

PUC-Rio  
Departamento de Informática  
Prof. Marcus Vinicius S. Poggi de Aragão  
Período: 2009.2  
1 de dezembro de 2009  
Horário: 2as-feiras e 4as-feiras de 15 às 17 horas

## ANÁLISE DE ALGORITMOS (INF 1721)

### 3ª Lista de Exercícios

1. Exercícios Livro *Algorithms*, Dasgupta, Papadimitriou e Vazirani.

- Cap. 5: 5.13, 5.14, 5.15;
- Cap. 6: 6.2, 6.3, 6.4, 6.13, 6.17, 6.19.

2. Considere a multiplicação de  $n$  matrizes  $A_1, \dots, A_n$ . Considere também que denota-se o produto das matrizes  $A_k \cdot A_{k+1} \cdot \dots \cdot A_q$  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  $A_k \cdot A_{k+1}$  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  $A_k \cdot A_{k+1}$  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:  $((A_1 \cdot A_2) \cdot A_3)$  e  $(A_1 \cdot (A_2 \cdot A_3))$ . (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 ?

3. 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  $t = 1, \dots, T$ .

- (a) Apresente o algoritmo que obtém a estratégia de compra de menor custo e atende às demandas dos seus clientes.
- (b) Analise a complexidade do algoritmo proposto. A complexidade é polinomial? Qual a complexidade menor possível que um algoritmo que resolve este problema pode ter?
- (c) 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.
4. 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  $(1, 1)$** .
- (a) 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).
- (b) 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?
- (c) 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.
5. Sejam  $T = \{t_1, \dots, t_n\}$  e  $P = \{p_1, \dots, p_k\}$  duas sequências de caracteres on  $k \leq n$ .
- (a) Proponha um algoritmo linear para determinar se  $P$  é uma subsequência de  $T$ , isto é, se os elementos de  $P$  aparecem em  $T$  na mesma ordem que em  $P$ , mas não necessariamente consecutivos.
- (b) Suponha que a resposta do item anterior é negativa. Proponha um algoritmo para encontrar a maior subsequência de  $P$  que está em  $T$ .
- (c) Considere que para cada elemento de  $T$  é associado um custo positivo  $c_i$ ,  $i = 1, \dots, n$ . Proponha um algoritmo para encontrar a subsequência de  $P$  que é subsequência de  $T$  onde a soma dos custos dos elementos de  $T$  é maximizada.
- (d) Analise a complexidade dos algoritmos propostos. Qual a complexidade menor possível de um algoritmo que resolve estes problemas?
6. Sejam  $P_1$ ,  $P_2$  e  $P_3$  três problemas tais que  $P_1 \alpha_n P_2 \alpha_{n^3 \log n} P_3$  (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.

- (a) O que você pode dizer sobre a complexidade de resolução de  $P_2$  ? Qual a complexidade do melhor algoritmo que você conhece para  $P_2$  ?
- (b) Todo algoritmo que resolve  $P_2$  tem que gastar pelo menos tempo quadrático ( $P_2$  é  $\Omega(n^2)$ ).
- (c)  $\Omega(n \log n)$  é um limite inferior para a complexidade de  $P_3$ .
- (d) Todo algoritmo que resolve  $P_1$  pode ser usado para resolver  $P_2$ .
- (e) Todo algoritmo que resolve  $P_3$  pode ser usado para resolver  $P_2$ .
- (f)  $P_2$  pode ser resolvido no pior caso em tempo  $O(n \log n)$ .
7. Defina as classes de problemas  $P$ ,  $NP$  e  $NP$ -*completo* Relacione estas classes e dê um exemplo de problema para cada classe.
8. 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  $P_2 \in NP$  - *completo*. Indique se cada afirmação abaixo é verdadeira, falsa ou se não se sabe.
- (a) Conhece-se uma redução de  $P_1$  para  $P_2$  que toma tempo polinomial ( $O(n^k)$ ).
- (b) Se existe um algoritmo determinístico polinomial para a resolução de  $P_2$  então podemos afirmar que  $P_1 \in NP$  - *completo* assim como  $P_2 \in P$ .
- (c)  $P_2$  é pelo menos tão difícil quanto 3 - *SAT*.
- (d) 3 - *SAT* é pelo menos tão difícil quanto  $P_2$ .
- (e) Existe uma redução de  $P_2$  para  $P_1$  que toma tempo polinomial.
9. Dado que você conhece um algoritmo determinístico polinomial para a resolução do problema (CMC) abaixo, USE ESTE CONHECIMENTO para provar que você também conhece um algoritmo determinístico polinomial para a resolução do problema (VMC) apresentado em seguida.
- (CMC) Dado um grafo orientado  $G = (V, A)$  com comprimentos positivos associados aos arcos  $(i, j) \in A$ , dois vértices  $i, j \in V$  e uma constante  $K$ . Pergunta-se se existe um caminho do vértice  $i$  ao vértice  $j$  que possua comprimento menor ou igual à  $K$ .
- (VMC) Dado um grafo orientado  $G = (V, A)$ , com comprimentos positivos associados aos arcos  $(i, j) \in A$ , e uma constante  $K$ . Pergunta-se se existe um par de vértices  $(i, j)$  para o qual exista um caminho de  $i$  a  $j$  e de  $j$  a  $i$  que tenha comprimento menor ou igual à  $K$ .
10. Prove que os problemas (de decisão) abaixo pertencem à  $NP$ .
- (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  $K$ )

