimalidade ao algorítmo.

Em relção especificamente ao problema proposto, seria conveniente observar a vantagem do uso do grafo adjunto transformado (LLG), ao invéz da própria rede G. Enquanto LLG tem o número de nós igual ao número de arcos em G, (em geral os nós de G tem baixo grau de incidência), se o algorítmo trabalhasse sobre G, teria seu tempo de processamento razoavelmente aumentado, uma vez que em G teria que analizar os arcos de uma distância de 1 e 2 do arco original. Uma outra vantagem reside na universalidade da formulação obtida, pois em G são os arcos e não os nós as unidades de informação. Uma vez obtido LLG, a formulação fica independente da distância em a interferência pode agir.

9.

REFERÊNCIAS BIBLIOGRÁFICAS:

(01) Santos, Rosângela da Nóbrega: Tese de Mestrado na Coordenação dos Programas de Pósgraduação em Engenharia, COPPE. "Obtenção do número cromático de um grafo".

(02) Matula, D.W.; Marble, G.; Isaacson, J. D.: em "Graph coloring algorithms" da Graph Theory and Computing. Editado pela R.C. Read Academic Press, New York,pags pags 109-122. 1972 apendice s.

(03) Wood, D.C.: em "A technique for colouring a graph aplicable to large scale timetabling problems". The computer journal v 12,pag 317-319; 1969, apêndice D.

(04) Knuth, Donald E., em "The art of computer programing" volume 1. Editado pela Addison-Wesley publishing co.

(05) Berztiss, A. T. , em "Data Structures, theory and pratice". Editado pela Academic Press, N York 1975.

(06) Even, S., "Algorithmic combinatorics", editado pela MacMillan Company em New York 1973.

(07) Deo,N, "Graph theory with aplications to engineering and computer science", editado pela Prentice Hall.

```
0001          1
0002        173
0003        100
0004   A0A1
0005   A0L4
0006   A0M0
0007   A1A2
0008   A1C8
0009   A1F5
0010   A1M1
0011   A2L6
0012   A3A4
0013   A3A5
0014   A3S0
0015   A3U0
0016   A4A5
0017   A4A6
0018   A4A7
0019   A4G2
0020   A4S5
0021   A4U1
0022   A5A8
0023   A5B1
0024   A5B6
0025   A5CL
0026   A5I7
0027   A6CN
0028   A7P6
0029   A7S7
0030   A8F9
0031   A9L5
0032   B0C4
0033   B1C1
0034   B2F0
0035   B3B4
0036   B3B5
0037   B3B6
0038   B3S1
0039   B4B5
0040   B4B6
0041   B4I1
0042   B4T2
0043   B5B6
0044   B5J4
0045   B5J7
0046   B6CL
0047   B7I0
0048   C0C1
0049   C0C9
0050   C0SJ
0051   C0T0
0052   C2C3
0053   C2SE
0054   C3CH
0055   C3P7
0056   C4C5
0057   C4C6
0058   C4P4
0059   C4R0
0060   C5CA
```

```
0061  C5CK
0062  C5P6
0063  C5U2
0064  C7C8
0065  C7M1
0066  C7S5
0067  CAU2
0068  CBS5
0069  CCF0
0070  CDX0
0071  CECF
0072  CEG1
0073  CFCH
0074  CFI5
0075  CFSK
0076  CGCH
0077  CGSD
0078  CHF3
0079  CHI5
0080  CHP2
0081  CHSG
0082  CHSH
0083  CHSI
0084  CHSN
0085  CHV0
0086  CIL3
0087  CIP7
0088  CJSL
0089  CKCL
0090  CKCM
0091  CKCN
0092  CKU2
0093  CLC0
0094  CMJ4
0095  E0P0
0096  F0F1
0097  F1F2
0098  F1F3
0099  F1P1
0100  F1SG
0101  F1SM
0102  F2SK
0103  F4X0
0104  F5P6
0105  F5X3
0106  F6L3
0107  F7F8
0108  F7U4
0109  F8I1
0110  F8I3
0111  F8J6
0112  FAS3
0113  G0J5
0114  G3P0
0115  H0L1
0116  H0X1
0117  H1J5
0118  I0I1
0119  I0S8
0120  I1J3
```

```
0121   I2I3
0122   I2L2
0123   I2T1
0124   I3J6
0125   I4P5
0126   I5P2
0127   I5S9
0128   I6I7
0129   I6L5
0130   I7S7
0131   J0S3
0132   J1J2
0133   J2J3
0134   J2J4
0135   J3J4
0136   J3SB
0137   J3SF
0138   J5J6
0139   J5J7
0140   J6J7
0141   J6L1
0142   J6SC
0143   J7SM
0144   L0L1
0145   L100
0146   L1SC
0147   L5L6
0148   M0M1
0149   M1S2
0150   M2X1
0151   P0P1
0152   P1SA
0153   P1X2
0154   P2SA
0155   P2V0
0156   P3S5
0157   P3X2
0158   P5P6
0159   S3S4
0160   S3S5
0161   S4S5
0162   S5S6
0163   S5X2
0164   S6S7
0165   SGSH
0166   SJSK
0167   SKSN
0168   SLSM
0169   T1T2
0170   T1U5
0171   U3X0
0172   X0X1
0173   X0X2
0174   X1X2
0175   Z0Z1
0176   Z1Z2
0177        13
0178   A0L4
0179   A0M0
0180   A1A2
```

```
0181   A1C8
0182   A1F5
0183   A1M1
0184   A2L6
0185   C7C8
0186   C7M1
0187   F5P6
0188   F5X3
0189   M0M1
0190   M1S2
0191        7
0192   A0A1
0193   A0M0
0194   A1A2
0195   A1C8
0196   A1F5
0197   A1M1
0198   M0M1
0199        9
0200   A0A1
0201   A0L4
0202   A1A2
0203   A1C8
0204   A1F5
0205   A1M1
0206   C7M1
0207   M0M1
0208   M1S2
0209       14
0210   A0A1
0211   A0L4
0212   A0M0
0213   A1C8
0214   A1F5
0215   A1M1
0216   A2L6
0217   C7C8
0218   C7M1
0219   F5P6
0220   F5X3
0221   L5L6
0222   M0M1
0223   M1S2
0224       14
0225   A0A1
0226   A0L4
0227   A0M0
0228   A1A2
0229   A1F5
0230   A1M1
0231   A2L6
0232   C7C8
0233   C7M1
0234   C7S5
0235   F5P6
0236   F5X3
0237   M0M1
0238   M1S2
0239       16
0240   A0A1
```

0241	A0L4	
0242	A0M0	
0243	A1A2	
0244	A1C8	
0245	A1M1	
0246	A2L6	
0247	A7P6	
0248	C5P6	
0249	C7C8	
0250	C7M1	
0251	F5P6	
0252	F5X3	
0253	M0M1	
0254	M1S2	
0255	P5P6	
0256		14
0257	A0A1	
0258	A0L4	
0259	A0M0	
0260	A1A2	
0261	A1C8	
0262	A1F5	
0263	A2L6	
0264	C7C8	
0265	C7M1	
0266	C7S5	
0267	F5P6	
0268	F5X3	
0269	M0M1	
0270	M1S2	
0271		8
0272	A0A1	
0273	A1A2	
0274	A1C8	
0275	A1F5	
0276	A1M1	
0277	A9L5	
0278	I6L5	
0279	L5L6	
0280		24
0281	A3A5	
0282	A3S0	
0283	A3U0	
0284	A4A5	
0285	A4A6	
0286	A4A7	
0287	A4G2	
0288	A4S5	
0289	A4U1	
0290	A5A8	
0291	A5B1	
0292	A5B6	
0293	A5CL	
0294	A5I7	
0295	A6CN	
0296	A7P6	
0297	A7S7	
0298	C7S5	
0299	C8S5	
0300	P3S5	

```
0301   S3S5
0302   S4S5
0303   S5S6
0304   S5X2
0305        24
0306   A3A4
0307   A3S0
0308   A3U0
0309   A4A5
0310   A4A6
0311   A4A7
0312   A4G2
0313   A4S5
0314   A4U1
0315   A5A8
0316   A5B1
0317   A5B6
0318   A5CL
0319   A5I7
0320   A8F9
0321   B1C1
0322   B3B6
0323   B4B6
0324   B5B6
0325   B6CL
0326   CKCL
0327   CLCO
0328   I6I7
0329   I7S7
0330        14
0331   A3A4
0332   A3A5
0333   A3U0
0334   A4A5
0335   A4A6
0336   A4A7
0337   A4G2
0338   A4S5
0339   A4U1
0340   A5A8
0341   A5B1
0342   A5B6
0343   A5CL
0344   A5I7
0345        14
0346   A3A4
0347   A3A5
0348   A3S0
0349   A4A5
0350   A4A6
0351   A4A7
0352   A4G2
0353   A4S5
0354   A4U1
0355   A5A8
0356   A5B1
0357   A5B6
0358   A5CL
0359   A5I7
0360        34
```

```
0361   A3A4
0362   A3A5
0363   A3S0
0364   A3U0
0365   A4A6
0366   A4A7
0367   A4G2
0368   A4S5
0369   A4U1
0370   A5A8
0371   A5B1
0372   A5B6
0373   A5CL
0374   A5I7
0375   A6CN
0376   A7P6
0377   A7S7
0378   A8F9
0379   B1C1
0380   B3B6
0381   B4B6
0382   B5B6
0383   B6CL
0384   C7S5
0385   CBS5
0386   CKCL
0387   CLCO
0388   I6I7
0389   I7S7
0390   P3S5
0391   S3S5
0392   S4S5
0393   S5S6
0394   S5X2
0395        25
0396   A3A4
0397   A3A5
0398   A3S0
0399   A3U0
0400   A4A5
0401   A4A7
0402   A4G2
0403   A4S5
0404   A4U1
0405   A5A8
0406   A5B1
0407   A5B6
0408   A5CL
0409   A5I7
0410   A6CN
0411   A7P6
0412   A7S7
0413   C7S5
0414   CBS5
0415   CKCN
0416   P3S5
0417   S3S5
0418   S4S5
0419   S5S6
0420   S5X2
```

```
0421        29
0422   A3A4
0423   A3A5
0424   A3S0
0425   A3U0
0426   A4A5
0427   A4A6
0428   A4G2
0429   A4S5
0430   A4U1
0431   A5A8
0432   A5B1
0433   A5B6
0434   A5CL
0435   A5I7
0436   A6CN
0437   A7P6
0438   A7S7
0439   C5P6
0440   C7S5
0441   CBS5
0442   F5P6
0443   I7S7
0444   P3S5
0445   P5P6
0446   S3S5
0447   S4S5
0448   S5S6
0449   S5X2
0450   S6S7
0451        24
0452   A3A4
0453   A3A5
0454   A3S0
0455   A3U0
0456   A4A5
0457   A4A6
0458   A4A7
0459   A4S5
0460   A4U1
0461   A5A8
0462   A5B1
0463   A5B6
0464   A5CL
0465   A5I7
0466   A6CN
0467   A7P6
0468   A7S7
0469   C7S5
0470   CBS5
0471   P3S5
0472   S3S5
0473   S4S5
0474   S5S6
0475   S5X2
0476        34
0477   A3A4
0478   A3A5
0479   A3S0
0480   A3U0
```

```
0481   A4A5
0482   A4A6
0483   A4A7
0484   A4G2
0485   A4U1
0486   A5A8
0487   A5B1
0488   A5B6
0489   A5CL
0490   A5I7
0491   A6CN
0492   A7P6
0493   A7S7
0494   C7C8
0495   C7M1
0496   C7S5
0497   CBS5
0498   FAS3
0499   J0S3
0500   P1X2
0501   P3S5
0502   P3X2
0503   S3S4
0504   S3S5
0505   S4S5
0506   S5S6
0507   S5X2
0508   S6S7
0509   X0X2
0510   X1X2
0511        24
0512   A3A4
0513   A3A5
0514   A3B0
0515   A3U0
0516   A4A5
0517   A4A6
0518   A4A7
0519   A4G2
0520   A4S5
0521   A5A8
0522   A5B1
0523   A5B6
0524   A5CL
0525   A5I7
0526   A6CN
0527   A7P6
0528   A7S7
0529   C7S5
0530   CBS5
0531   P3S5
0532   S3S5
0533   S4S5
0534   S5S6
0535   S5X2
0536        24
0537   A3A4
0538   A3A5
0539   A3B0
0540   A3U0
```

```
0541   A4A5
0542   A4A6
0543   A4A7
0544   A4G2
0545   A4S5
0546   A4U1
0547   A5B1
0548   A5B6
0549   A5CL
0550   A5I7
0551   A8F9
0552   B1C1
0553   B3B6
0554   B4B6
0555   B5B6
0556   B6CL
0557   CKCL
0558   CLCO
0559   I6I7
0560   I7S7
0561        25
0562   A3A4
0563   A3A5
0564   A3S0
0565   A3U0
0566   A4A5
0567   A4A6
0568   A4A7
0569   A4G2
0570   A4S5
0571   A4U1
0572   A5A8
0573   A5B6
0574   A5CL
0575   A5I7
0576   A8F9
0577   B1C1
0578   B3B6
0579   B4B6
0580   B5B6
0581   B6CL
0582   C0C1
0583   CKCL
0584   CLCO
0585   I6I7
0586   I7S7
0587        32
0588   A3A4
0589   A3A5
0590   A3S0
0591   A3U0
0592   A4A5
0593   A4A6
0594   A4A7
0595   A4G2
0596   A4S5
0597   A4U1
0598   A5A8
0599   A5B1
0600   A5CL
```

```
0601   A5I7
0602   A8F9
0603   B1C1
0604   B3B4
0605   B3B5
0606   B3B6
0607   B3S1
0608   B4B5
0609   B4B6
0610   B4I1
0611   B4T2
0612   B5B6
0613   B5J4
0614   B5J7
0615   B6CL
0616   CKCL
0617   CLCO
0618   I6I7
0619   I7S7
0620        28
0621   A3A4
0622   A3A5
0623   A3S0
0624   A3U0
0625   A4A5
0626   A4A6
0627   A4A7
0628   A4G2
0629   A4S5
0630   A4U1
0631   A5A8
0632   A5B1
0633   A5B6
0634   A5I7
0635   A8F9
0636   B1C1
0637   B3B6
0638   B4B6
0639   B5B6
0640   B6CL
0641   CSCK
0642   CKCL
0643   CKCM
0644   CKCN
0645   CKU2
0646   CLCO
0647   I6I7
0648   I7S7
0649        27
0650   A3A4
0651   A3A5
0652   A3S0
0653   A3U0
0654   A4A5
0655   A4A6
0656   A4A7
0657   A4G2
0658   A4S5
0659   A4U1
0660   A5A8
```

```
0661   A5B1
0662   A5B6
0663   A5CL
0664   A7S7
0665   A8F9
0666   B1C1
0667   B3B6
0668   B4B6
0669   B5B6
0670   B6CL
0671   CKCL
0672   CLCO
0673   I6I7
0674   I6L5
0675   I7S7
0676   S6S7
0677        12
0678   A3A4
0679   A4A5
0680   A4A6
0681   A4A7
0682   A4G2
0683   A4S5
0684   A4U1
0685   C5CK
0686   CKCL
0687   CKCM
0688   CKCN
0689   CKU2
0690        20
0691   A1F5
0692   A3A4
0693   A4A5
0694   A4A6
0695   A4A7
0696   A4G2
0697   A4S5
0698   A4U1
0699   A7S7
0700   C4C5
0701   C5CA
0702   C5CK
0703   C5P6
0704   C5U2
0705   F5P6
0706   F5X3
0707   I4P5
0708   I7S7
0709   P5P6
0710   S6S7
0711        16
0712   A3A4
0713   A4A5
0714   A4A6
0715   A4A7
0716   A4G2
0717   A4S5
0718   A4U1
0719   A5I7
0720   A7P6
```

```
0721   C5P6
0722   F5P6
0723   I6I7
0724   I7S7
0725   P5P6
0726   S5S6
0727   S6S7
0728         7
0729   A3A5
0730   A4A5
0731   A5A8
0732   A5B1
0733   A5B6
0734   A5CL
0735   A5I7
0736         4
0737   A2L6
0738   I6I7
0739   I6L5
0740   L5L6
0741         8
0742   C4C5
0743   C4C6
0744   C4P4
0745   C4R0
0746   C5CA
0747   C5CK
0748   C5P6
0749   C5U2
0750        11
0751   A3A5
0752   A4A5
0753   A5A8
0754   A5B1
0755   A5B6
0756   A5CL
0757   A5I7
0758   C0C1
0759   C0C9
0760   C0SJ
0761   C0T0
0762         7
0763   CCF0
0764   F0F1
0765   F1F2
0766   F1F3
0767   F1P1
0768   F1SG
0769   F1SM
0770        16
0771   A5B6
0772   B3B5
0773   B3B6
0774   B3S1
0775   B4B5
0776   B4B6
0777   B4I1
0778   B4T2
0779   B5B6
0780   B5J4
```

```
0781   B5J7
0782   B6CL
0783   F8I1
0784   I0I1
0785   I1J3
0786   I1T2
0787        18
0788   A5B6
0789   B3B4
