

PUC-Rio  
Departamento de Informática  
Prof. Marcus Vinicius S. Poggi de Aragão  
Horário: 4as-feiras de 13 às 16 horas - Sala 511 RDC  
25 de novembro de 2010  
Período: 2010.2

## PROJETO E ANÁLISE DE ALGORITMOS (INF 2926)

### 3ª Lista de Exercícios

- Sejam  $P_1$ ,  $P_2$  e  $P_3$  três problemas tais que  $P_1 \leq_n P_2 \leq_{n^3 \log n} P_3$  (i.e.,  $P_1$  é redutível a  $P_2$  em tempo linear e  $P_2$  a  $P_3$  em tempo  $n^3 \log n$ ). Assuma a hipótese de que  $P_1$  é  $\Omega(n \log n)$ . Assuma também que você conhece um algoritmo  $O(n^3)$  para resolver  $P_3$ . Discuta as afirmações abaixo.
  - O que você pode dizer sobre a complexidade de resolução de  $P_2$ ? Qual a complexidade do melhor algoritmo que você conhece para  $P_2$ ?
  - Todo algoritmo que resolve  $P_2$  tem que gastar pelo menos tempo quadrático ( $P_2$  é  $\Omega(n^2)$ ).
  - $\Omega(n \log n)$  é um limite inferior para a complexidade de  $P_3$ .
  - $P_2$  pode ser resolvido no pior caso em tempo  $O(n \log n)$ .
- Sejam  $P_1$  e  $P_2$  dois problemas tais que  $P_1 \leq_{2^n} P_2$ . Assuma a hipótese de que  $P_1 \in NP\text{-completo}$ . Assuma também que você conhece um algoritmo  $O(n^3)$  para resolver  $P_2$ . Discuta as afirmações abaixo.
  - $P_2 \in NP\text{-completo}$ .
  - $P_1$  tem algoritmo polinomial para a sua resolução.
  - Somente algoritmos de complexidade exponencial resolvem  $P_1$ .
  - Somente algoritmos de complexidade exponencial resolvem  $P_2$ .
- Defina as classes de problemas  $P$ ,  $NP$  e  $NP\text{-completo}$ . Relacione estas classes e dê um exemplo de problema para cada classe.
- Seja  $P$  o conjunto dos problemas para os quais existem algoritmos determinísticos polinomiais para a sua resolução. Seja  $NP$  o conjunto dos problemas para os quais existem algoritmos **não**-determinísticos polinomiais para a sua resolução. Naturalmente  $P$  está contido em  $NP$ . Considere os problemas  $P_1 \in P$  e  $P_2 \in NP\text{-completo}$ . Indique se cada afirmação abaixo é verdadeira, falsa ou se não se sabe.
  - Conhece-se uma redução de  $P_1$  para  $P_2$  que toma tempo polinomial ( $O(n^k)$ ).
  - Se existe um algoritmo determinístico polinomial para a resolução de  $P_2$  então podemos afirmar que  $P_1 \in NP\text{-completo}$  assim como  $P_2 \in P$ .
  - $P_2$  é pelo menos tão difícil quanto  $3\text{-SAT}$ .
  - $3\text{-SAT}$  é pelo menos tão difícil quanto  $P_2$ .

(e) Conhece-se uma redução de  $P_2$  para  $P_1$  que toma tempo polinomial.

5. Sabe-se que o problema (MC), abaixo, pertence a NP-completo. Use este conhecimento para provar que (MSS) também pertence a NP-completo.

Clique-Máximo (MC) - Dado um grafo não-orientado  $G = (V, A)$  e uma constante  $K$ . Pergunta-se se este grafo  $G$  possui um clique (isto é um sub-grafo completo) de cardinalidade maior ou igual à  $K$ .

Estável-Máximo (MSS) - Dado um grafo não-orientado  $G = (V, A)$  e uma constante  $K$ . Pergunta-se se este grafo  $G$  possui um conjunto de vértices independentes (isto é, um conjunto de vértices onde não existe aresta entre nenhum par do conjunto) de cardinalidade maior ou igual à  $K$ .

(Dica: Siga os passos para provar que um problema é NP-completo. A redução pedida aqui é muito, mas muito simples. Leia com cuidado e desenhe exemplos dos problemas).