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

12. Considere o problema abaixo:

(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  $d_i$ ,  $i = 1, \dots, q$  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  $K$ ).

Dado que este problema pertence a NP-completo responda:

- Quando temos um algoritmo polinomial determinístico para resolvê-lo ?
- Proponha um algoritmo para resolver este problema em tempo polinomial.
- Em que casos a solução obtida pelo seu algoritmo é correta ?
- Descreva um algoritmo que encontre sempre a resposta correta para este problema.

13. Sabe-se que o problema (CH), abaixo, pertence a NP-completo. Use este conhecimento para provar que (AGG) também pertence a NP-completo.

**Caminho Hamiltoniano (CH)** - Dado um grafo não-orientado  $G = (V, A)$ .

*Pergunta-se se este grafo  $G$  possui um caminho Hamiltoniano (isto é um caminho que começa em algum vértice, e passa exatamente uma vez em cada vértice do grafo, terminando em algum outro vértice qualquer).*

**Árvore Geradora com restrição de Grau (AGG)** - Dado um grafo não-orientado  $G = (V, A)$  e uma constante  $K$ .

Pergunta-se se este grafo  $G$  possui uma árvore geradora (uma árvore que conecta todos os vértices do grafo) cujo grau (na árvore) em nenhum dos vértices seja superior a  $K$ .

(Dica: Siga os passos para provar que um problema é NP-completo. Prove que (AGG) pertence a  $NP$  e em seguida use (CH) para mostrar que todos os problemas em  $NP$  se transformam em (AGG). Na transformação você deve mostrar que a resposta sim para o problema (CH) acontece se e somente se acontece a resposta sim na instância obtida pela transformação do problema (AGG). Não esqueça de explicitar as complexidades dos algoritmos de verificação e de transformação).

14. Seja  $G = (V, E)$  um grafo não-orientado. Considere a seguinte função que leva um subconjunto  $S$  de  $V$  em um valor real:  $f(S) = |S| - 2|E(S)|$ , onde  $E(S)$  é o conjunto das arestas de  $G = (V, E)$  que tem as duas extremidades em  $S$ . Podemos agora enunciar o problema: **Conjunto Independente Máximo (MAXSS)** - Dado um grafo não-orientado  $G = (V, E)$ . Encontrar  $S^*$  tal que  $\forall S \subseteq V$ ,

$$f(S^*) \geq f(S), \text{ onde } f(S) = |S| - 2|E(S)|.$$

### Algoritmo Constroi I

Passo 0: *Inicialização*

$$S \leftarrow \emptyset$$

Passo 1: *Iteração*

Para  $v = 1$  até  $|V|$  e  $v \in V - S$  faça

\* Se  $f(S \cup \{v\}) > f(S)$  então  
 $S \leftarrow S \cup \{v\}$ ;

### Algoritmo Constroi II

Passo 0: *Inicialização*

$$S \leftarrow \emptyset$$

Passo 1: *Iteração*

mingrau =  $|V|$ ;  $v_{min} = -1$

Para  $v = 1$  até  $|V|$  e  $v \in V - S$  faça

\* Se grau( $v$ ) < mingrau então  
mingrau = grau( $v$ );  $v_{min} = v$ ;

Passo 2: *Atualização*

– Se  $v_{min} \neq -1$  e  $f(S \cup \{v_{min}\}) > f(S)$  então  
 $S \leftarrow S \cup \{v_{min}\}$ ; Volte ao Passo 1;

- Escreva um algoritmo que enumere todos os subconjuntos  $S$  de  $V$  e encontre  $S^*$ . Analise sua complexidade.
- Escreva um algoritmo que enumere todos os subconjuntos  $S$  de  $V$  que não possuem arestas ligando dois dos seus elementos, e encontre  $S^*$ . Analise sua complexidade.
- Analise a complexidade do algoritmos I e II acima.
- Aplique os 4 algoritmos acima no grafo  $G = (V, E)$  onde  $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$  e  $E = \{(1, 2), (1, 3), (1, 4), (2, 4), (2, 5), (2, 7), (3, 4), (3, 6), (4, 6), (4, 8), (5, 6), (5, 7)\}$ .

