

## MC538/MC548 – PROJETO E ANÁLISE DE ALGORITMOS II

## Lista 4

1. Seja  $S$  o conjunto de sete números inteiros dado por  $S = \{11, 8, 7, 5, 4, 3, 2\}$ . Deseja-se saber se  $S$  contém um subconjunto  $S'$  tal que soma dos seus elementos seja exatamente igual a metade da soma dos elementos de  $S$ . Ou seja, trata-se de saber se o **problema da partição** para  $S$  tem uma solução e, em caso afirmativo, apresentar um subconjunto  $S'$  que responde ao problema.

Você deve resolver este problema usando um algoritmo de *backtracking* onde: **(i)** as soluções são representadas por tuplas de tamanho variável e, além disso, **(ii)** o valor do  $k$ -ésimo elemento da tupla é sempre menor do que o valor dos  $(k - 1)$  elementos que o antecedem na tupla para todo  $2 \leq k \leq 7$ .

Para explicar o seu raciocínio, a execução do algoritmo deve estar representada por meio da porção da árvore de espaço de estados que foi explorada. No interior de cada nó da árvore escreva qual é a solução parcial (tupla) construída até aquele momento. Além disso, coloque ao lado de cada nó da árvore um número que indica a ordem em que ele foi visitado, iniciando com o valor zero para a raiz. O algoritmo pára quando encontrar a primeira resposta ou concluir que ela não existe.

2. Escreva em um pseudo-código de alto nível, o algoritmo **recursivo** que implementa a solução da questão anterior.
3. Dado um grafo  $G = (V, E)$  direcionado e dois vértices  $u$  e  $v$ , considere o problema de se determinar todos os caminhos de  $u$  para  $v$ . Escreva um pseudo-código de um algoritmo de *backtracking* para este problema e explique o funcionamento do algoritmo. Argumente sobre a complexidade deste algoritmo.
4. Seja  $G = (V, E)$  um grafo completo não dirigido de 5 vértices com comprimento nas arestas dados pela matriz (simétrica)  $D$  abaixo. Nesta matriz, o elemento  $d_{ij}$  representa o comprimento da aresta  $(i, j)$ .

$$D = \begin{pmatrix} - & 8 & 12 & 14 & 10 \\ - & - & 7 & 4 & 16 \\ - & - & - & 18 & 15 \\ - & - & - & - & 9 \\ - & - & - & - & - \end{pmatrix}$$

Deseja-se encontrar o caminho hamiltoniano de comprimento mínimo neste grafo que tem o vértice  $\underline{1}$  como uma de suas extremidades. Para tanto, projetou-se um algoritmo de *branch-and-bound* onde as soluções (parciais ou completas) são representadas por tuplas começando com o número 1. Por exemplo, a tupla  $(1, 2, 3)$  representa a solução *parcial* formada pelas arestas  $(1, 2)$  e  $(2, 3)$ . **Note que uma tupla de tamanho 4 já é suficiente para definir por completo um caminho hamiltoniano !**

Lembrando que todo caminho hamiltoniano é também uma árvore geradora do grafo, pode-se usar como função classificadora a árvore geradora mínima do grafo. Na verdade, estando algumas arestas já incluídas na árvore geradora *a priori*, é possível construir a árvore geradora de comprimento mínimo da seguinte forma:

**Passo 1:** Seja  $F$  as arestas já incluídas na árvore *a priori*.

**Passo 2:** Ordene as arestas de  $(E - F)$  (ou seja, aquelas não incluídas *a priori* !) em ordem crescente do valor dos comprimentos.

**Passo 3:** Percorra as arestas nesta ordem e a cada iteração, inclua a aresta corrente na solução caso ela não forme um ciclo com as arestas anteriores.

**Passo 4:** Repita o passo 3 até que  $|V| - |F| - 1$  arestas tenham sido aceitas na solução.

Usando a *função classificadora* acima no algoritmo de *branch-and-bound* construa a porção da árvore de espaço de estados explorada pelo algoritmo. Ao lado de cada nó desta árvore indique qual o valor do limitante calculado pela *função classificadora* e no interior deste mesmo nó escreva qual a tupla que corresponde a ele.

