

PROJETO E ANÁLISE DE ALGORITMOS (INF 2926)

2ª Lista de Exercícios

1. Considere um mapa rodoviário e um motorista que tem que ir do vértice  $s$  ao  $t$ . O mapa é representado pelo grafo  $G = (V, E)$ , não-orientado onde os valores associados às arestas  $e \in E$ ,  $h_e$ , correspondem às altitudes das estradas correspondentes aos trechos. O motorista não gosta de altitude e quer fazer o caminho que minimiza a maior altitude que ele vai passar. Proponha um algoritmo que encontre esse caminho (Dica: utilize um algoritmo de árvore geradora mínima). Qual a complexidade do seu algoritmo ? Tem que ser  $O(n^2)$ .
2. Uma floresta é um grafo sem ciclos. Seja um grafo  $G = (V, E)$ , não-orientado onde os pesos das arestas  $e \in E$  são dados por  $w_e$ . Proponha um algoritmo para encontrar uma floresta com  $k$  arestas,  $K \leq n - 1$ , de peso total mínimo. Sua complexidade tem que ser  $O(n^2)$  ou inferior. Qual seria a complexidade utilizando o algoritmo de Kruskal ? Projete um algoritmo de complexidade  $O(Km \log K)$  para esse problema.
3. Uma 1-árvore geradora mínima pode ser obtida adicionando-se à árvore geradora mínima a menor aresta que não pertence à mesma. Suponha agora que essa aresta adicional precise estar conectada a um dado vértice  $v$ , isto é o único ciclo da 1 árvore deve conter o vértice  $v$  e após a remoção de uma aresta ligada ao vértice  $v$  deve restar uma árvore. Proponha um algoritmo para encontrar uma 1-árvore geradora mínima com esta restrição.
4. Considere que a árvore geradora de peso mínimo (AGM) de  $G = (V, E)$ , não-orientado onde os pesos das arestas  $e \in E$  são dados por  $w_e$ , é conhecida. Considere agora que um novo vértice foi acrescentado à  $G$  com arestas para todos os vértices em  $V$ . Proponha um algoritmo para encontrar a nova AGM. Seu algoritmo deve executar em  $O(n \log n)$  ou menos. Tente encontrar um algoritmo  $O(n)$ , existe.
5. Seja o grafo  $G = (V, E)$  onde  $V = \{1, 2, 3, 4, 5, 6\}$  e  $E = \{(1, 2, 5), (1, 3, 2), (1, 4, 2), (2, 5, 2), (3, 4, 1), (3, 6, 6), (4, 2, 1), (4, 6, 7), (5, 4, 1), (5, 6, 3)\}$ , onde os trios  $(u, v, l)$  indicam os vértice de partida, de chegada e a distância do arco.
  - (a) Aplique o algoritmo de Ford-Bellman ("Correção de Rótulos") para encontrar os c.m.c.'s do vértice 1 aos demais.
  - (b) Aplique o algoritmo de Dijkstra para encontrar os c.m.c.'s do vértice 1 aos demais.
  - (c) Suponha agora que os arcos não tem orientação, ou seja se existe o arco  $(u, v, l)$ , também existe o arco  $(v, u, l)$ , e aplique o algoritmo de Kruskal para encontrar a Árvore Geradora Mínima de  $G$ .

- (d) Ainda supondo que os arcos não tem orientação, aplique o algoritmo de Prim para encontrar a Árvore Geradora Mínima de  $G$ .
- (e) Ainda supondo que os arcos não tem orientação, aplique o algoritmo de Floyd-Warshall para encontrar os c.m.c.'s entre todos os pares de vértices.
6. Considere o algoritmo de Dijkstra para encontrar o caminho mais curto entre um vértice *fonte* e os demais vértices de um grafo orientado  $G = (V, E)$  onde a distância associada a um arco  $e \in E$  é dada por  $l_e = l_{vw}$  onde  $v$  e  $w$  são o vértice de partida e de chegada do arco  $e$ , respectivamente. (As complexidades devem ser obtidas em função de  $n = |V|$  e  $m = |E|$ ).

**Algoritmo Dijkstra** ( $s$  - fonte)

Passo 0: *Inicialização*

Seja  $S$  o conjunto de vértices com caminho mais curto a partir de  $s$  determinado, e  $\bar{S}$  seu complemento ( $\bar{S} = V - S$ ).

$S \leftarrow \emptyset$

$d(i) \leftarrow +\infty \forall i \in V$ ;

$d(s) \leftarrow 0$ ;  $pred(s) \leftarrow 0$ ;

Passo 1: *Iteração*

Enquanto  $S \subset V$  faça

1.1 Encontre  $v \in \bar{S}$  t.q.  $d(v) = \min_{w \in \bar{S}} d(w)$

1.2  $S \leftarrow S \cup \{v\}$ ;  $\bar{S} \leftarrow \bar{S} \setminus \{v\}$ ;

1.3 Para todo  $w \in \Gamma^+(v)$

Se  $d(v) + l_{vw} < d(w)$

então  $d(w) \leftarrow d(v) + l_{vw}$ ;  $pred(w) \leftarrow v$ ;

Responda aos itens abaixo:

