Departamento de Ciência de Computadores FCUP

Desenho e Análise de Algoritmos (CC211) 2012/13

Exame (15.01.2013) *duração: 3h30* Cotação: 3, 3.5, 4, 3.5, 2.5, 3.5

N.º | | Nome | |

1. Pretende-se uma função Posmin(v, k, n) para determinar o índice da posição que contém o menor elemento de um vetor v de n **inteiros** quando considerados apenas os elementos $v[k], v[k+1], \ldots, v[n]$. Os elementos do vetor são indexados de 1 a n. Se k for maior do que n ou menor do que 1, a função retorna -1. Caso contrário, retorna o índice da primeira ocorrência do mínimo.

a) Apresente em pseudocódigo a função Posmin(v, k, n). Justifique sucintamente, mas com rigor, a correção do algoritmo apresentado.

b) Assuma que $1 \leq k \leq n$ e que a comparação dos valores se efetua em tempo constante. Descreva duas instâncias que determinem a complexidade temporal assintótica do algoritmo no melhor e no pior caso e caracterize tal complexidade como função de k e n.

2. Considere o algoritmo apresentado abaixo, em que Posmin é a função descrita no problema **1.**

```
Func(v, n)
  Para cada k ← 1 até n − 1 fazer
      j ← Posmin(v, k, n);
      Se j ≠ k então
          aux ← v[k];
          v[k] ← v[j];
          v[j] ← aux;
```

a) Escreva o enunciado de um problema que tal algoritmo resolve.

b) Prove que o algoritmo resolve corretamente o problema enunciado. Comece por descrever com rigor o estado das variáveis à entrada do ciclo "Para" e no fim de cada iteração desse ciclo. Na prova, admita que Posmin está correta e que o vetor v guarda inteiros como no problema **1.**

c) Seja $T(v, n)$ o tempo de execução do algoritmo numa instância arbitrária. Prove que existem constantes c_1, c_2 e n_0 positivas (não dependentes de v) tais que $c_1 n^2 \leq T(v, n) \leq c_2 n^2$ para $n \geq n_0$.

d) Na continuação de **2c)**, diga para que valores de $p \in \mathbb{N}$, a complexidade temporal do algoritmo se pode caracterizar como $\Theta(n^p)$, $\Omega(n^p)$ ou $O(n^p)$. Explique.

3. Considere uma estrutura de dados Q semelhante à dada nas aulas para implementação de uma heap de mínimo (com informação adicional) e a função Heapify(i) apresentada abaixo.

```
Heapify(i)
  l ← Left(i);
  Se (l > Q.s) então l ← i;
  r ← Right(i);
  Se (r > Q.s) então r ← i;
  smallest ← i;
  Se (Q.x[l].k < Q.x[smallest].k) então
      smallest ← l;
  Se (Q.x[r].k < Q.x[smallest].k) então
      smallest ← r;
  Se (i ≠ smallest) então
      Swap(i, smallest); // trocar elementos
      Heapify(smallest);
```

A estrutura tem quatro campos x, y, s e m, sendo x e y vetores e s o número de elementos que estão na heap e m o número máximo de elementos que pode conter. Cada elemento de x (i.e., da heap) é um par de valores (k, v) em que k define a prioridade e v o identificador do elemento. A posição v de y contém o índice da posição associada a v em x. Sabe-se que v não está na heap sse $y[v] = 0$.

a) Que propriedades caracterizam uma heap de mínimo?

b) Escreva em pseudocódigo $\textsc{Left}(i)$, $\textsc{Right}(i)$ e $\textsc{Swap}(i,j)$, e ainda, uma função $\textsc{NotEmpty}()$ para verificar se a heap ainda guarda elementos. Devem ter complexidade $O(1)$.

c) Que finalidade tem $\textsc{Heapify}(i)$? Qual é a sua complexidade? Justifique.

d) Escreva em pseudocódigo uma função $\textsc{Func\_Heap}()$ para retirar da heap o elemento que tem prioridade mínima (i.e., o valor de k menor), mantendo as propriedades da estrutura de dados. A função retorna o valor de v correspondente e deve ter complexidade $O(\log_2(Q.s))$, se $Q.s \geq 1024$.

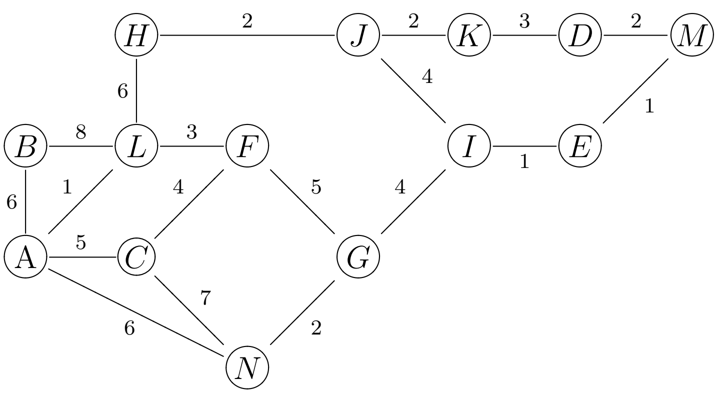
e) Suponha que $Q.m = 12$, $Q.s = 11$ e que o estado das variáveis é o seguinte

$Q.x$:	3 : 10	4 : 11	7 : 5	10 : 3	7 : 4	10 : 7	15 : 2	20 : 1	24 : 12	8 : 6	10 : 8	3 : 7

$Q.y$:	8	7	4	5	3	10	6	11	0	1	2	9

onde $k : v$ designa (k, v) e se supõe que a posição mais à esquerda tem índice 1. Represente $Q.x$ por uma árvore e indique o estado de $Q.x$, $Q.y$, $Q.m$ e $Q.s$ após a aplicação de $\textsc{Func\_Heap}()$.