**Lembre-se:** para uma tupla  $(x_1, \dots, x_p)$  as arestas  $(x_1, x_2), (x_2, x_3), \dots, (x_{p-1}, x_p)$  já devem ser consideradas como parte da árvore calculada pela função classificadora !

5. Considere o seguinte problema de escalonamento: Temos um conjunto  $J$  de tarefas, onde cada tarefa  $j \in J$  possui um tempo de processamento  $p_j$ , tempo de liberação  $r_j$  o qual representa o tempo em que a tarefa pode começar a ser executada, e uma previsão de término  $d_j$ . Se tivermos por exemplo uma tarefa  $j$  com  $p_j = 5, r_j = 100, d_j = 110$  significa que a tarefa deve ser executada de forma ininterrupta por 5 segundos, mas sua execução só poderá ocorrer após o tempo 100. A previsão de término da tarefa é o tempo 110.

Devemos escalonar as tarefas em uma única máquina e esta máquina só processa uma tarefa por vez. Em um escalonamento definimos o tempo de término da tarefa  $j$  como  $C_j$ . Definimos o atraso de uma tarefa como  $L_j = C_j - d_j$ .

O problema consiste em escalonar as tarefas em uma máquina de tal forma a minimizar o atraso máximo, ou seja, minimizar  $L_{max} = \max(L_1, \dots, L_n)$ .

Faça um algoritmo *branch-and-bound* para este problema. Este problema é para o caso não preemptivo, ou seja, após começar a execução de uma tarefa, esta deve ser executada de forma ininterrupta. Em um escalonamento preemptivo, uma tarefa pode ser interrompida e continuada depois.

Para calcular um limitante inferior de um nó na árvore de *branch-and-bound* use o seguinte algoritmo para o caso preemptivo do problema: Dentre as tarefas que podem ser executadas, ou seja, tem seus tempos de liberação menor do que o tempo atual, mantenha em execução a tarefa com menor previsão de tempo de término  $d_j$ . Esta regra de execução é um algoritmo ótimo para o caso preemptivo.

6. Considere o problema de se achar um conjunto independente máximo em um grafo. Formule este problema como um programa linear inteiro.
7. Considere o problema da cobertura por conjuntos: Temos um conjunto de elementos  $S$  e subconjuntos  $S_1, \dots, S_k$  de  $S$ , onde cada subconjunto  $S_i$  possui um peso  $p_i$ . Deseja-se achar

uma coleção  $C = \{S_p, \dots, S_q\}$  dos subconjuntos satisfazendo  $S = \cup_{S_i \in C} S_i$  e tal que a soma dos pesos dos subconjuntos de  $C$  seja mínima. Formule este problema como um programa linear inteiro.