0790   B3B6
0791   B3S1
0792   B4B5
0793   B4B6
0794   B4I1
0795   B4T2
0796   B5B6
0797   B5J4
0798   B5J7
0799   B6CL
0800   CMJ4
0801   J2J4
0802   J3J4
0803   J5J7
0804   J6J7
0805   J7SM
0806        20
0807   A3A5
0808   A4A5
0809   A5A8
0810   A5B1
0811   A5B6
0812   A5CL
0813   A5I7
0814   B3B4
0815   B3B5
0816   B3S1
0817   B4B5
0818   B4B6
0819   B4I1
0820   B4T2
0821   B5B6
0822   B5J4
0823   B5J7
0824   B6CL
0825   CKCL
0826   CLCO
0827        12
0828   A5B6
0829   B3B4
0830   B3B5
0831   B3B6
0832   B4B5
0833   B4B6
0834   B4I1
0835   B4T2
0836   B5B6
0837   B5J4
0838   B5J7
0839   B6CL
0840        22
```

```
0841   A5B6
0842   B3B4
0843   B3B5
0844   B3B6
0845   B3S1
0846   B4B6
0847   B4I1
0848   B4T2
0849   B5B6
0850   B5J4
0851   B5J7
0852   B6CL
0853   CMJ4
0854   F8I1
0855   I0I1
0856   I1J3
0857   J2J4
0858   J3J4
0859   J5J7
0860   J6J7
0861   J7SM
0862   T1T2
0863        24
0864   A3A5
0865   A4A5
0866   A5A8
0867   A5B1
0868   A5B6
0869   A5CL
0870   A5I7
0871   B3B4
0872   B3B5
0873   B3B6
0874   B3S1
0875   B4B5
0876   B4I1
0877   B4T2
0878   B5B6
0879   B5J4
0880   B5J7
0881   B6CL
0882   CKCL
0883   CLCO
0884   F8I1
0885   I0I1
0886   I1J3
0887   T1T2
0888        25
0889   A5B6
0890   B3B4
0891   B3B5
0892   B3B6
0893   B3S1
0894   B4B5
0895   B4B6
0896   B4T2
0897   B5B6
0898   B5J4
0899   B5J7
0900   B6CL
```

```
0901   B7I0
0902   F7F8
0903   F8I1
0904   F8I3
0905   F8J6
0906   I0I1
0907   I0S8
0908   I1J3
0909   J2J3
0910   J3J4
0911   J3S8
0912   J3SF
0913   T1T2
0914        18
0915   A5B6
0916   B3B4
0917   B3B5
0918   B3B6
0919   B3S1
0920   B4B5
0921   B4B6
0922   B4I1
0923   B5B6
0924   B5J4
0925   B5J7
0926   B6CL
0927   F8I1
0928   I0I1
0929   I1J3
0930   I2T1
0931   I1T2
0932   T1U5
0933        26
0934   A3A5
0935   A4A5
0936   A5A8
0937   A5B1
0938   A5B6
0939   A5CL
0940   A5I7
0941   B3B4
0942   B3B5
0943   B3B6
0944   B3S1
0945   B4B5
0946   B4B6
0947   B4I1
0948   B4T2
0949   B5J4
0950   B5J7
0951   B6CL
0952   CKCL
0953   CLCO
0954   CMJ4
0955   J2J4
0956   J3J4
0957   J5J7
0958   J6J7
0959   J7SM
0960        24
```

```
0961   A5B6
0962   B3B4
0963   B3B5
0964   B3B6
0965   B3S1
0966   B4B5
0967   B4B6
0968   B4I1
0969   B4T2
0970   B5B6
0971   B5J7
0972   B6CL
0973   CKCM
0974   CMJ4
0975   I1J3
0976   J1J2
0977   J2J3
0978   J2J4
0979   J3J4
0980   J3SB
0981   J3SF
0982   J5J7
0983   J6J7
0984   J7SM
0985        27
0986   A5B6
0987   B3B4
0988   B3B5
0989   B3B6
0990   B3S1
0991   B4B5
0992   B4B6
0993   B4I1
0994   B4T2
0995   B5B6
0996   B5J4
0997   B6CL
0998   CMJ4
0999   F1SM
1000   F8J6
1001   G0J5
1002   H1J5
1003   I3J6
1004   J2J4
1005   J3J4
1006   J5J6
1007   J5J7
1008   J6J7
1009   J6L1
1010   J6SC
1011   J7SM
1012   SLSM
1013        24
1014   A3A5
1015   A4A5
1016   A5A8
1017   A5B1
1018   A5B6
1019   A5CL
1020   A5I7
```

```
1021   B3B4
1022   B3B5
1023   B3B6
1024   B3B1
1025   B4B5
1026   B4B6
1027   B4I1
1028   B4T2
1029   B5B6
1030   B5J4
1031   B5J7
1032   C5CK
1033   CKCL
1034   CKCM
1035   CKCN
1036   CKU2
1037   CLCO
1038         5
1039   B4I1
1040   F8I1
1041   I0I1
1042   I0S8
1043   I1J3
1044         6
1045   A5B1
1046   B1C1
1047   C0C9
1048   C0SJ
1049   C0T0
1050   SJSK
1051         5
1052   B1C1
1053   C0C1
1054   C0SJ
1055   C0T0
1056   SJSK
1057         8
1058   B1C1
1059   C0C1
1060   C0C9
1061   C0T0
1062   CFSK
1063   F2SK
1064   SJSK
1065   SKSN
1066         5
1067   B1C1
1068   C0C1
1069   C0C9
1070   C0SJ
1071   SJSK
1072        14
1073   C2SE
1074   C3CH
1075   C3P7
1076   CFCH
1077   CGCH
1078   CHF3
1079   CHI5
1080   CHP2
```

```
1081  CHSG
1082  CHSH
1083  CHSI
1084  CHSN
1085  CHV0
1086  CIP7
1087        3
1088  C2C3
1089  C3CH
1090  C3P7
1091       26
1092  C2C3
1093  C2SE
1094  C3P7
1095  CECF
1096  CFCH
1097  CFI5
1098  CFSK
1099  CGCH
1100  CGSD
1101  CHF3
1102  CHI5
1103  CHP2
1104  CHSG
1105  CHSH
1106  CHSI
1107  CHSN
1108  CHV0
1109  CIP7
1110  F1F3
1111  F1SG
1112  I5P2
1113  I5S9
1114  P2SA
1115  P2V0
1116  SGSH
1117  SKSN
1118       15
1119  C2C3
1120  C2SE
1121  C3CH
1122  CFCH
1123  CGCH
1124  CHF3
1125  CHI5
1126  CHP2
1127  CHSG
1128  CHSH
1129  CHSI
1130  CHSN
1131  CHV0
1132  CIL3
1133  CIP7
1134       16
1135  A7P6
1136  B0C4
1137  C4C6
1138  C4P4
1139  C4R0
1140  C5CA
```

```
1141  C5CK
1142  C5P6
1143  C5U2
1144  CAU2
1145  CKCL
1146  CKCM
1147  CKCN
1148  CKU2
1149  F5P6
1150  P5P6
1151        8
1152  B0C4
1153  C4C5
1154  C4P4
1155  C4R0
1156  C5CA
1157  C5CK
1158  C5P6
1159  C5U2
1160        8
1161  B0C4
1162  C4C5
1163  C4C6
1164  C4R0
1165  C5CA
1166  C5CK
1167  C5P6
1168  C5U2
1169        8
1170  B0C4
1171  C4C5
1172  C4C6
1173  C4P4
1174  C5CA
1175  C5CK
1176  C5P6
1177  C5U2
1178       16
1179  A7P6
1180  B0C4
1181  C4C5
1182  C4C6
1183  C4P4
1184  C4R0
1185  C5CK
1186  C5P6
1187  C5U2
1188  CAU2
1189  CKCL
1190  CKCM
1191  CKCN
1192  CKU2
1193  F5P6
1194  P5P6
1195       21
1196  A5CL
1197  A6CN
1198  A7P6
1199  B0C4
1200  B6CL
```

```
1201   C4C5
1202   C4C6
1203   C4P4
1204   C4R0
1205   C5CA
1206   C5P6
1207   C5U2
1208   CAU2
1209   CKCL
1210   CKCM
1211   CKCN
1212   CKU2
1213   CLCO
1214   CMJ4
1215   F5P6
1216   P5P6
1217        21
1218   A1F5
1219   A4A7
1220   A7P6
1221   A7S7
1222   B0C4
1223   C4C5
1224   C4C6
1225   C4P4
1226   C4R0
1227   C5CA
1228   C5CK
1229   C5U2
1230   CAU2
1231   CKCL
1232   CKCM
1233   CKCN
1234   CKU2
1235   F5P6
1236   F5X3
1237   I4P5
1238   P5P6
1239        16
1240   A7P6
1241   B0C4
1242   C4C5
1243   C4C6
1244   C4P4
1245   C4R0
1246   C5CA
1247   C5CK
1248   C5P6
1249   CAU2
1250   CKCL
1251   CKCM
1252   CKCN
1253   CKU2
1254   F5P6
1255   P5P6
1256        16
1257   A0A1
1258   A1A2
1259   A1C8
1260   A1F5
```

```
1261   A1M1
1262   A4S5
1263   C7M1
1264   C7S5
1265   CBS5
1266   M0M1
1267   M1S2
1268   P3S5
1269   S3S5
1270   S4S5
1271   S5S6
1272   S5X2
1273        17
1274   A0A1
1275   A0M0
1276   A1A2
1277   A1C8
1278   A1F5
1279   A1M1
1280   A4S5
1281   C7C8
1282   C7S5
1283   CBS5
1284   M0M1
1285   M1S2
1286   P3S5
1287   S3S5
1288   S4S5
1289   S5S6
1290   S5X2
1291        27
1292   A1C8
1293   A1M1
1294   A3A4
1295   A4A5
1296   A4A6
1297   A4A7
1298   A4G2
1299   A4S5
1300   A4U1
1301   C7C8
1302   C7M1
1303   CBS5
1304   FAS3
1305   J0S3
1306   M0M1
1307   M1S2
1308   P1X2
1309   P3S5
1310   P3X2
1311   S3S4
1312   S3S5
1313   S4S5
1314   S5S6
1315   S5X2
1316   S6S7
1317   X0X2
1318   X1X2
1319         9
1320   C4C5
```

```
1321  C5CA
1322  C5CK
1323  C5P6
1324  C5U2
1325  CKCL
1326  CKCM
1327  CKCN
1328  CKU2
1329       23
1330  A3A4
1331  A4A5
1332  A4A6
1333  A4A7
1334  A4G2
1335  A4S5
1336  A4U1
1337  C7C8
1338  C7M1
1339  C7S5
1340  FAS3
1341  J0S3
1342  P1X2
1343  P3S5
1344  P3X2
1345  S3S4
1346  S3S5
1347  S4S5
1348  S5S6
1349  S5X2
1350  S6S7
1351  X0X2
1352  X1X2
1353        7
1354  B2F0
1355  F0F1
1356  F1F2
1357  F1F3
1358  F1P1
1359  F1SG
1360  F1SM
1361       10
1362  F4X0
1363  H0X1
1364  M2X1
1365  P1X2
1366  P3X2
1367  S5X2
1368  U3X0
1369  X0X1
1370  X0X2
1371  X1X2
1372       19
1373  C3CH
1374  CEG1
1375  CFCH
1376  CFI5
1377  CFSK
1378  CGCH
1379  CHF3
1380  CHI5
```

```
1381   CHP2
1382   CHSG
1383   CHSH
1384   CHSI
1385   CHSN
1386   CHV0
1387   F2SK
1388   I5P2
1389   I5S9
1390   SJSK
1391   SKSN
1392        4
1393   CECF
1394   CFCH
1395   CFI5
1396   CFSK
1397       27
1398   C2C3
1399   C3CH
1400   C3P7
1401   CECF
1402   CEG1
1403   CFI5
1404   CFSK
1405   CGCH
1406   CGSD
1407   CHF3
1408   CHI5
1409   CHP2
1410   CHSG
1411   CHSH
1412   CHSI
1413   CHSN
1414   CHV0
1415   F1F3
1416   F1SG
1417   F2SK
1418   I5P2
1419   I5S9
1420   P2SA
1421   P2V0
1422   SGSH
1423   SJSK
1424   SKSN
1425       21
1426   C3CH
1427   CECF
1428   CEG1
1429   CFCH
1430   CFSK
1431   CGCH
1432   CHF3
1433   CHI5
1434   CHP2
1435   CHSG
1436   CHSH
1437   CHSI
1438   CHSN
1439   CHV0
1440   F2SK
```

```
1441   I5P2
1442   I5S9
1443   P2SA
1444   P2V0
1445   SJSK
1446   SKSN
1447        21
1448   C0SJ
1449   C3CH
1450   CECF
1451   CEG1
1452   CFCH
1453   CFI5
1454   CGCH
1455   CHF3
1456   CHI5
1457   CHP2
1458   CHSG
1459   CHSH
1460   CHSI
1461   CHSN
1462   CHV0
1463   F1F2
1464   F2SK
1465   I5P2
1466   I5S9
1467   SJSK
1468   SKSN
1469        24
1470   C2C3
1471   C3CH
1472   C3P7
1473   CECF
1474   CFCH
1475   CFI5
1476   CFSK
1477   CGSD
1478   CHF3
1479   CHI5
1480   CHP2
1481   CHSG
1482   CHSH
1483   CHSI
1484   CHSN
1485   CHV0
1486   F1F3
1487   F1SG
1488   I5P2
1489   I5S9
1490   P2SA
1491   P2V0
1492   SGSH
1493   SKSN
1494        11
1495   C3CH
1496   CFCH
1497   CGCH
1498   CHF3
1499   CHI5
1500   CHP2
```

```
1501  CHSG
1502  CHSH
1503  CHSI
1504  CHSN
1505  CHV0
1506       28
1507  C2C3
1508  C3CH
1509  C3P7
1510  CECF
1511  CFCH
1512  CFI5
1513  CFSK
1514  CGCH
1515  CGSD
1516  CHI5
1517  CHP2
1518  CHSG
1519  CHSH
1520  CHSI
1521  CHSN
1522  CHV0
1523  F0F1
1524  F1F2
1525  F1F3
1526  F1P1
1527  F1SG
1528  F1SM
1529  I5P2
1530  I5S9
1531  P2SA
1532  P2V0
1533  SGSH
1534  SKSN
1535       24
1536  C2C3
1537  C3CH
1538  C3P7
1539  CECF
1540  CFCH
1541  CFI5
1542  CFSK
1543  CGCH
1544  CGSD
1545  CHF3
1546  CHP2
1547  CHSG
1548  CHSH
1549  CHSI
1550  CHSN
1551  CHV0
1552  F1F3
1553  F1SG
1554  I5P2
1555  I5S9
1556  P2SA
1557  P2V0
1558  SGSH
1559  SKSN
1560       25
```

```
1561   C2C3
1562   C3CH
1563   C3P7
1564   CECF
1565   CFCH
1566   CFIS
1567   CFSK
1568   CGCH
1569   CGSD
1570   CHF3
1571   CHIS
1572   CHSG
1573   CHSH
1574   CHSI
1575   CHSN
1576   CHV0
1577   F1F3
1578   F1SG
1579   I5P2
1580   I5S9
1581   P1SA
1582   P2SA
1583   P2V0
1584   SGSH
1585   SKSN
1586        28
1587   C2C3
1588   C3CH
1589   C3P7
1590   CECF
1591   CFCH
1592   CFIS
1593   CFSK
1594   CGCH
1595   CGSD
1596   CHF3
1597   CHIS
1598   CHP2
1599   CHSH
1600   CHSI
1601   CHSN
1602   CHV0
1603   F0F1
1604   F1F2
1605   F1F3
1606   F1P1
1607   F1SG
1608   F1SM
1609   I5P2
1610   I5S9
1611   P2SA
1612   P2V0
1613   SGSH
1614   SKSN
1615        24
1616   C2C3
1617   C3CH
1618   C3P7
1619   CECF
1620   CFCH
```

```
1621   CFI5
1622   CFSK
1623   CGCH
1624   CGSD
1625   CHF3
1626   CHI5
1627   CHP2
1628   CHSG
1629   CHSI
1630   CHSN
1631   CHV0
1632   F1F3
1633   F1SG
1634   I5P2
1635   I5S9
1636   P2SA
1637   P2V0
1638   SGSH
1639   SKSN
1640        24
1641   C2C3
1642   C3CH
1643   C3P7
1644   CECF
1645   CFCH
1646   CFI5
1647   CFSK
1648   CGCH
1649   CGSD
1650   CHF3
1651   CHI5
1652   CHP2
1653   CHSG
1654   CHSH
1655   CHSN
1656   CHV0
1657   F1F3
1658   F1SG
1659   I5P2
1660   I5S9
1661   P2SA
1662   P2V0
1663   SGSH
1664   SKSN
1665        26
1666   C2C3
1667   C3CH
1668   C3P7
1669   CECF
1670   CFCH
1671   CFI5
1672   CFSK
1673   CGCH
1674   CGSD
1675   CHF3
1676   CHI5
1677   CHP2
1678   CHSG
1679   CHSH
1680   CHSI
```

```
1681   CHV0
1682   F1F3
1683   F1SG
1684   F2SK
1685   I5P2
1686   I5S9
1687   P2SA
1688   P2V0
1689   SGSH
1690   SJSK
1691   SKSN
1692        24
1693   C2C3
1694   C3CH
1695   C3P7
1696   CECF
1697   CFCH
1698   CFI5
1699   CFSK
1700   CGCH
1701   CGSD
1702   CHF3
1703   CHI5
1704   CHP2
1705   CHSG
1706   CHSH
1707   CHSI
1708   CHSN
1709   F1F3
1710   F1SG
1711   I5P2
1712   I5S9
1713   P2SA
1714   P2V0
1715   SGSH
1716   SKSN
1717         3
1718   C3P7
1719   CIP7
1720   F6L3
1721         5