4. Seja G o grafo seguinte em que os pesos associados aos ramos representam **distâncias**.



Das alíneas 4a) e 4b), resolva apenas uma.

a) Aplique o algoritmo de Dijkstra para determinar um caminho mínimo de J para cada um dos restantes vértices do grafo. Para cada vértice $v \neq J$, deve indicar a distância mínima $\delta(J, v)$ e o vértice $prec[v]$ que precede v no caminho encontrado pelo algoritmo. Acrescente informação ao grafo que permita compreender como obteve o resultado.

b) Aplique o algoritmo de Prim para construir uma árvore de cobertura para G de peso total mínimo, partindo de J. Para cada nó $v \neq J$, deve indicar o vértice $pai[v]$ a que v ficou ligado nessa árvore. Acrescente informação ao grafo que permita compreender como obteve o resultado.

c) Por redução ao absurdo, prove que o ramo $\{I, J\}$ não pertence a **nenhuma** árvore de cobertura de G com peso total mínimo. Comece por justificar que o grafo que se obtém quando se retira um ramo $\{u, v\}$ a uma árvore de cobertura $\mathcal{T}$ de G tem exatamente duas componentes conexas.

d) Suponha que pretende determinar o caminho de comprimento mínimo entre um nó s e cada nó v de um grafo $\mathcal{G} = (\mathcal{V}, \mathcal{E}, d)$ não dirigido **conexo**, tal que $v \neq s$. Tem disponível uma função $\textsc{AlgoPrim}(\mathcal{G}, pai, s)$ que lhe permite construir uma árvore de cobertura de $\mathcal{G}$ de peso mínimo, com raíz em s, dando como resultado o vetor $pai[v]$ que identifica o vértice a que v ficou ligado na árvore. Justifique que o algoritmo seguinte não resolve corretamente o problema.

$\textsc{CaminhosMinimos}(s, \mathcal{G})$
 $\textsc{AlgoPrim}(\mathcal{G}, pai, s)$;
 Para cada $v \in \mathcal{V} \setminus \{s\}$ fazer
  $\textsc{escreveCaminho}(s, v, pai)$;

$\textsc{escreveCaminho}(s, v, pai)$
 Se $v \neq s$ então
  $\textsc{escreveCaminho}(s, pai[v], pai)$;
 escrever(v);

(CONTINUA)

N.º [ ] Nome [ ]

5. Considere o grafo dirigido simétrico que resulta do grafo representado no problema **4.** por substituição de cada ramo $\{u, v\}$ por dois arcos (u, v) e (v, u), com o mesmo peso. Suponha que os pesos designam as capacidades dos arcos.

a) Represente nesse grafo um fluxo máximo de C para J e justifique que é máximo.

b) Suponha que o fluxo que indicou era o obtido pelo algoritmo de Edmonds-Karp numa dada iteração (após aumento de fluxo). Descreva sucintamente quais seriam os passos seguintes nesse algoritmo.

6. Seja $G_A = (V, A, d)$ um grafo dirigido acíclico com pesos e $G_E = (V, E, d')$ o grafo não dirigido que resulta de G_A por substituição de cada arco $(u, v) \in A$ por um ramo não dirigido $\{u, v\}$, com peso $d'(\{u, v\}) = d(u, v)$. Seja Γ um conjunto finito de caminhos em G_A, sendo cada caminho $\gamma \in \Gamma$ dado pela sequência de vértices que o define. Pretende-se verificar se é possível reconstruir G_A a partir de G_E e de Γ.

(Como exemplo, suponha que $\Gamma = \{\texttt{MDKJH}, \texttt{MEIJH}, \texttt{DK}, \texttt{HLFG}, \texttt{CNA}, \texttt{LBA}\}$ e G_E é o grafo não dirigido representado no problema **4.**)

a) Seja $G_\Gamma = (V, A_\Gamma)$ o grafo dirigido **acíclico** formado por V e pelos arcos que constituem os caminhos de Γ. Prove que:

i. qualquer que seja o ramo $\{u, v\} \in E$, se v é acessível de u em G_Γ então $(u, v) \in A$ (se for u acessível de v então $(v, u) \in A$);

ii. se v é acessível de u em G_Γ, então a relação de acessibilidade não varia se se acrescentar o arco (u, v) a A_Γ.

iii(*). qualquer que seja o ramo $\{u, v\} \in E$, se v não é acessível de u em G_Γ nem u é acessível de v em G_Γ então nada se pode concluir sobre o ramo $\{u, v\}$.
(**Sugestão:** recorde que G_A é acíclico e mostre que, para um $\{u, v\}$ fixo existiriam sempre pelo menos dois grafos G_A possíveis; se necessário, use indução matemática)

b) Suponha que os vértices estão numerados de 1 a $|V|$ e que Γ é lido da entrada padrão (pode arbitrar a representação que entender para Γ). Apresente (em pseudocódigo) um algoritmo **polinomial** para determinar, para cada $v \in V$, o conjunto de vértices $\mathcal{C}_v$ dos quais v é acessível em G_Γ. Pode explorar o facto de G_Γ ser um grafo dirigido acíclico.

c) Na continuação da **6b)**, apresente um algoritmo **polinomial** para resolver o problema da reconstrução de G_A. O algoritmo deve produzir informação sobre a parte de G_A que se consegue reconstruir e os sobre os ramos sobrantes, se existirem. Suponha que G_E é dado por uma *matriz de adjacências simétrica* ($M[i, j] = M[j, i] = 1$ se $\{i, j\} \in E$, e $M[i, j] = M[j, i] = 0$ se $\{i, j\} \notin E$).

(FIM)