- (a) Considere que o passo **1.1** é realizado através do uso de um vetor que armazena os valores  $d(i)$ ,  $i = 1, \dots, n$ . Qual a complexidade desta implementação do algoritmo de Dijkstra ?
- (b) Qual a complexidade global do passo **1.1** ? E do item **1.3** ?
- (c) A implementação que utiliza uma d-Heap para armazenar  $d(i)$  faz uso das operações *remove - topo*( $H$ ) no passo **1.1** e *reduz - chave*( $H, posicao(v), novo - valor$ ) no passo **1.3**. Reescreva estes passos utilizando estas operações e analise a complexidade global do algoritmo.
- (d) Considere agora que todas as distâncias  $l_{vw}$  são inteiros e que  $C = \max_{(v,w) \in E} l_{vw}$  (qual a maior distância possível em um caminho ?). Observe que a distância  $d(v)$  do vértice selecionado no passo **1.1** a cada iteração é maior ou igual à do vértice selecionado na iteração anterior. Considere que *buckets* (ou caixas) são criadas para os valores de 0 a  $nC$ . Cada *bucket* utiliza uma lista duplamente encadeada para armazenar os vértices  $v$  tais que  $d(v)$  tem o valor correspondente ao *bucket*. Um vetor ( $d$ ) é utilizado para indicar em que *bucket* está armazenado.
- Escreva os passos **1.1** e **1.3** utilizando a estrutura descrita acima.
  - Analise a complexidade deste algoritmo. Qual seria a complexidade deste algoritmo quando as distâncias são dadas pelo número de arestas no caminho ?

7. Considere o problema de encontrar o caminho-mais-curto entre todos os pares de vértices onde os arcos do grafo orientado  $G = (V, E)$  podem assumir valores negativos.
  - (a) Proponha um algoritmo para determinar se o grafo possui ciclo negativo. Analise sua complexidade.
  - (b) Proponha um algoritmo que, nos casos em que um ou mais ciclos negativos estão presentes em  $G$ , indique os pares de vértices que possuem caminhos-mais-curtos entre eles em  $G$ , ou seja cmc's que não passam pelos ciclos negativos.
8. (C,L & R, 1990, ex. cap 24-1 ou C, L, R & S, 2001, ex. cap 23-1 ) “Second best minimum spanning tree”
9. (C,L & R, 1990, ex. cap 25-1 ou C, L, R & S, 2001, ex. cap 24-1 ) “Yen’s improvement to Bellman-Ford”
10. (C,L & R, 1990, ex. cap 25-3 ou C, L, R & S, 2001, ex. cap 24-3 ) “Arbitrage”
11. Prove que é verdade ou que é falso (nesse caso apresentando um contra-exemplo).
  - (a) Se todas as distâncias dos arcos são diferentes, então a árvore de c.m.c. (de  $s$  aos outros vértices do grafo) é **única**.
  - (b) Considere a distância dos c.m.c. de  $s$  aos outros vértices do grafo. Se a distância de cada arco é aumentada de  $k$  unidades (ou seja  $l'_{uv} \leftarrow l_{uv} + k$  para todo arco  $e = (u, v) \in E$ ), as distâncias dos c.m.c. aumentam de um múltiplo de  $k$ .
  - (c) Se forem retiradas a orientações dos arcos de um grafo orientado  $G$  (i.e. passa ser possível passar nos dois sentidos), as distâncias dos c.m.c.'s permanecem as mesmas.
  - (d) Entre todos os c.m.c.'s existentes entre dois vértices em um grafo, o algoritmo de Dijkstra sempre acha o c.m.c. que possui o menor número de arestas.
12. Considere o c.m.c. entre um par de vértices em um grafo orientado  $G = (V, E)$ ,  $s$  e  $t$  por exemplo. Um arco vital com respeito a esse c.m.c. é um arco que, se retirado do grafo, a distância do c.m.c. de  $s$  a  $t$  aumenta. O arco mais vital é aquele cuja retirada causa o maior aumento.
  - Proponha um algoritmo para encontrar o arco mais vital dados  $G = (V, E)$ ,  $s$  e  $t$ .
  - Analise a complexidade do seu algoritmo.
13. Seja  $G = (V, E)$  um grafo orientado e acíclico, com distâncias  $l_e$  para  $e \in E$ , e  $s$  um vértice a partir do qual existe caminho para todos os demais vértices do grafo. Proponha um algoritmo com complexidade  $O(m)$ ,  $m = |E|$ , para encontrar os c.m.c.'s de  $s$  aos outros vértices do grafo. (Dica: use ordenação topológica.)
14. Suponha que foram obtidos os c.m.c.'s de  $s$  aos outros vértices de  $G$  e a árvore de c.m.c. é conhecida. Suponha agora que as distâncias de todos os arcos deve ser aumentada de  $k$  unidades (ou seja  $l'_{uv} \leftarrow l_{uv} + k$  para todo arco  $e = (u, v) \in E$ ). Proponha um algoritmo  $O(m)$  para obter os novos c.m.c.'s.
15. 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)  $P_2$  pode ser resolvido no pior caso em tempo  $O(n \log n)$ .
16. 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.
- (a)  $P_2 \in NP\text{-completo}$ .
- (b)  $P_1$  tem algoritmo polinomial para a sua resolução.
- (c) Somente algoritmos de complexidade exponencial resolvem  $P_1$ .
- (d) Somente algoritmos de complexidade exponencial resolvem  $P_2$ .
17. Defina as classes de problemas  $P$ ,  $NP$  e  $NP\text{-completo}$  Relacione estas classes e dê um exemplo de problema para cada classe.
18. Seja  $P$  o conjunto dos problemas para os quais existem algoritmos determinísticos polinomial para a sua resolução . Seja  $NP$  o conjunto dos problemas para os quais existem algoritmos **não**-determinísticos polinomial 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.
- (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  $P_1 \in NP\text{-completo}$  assim como  $P_2 \in P$ .
- (c)  $P_2$  é pelo menos tão difícil quanto  $3\text{-SAT}$ .
- (d)  $3\text{-SAT}$  é pelo menos tão difícil quanto  $P_2$ .
- (e) Conhece-se uma redução de  $P_2$  para  $P_1$  que toma tempo polinomial.
19. 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).
20. 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  $K$ )

21. 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).

22. 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.