6. Prove que os problemas abaixo pertencem à  $NP$ . Em seguida, apresente uma relaxação (aceitável, a melhor que você pode conseguir) e calculável em tempo polinomial para as versões de otimização de cada um deles.

- (a) Árvore Geradora Mínima com restrição de Grau (AGG) - Dado um grafo não-orientado  $G = (V, E)$ , pesos  $w_e \in E$  e constantes  $K$  e  $C$ .

Pergunta-se se este grafo  $G$  possui uma árvore geradora cujo grau em nenhum dos vértices seja superior a  $K$  e cuja soma dos pesos das arestas nesta árvore seja no máximo  $C$ .

(minimize  $C$ )

- (b) Clique-Máximo (MC) - Dado um grafo não-orientado  $G = (V, A)$  e uma constante  $K$ . Pergunta-se se este grafo  $G$  possui um clique (isto é um sub-grafo completo) de cardinalidade maior ou igual à  $K$ .

(maximize  $K$ )

- (c) (MS): Dados um conjunto de máquinas  $M = \{m_1, m_2, \dots, m_p\}$  e um conjunto de tarefas  $T = \{t_1, t_2, \dots, t_q\}$  cada uma com uma duração  $d_i \in Z$ ,  $i = 1, \dots, q$  onde  $d_i$  associada e uma constante  $K$ . Considerando-se que as tarefas podem ser atribuídas à qualquer máquina indistintamente. Pergunta-se se existe uma atribuição das tarefas às  $p$  máquinas tal que o instante em que a última tarefa é terminada é menor ou igual que  $K$  (Isto é, a duração total é inferior ou igual a  $K$ ).

(minimize  $K$ )

7. Considere a multiplicação de  $n$  matrizes  $A_1, \dots, A_n$ . Considere também que denota-se o produto das matrizes  $A_k.A_{k+1} \dots A_q$  por  $A_{k..q}$  e que as dimensões das matrizes são dadas por  $d_k \times d_{k+1}$  para a matriz  $A_k$ . Assim,  $A_{1..n}$  terá dimensão  $d_1 \times d_{n+1}$ .

Aqui a multiplicação de pares consecutivos de matrizes  $A_k.A_{k+1}$  é feita calculando

$$a_{i,j}^{k..k+1} = \sum_{p=1}^{d_{k+1}} a_{i,p}^k \cdot a_{p,j}^{k+1}$$

para todo par  $(i, j)$  que é elemento de  $A_k.A_{k+1}$  e onde  $a_{i,j}^k$  representa o elemento  $(i, j)$  da matriz  $A_k$ .

Observe que a multiplicação de 3 matrizes  $A_1$ ,  $A_2$  e  $A_3$ , pode ser feita de duas maneiras:  $((A_1.A_2).A_3)$  e  $(A_1.(A_2.A_3))$ . (De quantas maneiras pode-se obter o produto de  $n$  matrizes ?) Observe também que para cada maneira de se multiplicar  $n$  matrizes pode-se ter que realizar um número diferente de multiplicações. Quantas são ?

Apresente um algoritmo para determinar a maneira de se multiplicar as  $n$  matrizes que utiliza o menor número de multiplicações. Analise a complexidade do algoritmo proposto. A complexidade é polinomial ? Qual a complexidade menor possível que um algoritmo que resolve este problema pode ter ?

8. Um comerciante possui um armazém que utiliza para suprir seus clientes de um único produto. O seu armazém pode guardar até  $C$  unidades do produto. Para as próximas  $T$  semanas o comerciante TEM que atender às demandas dos seus clientes que somam  $d_t$  para a semana  $t$ , onde  $t = 1, 2, \dots, T$ . Além disso, ele possui  $s_0 (\leq C)$  unidades em estoque antes do início da primeira semana, e já negociou com os fornecedores os preços unitários  $p_t$  ( $t = 1, 2, \dots, T$ ). Ele deseja planejar o atendimento dos seus clientes de modo a gastar o mínimo possível com a compra do produto.

Ajude ao comerciante a definir a sua estratégia ótima de compra do produto nas semanas  $t = 1, \dots, T$ .