1722   C2C3
1723   C3CH
1724   C3P7
1725   CIL3
1726   F6L3
1727         3
1728   F1SM
1729   J7SM
1730   SLSM
1731        23
1732   A3A5
1733   A4A5
1734   A5A8
1735   A5B1
1736   A5B6
1737   A5CL
1738   A5I7
1739   A6CN
1740   B3B6
```

```
1741   B4B6
1742   B5B6
1743   B6CL
1744   C4C5
1745   C5CA
1746   C5CK
1747   C5P6
1748   C5U2
1749   CAU2
1750   CKCM
1751   CKCN
1752   CKU2
1753   CLCO
1754   CMJ4
1755        17
1756   A5CL
1757   A6CN
1758   B5J4
1759   B6CL
1760   C4C5
1761   C5CA
1762   C5CK
1763   C5P6
1764   C5U2
1765   CAU2
1766   CKCL
1767   CKCN
1768   CKU2
1769   CLCO
1770   CMJ4
1771   J2J4
1772   J3J4
1773        15
1774   A4A6
1775   A5CL
1776   A6CN
1777   B6CL
1778   C4C5
1779   C5CA
1780   C5CK
1781   C5P6
1782   C5U2
1783   CAU2
1784   CKCL
1785   CKCM
1786   CKU2
1787   CLCO
1788   CMJ4
1789        14
1790   A5CL
1791   A6CN
1792   B6CL
1793   C4C5
1794   C5CA
1795   C5CK
1796   C5P6
1797   C5U2
1798   CAU2
1799   CKCL
1800   CKCM
```

```
1801   CKCN
1802   CLCO
1803   CMJ4
1804        16
1805   A3A5
1806   A4A5
1807   A5A8
1808   A5B1
1809   A5B6
1810   A5CL
1811   A5I7
1812   B3B6
1813   B4B6
1814   B5B6
1815   B6CL
1816   C5CK
1817   CKCL
1818   CKCM
1819   CKCN
1820   CKU2
1821        17
1822   B3B5
1823   B4B5
1824   B5B6
1825   B5J4
1826   B5J7
1827   C5CK
1828   CKCL
1829   CKCM
1830   CKCN
1831   CKU2
1832   I1J3
1833   J1J2
1834   J2J3
1835   J2J4
1836   J3J4
1837   J3SB
1838   J3SF
1839         5
1840   F1P1
1841   G3P0
1842   P0P1
1843   P1SA
1844   P1X2
1845        16
1846   B2F0
1847   CCF0
1848   CHF3
1849   CHSG
1850   F1F2
1851   F1F3
1852   F1P1
1853   F1SG
1854   F1SM
1855   F2SK
1856   J7SM
1857   P0P1
1858   P1SA
1859   P1X2
1860   SGSH
```

```
1861   SLSM
1862        19
1863   B2F0
1864   CCF0
1865   CFSK
1866   CHF3
1867   CHSG
1868   F0F1
1869   F1F3
1870   F1P1
1871   F1SG
1872   F1SM
1873   F2SK
1874   J7SM
1875   P0P1
1876   P1SA
1877   P1X2
1878   SGSM
1879   SJSK
1880   SKSN
1881   SLSM
1882        25
1883   B2F0
1884   C3CH
1885   CCF0
1886   CFCH
1887   CGCH
1888   CHF3
1889   CHI5
1890   CHP2
1891   CHSG
1892   CHSM
1893   CHSI
1894   CHSN
1895   CHV0
1896   F0F1
1897   F1F2
1898   F1P1
1899   F1SG
1900   F1SM
1901   F2SK
1902   J7SM
1903   P0P1
1904   P1SA
1905   P1X2
1906   SGSM
1907   SLSM
1908        23
1909   B2F0
1910   CCF0
1911   CHF3
1912   CHSG
1913   E0P0
1914   F0F1
1915   F1F2
1916   F1F3
1917   F1SG
1918   F1SM
1919   F2SK
1920   G3P0
```

```
1921   J7SM
1922   P0P1
1923   P1SA
1924   P1X2
1925   P2SA
1926   P3X2
1927   S5X2
1928   SGSH
1929   SLSM
1930   X0X2
1931   X1X2
1932        25
1933   B2F0
1934   C3CH
1935   CCF0
1936   CFCH
1937   CGCH
1938   CHF3
1939   CHI5
1940   CHP2
1941   CHSG
1942   CHSH
1943   CHSI
1944   CHSN
1945   CHV0
1946   F0F1
1947   F1F2
1948   F1F3
1949   F1P1
1950   F1SM
1951   F2SK
1952   J7SM
1953   P0P1
1954   P1SA
1955   P1X2
1956   SGSH
1957   SLSM
1958        20
1959   B2F0
1960   B5J7
1961   CCF0
1962   CHF3
1963   CHSG
1964   CJSL
1965   F0F1
1966   F1F2
1967   F1F3
1968   F1P1
1969   F1SG
1970   F2SK
1971   J5J7
1972   J6J7
1973   J7SM
1974   P0P1
1975   P1SA
1976   P1X2
1977   SGSH
1978   SLSM
1979        14
1980   C0SJ
```

```
1981  CECF
1982  CFCH
1983  CFIS
1984  CFSK
1985  CHSN
1986  F0F1
1987  F1F2
1988  F1F3
1989  F1P1
1990  F1SG
1991  F1SM
1992  SJSK
1993  SKSN
1994       10
1995  CDX0
1996  H0X1
1997  M2X1
1998  P1X2
1999  P3X2
2000  S5X2
2001  U3X0
2002  X0X1
2003  X0X2
2004  X1X2
2005       16
2006  A0A1
2007  A1A2
2008  A1C8
2009  A1F5
2010  A1M1
2011  A4A7
2012  A7P6
2013  A7S7
2014  C4C5
2015  C5CA
2016  C5CK
2017  C5P6
2018  C5U2
2019  F5X3
2020  I4P5
2021  P5P6
2022        9
2023  A0A1
2024  A1A2
2025  A1C8
2026  A1F5
2027  A1M1
2028  A7P6
2029  C5P6
2030  F5P6
2031  P5P6
2032        2
2033  CIL3
2034  CIP7
2035       13
2036  B4I1
2037  F7U4
2038  F8I1
2039  F8I3
2040  F8J6
```

```
2041   I0I1
2042   I1J3
2043   I2I3
2044   I3J6
2045   J5J6
2046   J6J7
2047   J6L1
2048   J6SC
2049         4
2050   F7F8
2051   F8I1
2052   F8I3
2053   F8J6
2054        23
2055   B3B4
2056   B4B5
2057   B4B6
2058   B4I1
2059   B4T2
2060   B7I0
2061   F7F8
2062   F7U4
2063   F8I3
2064   F8J6
2065   I0I1
2066   I0S8
2067   I1J3
2068   I2I3
2069   I3J6
2070   J2J3
2071   J3J4
2072   J3SB
2073   J3SF
2074   J5J6
2075   J6J7
2076   J6L1
2077   J6SC
2078        15
2079   B4I1
2080   F7F8
2081   F7U4
2082   F8I1
2083   F8J6
2084   I0I1
2085   I1J3
2086   I2I3
2087   I2L2
2088   I2T1
2089   I3J6
2090   J5J6
2091   J6J7
2092   J6L1
2093   J6SC
2094        22
2095   B4I1
2096   B5J7
2097   F7F8
2098   F7U4
2099   F8I1
2100   F8I3
```

```
2101  G0J5
2102  H0L1
2103  H1J5
2104  I0I1
2105  I1J3
2106  I2I3
2107  I3J6
2108  J5J6
2109  J5J7
2110  J6J7
2111  J6L1
2112  J6SC
2113  J7SM
2114  L0L1
2115  L100
2116  L1SC
2117       10
2118  A4S5
2119  C7S5
2120  C8S5
2121  J0S3
2122  P3S5
2123  S3S4
2124  S3S5
2125  S4S5
2126  S5S6
2127  S5X2
2128       10
2129  B5J7
2130  F8J6
2131  H1J5
2132  I3J6
2133  J5J6
2134  J5J7
2135  J6J7
2136  J6L1
2137  J6SC
2138  J7SM
2139        5
2140  E0P0
2141  F1P1
2142  P0P1
2143  P1SA
2144  P1X2
2145       13
2146  F8J6
2147  H0X1
2148  I3J6
2149  J5J6
2150  J6J7
2151  J6L1
2152  J6SC
2153  L0L1
2154  L100
2155  L1SC
2156  M2X1
2157  X0X1
2158  X1X2
2159       15
2160  CDX0
```

```
2161   F4X0
2162   H0L1
2163   J6L1
2164   L0L1
2165   L100
2166   L1SC
2167   M2X1
2168   P1X2
2169   P3X2
2170   S5X2
2171   U3X0
2172   X0X1
2173   X0X2
2174   X1X2
2175        10
2176   B5J7
2177   F8J6
2178   G0J5
2179   I3J6
2180   J5J6
2181   J5J7
2182   J6J7
2183   J6L1
2184   J6SC
2185   J7SM
2186        16
2187   B3B4
2188   B4B5
2189   B4B6
2190   B4I1
2191   B4T2
2192   B7I0
2193   F7F8
2194   F8I1
2195   F8I3
2196   F8J6
2197   I0S8
2198   I1J3
2199   J2J3
2200   J3J4
2201   J3SB
2202   J3SF
2203         5
2204   B4I1
2205   B7I0
2206   F8I1
2207   I0I1
2208   I1J3
2209        20
2210   B3B4
2211   B4B5
2212   B4B6
2213   B4I1
2214   B4T2
2215   B5J4
2216   B7I0
2217   CMJ4
2218   F7F8
2219   F8I1
2220   F8I3
```

```
2221   F8J6
2222   I0I1
2223   I0S8
2224   J1J2
2225   J2J3
2226   J2J4
2227   J3J4
2228   J3S8
2229   J3SF
2230        13
2231   F7F8
2232   F8I1
2233   F8I3
2234   F8J6
2235   I2L2
2236   I2T1
2237   I3J6
2238   J5J6
2239   J6J7
2240   J6L1
2241   J6SC
2242   T1T2
2243   T1U5
2244        6
2245   F8I3
2246   I2I3
2247   I2T1
2248   I3J6
2249   T1T2
2250   T1U5
2251        7
2252   B4T2
2253   F8I3
2254   I2I3
2255   I2L2
2256   I3J6
2257   T1T2
2258   T1U5
2259       20
2260   B5J7
2261   F7F8
2262   F8I1
2263   F8I3
2264   F8J6
2265   G0J5
2266   H0L1
2267   H1J5
2268   I2I3
2269   I2L2
2270   I2T1
2271   J5J6
2272   J5J7
2273   J6J7
2274   J6L1
2275   J6SC
2276   J7SM
2277   L0L1
2278   L100
2279   L1SC
2280        4
```

```
2281   A7P6
2282   C5P6
2283   F5P6
2284   P5P6
2285        18
2286   C3CH
2287   CECF
2288   CFCH
2289   CFI5
2290   CFSK
2291   CGCH
2292   CHF3
2293   CHI5
2294   CHP2
2295   CHSG
2296   CHSH
2297   CHSI
2298   CHSN
2299   CHV0
2300   I5S9
2301   P1SA
2302   P2SA
2303   P2V0
2304        17
2305   C3CH
2306   CECF
2307   CFCH
2308   CFI5
2309   CFSK
2310   CGCH
2311   CHF3
2312   CHI5
2313   CHP2
2314   CHS0
2315   CHSH
2316   CHSI
2317   CHSN
2318   CHV0
2319   I5P2
2320   P2SA
2321   P2V0
2322        13
2323   A3A5
2324   A4A5
2325   A5A8
2326   A5B1
2327   A5B6
2328   A5CL
2329   A5I7
2330   A7S7
2331   A9L5
2332   I6L5
2333   I7S7
2334   L5L6
2335   S6S7
2336         6
2337   A2L6
2338   A5I7
2339   A9L5
2340   I6I7
```

```
2341   I7S7
2342   L5L6
2343        14
2344   A3A5
2345   A4A5
2346   A4A7
2347   A5A8
2348   A5B1
2349   A5B6
2350   A5CL
2351   A5I7
2352   A7P6
2353   A7S7
2354   I6I7
2355   I6L5
2356   S5S6
2357   S6S7
2358        10
2359   A4S5
2360   C7S5
2361   CBS5
2362   FAS3
2363   P3S5
2364   S3S4
2365   S3S5
2366   S4S5
2367   S5S6
2368   S5X2
2369         8
2370   B5J4
2371   CMJ4
2372   I1J3
2373   J2J3
2374   J2J4
2375   J3J4
2376   J3SB
2377   J3SF
2378        11
2379   B4I1
2380   B5J4
2381   CMJ4
2382   F8I1
2383   I0I1
2384   I1J3
2385   J1J2
2386   J2J4
2387   J3J4
2388   J3SB
2389   J3SF
2390        13
2391   B3B5
2392   B4B5
2393   B5B6
2394   B5J4
2395   B5J7
2396   CKCM
2397   CMJ4
2398   I1J3
2399   J1J2
2400   J2J3
```

```
2401   J3J4
2402   J3SB
2403   J3SF
2404        16
2405   B3B5
2406   B4B5
2407   B4I1
2408   B5B6
2409   B5J4
2410   B5J7
2411   CKCM
2412   CMJ4
2413   F8I1
2414   I0I1
2415   I1J3
2416   J1J2
2417   J2J3
2418   J2J4
2419   J3SB
2420   J3SF
2421        11
2422   B4I1
2423   B5J4
2424   CMJ4
2425   F8I1
2426   I0I1
2427   I1J3
2428   J1J2
2429   J2J3
2430   J2J4
2431   J3J4
2432   J3SF
2433        11
2434   B4I1
2435   B5J4
2436   CMJ4
2437   F8I1
2438   I0I1
2439   I1J3
2440   J1J2
2441   J2J3
2442   J2J4
2443   J3J4
2444   J3SB
2445        18
2446   B5J7
2447   F7F8
2448   F8I1
2449   F8I3
2450   F8J6
2451   G0J5
2452   H0L1
2453   H1J5
2454   I2I3
2455   I3J6
2456   J5J7
2457   J6J7
2458   J6L1
2459   J6SC
2460   J7SM
```

```
2461   L0L1
2462   L100
2463   L1SC
2464        16
2465   B3B5
2466   B4B5
2467   B5B6
2468   B5J4
2469   B5J7
2470   F1SM
2471   F8J6
2472   G0J5
2473   H1J5
2474   I3J6
2475   J5J6
2476   J6J7
2477   J6L1
2478   J6SC
2479   J7SM
2480   SLSM
2481        24
2482   B3B5
2483   B4B5
2484   B5B6
2485   B5J4
2486   B5J7
2487   F1SM
2488   F7F8
2489   F8I1
2490   F8I3
2491   F8J6
2492   G0J5
2493   H0L1
2494   H1J5
2495   I2I3
2496   I3J6
2497   J5J6
2498   J5J7
2499   J6L1
2500   J6SC
2501   J7SM
2502   L0L1
2503   L100
2504   L1SC
2505   SLSM
2506        19
2507   B5J7
2508   F7F8
2509   F8I1
2510   F8I3
2511   F8J6
2512   G0J5
2513   H0L1
2514   H0X1
2515   H1J5
2516   I2I3
2517   I3J6
2518   J5J6
2519   J5J7
2520   J6J7
```

```
2521   J6SC
2522   J7SM
2523   L0L1
2524   L100
2525   L1SC
2526       18
2527   B5J7
2528   F7F8
2529   F8I1
2530   F8I3
2531   F8J6
2532   G0J5
2533   H0L1
2534   H1J5
2535   I2I3
2536   I3J6
2537   J5J6
2538   J5J7
2539   J6J7
2540   J6L1
2541   J7SM
2542   L0L1
2543   L100
2544   L1SC
2545       22
2546   B3B5
2547   B4B5
2548   B5B6
2549   B5J4
2550   B5J7
2551   CJSL
2552   F0F1
2553   F1F2
2554   F1F3
2555   F1P1
2556   F1SG
2557   F1SM
2558   F8J6
2559   G0J5
2560   H1J5
2561   I3J6
2562   J5J6
2563   J5J7
2564   J6J7
2565   J6L1
2566   J6SC
2567   SLSM
2568       10
2569   F8J6
2570   H0L1
2571   H0X1
2572   I3J6
2573   J5J6
2574   J6J7
2575   J6L1
2576   J6SC
2577   L100
2578   L1SC
2579       10
2580   F8J6
```

```
2581   H0L1
2582   H0X1
2583   I3J6
2584   J5J6
2585   J6J7
2586   J6L1
2587   J6SC
2588   L0L1
2589   L1SC
2590        10
2591   F8J6
2592   H0L1
2593   H0X1
2594   I3J6
2595   J5J6
2596   J6J7
2597   J6L1
2598   J6SC
2599   L0L1
2600   L100
2601         5
2602   A1A2
2603   A2L6
2604   A9L5
2605   I6I7
2606   I6L5
2607        11
2608   A0A1
2609   A0L4
2610   A0M0
2611   A1A2
2612   A1C8
2613   A1F5
2614   A1M1
2615   C7C8
2616   C7M1
2617   C7S5
2618   M1S2
2619        10
2620   A0A1
2621   A0M0
2622   A1A2
2623   A1C8
2624   A1F5
2625   A1M1
2626   C7C8
2627   C7M1
2628   C7S5
2629   M0M1
2630        11
2631   CDX0
2632   F4X0
2633   H0L1
2634   H0X1
2635   P1X2
2636   P3X2
2637   S5X2
2638   U3X0
2639   X0X1
2640   X0X2
```