8. Suponha que você está modelando um PLI em que deve-se escolher um conjunto de investimentos  $\{1, \dots, 7\}$ . Usando variáveis binárias modele as seguintes restrições:
- (a) Não se pode aplicar em todos os investimentos.
  - (b) Deve-se escolher pelo menos um dos investimentos.
  - (c) O investimento 1 não pode ser escolhido se o investimento 3 for escolhido.
  - (d) O investimento 4 só pode ser escolhido caso o investimento 2 seja escolhido.
  - (e) Ambos investimentos 1 e 5 devem ser escolhidos ou nenhum dos dois pode ser escolhido.
9. Hermes&Renato devem apresentar propagandas de seu programa de TV pelo menos 4 vezes por dia. As propagandas são apresentadas durante outros programas da emissora. Por isso, Hermes&Renato devem aparecer nas gravações destes programas para fazerem a sua propaganda. Suponha que uma gravação de um programa, seja ela qual for, tem duração de 1 hora. Além disso, são várias as gravações de cada programa em um dia. Estas gravações iniciam-se sempre nas horas cheias, começando às 10 da manhã e terminando às 7 da noite. A escolha de Hermes&Renato por qual gravação e de qual programa eles pretendem participar é influenciada pela reputação do programa, quem é a apresentadora, horário da gravação etc. Vamos denotar por  $p_{ik}$  o peso da preferência de Hermes&Renato pela gravação do programa  $k$  que começa na hora  $i$ .
- (a) Formule um PLI que gere um escalonamento de quais gravações Hermes&Renato devem participar de tal forma que se maximize a soma das preferências da dupla. Explique cada uma das restrições do seu modelo.
  - (b) Modifique a formulação para que Hermes&Renato não participem de mais de duas gravações seguidas sem antes terem um intervalo.
  - (c) Modifique a formulação para que Hermes&Renato tenham um escalonamento em que suas gravações comecem o mais tarde possível.
10. Seja um grafo não-orientado  $G = (V, E)$  com custos não-negativos  $w_e$  associados a toda aresta  $e \in E$ . Dado um subconjunto  $U$  de vértices de  $V$ , o *corte* definido por  $U$  em  $G$  é o conjunto de arestas  $(i, j)$  onde  $i \in U$  e  $j \notin U$ , o qual é denotado por  $\delta(U)$ .
- O *problema do corte máximo* em  $G$  é  $\mathcal{NP}$ -difícil. Neste problema deve-se encontrar o subconjunto  $U$  de vértices tal que a soma das arestas em  $\delta(U)$  é maximizado. Formule este problema usando Programação Linear Inteira. Explique o significado de cada variável e de cada restrição do seu modelo.
- Dica:* use variáveis binárias para representar os vértices que estão em  $U$  e as arestas que estão em  $\delta(U)$ .
11. Uma firma encontra-se em fase de expansão e esta planejando a sua mudança para um novo prédio onde ocupará  $m$  andares. Cada andar dispõe de  $W$  m<sup>2</sup> de área útil. A firma é

subdividida em  $n$  setores. Cada setor  $i \in \{1, \dots, n\}$  necessita de  $w_i$  m<sup>2</sup> de área para se instalar. Supõe-se que esta área é inferior à área de um andar e que um mesmo setor não será distribuído em mais de um andar. Além disso, a necessidade de interação entre o pessoal de dois setores  $i$  e  $j$  foi medida pelos administradores da firma. O problema é encontrar uma alocação dos setores aos andares que minimize o *custo* total de manter separados os diversos pares de setores da firma. Para um par de setores  $i$  e  $j$  o custo da mudança será  $c_{ij} > 0$  se  $i$  e  $j$  ficarem em andares distintos (independentemente de quantos andares os separam) e zero caso eles fiquem no mesmo andar. Formule este problema usando Programação Linear Inteira. Explique o significado de cada variável e de cada restrição do seu modelo.

12. Uma biblioteca recebeu  $n$  livros novos. Para cada livro  $k$  são conhecidos a sua altura  $a_k$  e a sua largura  $\ell_k$ . Foi decidido que os livros serão armazenados em estantes de altura  $H$  e de largura  $L$  fixas. Existe uma flexibilidade de se instalar uma quantidade variável de prateleiras nas estantes. Assim, se forem instaladas  $i$  prateleiras em uma estante, haverá  $i + 1$  vãos na estante disponíveis para colocar os livros naquela estante. Note que cada uma das prateleiras terá largura igual à largura da estante, ou seja,  $L$ .

Para instalar uma prateleira tem-se um custo  $c_p$  enquanto que cada estante custa  $c_e$ . O objetivo é determinar quantas estantes comprar e quantas prateleiras instalar em cada uma delas de modo a minimizar o custo da biblioteca. Para tanto, pede-se que o problema seja equacionado usando Programação Linear Inteira. Explique o significado de cada variável e de cada restrição do seu modelo.

Supõe-se que: (1) todos livros devem ser guardados em pé, (2) a profundidade das prateleiras é suficiente para acomodar qualquer livro, (3) a espessura de uma prateleira é desprezível e (4) existe uma quantidade conhecida  $j_{\max}$  de estantes que já se sabe ser capaz de armazenar todos os livros.

*Dicas:* a altura mínima de um vão entre duas prateleiras (ou entre o fundo ou o topo da estante e uma prateleira) não deve ser inferior à altura de um livro. Logo, é possível estabelecer um limite máximo sobre o número de prateleiras em uma estante.