- Apresente o algoritmo que obtém a estratégia de compra de menor custo e atende às demandas dos seus clientes.
  - Analise a complexidade do algoritmo proposto. A complexidade é polinomial ? Qual a complexidade menor possível que um algoritmo que resolve este problema pode ter ?
  - Execute o seu algoritmo sobre a seguinte instância:  $C = 12$ ,  $T = 5$ ,  $s_0 = 3$ ,  $d_1 = 7$ ,  $d_2 = 4$ ,  $d_3 = 15$ ,  $d_4 = 10$ ,  $d_5 = 7$  e  $p_1 = 3$ ,  $p_2 = 4$ ,  $p_3 = 7$ ,  $p_4 = 6$ ,  $p_5 = 8$ . Informe quanto o comerciante deve comprar em cada semana e o seu custo total.
9. Considere um tabuleiro de xadrez e um rei que está inicialmente na posição  $(1, 1)$  (as posições do tabuleiro são representadas por  $(i, j)$  onde  $1 \leq i \leq 8$  e  $1 \leq j \leq 8$ ). Para cada posição do tabuleiro estão associados um prêmio  $p_{ij}$  e um consumo  $q_{ij}$  (o prêmio pode ser em USD(!) e o consumo em litros de gasolina, por exemplo). Os prêmios e os consumos assumem somente valores positivos. O rei tem inicialmente  $Q$  unidades para consumir e pode passar quantas vezes quiser em cada posição do tabuleiro e a cada vez receber o prêmio e, naturalmente, consumir os seus recursos. Ao final (do passeio) **o rei tem que estar de volta na posição  $(1, 1)$** .

- Proponha um algoritmo para determinar o caminho que o rei deve fazer para obter o maior total possível em prêmios.  
(Dica: Suponha que você conhece a solução que obtém o maior total em prêmios dado que o rei está em cada uma das posições do tabuleiro e para cada consumo possível. Escreva agora o teorema do passo indutivo reforçando a hipótese indutiva. A prova por indução matemática (simples) de que você sabe resolver o problema do rei leva ao algoritmo).
- Analise a complexidade do algoritmo proposto. A complexidade é polinomial ? Qual a complexidade menor possível que um algoritmo que resolve este problema pode ter ? Qual a complexidade deste problema ?
- Suponha que  $Q = 7$  e que  $q_{ij} = 1$  e  $p_{ij} = 2*i + 3*j$  para todo  $(i, j)$ . Determine o caminho em que o rei acumula o maior total possível de prêmios. Repita o cálculo modificando apenas  $p_{22} = 100$  e mantendo os demais valores.

10. Considere que um ladrão possui um caminhão (:-)) cuja capacidade de peso é  $P$  e de volume é  $V$ . Este ladrão está assaltando um almoxarifado de um museu que contém  $n$  objetos. Em visita anterior, o ladrão levantou para cada objeto  $j$  o seu preço  $f_j$ , o seu peso  $p_j$  e o seu volume  $v_j$  (para  $j = 1, \dots, n$ ). Para escolher os objetos que é capaz de colocar no seu caminhão que maximizam o valor total do furto, o ladrão utilizou um algoritmo de programação dinâmica que utiliza a seguinte recursão:

$$F(j, p, v) = \max\{F(j-1, p, v), F(j-1, p-p_j, v-v_j) + f_j\}$$

- (a) Explique como esta recursão é utilizada pelo ladrão para obter o furto ótimo. Explique o que representa  $F(j, p, v)$ , como é obtida e para que valores de  $j$ ,  $p$  e  $v$  a cada iteração (o faz um iteração e como é a sua inicialização).
- (b) Qual é a complexidade do algoritmo resultante ?
- (c) Aplique este algoritmo na seguinte instância de furto:  $P = 3$ ,  $V = 6$ ,  $n = 3$ ,  $f_1 = 4$ ,  $p_1 = 2$ ,  $v_1 = 2$ ,  $f_2 = 5$ ,  $p_2 = 2$ ,  $v_2 = 2$ ,  $f_3 = 3$ ,  $p_3 = 1$  e  $v_3 = 2$ .
11. Considere o seguinte jogo: Dada um sequência de  $n$  ( $n$  é par) cartas enfileiradas com valores positivos e inteiros  $v_1, v_2, \dots, v_n$  abertos (conhecidos). Dois jogadores  $J1$  e  $J2$  alternam a vez de jogar. Uma jogada consiste em pegar uma carta de uma das pontas (a carta 1 ou a  $n$  no início, a 2 ou a  $n$  se o primeiro a jogar pegou a carta 1, ou 1 ou a  $n-1$  se ele pegou a carta  $n$ ). Vence aquele cuja soma dos valores das cartas que pegou for maior.
- (a) Apresente uma sequência de cartas onde o primeiro jogador não pega, no final, o maior valor total possível se ele escolhe a carta de maior valor na primeira jogada. (Ou seja, mostre que a estratégia gulosa não funciona para este problema.)
- (b) Proponha um algoritmo que calcule a estratégia ótima para o primeiro jogador. Este algoritmo deve executar em  $O(n^2)$ . Uma estratégia ótima é a informação necessária para que o jogador 1 determine cada uma das suas jogadas em  $O(1)$  de modo que obtenha no final o maior valor possível.
12. Considere o problema abaixo, para o qual existe uma prova de que sua versão decisão é NP-completo.