```
2641   X1X2
2642         15
2643   E0P0
2644   F0F1
2645   F1F2
2646   F1F3
2647   F1P1
2648   F1SG
2649   F1SM
2650   G3P0
2651   P1SA
2652   P1X2
2653   P2SA
2654   P3X2
2655   S5X2
2656   X0X2
2657   X1X2
2658         18
2659   CHP2
2660   E0P0
2661   F0F1
2662   F1F2
2663   F1F3
2664   F1P1
2665   F1SG
2666   F1SM
2667   G3P0
2668   I5P2
2669   P0P1
2670   P1X2
2671   P2SA
2672   P2V0
2673   P3X2
2674   S5X2
2675   X0X2
2676   X1X2
2677         28
2678   A4S5
2679   C7S5
2680   CBS5
2681   COX0
2682   E0P0
2683   F0F1
2684   F1F2
2685   F1F3
2686   F1P1
2687   F1SG
2688   F1SM
2689   F4X0
2690   G3P0
2691   H0X1
2692   M2X1
2693   P0P1
2694   P1SA
2695   P2SA
2696   P3S5
2697   P3X2
2698   S3S5
2699   S4S5
2700   S5S6
```

```
2701   S5X2
2702   U3X0
2703   X0X1
2704   X0X2
2705   X1X2
2706         19
2707   C3CH
2708   CFCH
2709   CFI5
2710   CGCH
2711   CHF3
2712   CHI5
2713   CHP2
2714   CHSG
2715   CHSH
2716   CHSI
2717   CHSN
2718   CHV0
2719   F1P1
2720   I5P2
2721   I5S9
2722   P0P1
2723   P1SA
2724   P1X2
2725   P2V0
2726         16
2727   C3CH
2728   CFCH
2729   CFI5
2730   CGCH
2731   CHF3
2732   CHI5
2733   CHP2
2734   CHSG
2735   CHSH
2736   CHSI
2737   CHSN
2738   CHV0
2739   I5P2
2740   I5S9
2741   P1SA
2742   P2SA
2743         23
2744   A3A4
2745   A4A5
2746   A4A6
2747   A4A7
2748   A4G2
2749   A4S5
2750   A4U1
2751   C7C8
2752   C7M1
2753   C7S5
2754   CBS5
2755   FAS3
2756   J0S3
2757   P1X2
2758   P3X2
2759   S3S4
2760   S3S5
```

```
2761  S4S5
2762  S5S6
2763  S5X2
2764  S6S7
2765  X0X2
2766  X1X2
2767       20
2768  A4S5
2769  C7S5
2770  CBS5
2771  CDX0
2772  F1P1
2773  F4X0
2774  H0X1
2775  M2X1
2776  P0P1
2777  P1SA
2778  P1X2
2779  P3S5
2780  S3S5
2781  S4S5
2782  S5S6
2783  S5X2
2784  U3X0
2785  X0X1
2786  X0X2
2787  X1X2
2788       12
2789  A1F5
2790  A4A7
2791  A7P6
2792  A7S7
2793  C4C5
2794  C5CA
2795  C5CK
2796  C5P6
2797  C5U2
2798  F5P6
2799  F5X3
2800  I4P5
2801       10
2802  A4S5
2803  C7S5
2804  CBS5
2805  FAS3
2806  J0S3
2807  P3S5
2808  S3S5
2809  S4S5
2810  S5S6
2811  S5X2
2812       23
2813  A3A4
2814  A4A5
2815  A4A6
2816  A4A7
2817  A4G2
2818  A4S5
2819  A4U1
2820  C7C8
```

```
2821   C7M1
2822   C7S5
2823   CBS5
2824   FAS3
2825   J0S3
2826   P1X2
2827   P3S5
2828   P3X2
2829   S3S4
2830   S4S5
2831   S5S6
2832   S5X2
2833   S6S7
2834   X0X2
2835   X1X2
2836        23
2837   A3A4
2838   A4A5
2839   A4A6
2840   A4A7
2841   A4G2
2842   A4S5
2843   A4U1
2844   C7C8
2845   C7M1
2846   C7S5
2847   CBS5
2848   FAS3
2849   J0S3
2850   P1X2
2851   P3S5
2852   P3X2
2853   S3S4
2854   S3S5
2855   S5S6
2856   S5X2
2857   S6S7
2858   X0X2
2859   X1X2
2860        25
2861   A3A4
2862   A4A5
2863   A4A6
2864   A4A7
2865   A4G2
2866   A4S5
2867   A4U1
2868   A7S7
2869   C7C8
2870   C7M1
2871   C7S5
2872   CBS5
2873   FAS3
2874   I7S7
2875   J0S3
2876   P1X2
2877   P3S5
2878   P3X2
2879   S3S4
2880   S3S5
```

```
2881   S4S5
2882   S5X2
2883   S6S7
2884   X0X2
2885   X1X2
2886         32
2887   A3A4
2888   A4A5
2889   A4A6
2890   A4A7
2891   A4G2
2892   A4S5
2893   A4U1
2894   C7C8
2895   C7M1
2896   C7S5
2897   CBS5
2898   CDX0
2899   F1P1
2900   F4X0
2901   FAS3
2902   H0X1
2903   J0S3
2904   M2X1
2905   P0P1
2906   P1SA
2907   P1X2
2908   P3S5
2909   P3X2
2910   S3S4
2911   S3S5
2912   S4S5
2913   S5S6
2914   S6S7
2915   U3X0
2916   X0X1
2917   X0X2
2918   X1X2
2919         14
2920   A4A7
2921   A4S5
2922   A5I7
2923   A7P6
2924   A7S7
2925   C7S5
2926   CBS5
2927   I6I7
2928   I7S7
2929   P3S5
2930   S3S5
2931   S4S5
2932   S5S6
2933   S5X2
2934         17
2935   C3CH
2936   CFCH
2937   CGCH
2938   CHF3
2939   CHI5
2940   CHP2
```

```
2941   CHSG
2942   CHSH
2943   CHSI
2944   CHSN
2945   CHVO
2946   F0F1
2947   F1F2
2948   F1F3
2949   F1P1
2950   F1SG
2951   F1SM
2952        12
2953   C0C1
2954   C0C9
2955   C0SJ
2956   C0T0
2957   CECF
2958   CFCH
2959   CFI5
2960   CFSK
2961   CHSN
2962   F1F2
2963   F2SK
2964   SKSN
2965        18
2966   C0SJ
2967   C3CH
2968   CECF
2969   CFCH
2970   CFI5
2971   CFSK
2972   CGCH
2973   CHF3
2974   CHI5
2975   CHP2
2976   CHSG
2977   CHSH
2978   CHSI
2979   CHSN
2980   CHVO
2981   F1F2
2982   F2SK
2983   SJSK
2984        11
2985   B5J7
2986   CJSL
2987   F0F1
2988   F1F2
2989   F1F3
2990   F1P1
2991   F1SG
2992   F1SM
2993   J5J7
2994   J6J7
2995   J7SM
2996         9
2997   B3B4
2998   B4B5
2999   B4B6
3000   B411
```

```
3001  B4T2
3002  I2I3
3003  I2L2
3004  I2T1
3005  T1U5
3006       5
3007  B4T2
3008  I2I3
3009  I2L2
3010  I2T1
3011  T1T2
3012       10
3013  CDX0
3014  F4X0
3015  H0X1
3016  M2X1
3017  P1X2
3018  P3X2
3019  S5X2
3020  X0X1
3021  X0X2
3022  X1X2
3023       11
3024  CDX0
3025  F4X0
3026  H0L1
3027  H0X1
3028  M2X1
3029  P1X2
3030  P3X2
3031  S5X2
3032  U3X0
3033  X0X2
3034  X1X2
3035       20
3036  A4S5
3037  C7S5
3038  CBS5
3039  CDX0
3040  F1P1
3041  F4X0
3042  H0X1
3043  M2X1
3044  P0P1
3045  P1SA
3046  P1X2
3047  P3S5
3048  P3X2
3049  S3S5
3050  S4S5
3051  S5S6
3052  S5X2
3053  U3X0
3054  X0X1
3055  X1X2
3056       21
3057  A4S5
3058  C7S5
3059  CBS5
3060  CDX0
```

```
3061  F1P1
3062  F4X0
3063  H0L1
3064  H0X1
3065  M2X1
3066  P0P1
3067  P1SA
3068  P1X2
3069  P3S5
3070  P3X2
3071  S3S5
3072  S4S5
3073  S5S6
3074  S5X2
3075  U3X0
3076  X0X1
3077  X0X2
3078        1
3079  Z1Z2
3080        1
3081  Z0Z1
3082  03                  LER CORES DE LLG
3083  0                         NUMERO DE CORES DO NO A0A1 DE LLG
3084  12                        NUMERO DE CORES DO NO A0L4 DE LLG
3085  20
3086  21
3087  22
3088  23
3089  25
3090  26
3091  27
3092  28
3093  31
3094  32
3095  40
3096  41
3097  16                        NUMERO DE CORES DO NO A0M0 DE LLG
3098  20
3099  21
3100  22
3101  23
3102  25
3103  26
3104  27
3105  28
3106  56
3107  57
3108  58
3109  59
3110  66
3111  67
3112  68
3113  69
3114  00                        NUMERO DE CORES DO NO A1A2 DE LLG
3115  06                        NUMERO DE CORES DO NO A1C8 DE LLG
3116  19
3117  24
3118  52
3119  53
3120  60
```

```
3121  61
3122  08                    NUMERO DE CORES DO NO A1F5 DE LLG
3123  33
3124  34
3125  35
3126  36
3127  42
3128  43
3129  44
3130  45
3131  00                    NUMERO DE CORES DO NO A1M1 DE LLG
3132  22                    NUMERO DE CORES DO NO A2L6 DE LLG
3133  20
3134  21
3135  29
3136  30
3137  31
3138  32
3139  38
3140  39
3141  40
3142  41
3143  62
3144  63
3145  72
3146  73
3147  77
3148  78
3149  88
3150  89
3151  91
3152  92
3153  94
3154  95
3155  00                    NUMERO DE CORES DO NO A3A4 DE LLG
3156  00                    NUMERO DE CORES DO NO A3A5 DE LLG
3157  04                    NUMERO DE CORES DO NO A3S0 DE LLG
3158  29
3159  30
3160  38
3161  39
3162  01                    NUMERO DE CORES DO NO A3U0 DE LLG
3163  11
3164  00                    NUMERO DE CORES DO NO A4A5 DE LLG
3165  00                    NUMERO DE CORES DO NO A4A6 DE LLG
3166  00                    NUMERO DE CORES DO NO A4A7 DE LLG
3167  00                    NUMERO DE CORES DO NO A4G2 DE LLG
3168  22                    NUMERO DE CORES DO NO A4S5 DE LLG
3169  19
3170  20
3171  22
3172  23
3173  24
3174  25
3175  27
3176  28
3177  29
3178  30
3179  35
3180  40
```

```
3181  41
3182  48
3183  49
3184  50
3185  51
3186  56
3187  57
3188  58
3189  59
3190  67
3191  00                    NUMERO DE CORES DO NO A4U1 DE LLG
3192  00                    NUMERO DE CORES DO NO A5A8 DE LLG
3193  00                    NUMERO DE CORES DO NO A5B1 DE LLG
3194  00                    NUMERO DE CORES DO NO A5B6 DE LLG
3195  00                    NUMERO DE CORES DO NO A5CL DE LLG
3196  00                    NUMERO DE CORES DO NO A5I7 DE LLG
3197  22                    NUMERO DE CORES DO NO A6CN DE LLG
3198  12
3199  13
3200  16
3201  17
3202  31
3203  32
3204  33
3205  34
3206  36
3207  37
3208  38
3209  39
3210  44
3211  45
3212  46
3213  47
3214  55
3215  60
3216  61
3217  62
3218  63
3219  67
3220  03                    NUMERO DE CORES DO NO A7P6 DE LLG
3221  11
3222  68
3223  69
3224  00                    NUMERO DE CORES DO NO A7S7 DE LLG
3225  00                    NUMERO DE CORES DO NO A8F9 DE LLG
3226  09                    NUMERO DE CORES DO NO A9L5 DE LLG
3227  20
3228  22
3229  75
3230  79
3231  80
3232  82
3233  83
3234  94
3235  95
3236  12                    NUMERO DE CORES DO NO B0C4 DE LLG
3237  29
3238  30
3239  31
3240  32
```

```
3241  38
3242  39
3243  40
3244  41
3245  62
3246  63
3247  72
3248  73
3249  00                    NUMERO DE CORES DO NO B1C1 DE LLG
3250  06                    NUMERO DE CORES DO NO B2F0 DE LLG
3251  23
3252  33
3253  44
3254  45
3255  48
3256  49
3257  00                    NUMERO DE CORES DO NO B3B4 DE LLG
3258  00                    NUMERO DE CORES DO NO B3B5 DE LLG
3259  00                    NUMERO DE CORES DO NO B3B6 DE LLG
3260  00                    NUMERO DE CORES DO NO B3S1 DE LLG
3261  00                    NUMERO DE CORES DO NO B4B5 DE LLG
3262  00                    NUMERO DE CORES DO NO B4B6 DE LLG
3263  08                    NUMERO DE CORES DO NO B4I1 DE LLG
3264  17
3265  18
3266  21
3267  22
3268  42
3269  43
3270  51
3271  52
3272  00                    NUMERO DE CORES DO NO B4T2 DE LLG
3273  00                    NUMERO DE CORES DO NO B5B6 DE LLG
3274  15                    NUMERO DE CORES DO NO B5J4 DE LLG
3275  6
3276  7
3277  8
3278  16
3279  17
3280  23
3281  26
3282  44
3283  45
3284  59
3285  61
3286  35
3287  36
3288  37
3289  39
3290  15                    NUMERO DE CORES DO NO B5J7 DE LLG
3291  3
3292  4
3293  5
3294  12
3295  13
3296  16
3297  17
3298  23
3299  26
3300  44
```

```
3301  45
3302  59
3303  61
3304  48
3305  63
3306  00                    NUMERO DE CORES DO NO B6CL DE LLG
3307  04                    NUMERO DE CORES DO NO B7I0 DE LLG
3308  42
3309  43
3310  51
3311  52
3312  00                    NUMERO DE CORES DO NO C0C1 DE LLG
3313  02                    NUMERO DE CORES DO NO C0C9 DE LLG
3314  25
3315  30
3316  02                    NUMERO DE CORES DO NO C0SJ DE LLG
3317  64
3318  66
3319  10                    NUMERO DE CORES DO NO C0T0 DE LLG
3320  36
3321  37
3322  38
3323  39
3324  43
3325  44
3326  45
3327  46
3328  60
3329  62
3330  00                    NUMERO DE CORES DO NO C2C3 DE LLG
3331  00                    NUMERO DE CORES DO NO C2SE DE LLG
3332  02                    NUMERO DE CORES DO NO C3CH DE LLG
3333  52
3334  56
3335  03                    NUMERO DE CORES DO NO C3P7 DE LLG
3336  21
3337  38
3338  44
3339  00                    NUMERO DE CORES DO NO C4C5 DE LLG
3340  04                    NUMERO DE CORES DO NO C4C6 DE LLG
3341  46
3342  47

3343  52
3344  53                    NUMERO DE CORES DO NO C4P4 DE LLG
3345  58
3346  58
3347  57
3348  58
3349  59
3350  66
3351  67
3352  68
3353  69
3354  04                    NUMERO DE CORES DO NO C4R0 DE LLG
3355  48
3356  49
3357  54
3358  55
3359  05                    NUMERO DE CORES DO NO C5CA DE LLG
3360  12
```

```
3361  22
3362  23
3363  27
3364  28
3365  06                    NUMERO DE CORES DO NO C5CK DE LLG
3366  12
3367  17
3368  20
3369  21
3370  25
3371  26
3372  13                    NUMERO DE CORES DO NO C5P6 DE LLG
3373  14
3374  33
3375  34
3376  35
3377  36
3378  42
3379  43
3380  44
3381  45
3382  48
