Departamento de Ciência de Computadores FCUP

Desenho e Análise de Algoritmos (CC211) 2012/13

Exame (29.01.2013) *duração: 3h + 30 minutos*

UMA RESOLUÇÃO Cotação: (1.5+3+1), (2+1), (1+3+0.5+1), (1.5+1),(1+1+1.5),(1+2)

1. Seja $\mathcal{G} = (\mathcal{V}, \mathcal{A}, p, \{s, t\})$ uma rede, em que s e t são os nós origem e destino, $s \neq t$, e $p : \mathcal{A} \to \mathbb{Z}^+$ define os valores nos arcos. Para cada percurso γ_{uv} em $\mathcal{G}$, com origem u e fim v, designe-se por $\mathcal{P}(\gamma_{uv})$ o valor *máximo* nos arcos que o constituem, i.e., $\mathcal{P}(\gamma_{uv}) = \max\{p(x, y) \mid (x, y) \text{ é arco de } \gamma_{uv}\}$. Dizemos que γ_{uv} **é *ótimo* sse $\mathcal{P}(\gamma_{uv})$ for *mínimo* quando considerados todos os percursos alternativos de** u **para** v. Pretendemos encontrar um percurso ótimo $\gamma^\star_{st}$ de s para t.

a) Averigue a veracidade de cada uma das afirmações seguintes sobre $\gamma^\star_{st}$, justificando a resposta:

1. Se $\gamma^\star_{st}$ contiver ciclos, existe um percurso ϕ_{st} sem ciclos tal que $\mathcal{P}(\gamma^\star_{st}) = \mathcal{P}(\phi_{st})$, ou seja, se existe um percurso ótimo de s para t então existe um caminho ótimo de s para t.

   Resposta:

   A afirmação é verdadeira. Se $\gamma^\star_{st}$ contiver ciclos, então existe v tal que $\gamma^\star_{vv}$ é um ciclo contido em $\gamma^\star_{st}$. Assim, $\gamma^\star_{st}$ pode-se decompor como $\gamma^\star_{sv}\gamma^\star_{vv}\gamma^\star_{vt}$ e, por definição de $\mathcal{P}$, tem-se $\mathcal{P}(\gamma^\star_{st}) = \max\{\mathcal{P}(\gamma^\star_{sv}), \mathcal{P}(\gamma^\star_{vv}), \mathcal{P}(\gamma^\star_{vt})\}$ (aqui, $\gamma^\star_{vt} = \epsilon$ se $v = t$, e $\gamma^\star_{sv} = \epsilon$ se $s = v$, e $\mathcal{P}(\epsilon) = 0$). Se retirarmos o ciclo $\gamma^\star_{vv}$, obtemos o percurso $\gamma'_{st} = \gamma^\star_{sv}\gamma^\star_{vt}$ de s para t, com $\mathcal{P}(\gamma'_{st}) \leq \mathcal{P}(\gamma^\star_{st})$. Por outro lado, como $\gamma^\star_{st}$ é ótimo, $\mathcal{P}(\gamma'_{st}) \geq \mathcal{P}(\gamma^\star_{st})$. Logo, $\mathcal{P}(\gamma'_{st}) = \mathcal{P}(\gamma^\star_{st})$, o percurso γ'_{st} é ótimo e tem menos arcos do que $\gamma^\star_{st}$. Se γ'_{st} ainda contiver ciclos, podemos aplicar o mesmo procedimento a γ'_{st} para continuar a reduzir o número de arcos do percurso ótimo. Mas, como um percurso tem um número de arcos finito e $s \neq t$, este procedimento terá de terminar, resultando um percurso com pelo menos um arco e sem ciclos, ou seja um caminho.

2. Se $\gamma^\star_{st}$ for um caminho com dois ou mais arcos e que passa num vértice v (fixo), então existem *caminhos* ótimos γ_{sv} e γ_{vt} tais que o percurso $\gamma_{sv}\gamma_{vt}$ de s para t é ótimo (i.e., $\mathcal{P}(\gamma^\star_{st}) = \mathcal{P}(\gamma_{sv}\gamma_{vt})$).

