

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.
 - i. Escreva os passos **1.1** e **1.3** utilizando a estrutura descrita acima.
 - ii. 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.
- Proponha um algoritmo para determinar se o grafo possui ciclo negativo. Analise sua complexidade.
 - 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).
- Se todas as distâncias dos arcos são diferentes, então a árvore de c.m.c. (de s aos outros vértices do grafo) é **única**.
 - 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 .
 - Se forem retiradas as 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.
 - 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.