3383  49
3384  54
3385  55
3386  01                    NUMERO DE CORES DO NO C5U2 DE LLG
3387  10
3388  00                    NUMERO DE CORES DO NO C7C8 DE LLG
3389  02                    NUMERO DE CORES DO NO C7M1 DE LLG
3390  65
3391  74
3392  08                    NUMERO DE CORES DO NO C7S5 DE LLG
3393  15
3394  16
3395  17
3396  18
3397  31
3398  32
3399  33
3400  34
3401  01                    NUMERO DE CORES DO NO CAU2 DE LLG
3402  10
3403  04                    NUMERO DE CORES DO NO CBS5 DE LLG
3405  57
3406  61
3407  62
3408  06                    NUMERO DE CORES DO NO CCF0 DE LLG
3409  44
3410  45
3411  48
3412  49
3413  64
3414  66
3415  04                    NUMERO DE CORES DO NO CDX0 DE LLG
3416  50
3417  51
3418  57
3419  58
3420  00                    NUMERO DE CORES DO NO CECF DE LLG
```

```
0001  ?LPS,"MGFGNA"
0002  BEGIN
0003      BYTE (150) BUFFER;
0004      WORD (20000) MEMAUX = (20000:=1);
0005      WORD (1) NOME.DO.NO POS MEMAUX;
0006  &
0007  &   DECLARACAO DE PONTEIROS:
0008  &
0009      WORD     PLIGACOES
0010               ,PCOR
0011               ,PSOLUCOES
0012               ,PDISPONIBILIDADE
0013               ,LIVRE
0014               ,SALVA.LIVRE1
0015               ,SALVA.LIVRE2
0016               ,SALVA.LIVRE3
0017               ,INICIO.DAS.CORES
0018               ,INICIO.DAS.LIGACOES
0019               ,INICIO.DAS.CORES.DA.LIGACAO
0020               ,INICIO.DA.SOLUCAO
0021               ,INICIO.DA.SOLUCAO.DA.LIGACAO
0022               ,INICIO.DA.DISPONIBILIDADE
0023               ,INICIO.DA.ORDEM.DOS.NOS
0024               ,INICIO.DOS.NOS
0025               ;
0026  &
0027  &   DECLARACAO DOS CONTADORES:
0028  &
0029      WORD     NUMERO.DE.NOS
0030               ,NUMERO.DE.NOS.DE.LLG
0031               ,NUMERO.MAXIMO.DE.CORES
0032               ,NUMERO.DE.CORES
0033               ,NUMERO.DE.LIGACOES
0034               ,NLIGACOES
0035               ,NLIGACOES2
0036               ,NUMERO.DE.CORES.DA.LIGACAO
0037               ,NUMERO.DE.DISPONIVEIS
0038               ;
0039  ?P
0040  &
0041  &   DECLARACAO DAS VARIAVEIS PRINCIPAIS:
0042  &
0043      WORD     NUMERO
0044               ,NUMERO.DE.ORDEM.DO.NO
0045               ,NUMERO.DA.SOLUCAO = 0
0046               ,NSOLUCAO2 = 0
0047               ,TAMANHO.DA.SOLUCAO
0048               ,TAMANHO.DA.SOLUCAO.DA.LIGACAO
0049               ,LIGACAO
0050               ,NO
0051               ,NOE
0052               ,NOD
0053               ,COR
0054               ,COR.DA.LIGACAO
0055               ,OPERACAO
0056               ;
0057  &
0058  &   DECLARACAO DAS VARIAVEIS SECUNDARIAS E FLAGS
0059  &
0060      WORD     FLAGTV
```

```
0061              ,FLAGSOLUCAO
0062              ,F.ORDENA.N = 1
0063              ,F.ORDENA.C = 1
0064              ,FLAGDUMP = 1
0065              ,ESTADO = 0
0066              ,COND = 0
0067              ,QUATRO = 4
0068              ,I
0069              ,J
0070              ,K
0071              ,L
0072              ,M
0073              ;
0074  &
0075  &  DECLARACAO DAS CONSTANTES:
0076  &
0077      CONSTANT     FALSO = 0
0078                  ,VERDADE = 1
0079                  ,BRANCO = " "
0080                  ;
0081  ?P
0082      BYTE (30) PEDE.NUM.LIGACOES = (" QUANTAS LIGACOES TEM O      ?");
0083      BYTE (30) PEDE.NUMERO.DE.NOS= (" QUAL O NUMERO DE NOS ?*******");
0084      BYTE (30) PEDE.NOME.DO.NO   = (" QUAL O NOME DO PROXIMO NO ?.,");
0085      BYTE (30) PEDE.NUMERO.CORES = (" QUANTAS CORES TEM O NO XXXX ?");
0086      BYTE (30) PEDE.COR          = (" QUAL A PROXIMA COR DO XXXX  ?");
0087      BYTE (30) PEDE.LIGACAO      = (" QUAL PROXIMA LIGACAO DO XXXX?");
0088      BYTE (30) PEDE.UNIVERSO.COR = (" QUANTAS CORES NO UNIVERSO ?==");
0089      BYTE (40) PEDE.INCREMENTO   = (" QUANTAS CORES A ACRESCENTAR AO NO XXXX?");
0090      BYTE (25) SOLUC             = (" SOLUCOES PARA O NO XXXX=");
0091  ?P
0092  PROCEDURE BINDEC(WORD NUMER,ENDER; BYTE TAM,TIPO); EXTERNAL;
0093  WORD PROCEDURE DECBIN(WORD END1; BYTE TAM);       EXTERNAL;
0094  WORD PROCEDURE HEXBIN(WORD END1; BYTE TAM);       EXTERNAL;
0095  WORD PROCEDURE ALLOCATE( WORD CONTR,TV,NN,V0,V1,V2);EXTERNAL;
0096  PROCEDURE FIMPROG(WORD ENDCOND);                  SUPERVISOR 3;
0097  BYTE PROCEDURE CBT(WORD END1,END2; BYTE TAM);     EXTERNAL;
0098  PROCEDURE ABREENTRADA;
0099  BEGIN
0100      FLAGTV:=ALLOCATE(08005,"T*","  ","  ",-1,-1) AND 00FF;
0101      COND:=OPEN(01905);
0102      COND:=ALLOCATE(08006,"T*","  ","  ",-1,-1);
0103      COND:=OPEN(01906);
0104      COND:=OPEN(01907);
0105  END;
0106  ?P
0107  PROCEDURE ERRO.DE.ESTADO;
0108  BEGIN
0109      BYTE (43) ERRO1 = (" OPERACAO INCOMPATIVEL COM O ESTADO ATUAL. ");
0110      BYTE (540) ERRO2 =
0111          (" ESTADO = 0 => AINDA NAO LEU G.                              "
0112          ," ESTADO = 1 => LEU APENAS O GRAFO G.                         "
0113          ," ESTADO = 2 => APENAS LEU OU OBTEVE O GRAGO LLG.             "
0114          ," ESTADO = 3 => LEU AS CORES DOS NOS DE LLG.                  "
0115          ," ESTADO = 4 => LEU OS TAMANHOS DA SOLUCAO.                   "
0116          ," ESTADO = 5 => OBTEVE AS CORES DISPONIVEIS AOS NOS DE LLG.   "
0117          ," ESTADO = 6 => ORDENOU AS CORES DISPONIVEIS AOS NOS DE LLG.  "
0118          ," ESTADO = 7 => ORDENOU OS NOS DE LLG.                        "
0119          ," ESTADO = 8 => JA' OBTEVE ALGUMA SOLUCAO.                    ");
0120      WRITE(6,.ERRO1,43);
```

```
0121      WRITE(6,ERRO2+ESTADO*60,60);
0122  END;
0123  ?P
0124  WORD PROCEDURE LE.NUMERO;
0125  BEGIN
0126      WORD FLAGHEX,INIHEX,TAM,I;
0127      BYTE BYT;
0128  &
0129  &   LE O NUMERO
0130  &
0131      FILL(BUFFER,90,BRANCO);
0132      BUFFER(0):=BUFFER(80):="*";
0133      WHILE BUFFER(0) = "*" DO
0134      BEGIN
0135          TAM:=80;
0136          COND:=READ(5,BUFFER,TAM);
0137      END;
0138      I:=0; WHILE BUFFER(I) = " " DO INCR I;
0139      TAM:=I; WHILE BUFFER(TAM) <> " " DO INCR TAM;
0140  &
0141  &   TESTA SE EH HEXADECIMAL OU NAO.
0142  &
0143      IF TAM > 0 THEN BEGIN
0144          FLAGHEX:=0;
0145          IF BUFFER(I) = "0"
0146          THEN BEGIN
0147  &
0148  &   NUMERO HEXADECIMAL
0149  &
0150              FLAGHEX:=-1;
0151              INIHEX:=I+1;
0152              I:=TAM-INIHEX;
0153              BYT:=I;
0154              NUMERO:=HEXBIN(BUFFER+INIHEX,BYT);
0155              I:=TAM;
0156          END;
0157  &
0158  &   RETIRA BRANCOS A DIREITA DO NUMERO DECIMAL
0159  &
0160          IF FLAGHEX > -1 THEN BEGIN
0161  &
0162  &   NUMERO E DECIMAL:
0163  &
0164              INIHEX:=TAM-I;
0165              BYT:=INIHEX;
0166              NUMERO:=DECBIN(BUFFER+I,BYT);
0167              I:=TAM;
0168          END;
0169  &
0170  &   SETA CONDICAO DE RETORNO:
0171  &
0172          IF FLAG THEN LE.NUMERO := FALSO
0173                  ELSE LE.NUMERO := VERDADE;
0174      END;
0175  END;
0176  ?P
0177  PROCEDURE SAIDA( WORD ENDERECO,TAM);
0178  BEGIN
0179      IF FLAGTV = 0
0180      THEN COND:=WRITE(5,ENDERECO,TAM);
```

```
0181  END;
0182  ?P
0183  PROCEDURE LE.FATOR.DE.ORDENACAO;
0184  BEGIN
0185      BYTE (200) MSG1 = (" FATOR DE ORDENACAO INDICA O TIPO DE    "
0186                        ," ORDENACAO DESEJADA:                    "
0187                        ,"       FATOR > 0 => ORDENACAO CRESCENTE,"
0188                        ,"             = 0 => SEM ORDEM NENHMA,   "
0189                        ,"             < 0 => ORDEM DECRESCENTE.  ");
0190      BYTE (47) MSG2 = (" QUAL O FATOR DE ORDENACAO PARA OS NOS DE LLG? ");
0191      BYTE (54) MSG3 = (" QUAL O FATOR DE ORDENACAO P AS CORES DOS NOS DE LLG? ");
0192      WORD J;
0193      FOR J:=0 STEP 40 UNTIL 160 DO
0194      SAIDA(@MSG1+J,40);
0195      REPEAT SAIDA(@MSG2,47)
0196      UNTIL LE.NUMERO = VERDADE;
0197      F.ORDENA.N:=NUMERO;
0198      REPEAT SAIDA(@MSG3,54)
0199      UNTIL LE.NUMERO = VERDADE;
0200      F.ORDENA.C:=NUMERO;
0201  END;
0202  ?P
0203  PROCEDURE LE.NOME.DO.NO(WORD ENDERECO,DESVIO,FATOR);
0204  BEGIN
0205      WORD I;
0206      BYTE M;
0207      ENDERECO:=ENDERECO+DESVIO*FATOR;
0208      BUFFER(0):="*";
0209      WHILE BUFFER(0) = "*" DO
0210      BEGIN
0211          I:=80;
0212          COND:=READ(5,@BUFFER,@I);
0213      END;
0214      M:=FATOR;
0215      MBT(@BUFFER,ENDERECO,M);
0216  END;
0217  ?P
0218  PROCEDURE DUMP1( WORD END1,NUM; BYTE TAM);
0219  BEGIN
0220      FILL(@BUFFER,80,BRANCO);
0221      MBT(END1,@BUFFER,TAM);
0222      BINDEC(NUM,@BUFFER+25,6,1);
0223      WRITE(7,@BUFFER,60);
0224  END;
0225  ?P
0226  PROCEDURE DUMP;
0227  BEGIN
0228      PROCEDURE DDUMP;
0229      BEGIN
0230          FILL(@BUFFER,80,BRANCO);
0231          BINDEC(I,@BUFFER+1,6,1);
0232          K:=0;
0233          FOR J:=9 STEP 7 UNTIL 72 DO
0234          BEGIN
0235              BINDEC(MEMAUX(I+K),@BUFFER+J,6,1);
0236              INCR K;
0237          END;
0238          WRITE(7,@BUFFER,80);
0239      END;
0240      BYTE (13) PDISP = (" DISPONIVEIS ");
```

```
0241      DUMP1(@APEDE.NUMERO.DE.NOS+7,NUMERO.DE.NOS,15);
0242      DUMP1(@APEDE.UNIVERSO.COR+8,NUMERO.MAXIMO.DE.CORES,19);
0243      DUMP1(@APEDE.NUMERO.CORES+8,PCOR,7);
0244      DUMP1(@APEDE.NUM.LIGACOES+8,PLIGACOES,10);
0245      DUMP1(@APDISP,PDISPONIBILIDADE,13);
0246      DUMP1(@APEDE.INCREMENTO+14,PSOLUCOES,15);
0247      FOR I:=0 STEP 10 UNTIL LIVRE+10 DO
0248      DDUMP;
0249      J:=PSOLUCOES-NUMERO.DE.NOS-10;
0250      J:=J/10;
0251      J:=J*10;
0252      FOR I:=J STEP 10 UNTIL 4990 DO
0253      DDUMP;
0254  END;
0255  ?P
0256  PROCEDURE DUMPGERAL(WORD NUM);
0257  BEGIN
0258      WORD I,J;
0259      IF FLAGDUMP >= 0
0260      THEN BEGIN
0261          FILL(@BUFFER,130,BRANCO);
0262          BINDEC(NUM,@BUFFER+1,5,1);
0263          BINDEC(NUMERO.DE.ORDEM.DO.NO,@BUFFER+7,5,1);
0264          BINDEC(NO,@BUFFER+13,5,1);
0265          BINDEC(LIGACAO,@BUFFER+19,5,1);
0266          BINDEC(FLAGSOLUCAO,@BUFFER+25,5,1);
0267          J:=35;
0268          FOR I:=0 TO TAMANHO.DA.SOLUCAO-1 DO
0269          BEGIN
0270              BINDEC(MEMAUX(INICIO.DA.SOLUCAO+I),@BUFFER+J,5,1);
0271              J:=J+6;
0272          END;
0273          WRITE(7,@BUFFER,80);
0274      END;
0275  END;
0276  ?P
0277  PROCEDURE LE.NUMERO.DE.NOS.E.DE.CORES;
0278  BEGIN
0279  &
0280  &  LE O NUMERO DE NOS.
0281  &
0282      REPEAT SAIDA(@APEDE.NUMERO.DE.NOS,30)
0283      UNTIL LE.NUMERO = VERDADE;
0284      NUMERO.DE.NOS:=NUMERO;
0285      LIVRE:=NUMERO;
0286      FILL(@MEMAUX,NUMERO*2,BRANCO);
0287  &
0288  &  LE O NUMERO DE CORES DO UNIVERSO.
0289  &
0290      REPEAT SAIDA(@APEDE.UNIVERSO.COR,30)
0291      UNTIL LE.NUMERO = VERDADE;
0292      NUMERO.MAXIMO.DE.CORES:=NUMERO;
0293  &
0294  &  SETA INICIO DOS PONTEIROS DE CORES, LIGACOES, SOLUCOES E DISPONIBILIDADE.
0295  &
0296      PLIGACOES:=(@PLIGACOES-@MEMAUX)/2-1;
0297      PCOR:=PLIGACOES-NUMERO.DE.NOS;
0298      PSOLUCOES:=PCOR-NUMERO.DE.NOS;
0299      PDISPONIBILIDADE:=PSOLUCOES-NUMERO.DE.NOS;
0300  END;
```

```
0301  ?P
0302  PROCEDURE LE.LIGACOES(WORD QUATRO);
0303  BEGIN
0304  &
0305  &  LEITURA DAS LIGACOES:
0306  &
0307      FOR NO:=0 TO NUMERO.DE.NOS-1 DO
0308      BEGIN
0309          BYTE QUATROB;
0310          LABEL L1;
0311  &
0312  &  PEDE O NUMERO DE LIGACOES DO NO ATUAL.