- (e) Explícite no algoritmo I o cálculo do valor da função  $f(S)$ . Descreva as estruturas de dados utilizadas e analise a complexidade do algoritmo resultante. Repita a análise para o algoritmo II.
- (f) Seria interessante o algoritmo I gerar soluções tomando decisões aleatorizadas ? Como isso seria feito ? E para o algoritmo II ?
- (g) O que é busca local ? Proponha um algoritmo de busca local para melhorar a solução  $S$  obtida pelos algoritmos I e II. Analise sua complexidade e aplique nas soluções obtidas para o grafo acima.
- (h) Compare as soluções obtidas pelos algoritmos enumerativos e os algoritmos I e II.

15. Seja o Problema do Caixeiro Viajante(TSP):

*(TSP): Dado um grafo completo  $G = (V, E)$ , custos  $c(i, j) \geq 0$  associadas às arestas  $(i, j)$  em  $E$ . Deseja-se encontrar uma seqüência de visitaç o dos v rtices de  $G$  iniciando e terminando em um mesmo v rtice (1), e passando exatamente uma vez por todos os demais v rtices onde a soma dos custos das arestas usadas na visitaç o seja **m nima**.*

Seja  $Z^*$  o custo de uma visitaç o de custo m nimo.

- (a) O custo da  rvore geradora de  $G$    maior, menor ou igual a  $Z^*$  ?
- (b) Dado que um conjunto de arestas  $E1$  tem que ser todo usado na visitaç o e que um conjunto de arestas  $E0$  n o pode ser usado, sugira uma forma de estimar o menor custo que essa visitaç o pode ter.
- (c) O custo da visitaç o  $1 \rightarrow 2 \rightarrow \dots \rightarrow n \rightarrow 1$    maior, menor ou igual a  $Z^*$  ?

16. Seja o Problema da  rvore Geradora com Restriç es de Capacidade:

*(CAP-MST): Dado um grafo  $G = (V \cup \{r\}, E)$ , custos  $c(i, j) \geq 0$  associadas  s arestas  $(i, j)$  em  $E$ , demandas  $q(v)$  associados aos v rtices  $v$  em  $V$ , e uma capacidade  $Q$ . Deseja-se encontrar uma  rvore geradora de  $G$ , cujas sub- rvores enraizadas no v rtice  $r$  possuam a soma das demandas dos seus v rtices inferiores ou iguais    $Q$ , cujo custo total de suas arestas seja **m nimo**. Considere a seguinte estrutura de dados para representar uma soluç o:*

- Um vetor com  $n = |V|$  posiç es que indica em que  rvore o v rtice  $v$  est . Isto  ,  $A[v] = k$  significa que o v rtice  $v$  est  na  rvore  $k$ .
- (a) Escreva um algoritmo para determinar se uma soluç o representada por um vetor  $A$    vi vel. Analise sua complexidade.
- (b) Escreva um algoritmo para determinar o valor da soluç o representada por  $A$ . Analise a sua complexidade.
- (c) Considere que um v rtice pode pertencer    $n$   rvores diferentes, logo  $A[v] \in \{1, \dots, n\}$  para  $v = 1, \dots, n$ . Supondo que uma vizinhança de uma soluç o arbitr ria  $A$    definida como todas as soluç es em que somente um v rtice troca de  rvore, investigar a vizinhança de  $A$    equivalente   avaliar o valor da soluç o representada por  $A$  onde modifica-se somente o valor do elemento  $A[v]$  onde este assume os valores  $k = 1, \dots, n, k \neq v$ , para  $v = 1, \dots, n$ .
  - i. Qual   a complexidade do algoritmo para encontrar a soluç o vizinha de menor valor, caso seja calculada o valor da soluç o de casa soluç o vizinha ?
  - ii. Esta complexidade pode ser reduzida ? No caso afirmativo, proponha um algoritmo com complexidade menor.