   Resposta:

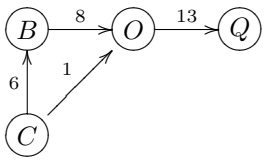
   A afirmação é verdadeira. Se o caminho $\gamma^\star_{st}$ passa em v, podemos decompor $\gamma^\star_{st}$ como $\gamma^\star_{sv}\gamma^\star_{vt}$, sendo $\gamma^\star_{sv}$ e $\gamma^\star_{vt}$ caminhos. Substituimos $\gamma^\star_{sv}$ e $\gamma^\star_{vt}$ por caminhos ótimos γ_{sv} e γ_{vt}, se não o forem, e $\gamma_{sv}\gamma_{vt}$ será um percuso tal que $\mathcal{P}(\gamma^\star_{st}) \geq \mathcal{P}(\gamma_{sv}\gamma_{vt})$, porque $\mathcal{P}(\gamma^\star_{st}) = \max\{\mathcal{P}(\gamma^\star_{sv}), \mathcal{P}(\gamma^\star_{vt})\} \geq \max\{\mathcal{P}(\gamma_{sv}), \mathcal{P}(\gamma_{vt})\} = \mathcal{P}(\gamma_{sv}\gamma_{vt})$. Mas, como $\gamma^\star_{st}$ é ótimo, então $\mathcal{P}(\gamma^\star_{st}) \leq \mathcal{P}(\gamma_{sv}\gamma_{vt})$, e consequentemente, $P(\gamma^\star_{st}) = \mathcal{P}(\gamma_{sv}\gamma_{vt})$.

3. Se $\gamma^\star_{st}$ for um caminho com dois ou mais arcos que passa num vértice v (fixo), pelo menos um dos dois subcaminhos $\gamma^\star_{sv}$ e $\gamma^\star_{vt}$ que constituem $\gamma^\star_{st}$ é ótimo, mas o outro pode ser ótimo ou não.

   Resposta:

   A afirmação é verdadeira. Se o caminho $\gamma^\star_{st}$ passa em v, podemos decompor $\gamma^\star_{st}$ como $\gamma^\star_{sv}\gamma^\star_{vt}$, sendo $\gamma^\star_{sv}$ e $\gamma^\star_{vt}$ caminhos. Se estes subcaminhos fossem ambos não ótimos então, se os substituissemos por caminhos ótimos γ_{sv} e γ_{vt}, obteriamos um percurso $\gamma_{sv}\gamma_{vt}$ de s para t, tal que $\mathcal{P}(\gamma^\star_{st}) > \mathcal{P}(\gamma_{sv}\gamma_{vt})$, porque $\mathcal{P}(\gamma^\star_{st}) = \max\{\mathcal{P}(\gamma^\star_{sv}), \mathcal{P}(\gamma^\star_{vt})\} > \max\{\mathcal{P}(\gamma_{sv}), \mathcal{P}(\gamma_{vt})\} = \mathcal{P}(\gamma_{sv}\gamma_{vt})$. Assim, $\gamma^\star_{st}$ não seria ótimo, contrariando o pressuposto de que era. Portanto, pelo menos um dos sub-caminhos $\gamma^\star_{sv}$ e $\gamma^\star_{vt}$ tem de ser ótimo se $\gamma^\star_{sv}$ for ótimo.

   É verdade que um dos dois sub-caminhos pode não ser ótimo. Por exemplo, na rede seguinte, $CBOQ$ é um caminho ótimo de C para Q (porque $\mathcal{P}(CBOQ) = \mathcal{P}(COQ) = 13$), contém CBO como sub-caminho, e CBO não é um caminho ótimo de C para Q (porque $\mathcal{P}(CBO) = 8 > \mathcal{P}(CO) = 1$).

b) Escreva um algoritmo para determinar um caminho ótimo $\gamma^{\star}_{st}$ de s para t, baseado numa adaptação do algoritmo de Dijkstra. Enuncie uma propriedade que justifique a correção desse algoritmo, relacionando-a com **1a)**.