0313  &
0314          QUATROB:=QUATRO;
0315          MBT(@NOME.DO.NO+NO*QUATRO,@PEDE.LIGACAO+25,QUATROB);
0316          MBT(@NOME.DO.NO+NO*QUATRO,@PEDE.NUM.LIGACOES+24,QUATROB);
0317          SALVA.LIVRE1:=LIVRE;
0318  L1:     LIVRE:=SALVA.LIVRE1;
0319          REPEAT SAIDA(@PEDE.NUM.LIGACOES,30)
0320          UNTIL LE.NUMERO = VERDADE;
0321          MEMAUX(PLIGACOES-NO):=LIVRE;
0322          MEMAUX(LIVRE):=NLIGACOES:=NUMERO;
0323          INCR LIVRE;
0324  &
0325  &  LOOP DE LEITURA DOS NOMES DAS LIGACOES;
0326  &
0327          FOR K:=1 TO NLIGACOES DO
0328          BEGIN
0329              SAIDA(@PEDE.LIGACAO,30);
0330              LE.NOME.DO.NO(@BUFFER,0,QUATRO);
0331              J:=-1;
0332  &
0333  &  RECONHECE SE EXISTE A LIGACAO FORNECIDA:
0334  &
0335              FOR I:=0 TO NUMERO.DE.NOS-1 DO
0336              IF CBT(@NOME.DO.NO+I*QUATRO,@BUFFER,QUATROB) = 0 AND I <> NO
0337              THEN BEGIN
0338  &
0339  &  ARMAZENA O NUMERO DO NO CORRESPONDENTE AO NOME DA LIGACAO DADA.
0340  &
0341                  J:=I;
0342                  MEMAUX(LIVRE):=I;
0343                  INCR LIVRE;
0344                  I:=NUMERO.DE.NOS;
0345              END;
0346              IF J < 0 THEN GO TO L1;
0347          END;
0348      END;
0349  END;
0350  ?P
0351  PROCEDURE LE.G;
0352  BEGIN
0353  &
0354  &  OBSERVA SE O ESTADO E' COMPATIVEL.
0355  &
0356      IF ESTADO <> 0
0357      THEN BEGIN
0358          ERRO.DE.ESTADO;
0359          RETURN;
0360      END;
```

```
0361  &
0362  &  LE O NUMERO DE NOS E DE CORES DE G.
0363  &
0364      LE.NUMERO.DE.NOS.E.DE.CORES;
0365  &
0366  &  LE O NOME DE CADA NO.
0367  &
0368      FOR NO:=0 TO NUMERO.DE.NOS-1 DO
0369      BEGIN
0370          SAIDA(@PEDE.NOME.DO.NO,30);
0371          LE.NOME.DO.NO(@NOME.DO.NO,NO,2);
0372      END;
0373  &
0374  &  LEITURA DAS LIGACOES EM G.
0375  &
0376      LE.LIGACOES(2);
0377      ESTADO:=1;
0378  END;
0379  ?P
0380  PROCEDURE LE.LLG;
0381  BEGIN
0382  &
0383  &  OBSERVA SE O ESTADO ESTA COMPATIVEL.
0384  &
0385      IF ESTADO <> 0
0386      THEN BEGIN
0387          ERRO.DE.ESTADO;
0388          RETURN;
0389      END;
0390  &
0391  &  LE O NUMERO DE NOS DE LL(G);
0392  &
0393      LE.NUMERO.DE.NOS.E.DE.CORES;
0394      LIVRE:=NUMERO.DE.NOS*2;
0395      FILL(@MEMAUX,NUMERO.DE.NOS*4,BRANCO);
0396  &
0397  &  LOOP DE LEITURA SOBRE OS NOS.
0398  &
0399      FOR NO:=0 TO NUMERO.DE.NOS-1 DO
0400      BEGIN
0401          SAIDA(@PEDE.NOME.DO.NO,30);
0402          LE.NOME.DO.NO(@NOME.DO.NO,NO,4);
0403      END;
0404      LE.LIGACOES(4);
0405      ESTADO:=2;
0406  END;
0407  ?P
0408  PROCEDURE GRAVA.LLG;
0409  BEGIN
0410      PROCEDURE ESCREVA(WORD PALAVRA);
0411      BEGIN
0412          FILL(@BUFFER,10,BRANCO);
0413          BINDEC(PALAVRA,@BUFFER,6,1);
0414          WRITE(8,@BUFFER,10);
0415      END;
0416  &
0417  &  TESTA A COMPATIBILIDADE DO ESTADO.
0418  &
0419      IF ESTADO < 2
0420      THEN BEGIN
```

```
0421            ERRO.DE.ESTADO;
0422            RETURN;
0423        END;
0424    &
0425    &   TENTA CRIAR O ARQUIVO DE SAIDA.
0426    &
0427        IF CREATE(Ö1008,"D ",Ö0102) <> 0
0428        THEN RETURN;
0429        IF OPEN(Ö1908) <> 0
0430        THEN RETURN;
0431    &
0432    &   GRAVACAO DO COMANDO DE LEITURA.
0433    &
0434        ESCREVA(01);                        &   GRAVA O COMANDO DE LEITURA,
0435        ESCREVA(NUMERO.DE.NOS);             &   O NUMERO DE NOS,
0436        ESCREVA(NUMERO.MAXIMO.DE.CORES);&   O NUMERO MAXIMO DE CORES,
0437        FOR NO:=0 TO NUMERO.DE.NOS-1 DO
0438            WRITE(8,ÄNOME.DO.NO+NO*4,4);&   E OS NOMES DOS NOS.
0439    &
0440    &   GRAVACAO DAS LIGACOES
0441    &
0442        FOR NO:=0 TO NUMERO.DE.NOS-1 DO
0443        BEGIN
0444            INICIO.DAS.LIGACOES:=MEMAUX(PLIGACOES+NO);
0445            NLIGACOES:=MEMAUX(INICIO.DAS.LIGACOES);
0446            ESCREVA(NLIGACOES);                         &   GRAVA O NUMERO DE LIG
0447            FOR I:=1 TO NLIGACOES DO                    &   E O NOME DAS LIGACOES
0448            BEGIN
0449                LIGACAO:=MEMAUX(INICIO.DAS.LIGACOES+I);
0450                WRITE(8,ÄNOME.DO.NO+LIGACAO*4,4);
0451            END;
0452        END;
0453    ?P
0454    &
0455    &   SE TEM AS CORES JA DEFINIDAS, ENTAO AS GRAVA TAMBEM.
0456    &
0457        IF ESTADO > 3
0458        THEN BEGIN
0459            ESCREVA(02);                        &   GRAVA O COMANDO DE LER CORES.
0460            FOR NO:=0 TO NUMERO.DE.NOS-1 DO
0461            BEGIN
0462                J:=MEMAUX(PCOR+NO)+1;
0463                NUMERO.DE.CORES:=MEMAUX(J);
0464                ESCREVA(NUMERO.DE.CORES);   &   GRAVA O NUMERO DE CORES.
0465                FOR I:=1 TO NUMERO.DE.CORES DO
0466                BEGIN
0467                    COR:=MEMAUX(J+I);
0468                    ESCREVA(COR);           &   GRAVA AS CORES DE CADA NO.
0469                END;
0470            END;
0471        END;
0472    &
0473    &   SE JA TEM OS TAMANHOS DA SOLUCAO, ENTAO TAMBEM OS GRAVA.
0474    &
0475        IF ESTADO > 4
0476        THEN BEGIN
0477            ESCREVA(04);                        &   GRAVA COMANDO LER TAM SOLUCAO.
0478            FOR NO:=0 TO NUMERO.DE.NOS-1 DO
0479            BEGIN
0480                J:=MEMAUX(PCOR+NO);
```

```
0481                I:=MEMAUX(J);
0482                ESCREVA(I);                    &   GRAVA O TAMANHO DA SOLUCAO.
0483            END;
0484        END;
0485    END;
0486    ?P
0487    PROCEDURE LE.SOLUCAO.DE.PARTIDA;
0488    BEGIN
0489        BYTE (47) PEDE.SOLUCAO =
0490            (" ENTRE COM A XXXX-ESIMA COR DA SOLUCAO DE XXXX ");
0491        BYTE (27) ERRO.DE.SOLUCAO = (" ESTA SOLUCAO NAO E VALIDA ");
0492        WORD I,J,K,L;
0493    &
0494    &   TESTA SE O ESTADO E' COMPATIVEL.
0495    &
0496        IF ESTADO < 8
0497        THEN BEGIN
0498            ERRO.DE.ESTADO;
0499            RETURN;
0500        END;
0501    &
0502    &   LOOP DE LEITURA SOBRE OS NOS
0503    &
0504        FOR NO:=0 TO NUMERO.DE.NOS-1 DO
0505        BEGIN
0506            I:=MEMAUX(PCOR-NO);
0507            TAMANHO.DA.SOLUCAO:=MEMAUX(I);
0508            INICIO.DA.DISPONIBILIDADE:=MEMAUX(PDISPONIBILIDADE-NO);
0509            INICIO.DA.SOLUCAO:=MEMAUX(PSOLUCOES-NO);
0510            NUMERO.DE.DISPONIVEIS:=MEMAUX(INICIO.DA.DISPONIBILIDADE);
0511            MBT(ANOME.DO.NO+NO*4,APEDE.SOLUCAO+42,4);
0512            L:=0;
0513            K:=J:=1;
0514            FOR I:=1 TO TAMANHO.DA.SOLUCAO DO
0515            BEGIN
0516                LABEL L1,L2;
0517                BINDEC(I,APEDE.SOLUCAO+13,4,1);
0518    L1:         REPEAT SAIDA(APEDE.SOLUCAO,47)
0519                UNTIL LE.NUMERO = VERDADE;
0520                K:=J;
0521    L2:         IF J > NUMERO.DE.DISPONIVEIS
0522                THEN BEGIN
0523                    J:=K;
0524                    SAIDA(AERRO.DE.SOLUCAO,26);
0525                    GO TO L1;
0526                END;
0527                IF NUMERO = MEMAUX(INICIO.DA.DISPONIBILIDADE+J)
0528                THEN INCR J
0529                ELSE BEGIN
0530                    INCR J;      & AINDA NAO ENCONTROU A COR DENTRE
0531                    GO TO L2;    & AS DISPONIVEIS.
0532                END;
0533                MEMAUX(INICIO.DA.SOLUCAO+L):=NUMERO;
0534                INCR L;
0535            END;
0536        END;
0537    END;
0538    ?P
0539    PROCEDURE OBTEM.LLG;
0540    BEGIN
```

```
0541  &
0542  &  OBSERVA SE O ESTADO ESTA COMPATIVEL.
0543  &
0544      IF ESTADO <> 1
0545      THEN BEGIN
0546          ERRO.DE.ESTADO;
0547          RETURN;
0548      END;
0549  &
0550  &  OBTEM OS NOS DE LLG.
0551  &
0552      INICIO.DOS.NOS:=LIVRE;
0553      NUMERO.DE.NOS.DE.LLG:=0;
0554      FOR NO:=0 TO NUMERO.DE.NOS-1 DO
0555      BEGIN
0556          INICIO.DAS.LIGACOES:=MEMAUX(PLIGACOES+NO);
0557          NLIGACOES:=MEMAUX(INICIO.DAS.LIGACOES);
0558          FOR I:=1 TO NLIGACOES DO
0559          BEGIN
0560              LIGACAO:=MEMAUX(INICIO.DAS.LIGACOES+I);
0561              IF LIGACAO > NO
0562              THEN BEGIN
0563                  MEMAUX(LIVRE):=NO;
0564                  MEMAUX(LIVRE+1):=LIGACAO;
0565                  LIVRE:=LIVRE+2;
0566                  INCR NUMERO.DE.NOS.DE.LLG;
0567              END;
0568          END;
0569      END;
0570  ?P
0571  &
0572  &  OBTEM AS LIGACOES DE LG
0573  &
0574      FOR NO:=0 TO NUMERO.DE.NOS.DE.LLG-1 DO
0575      BEGIN
0576          I:=INICIO.DOS.NOS+NO*2;
0577          NOE:=MEMAUX(I);
0578          NOD:=MEMAUX(I+1);
0579          INICIO.DAS.LIGACOES:=MEMAUX(PCOR+NO):=LIVRE;
0580          NLIGACOES:=0;
0581          INCR LIVRE;
0582          FOR I:=0 TO NUMERO.DE.NOS.DE.LLG-1 DO
0583          BEGIN
0584              J:=INICIO.DOS.NOS+I*2;
0585              K:=MEMAUX(J);
0586              L:=MEMAUX(J+1);
0587              IF (NOE=K OR NOE=L OR NOD=K OR NOE=L) AND I <> NO
0588              THEN BEGIN
0589                  INCR NLIGACOES;
0590                  MEMAUX(LIVRE):=I;
0591                  INCR LIVRE;
0592              END;
0593          END;
0594  ?P
0595  &
0596  &  OBTEM AS LIGACOES DE LLG
0597  &
0598          NLIGACOES2:=NLIGACOES;
0599          FOR M:=1 TO NLIGACOES2 DO
0600          BEGIN
```

```
0601  &
0602  &  EXAMINA AS LIGACOES DAS LIGACOES.
0603  &
0604            I:=INICIO.DOS.NOS+MEMAUX(INICIO.DAS.LIGACOES+M)*2;
0605            NOE:=MEMAUX(I);
0606            NOD:=MEMAUX(I+1);
0607            FOR I:=0 TO NUMERO.DE.NOS.DE.LLG-1 DO
0608            BEGIN
0609                J:=INICIO.DOS.NOS+I*2;
0610                K:=MEMAUX(J);
0611                L:=MEMAUX(J+1);
0612                IF (NOE=K OR NOE=L OR NOD=K OR NOD=L) AND I <> NO
0613                THEN BEGIN
0614                    INCR NLIGACOES;
0615                    MEMAUX(LIVRE):=I;
0616                    INCR LIVRE;
0617                END;
0618            END;
0619        END;
0620        MEMAUX(INICIO.DAS.LIGACOES):=NLIGACOES;
0621    END;
0622  ?P
0623  &
0624  &  MONTA OS NOMES DOS NOS DE LLG.
0625  &
0626    FOR NO:=0 TO NUMERO.DE.NOS.DE.LLG-1 DO
0627    BEGIN
0628        NOE:=MEMAUX(INICIO.DOS.NOS+NO*2);
0629        NOD:=MEMAUX(INICIO.DOS.NOS+NO*2+1);
0630        NOE:=MEMAUX(NOE);
0631        NOD:=MEMAUX(NOD);
0632        MEMAUX(INICIO.DOS.NOS+NO*2):=NOE;
0633        MEMAUX(INICIO.DOS.NOS+NO*2+1):=NOD;
0634    END;
0635  &
0636  &  MOVE TUDO PARA O INICIO DA MEMORIA AUXILIAR;
0637  &
0638    FOR I:=0 TO NUMERO.DE.NOS.DE.LLG*2-1 DO
0639    MEMAUX(I):=MEMAUX(INICIO.DOS.NOS+I);
0640    LIVRE:=NUMERO.DE.NOS.DE.LLG*2;
0641  ?P
0642  &
0643  &  MONTA TODAS AS LIGACOES NO INICIO DA MEMORIA
0644  &
0645    FOR NO:=0 TO NUMERO.DE.NOS.DE.LLG-1 DO
0646    BEGIN
0647        INICIO.DAS.LIGACOES:=MEMAUX(PCOR-NO);
0648  &
0649  &  CLASSIFICA AS LIGACOES
0650  &
0651        NLIGACOES:=MEMAUX(INICIO.DAS.LIGACOES);
0652        I:=J:=1;
0653        WHILE J < NLIGACOES DO
0654        IF MEMAUX(INICIO.DAS.LIGACOES+J) > MEMAUX(INICIO.DAS.LIGACOES+J+1)
0655        THEN BEGIN
0656            K:=MEMAUX(INICIO.DAS.LIGACOES+J);
0657            L:=MEMAUX(INICIO.DAS.LIGACOES+J+1);
0658            MEMAUX(INICIO.DAS.LIGACOES+J):=L;
0659            MEMAUX(INICIO.DAS.LIGACOES+J+1):=K;
0660            IF J > 1
```

```
0661                THEN DECR J
0662                ELSE BEGIN
0663                    INCR I;
0664                    J:=I;
0665                END;
0666            END ELSE BEGIN
0667                INCR I;
0668                J:=I;
0669            END;
0670    &
0671    &   MOVE AS LIGACOES PARA O INICIO DA MEMORIA E RETIRA AS DUPLICADAS
0672    &
0673            MEMAUX(PLIGACOES-NO):=LIVRE;
0674            I:=0;
0675            MEMAUX(LIVRE):=-1;
0676            FOR J:=1 TO NLIGACOES DO
0677            IF MEMAUX(INICIO.DAS.LIGACOES+J) <> MEMAUX(LIVRE+I)
0678            THEN BEGIN
0679                MEMAUX(LIVRE+I+1):=MEMAUX(INICIO.DAS.LIGACOES+J);
0680                INCR I;
0681            END;
0682            MEMAUX(LIVRE):=I;
0683            LIVRE:=LIVRE+I+1;
0684        END;
0685        NUMERO.DE.NOS:=NUMERO.DE.NOS.DE.LLG;
0686        PCOR:=PLIGACOES-NUMERO.DE.NOS;
0687        PSOLUCOES:=PCOR-NUMERO.DE.NOS;
0688        PDISPONIBILIDADE:=PSOLUCOES-NUMERO.DE.NOS;
0689        ESTADO:=2;
0690    END;
0691    ?P
0692    BYTE PROCEDURE LE.CORES.DOS.NOS.DE.LLG;
0693    BEGIN
0694        WORD NCORES;
0695    &
0696    &   OBSERVA SE O ESTADO E' COMPATIVEL.