### Árvore Geradora Mínima com restrição de Grau (AGG):

Instância: Um grafo não-orientado  $G = (V, E)$ , pesos  $w_e \in E$  e um inteiro  $K$ .

Problema: Encontrar uma árvore geradora de  $G$  onde nenhum dos vértices tem grau superior a  $K$  e cuja soma dos pesos das arestas nesta árvore geradora seja mínima.

- (a) Enuncie a versão decisão de AGG e prove que pertence a NP.
- (b) Sejam os algoritmos abaixo:
- i. Selecione as menores  $|V| - 1$  arestas de  $E$  e calcule a soma das mesmas. Retorne este valor e este conjunto de arestas.
  - ii. Encontre o conjunto das  $|V| - 1$  arestas que correspondem à árvore geradora mínima de  $G$  e calcule a soma das mesmas. Retorne este valor e este conjunto de arestas.
  - iii. Ordene as arestas em ordem não-decrescente de valor. Inicialize  $T$ , um conjunto de arestas, como um conjunto vazio. Percorra a lista ordenada de arestas, inserindo em  $T$  sempre as arestas que não formam um ciclo no grafo  $G' = (V, T)$ . Quando  $|T| = |V| - 1$  pare. Retorne  $T$  e o valor da soma das arestas em  $T$ .

- iv. Ordene as arestas em ordem não-decrescente de valor. Inicialize  $T$ , um conjunto de arestas, como um conjunto vazio. Percorra a lista ordenada de arestas, inserindo em  $T$  sempre as arestas que não formam um ciclo no grafo  $G' = (V, T)$  e não tornem o grau de nenhum vértice em  $G'$  maior que  $K$ . Quando  $|T| = |V| - 1$  pare. Retorne  $T$  e o valor da soma das arestas em  $T$ .
- (c) Para cada um dos algoritmos acima, apresente sua complexidade (em função de  $n = |V|$  e  $m = |E|$ ) e justifique-as através da descrição das estruturas de dados utilizadas e da complexidade de suas respectivas operações.
- (d) (0.5) Quais dos algoritmos acima retornam uma solução (viável) para o problema (AGG) ?
- (e) Quais dos algoritmos acima retornam um valor garantidamente inferior (melhor) ao valor da solução ótima do problema ? Ou seja, quais deles são classificados como relaxações ? Qual deles seria a melhor relaxação ? Por que ?
- (f) Aplique os seu algoritmos iii) e iv) na instância abaixo.  
 Grafo:  $|V| = 7$ ,  $K = 3$ , pesos das arestas existentes  $w_{12} = 3$ ,  $w_{23} = 4$ ,  $w_{34} = 5$ ,  $w_{45} = 6$ ,  $w_{56} = 7$ ,  $w_{16} = 2$ ,  $w_{17} = 1$ ,  $w_{27} = 1$ ,  $w_{37} = 1$ ,  $w_{47} = 1$ ,  $w_{57} = 1$ ,  $w_{67} = 1$ .
- (g) Qual o novo valor obtido pelo algoritmo em iii) quando obriga-se a aresta (5,6) a pertencer ao conjunto resposta  $T$  ?

13. Considere os seguintes problemas:

(GBS) Instância: Um alfabeto finito  $\Sigma$ , um *string*  $x \in \Sigma^*$ , e um inteiro positivo  $K$ .