0697    &
0698        IF ESTADO <> 2
0699        THEN BEGIN
0700            ERRO.DE.ESTADO;
0701            RETURN;
0702        END;
0703        FOR NO:=0 TO NUMERO.DE.NOS-1 DO
0704        BEGIN
0705            LABEL L1;
0706            SALVA.LIVRE1:=LIVRE;
0707    L1:     LIVRE:=SALVA.LIVRE1;
0708            MEMAUX(PCOR-NO):=LIVRE;
0709            MEMAUX(LIVRE):=0;
0710            INCR LIVRE;
0711    &
0712    &   LE O NUMERO DE CORES ATUALMENTE ASSOCIADO A CADA NO.
0713    &
0714            MBT(ANOME.DO.NO+NO*4,APEDE.NUMERO.CORES+24,4);
0715            MBT(ANOME.DO.NO+NO*4,APEDE.COR+23,4);
0716            SAIDA(APEDE.NUMERO.CORES,30);
0717            IF LE.NUMERO = FALSO
0718            THEN GO TO L1
0719            ELSE BEGIN
0720                MEMAUX(LIVRE):=NCORES:=NUMERO;
```

```
0721              INCR LIVRE;
0722          END;
0723          FOR COR:=1 TO NCORES DO
0724          BEGIN
0725  &
0726  &   LE AS CORES DE CADA NO
0727  &
0728              SAIDA(APEDE.COR,30);
0729              IF LE.NUMERO = FALSO
0730              THEN GO TO L1
0731              ELSE BEGIN
0732                  IF NUMERO < 0 OR NUMERO > NUMERO.MAXIMO.DE.CORES
0733                  THEN GO TO L1;
0734                  MEMAUX(LIVRE):=NUMERO;
0735                  INCR LIVRE;
0736              END;
0737          END;
0738          LE.CORES.DOS.NOS.DE.LLG:=VERDADE;
0739      END;
0740      ESTADO:=3;
0741  END;
0742  ?P
0743  PROCEDURE LE.OS.TAMANHOS.DA.SOLUCAO;
0744  BEGIN
0745  &
0746  &   OBSERVA SE O ESTADO E' COMPATIVEL
0747  &
0748      IF ESTADO <> 3
0749      THEN BEGIN
0750          ERRO.DE.ESTADO;
0751          RETURN;
0752      END;
0753  &
0754  &   LE O INCREMENTO DE CORES DESEJADO A CADA NO.
0755  &
0756      FOR NO:=0 TO NUMERO.DE.NOS-1 DO
0757      BEGIN
0758          MBT(ANOME.DO.NO+NO*4,APEDE,INCREMENTO+35,4);
0759          REPEAT SAIDA(APEDE.INCREMENTO,40)
0760          UNTIL LE.NUMERO = VERDADE;
0761          I:=MEMAUX(PCOR-NO);
0762          MEMAUX(I):=NUMERO;
0763      END;
0764      ESTADO:=4;
0765  END;
0766  ?P
0767  PROCEDURE OBTEM.AS.CORES.DISPONIVEIS.EM.LLG;
0768  BEGIN
0769      WORD I,J;
0770  &
0771  &   OBSERVA SE O ESTADO E' COMPATIVEL.
0772  &
0773      IF ESTADO <> 4
0774      THEN BEGIN
0775          ERRO.DE.ESTADO;
0776          RETURN;
0777      END;
0778  &
0779  &   PARA CADA NO, TENTA OBTER AS CORES DISPONIVEIS
0780  &
```

```
0781        FOR NO:=0 TO NUMERO.DE.NOS-1 DO
0782        BEGIN
0783            MEMAUX(PDISPONIBILIDADE-NO):=LIVRE;
0784            MEMAUX(LIVRE):=0;
0785    &
0786    &   PREVIAMENTE SUPOE TODO O UNIVERSO DE CORES COMO DISPONIVEL.
0787    &
0788            FOR J:=1 TO NUMERO.MAXIMO.DE.CORES DO
0789            MEMAUX(LIVRE+J):=J;
0790            INICIO.DAS.LIGACOES:=MEMAUX(PLIGACOES-NO);
0791            NUMERO.DE.LIGACOES:=MEMAUX(INICIO.DAS.LIGACOES);
0792            LIGACAO:=NO;
0793    &
0794    &   PARA CADA NO DA LIGACAO...
0795    &
0796            FOR J:=0 TO NUMERO.DE.LIGACOES DO
0797            BEGIN
0798                INICIO.DAS.CORES:=MEMAUX(PCOR-LIGACAO)+1;
0799                NUMERO.DE.CORES:=MEMAUX(INICIO.DAS.CORES);
0800    &
0801    &   ...SUBTRAI DO UNIVERSO DISPONIVEL AS CORES DESTA LIGACAO
0802    &
0803                FOR I:=1 TO NUMERO.DE.CORES DO
0804                BEGIN
0805                    COR:=MEMAUX(INICIO.DAS.CORES+I);
0806                    IF COR > 0 AND COR <= NUMERO.MAXIMO.DE.CORES
0807                    THEN MEMAUX(LIVRE+COR):=-1;
0808                END;
0809                LIGACAO:=MEMAUX(INICIO.DAS.LIGACOES+J+1);
0810            END;
0811    &
0812    &   CONPACTA O UNIVERSO RESULTANTE, GERANDO AS CORES DISPONIVEIS
0813    &
0814            I:=0;
0815            FOR J:=1 TO NUMERO.MAXIMO.DE.CORES DO
0816            IF MEMAUX(LIVRE+J) = J
0817            THEN BEGIN
0818                INCR I;
0819                MEMAUX(LIVRE+I):=J;
0820            END;
0821            MEMAUX(LIVRE):=I;
0822            LIVRE:=LIVRE+I+1;
0823        END;
0824        ESTADO:=5;
0825    END;
0826    ?P
0827    PROCEDURE PEGA.INFORMACOES.DO.NO;
0828    BEGIN
0829        INICIO.DAS.CORES:=MEMAUX(PCOR-NO);
0830        TAMANHO.DA.SOLUCAO:=MEMAUX(INICIO.DAS.CORES);
0831        NUMERO.DE.CORES:=MEMAUX(INICIO.DAS.CORES+1);
0832        INICIO.DA.SOLUCAO:=MEMAUX(PSOLUCOES-NO);
0833        INICIO.DAS.LIGACOES:=MEMAUX(PLIGACOES-NO);
0834        NUMERO.DE.LIGACOES:=MEMAUX(INICIO.DAS.LIGACOES);
0835        INICIO.DA.DISPONIBILIDADE:=MEMAUX(PDISPONIBILIDADE-NO);
0836        NUMERO.DE.DISPONIVEIS:=MEMAUX(INICIO.DA.DISPONIBILIDADE);
0837    END;
0838    ?P
0839    PROCEDURE ORDENA.AS.CORES.DISPONIVEIS.EM.LLG;
0840    BEGIN
```

```
0841  &
0842  &  OBSERVA SE O ESTADO E' COMPATIVEL.
0843  &
0844      IF ESTADO <> 5
0845      THEN BEGIN
0846          ERRO.DE.ESTADO;
0847          RETURN;
0848      END;
0849  &
0850  &  VOU OBTER PARA CADA NO OS PESOS DE SUAS CORES
0851  &
0852      FOR NO:=0 TO NUMERO.DE.NOS-1 DO
0853      BEGIN
0854  &
0855  &  OBTENHO O INICIO DAS CORES DESTE NO, E SEU NUMERO DE CORES
0856  &
0857          PEGA.INFORMACOES.DO.NO;
0858  &
0859  &  INICIALIZO  A AREA DE PESO DAS CORES
0860  &
0861          FOR J:=1 TO NUMERO.DE.DISPONIVEIS DO
0862          MEMAUX(LIVRE+J):=0;
0863          MEMAUX(LIVRE+NUMERO.DE.DISPONIVEIS+1):=32767;
0864  &
0865  &  VOU VARRER TODAS AS LIGACOES DESTE NO, PARA CALCULAR OS
0866  &  PESOS DE SUAS CORES
0867  &
0868          FOR J:=1 TO NUMERO.DE.LIGACOES DO
0869          BEGIN
0870              LIGACAO:=MEMAUX(INICIO.DAS.LIGACOES+J);
0871  &
0872  &  OBTEM O INICIO E O NUMERO DE CORES DA LIGACAO.
0873  &
0874              INICIO.DAS.CORES.DA.LIGACAO:=MEMAUX(PDISPONIBILIDADE=LIGACAO);
0875              NUMERO.DE.CORES.DA.LIGACAO:=MEMAUX(INICIO.DAS.CORES.DA.LIGACAO);
0876  &
0877  &  CALCULA O PESO DAS CORES.
0878  &
0879              FOR I:=1 TO NUMERO.DE.CORES.DA.LIGACAO DO
0880              BEGIN
0881                  COR.DA.LIGACAO:=MEMAUX(INICIO.DAS.CORES.DA.LIGACAO+I);
0882                  FOR K:=1 TO NUMERO.DE.DISPONIVEIS DO
0883                  IF COR.DA.LIGACAO = MEMAUX(INICIO.DA.DISPONIBILIDADE+K)
0884                  THEN BEGIN
0885                      INCR MEMAUX(LIVRE+K);
0886                      K:=NUMERO.DE.DISPONIVEIS;
0887                  END;
0888              END;
0889          END;
0890  &
0891  &  ORDENACAO DAS CORES DO NO
0892  &
0893          J:=1;
0894          I:=1;
0895          WHILE J < NUMERO.DE.DISPONIVEIS DO
0896          IF MEMAUX(LIVRE+J)*F.ORDENA.C > MEMAUX(LIVRE+J+1)*F.ORDENA.C
0897          THEN BEGIN
0898              K:=MEMAUX(LIVRE+J);
0899              MEMAUX(LIVRE+J):=MEMAUX(LIVRE+J+1);
0900              MEMAUX(LIVRE+J+1):=K;
```

```
0901                K:=MEMAUX(INICIO.DA.DISPONIBILIDADE+J);
0902                MEMAUX(INICIO.DA.DISPONIBILIDADE+J):=
0903                      MEMAUX(INICIO.DA.DISPONIBILIDADE+J+1);
0904                MEMAUX(INICIO.DA.DISPONIBILIDADE+J+1):=K;
0905                IF J > 1
0906                THEN DECR J
0907                ELSE BEGIN
0908                     INCR I;
0909                     J:=I;
0910                END;
0911           END ELSE BEGIN
0912                INCR I;
0913                J:=1;
0914           END;
0915      END;
0916      ESTADO:=6;
0917 END;
0918 ?P
0919 WORD PROCEDURE COMBINACAO(WORD N,P);
0920 BEGIN
0921      IF P = 0
0922      THEN COMBINACAO:=0
0923      ELSE BEGIN
0924           J:=1;
0925           FOR I:=1 TO P DO
0926           BEGIN
0927              J:=J*N/I;
0928              N:=N-1;
0929           END;
0930           COMBINACAO:=IF J < 0 THEN 32767
0931                                ELSE J;
0932      END;
0933 END;
0934 ?P
0935 PROCEDURE ORDENA.NOS.DE.LLG;
0936 BEGIN
0937 &
0938 &  OBSERVA SE O ESTADO E' COMPATIVEL.
0939 &
0940      IF ESTADO <> 6
0941      THEN BEGIN
0942           ERRO.DE.ESTADO;
0943           RETURN;
0944      END;
0945 &
0946 &  SETA INICIO DA ORDEM DOS NOS;
0947 &
0948      INICIO.DA.ORDEM.DOS.NOS:=LIVRE;
0949      FOR NO:=0 TO NUMERO.DE.NOS-1 DO
0950      BEGIN
0951 &
0952 &  SETA OS PONTEIROS DO NO
0953 &
0954           MEMAUX(LIVRE):=NO;
0955 &
0956 &  CALCULA OS PESOS DOS NOS
0957 &
0958           PEGA.INFORMACOES.DO.NO;
0959           MEMAUX(LIVRE+NUMERO.DE.NOS):=COMBINACAO(NUMERO.DE.DISPONIVEIS,
0960                                                   TAMANHO.DA.SOLUCAO);
```

```
0961          INCR LIVRE;
0962      END;
0963  &
0964  &   ORDENA OS PONTEIROS DOS NOS EM FORMA DECRESCENTE, DE ACORDO COM OS
0965  &   PESOS OBTIDOS PARA CADA NO.
0966  &
0967      I:=J:=0;
0968      WHILE J < NUMERO.DE.NOS-1 DO
0969      IF MEMAUX(LIVRE+J)*F.ORDENA.N > MEMAUX(LIVRE+J+1)*F.ORDENA.N
0970      THEN BEGIN
0971          K:=MEMAUX(INICIO.DA.ORDEM.DOS.NOS+J);
0972          MEMAUX(INICIO.DA.ORDEM.DOS.NOS+J):=MEMAUX(INICIO.DA.ORDEM.DOS.NOS+J+1);
0973          MEMAUX(INICIO.DA.ORDEM.DOS.NOS+J+1):=K;
0974          K:=MEMAUX(LIVRE+J);
0975          MEMAUX(LIVRE+J):=MEMAUX(LIVRE+J+1);
0976          MEMAUX(LIVRE+J+1):=K;
0977          IF J > 0
0978          THEN DECR J
0979          ELSE J:=I:=I+1;
0980      END ELSE J:=I:=I+1;
0981      ESTADO:=7;
0982  END;
0983  ?P
0984  PROCEDURE TESTA.COMPATIBILIDADE.GERAL;
0985  BEGIN
0986  &                          0123456789.123456789.123456789.123456789.123456789.12345678
0987    BYTE (43) ERRO1 = (" A COR XXXX NAO ESTA DISPONIVEL AO NO XXXX.");
0988    BYTE (54) ERRO2 = (" A COR XXXX COMO SOLUCAO AO NO XXXX E' COR DO NO XXXX.");
0989    BYTE (61) ERRO3 = (" A COR XXXX E' SOLUCAO AOS NOS XXXX E XXXX, QUE SAO ",
0990                       "VIZINHOS.");
0991  &
0992  &   SE NAO FOI PEDIDO O RASTREAMENTO, ABANDONAR O TESTE.
0993  &
0994      IF FLAGDUMP < 0 THEN RETURN;
0995      FOR NO:=0 TO NUMERO.DE.NOS-1 DO
0996      BEGIN
0997  &
0998  &   PEGA INFORMACOES DO NO..
0999  &
1000          PEGA.INFORMACOES.DO.NO;
1001          MBT(ANOME.DO.NO+NO*4,AERRO1+38,4);
1002          MBT(ANOME.DO.NO+NO*4,AERRO2+31,4);
1003          MBT(ANOME.DO.NO+NO*4,AERRO3+31,4);
1004  &
1005  &   TESTA COMAPTIBILIDADE DA SOLUCAO COM A DISPONIBILIDADE
1006  &
1007          FOR I:=0 TO TAMANHO.DA.SOLUCAO-1 DO
1008          BEGIN
1009              M:=-1;
1010              COR:=MEMAUX(INICIO.DA.SOLUCAO+I);
1011              FOR J:=1 TO NUMERO.DE.DISPONIVEIS DO
1012              IF MEMAUX(INICIO.DA.DISPONIBILIDADE+J) = COR
1013              THEN BEGIN
1014                  M:=J;
1015                  J:=NUMERO.DE.DISPONIVEIS;
1016              END;
1017              IF M < 0
1018              THEN BEGIN
1019                  BINDEC(COR,AERRO1+7,4,1);
1020                  WRITE(7,AERRO1,43);
```

```
1021                END;
1022            END;
1023    ?P
1024    &
1025    &   OBSERVA AS LIGACOES
1026    &
1027            FOR NLIGACOES:=1 TO NUMERO.DE.LIGACOES DO
1028            BEGIN
1029                LIGACAO:=MEMAUX(INICIO.DAS.LIGACOES+NLIGACOES);
1030                INICIO.DAS.CORES.DA.LIGACAO:=MEMAUX(PCOR+LIGACAO);
1031                INICIO.DA.SOLUCAO.DA.LIGACAO:=MEMAUX(PSOLUCOES+LIGACAO);
1032                TAMANHO.DA.SOLUCAO.DA.LIGACAO:=
1033                    MEMAUX(INICIO.DAS.CORES.DA.LIGACAO);
1034                NUMERO.DE.CORES.DA.LIGACAO:=
1035                    MEMAUX(INICIO.DAS.CORES.DA.LIGACAO+1);
1036                MBT(ANOME.DO.NO+LIGACAO*4,AERRO2+49,4);
1037                MBT(ANOME.DO.NO+LIGACAO*4,AERRO3+38,4);
1038    &
1039    &   CORES UTILIZADAS PELA LIGACAO
1040    &
1041                FOR I:=0 TO TAMANHO.DA.SOLUCAO-1 DO
1042                BEGIN
1043                    COR:=MEMAUX(INICIO.DA.SOLUCAO+I);
1044                    FOR J:=2 TO NUMERO.DE.CORES.DA.LIGACAO+1 DO
1045                    IF MEMAUX(INICIO.DAS.CORES.DA.LIGACAO+J) = COR
1046                    THEN BEGIN
1047                        BINDEC(COR,AERRO2+7,4,1);
1048                        WRITE(7,AERRO2,54);
1049                        J:=NUMERO.DE.CORES.DA.LIGACAO;
1050                    END;
1051    &
1052    &   CORES DA SOLUCAO DA LIGACAO
1053    &
1054                    FOR J:=0 TO TAMANHO.DA.SOLUCAO.DA.LIGACAO-1 DO
1055                    IF MEMAUX(INICIO.DA.SOLUCAO.DA.LIGACAO+J) = COR
1056                    THEN BEGIN
1057                        BINDEC(COR,AERRO3+7,4,1);
1058                        WRITE(7,AERRO3,61);
1059                        J:=TAMANHO.DA.SOLUCAO.DA.LIGACAO;
1060                    END;
1061                END;
1062            END;
1063        END;
1064    END;
1065    ?P
1066    BYTE PROCEDURE PRIMEIRA.SOLUCAO;
1067    BEGIN
1068    &
1069    &   GERA A PRIMEIRA SOLUCAO PARA O NO ATUAL
1070    &
1071        PEGA.INFORMACOES.DO.NO;
1072        IF NUMERO.DE.DISPONIVEIS < TAMANHO.DA.SOLUCAO
1073    &
1074    &   NAO HA NENHUMA SOLUCAO POSSIVEL PARA ESTE NO.