Questão: Existe uma sequência de  $K$  ou menos *swaps* (troca de dois caracteres adjacentes) que transforma  $x$  em um *string*  $y$  onde caracteres iguais formam sequências contínuas ? (I.e., sequências como *aba* não ocorrem em  $Y$ ).

(PART) Instância: Dado um conjunto de elementos  $X = \{x_1, x_2, \dots, x_n\}$  e seus respectivos tamanhos  $T = \{t_1, t_2, \dots, t_n\}$ , onde  $t_i$  é um inteiro para todo  $i$ .

Questão: Existe um subconjunto  $S$  de  $X$  tal que  $\sum_{x_i \in S} t_i = \sum_{x_i \in X-S} t_i$  ?

(CMC) Instância: Dados um grafo orientado  $G = (V, A)$  com comprimentos positivos associados aos arcos  $(i, j) \in A$ , dois vértices  $s, t \in V$  e um inteiro  $K$ .

Questão: Existe um caminho do vértice  $s$  ao vértice  $t$  que possua comprimento menor ou igual à  $K$  ?

(a) Prove que os problemas acima pertencem à  $NP$ .

(b) Dado que (PART), acima, é NP-completo, prove que (HSP), abaixo, também é NP-completo. (Um problema  $P_1$  NP-completo é um problema que pertence a NP e no qual é possível transformar, em tempo polinomial, qualquer problema em NP nele, i.e.  $P_a \alpha_{poly} P_1 \forall P_a \in NP$ ).

(HSP) Instância: Dado um conjunto de elementos  $N = \{1, 2, \dots, n\}$ , e respectivos pesos inteiro  $A = \{a_1, a_2, \dots, a_n\}$  e  $B = \{b_1, b_2, \dots, b_n\}$ , e uma constante  $C$ .

Questão: Existe um subconjunto  $S$  de  $N$  tal que

$$\frac{\sum_{i \in S} a_i}{\sum_{i \in S} b_i}$$

é maior ou igual a  $C$ .

Dica: observe que a maior razão é aquela que tem o menor denominador (1 por exemplo). Construa uma instância de (HSP) equivalente a uma instância de (PART).

14. Considere a versão otimização do problema (GBS), esta pertence a NP-difícil:

(GBS) Instância: Um alfabeto finito  $\Sigma$ , um *string*  $x \in \Sigma^*$ , e um inteiro positivo  $K$ . Problema: **Encontrar uma sequência com número mínimo** de *swaps* (troca de dois caracteres adjacentes) que transforma  $x$  em um *string*  $y$  onde caracteres iguais formam sequências contínuas? (I.e., sequências como *aba* não ocorrem em  $Y$ ).

Alg 1: Ordene os elementos de  $\Sigma$  presentes em  $x$  em ordem lexicográfica (alfabética) e obtenha o *string* final,  $y$ , fazendo cada caractere aparecer exatamente o número de vezes que aparece em  $x$ . Começando da esquerda, faça os *swaps* necessários em  $x$  para colocar o caractere mais próximo na posição que está em  $y$ , obtendo um novo *string*  $x$ . Repita até que o  $x$  esteja idêntico a  $y$ . Contabilize o número de *swaps* e retorne este valor.

Alg 2: Para cada par de caracteres iguais presentes em  $x$ , encontre o número de *swaps* necessários para colocá-los juntos. Encontre o número mínimo de *swaps* para cada caractere diferente. Retorne o maior destes valores (mínimos para cada caractere diferente).

(a) Execute os algoritmos acima na instância:

*cbaaababdbacca*

(b) Analise a complexidade dos algoritmos 1 e 2 acima. O tamanho da entrada  $n$  é o número de caracteres em  $x$ . (Encontre as funções  $\Omega$  e  $O$ ).

(c) Qual dos algoritmos acima obtém uma “solução viável” para o GBS? Por que este não obtém consistentemente o menor número possível de *swaps*?

(d) Qual dos algoritmos retorna um limite inferior para o número de *swaps*? (I.e. é uma relaxação). Prove que o valor que este algoritmo retorna é garantidamente um limite inferior (argumente!).

(e) Proponha um algoritmo que encontre sempre o número mínimo de *swaps*, ou seja a resposta ótima para GBS.