1075    &
1076        THEN PRIMEIRA.SOLUCAO:=FALSO
1077        ELSE BEGIN
1078    &
1079    &   SETA O PONTEIRO SOLUCAO
1080    &
```

```
        IF TAMANHO.DA.SOLUCAO > 0
        THEN BEGIN
            SALVA.LIVRE2:=MEMAUX(PSOLUCOES-NO);
            MEMAUX(PSOLUCOES-NO):=INICIO.DA.SOLUCAO:=SALVA.LIVRE1:=
            IF SALVA.LIVRE2 < 0
            THEN LIVRE
            ELSE SALVA.LIVRE2;
            FOR I:=1 TO TAMANHO.DA.SOLUCAO DO
            BEGIN
                MEMAUX(SALVA.LIVRE1):=MEMAUX(INICIO.DA.DISPONIBILIDADE+I);
                INCR SALVA.LIVRE1;
            END;
            IF SALVA.LIVRE2 < 0 THEN LIVRE:=SALVA.LIVRE1;
        END ELSE MEMAUX(PSOLUCOES-NO):=-1;
        PRIMEIRA.SOLUCAO:=VERDADE;
    END;
END;
?P
BYTE PROCEDURE PROXIMA.SOLUCAO;
BEGIN
&
&   RECONHECE O INICIO DA SOLUCAO ATUAL
&
    PEGA.INFORMACOES.DO.NO;
    IF TAMANHO.DA.SOLUCAO <= 0
    THEN BEGIN                              & RETORNO
        PROXIMA.SOLUCAO:=VERDADE;           &   TAMANHO DA SOLUCAO
        RETURN;                             &   EH ZERO.
    END;
    IF INICIO.DA.SOLUCAO < 0
    THEN BEGIN
        PROXIMA.SOLUCAO:=PRIMEIRA.SOLUCAO;
        RETURN;
    END;
&
&   OBSERVA A POSICAO RELATIVA DAS CORES DA SOLUCAO ATUAL
&
    FOR I:=0 TO TAMANHO.DA.SOLUCAO-1 DO
    BEGIN
        J:=-1;
        FOR K:=0 TO NUMERO.DE.DISPONIVEIS-1 DO
        IF MEMAUX(INICIO.DA.SOLUCAO+I) =
           MEMAUX(INICIO.DA.DISPONIBILIDADE+K+1)
        THEN BEGIN
            J:=K;
            K:=NUMERO.DE.DISPONIVEIS;
        END;
        IF J < 0                                & RETORNO
        THEN BEGIN                              &   ALGUMA COR DA SOLUCAO
            PROXIMA.SOLUCAO:=FALSO;             &   ATUAL NAO FOI RECONHECIDA
            RETURN;                             &   DENTRE AS DISPONIVEIS.
        END;
        MEMAUX(LIVRE+I):=J;
    END;
    MEMAUX(LIVRE+TAMANHO.DA.SOLUCAO):=NUMERO.DE.DISPONIVEIS;
?P
&
&   PEGA A ULTIMA COR FORA DE SEQUENCIA
&
    K:=-1;
```

```
    FOR I:=0 TO TAMANHO.DA.SOLUCAO-1 DO
    IF MEMAUX(LIVRE+I)+1 < MEMAUX(LIVRE+I+1)
    THEN K:=I;
    IF K < 0                                        & RETORNO
    THEN BEGIN                                      &   NAO HA MAIS SOLUCAO PARA
        PROXIMA.SOLUCAO:=FALSO;                     &   ESTE NO, A ULTIMA JA FOI
        RETURN;                                     &   ENCONTRADA.
    END;
&
&   OBTEM A NOVA SOLUCAO
&
    J:=MEMAUX(LIVRE+K);
    FOR I:=K TO TAMANHO.DA.SOLUCAO-1 DO
    BEGIN
        INCR J;
        MEMAUX(LIVRE+I):=J;
    END;
&
&   ARMAZENA A NOVA SOLUCAO
&
    FOR I:=0 TO TAMANHO.DA.SOLUCAO-1 DO
    BEGIN
        J:=MEMAUX(LIVRE+I);
        MEMAUX(INICIO.DA.SOLUCAO+I):=MEMAUX(INICIO.DA.DISPONIBILIDADE+J+1);
    END;
    PROXIMA.SOLUCAO:=VERDADE;
END;
?P
PROCEDURE LISTA.SOLUCAO.GERAL;
BEGIN
    BYTE (45) CABEC = (00C,20:" ","  SOLUCAO NUMERO: XXXXXX ");
    WORD POSICAO.NA.LINHA,FLAGLIN;
    INCR NUMERO.DA.SOLUCAO;
    BINDEC(NUMERO.DA.SOLUCAO,ACABEC+38,6,1);
    WRITE(7,ACABEC,45);
    FOR NO:=0 TO NUMERO.DE.NOS-1 DO
    BEGIN
        PEGA.INFORMACOES.DO.NO;
        FILL(ABUFFER,130,BRANCO);
        MBT(ASOLUC,ABUFFER,25);
        MBT(ANOME.DO.NO+NO*4,ABUFFER+20,4);
        BINDEC(NO,ABUFFER+27,6,1);
        BUFFER(35):=">";
        POSICAO.NA.LINHA:=FLAGLIN:=40;
        FOR I:=0 TO TAMANHO.DA.SOLUCAO-1 DO
        BEGIN
            BINDEC(MEMAUX(INICIO.DA.SOLUCAO+I),ABUFFER+POSICAO.NA.LINHA,6,1);
            POSICAO.NA.LINHA:=I+8;
            IF POSICAO.NA.LINHA > 122
            THEN BEGIN
                WRITE(7,ABUFFER,130);
                POSICAO.NA.LINHA:=40;
                FLAGLIN:=0;
                FILL(ABUFFER,130,BRANCO);
            END ELSE FLAGLIN:=1;
        END;
        IF FLAGLIN <> 0
        THEN WRITE(7,ABUFFER,130);
    END;
END;
```

```
?P
BYTE PROCEDURE SITUACAO.DA.SOLUCAO.ATUAL;
BEGIN
&
&   VARRE AS LIGACOES DO NO ATUAL
&
    FLAGSOLUCAO:=VERDADE;
    FOR J:=1 TO NUMERO.DE.LIGACOES DO
    BEGIN
        LIGACAO:=MEMAUX(INICIO.DAS.LIGACOES+J);
        DUMPGERAL(1); &**********************************************************
&
&   OBSERVA SE ESTA LIGACAO JA TEM SOLUCAO DEFINIDA
&
        K:=-1;
        FOR I:=0 TO NUMERO.DE.ORDEM.DO.NO-1 DO
        IF LIGACAO = MEMAUX(INICIO.DA.ORDEM.DOS.NOS+I)
        THEN I:=K:=NUMERO.DE.ORDEM.DO.NO;
        IF K > 0
        THEN BEGIN
&
&   ... E SE FOR O CASO, VE SE A SOLUCAO DE UM
&       NAO CONFLITA COM A DO OUTRO.
&
            INICIO.DAS.CORES.DA.LIGACAO:=MEMAUX(PCOR-LIGACAO);
            TAMANHO.DA.SOLUCAO.DA.LIGACAO:=MEMAUX(INICIO.DAS.CORES.DA.LIGACAO);
            INICIO.DA.SOLUCAO.DA.LIGACAO:=MEMAUX(PSOLUCOES-LIGACAO);
&
&   ... COMPARA AS DUAS SOLUCOES, (LIGACAO E NO ATUAL)
&
            FOR I:=0 TO TAMANHO.DA.SOLUCAO-1 DO
            FOR K:=0 TO TAMANHO.DA.SOLUCAO.DA.LIGACAO-1 DO
            IF MEMAUX(INICIO.DA.SOLUCAO+I)=MEMAUX(INICIO.DA.SOLUCAO.DA.LIGACAO+K)
            THEN BEGIN
                I:=TAMANHO.DA.SOLUCAO;
                K:=TAMANHO.DA.SOLUCAO.DA.LIGACAO;
                J:=NUMERO.DE.LIGACOES+1;
                FLAGSOLUCAO:=FALSO;
            END;
        END;
    END;
    DUMPGERAL(2); &**********************************************************
    SITUACAO.DA.SOLUCAO.ATUAL:=FLAGSOLUCAO;
END;
?P
BYTE PROCEDURE RETROCEDE;
BEGIN
&
&   REALIZA UM BACK TRACK
&
    RETROCEDE:=VERDADE;
    FLAGSOLUCAO:=FALSO;
    WHILE FLAGSOLUCAO = FALSO DO
    IF NUMERO.DE.ORDEM.DO.NO > 0
    THEN BEGIN
        J:=-1;
        FOR I:=1 TO NUMERO.DE.ORDEM.DO.NO DO
        FOR L:=1 TO NUMERO.DE.LIGACOES DO
        IF MEMAUX(INICIO.DA.ORDEM.DOS.NOS+I) =
            MEMAUX(INICIO.DAS.LIGACOES+L)
```

```
        THEN J:=I;
        IF J >= 0
        THEN NUMERO.DE.ORDEM.DO.NO:=J
        ELSE DECR NUMERO.DE.ORDEM.DO.NO;
        NO:=MEMAUX(INICIO.DA.ORDEM.DOS.NOS+NUMERO.DE.ORDEM.DO.NO);
        FLAGSOLUCAO:=PROXIMA.SOLUCAO;
    END ELSE BEGIN
        FLAGSOLUCAO:=VERDADE;
        RETROCEDE:=FALSO;
    END;
    FLAGSOLUCAO:=RETROCEDE;
    DUMPGERAL(4); &*****************************************************************
END;



BYTE PROCEDURE AVANCA;
BEGIN
    INCR NUMERO.DE.ORDEM.DO.NO;
    J:=NUMERO.DE.ORDEM.DO.NO+INICIO.DA.ORDEM.DOS.NOS;
    NO:=MEMAUX(J);
    AVANCA:=FLAGSOLUCAO:=IF NUMERO.DE.ORDEM.DO.NO < NUMERO.DE.NOS
                         THEN PRIMEIRA.SOLUCAO
                         ELSE VERDADE;
    DUMPGERAL(3); &*****************************************************************
END;
?P
BYTE PROCEDURE PROXIMA.SOLUCAO.GERAL;
BEGIN
    LABEL L0,L1,L2,L3,L4;
&
&   RECONHECE O PRIMEIRO NO.
&
    NO:=MEMAUX(INICIO.DA.ORDEM.DOS.NOS+NUMERO.DE.ORDEM.DO.NO);
&
&   VOU OBSERVAR A PROXIMA SOLUCAO DESTE NO.
&
L0:     IF PROXIMA.SOLUCAO = FALSO THEN GO TO L3;
&
&   OBSERVA SE E' VALIDO
&
L1:     IF NUMERO.DE.ORDEM.DO.NO >= NUMERO.DE.NOS THEN GO TO L4;
&
&   OBSERVA A SITUACAO DA SOLUCAO DESTE NO
&
        IF SITUACAO.DA.SOLUCAO.ATUAL = FALSO THEN GO TO L0;
&
&   AVANCA AO PROXIMO NO
&
        IF AVANCA = VERDADE THEN GO TO L1;
&
&   NAO CONSEGUIU OBTER MAIS SOLUCAO GERAL
&
L2:     PROXIMA.SOLUCAO.GERAL:=FALSO;
        RETURN;
&
```

```
&  TENTA RETROCEDER
&
L3:     IF RETROCEDE = FALSO OR
           TAMANHO.DA.SOLUCAO <= 0 THEN GO TO L2;
        GO TO L1;
L4:     PROXIMA.SOLUCAO.GERAL:=VERDADE;
        ESTADO:=8;
        TESTA.COMPATIBILIDADE.GERAL;
END;
?P
PROCEDURE ESCOLHA.A.OPERACAO;
BEGIN
&
&  OFERECE UM CONJUNTO DE OPERACOES A ESCOLHER, E LE A
&  OPERACAO ESCOLHIDA.
&
   BYTE (738) ESCOLHA =
              (" 00 - LER G.                                  "
              ," 01 - LER LLG.                                "
              ," 02 - OBTEM LLG A PARTIR DE G.                "
              ," 03 - LER AS CORES DOS NOS DE LLG.            "
              ," 04 - LER OS TAMANHOS DA SOLUCAO.             "
              ," 05 - GRAVA LLG.                              "
              ," 06 - OBTEM AS CORES DISPONIVEIS DE LLG.      "
              ," 07 - ORDENA AS CORES DISPONIVEIS EM LLG.     "
              ," 08 - ORDENA OS NOS DE LLG.                   "
              ," 09 - OBTEM A PRIMEIRA SOLUCAO GERAL.         "
              ," 10 - LE UMA SOLUCAO DE PARTIDA.              "
              ," 11 - OBTEM A PROXIMA SOLUCAO GERAL.          "
              ," 12 - OBTEM TODAS AS SOLUCOES GERAIS.         "
              ," 13 - DUMP DA MEMORIA AUXILIAR.               "
              ," 14 - LIGA RASTREAMENTO.                      "
              ," 15 - ESTABELECE FATORES DE ORDENACAO.        "
              ," 16 - TERMINA O PROGRAMA.                     "
              ," **** QUAL A OPERACAO ESCOLHIDA ?  ****  ");
   LABEL L1;
L1:     FOR I:=0 STEP 41 UNTIL 697 DO
        SAIDA(ADESCOLHA+I,41);
        IF LE.NUMERO = FALSO
        THEN GO TO L1;
        IF NUMERO < 0 OR NUMERO > 16
        THEN GO TO L1;
        OPERACAO:=NUMERO;
END;
?P
PROCEDURE EXECUTE.A.OPERACAO.ESCOLHIDA;
BEGIN
&
&  EXECUTA A OPERACAO ESCOLHIDA.
&
   CASE OPERACAO OF
   BEGIN
        LE.G;
        LE.LLG;
        OBTEM.LLG;
        LE.CORES.DOS.NOS.DE.LLG;
        LE.OS.TAMANHOS.DA.SOLUCAO;
        GRAVA.LLG;
        OBTEM.AS.CORES.DISPONIVEIS.EM.LLG;
        ORDENA.AS.CORES.DISPONIVEIS.EM.LLG;
```

```
        ORDENA.NOS.DE.LLG;
        BEGIN
            IF ESTADO <> 7
            THEN BEGIN
                ERRO.DE.ESTADO;
                RETURN;
            END;
            NUMERO.DE.ORDEM.DO.NO:=0;
            IF PROXIMA.SOLUCAO.GERAL = VERDADE
            THEN LISTA.SOLUCAO.GERAL;
        END;
        LE.SOLUCAO.DE.PARTIDA;
        BEGIN
            IF ESTADO < 8
            THEN BEGIN
                ERRO.DE.ESTADO;
                RETURN;
            END;
            NUMERO.DE.ORDEM.DO.NO:=NUMERO.DE.NOS-1;
            IF PROXIMA.SOLUCAO.GERAL = VERDADE
            THEN LISTA.SOLUCAO.GERAL;
        END;
        BEGIN
            IF ESTADO < 8
            THEN BEGIN
                ERRO.DE.ESTADO;
                RETURN;
            END;
            NUMERO.DE.ORDEM.DO.NO:=NUMERO.DE.NOS-1;
            NSOLUCAO2:=0;
            WHILE PROXIMA.SOLUCAO.GERAL = VERDADE AND NSOLUCAO2 < 20 DO
            BEGIN
                INCR NSOLUCAO2;
                LISTA.SOLUCAO.GERAL;
                NUMERO.DE.ORDEM.DO.NO:=NUMERO.DE.NOS-1;
            END;
        END;
        DUMP;
        FLAGDUMP:=-FLAGDUMP;
        LE.FATOR.DE.ORDENACAO;
        BEGIN
            COND:=0;
            FIMPROG(ACOND);
        END;
    END;
END;
?P
    FLAGDUMP:=-1;
    ABREENTRADA;
    WHILE VERDADE = VERDADE DO
    BEGIN
        ESCOLHA.A.OPERACAO;
        EXECUTE.A.OPERACAO.ESCOLHIDA;
    END;
END;
FIM DO ARQUIVO.
```