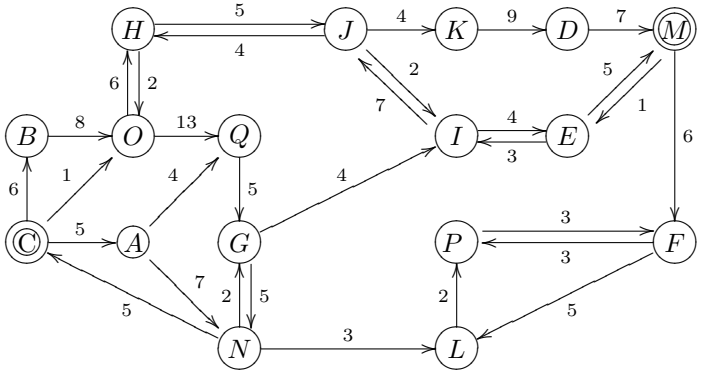
Resposta:

```
DIJKSTRA_ADAPTADO(s, G)
    Para cada v ∈ V fazer
        d[v] ← ∞;
        pai[v] ← NIL;
    d[s] ← 0;
    Q ← MK_PQ_HEAPMIN(cap, V);
    Enquanto(PQ_NOT_EMPTY(Q) ) fazer
        v ← EXTRACTMIN(Q);
        Se (v = t ou d[v] = ∞) então sai do ciclo;
        Para cada w ∈ Adjs[v] fazer
            Se d[w] > max(d[v], p(v, w)) então
                d[w] ← max(d[v], p(v, w));
                DECREASEKEY(Q, w, d[w]);
                pai[w] ← v;
    Se d[t] < ∞ então ESCREVECAMINHO(t, pai);
    senão escrever("Não existe caminho");
```

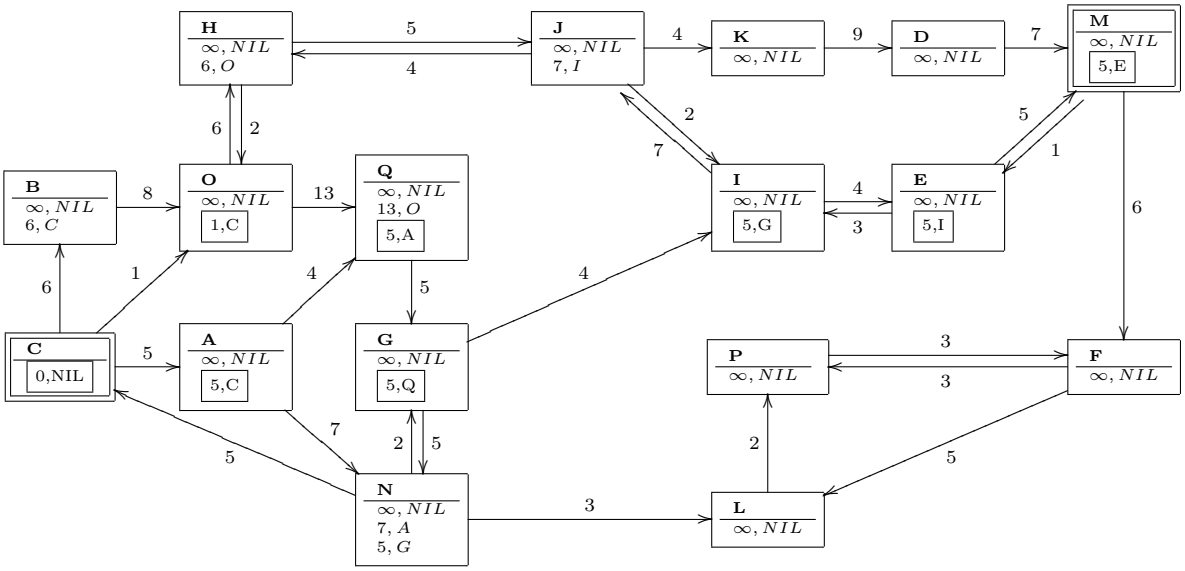
```
ESCREVECAMINHO(v, pai)
    Se pai[v] ≠ NIL então
        ESCREVECAMINHO(pai[v], pai);
    escrever(v);
```

Este algoritmo explora a existência de um caminho ótimo formado por subcaminhos ótimos se existir um percurso ótimo (duas primeiras afirmações **1a)**). Se $\delta(s, v)$ for o valor ótimo de $\mathcal{P}$ para os caminhos ótimos de s para v, então, em cada passo, $d[v]$ é um majorante de $\delta(s, v)$ e corresponde ao valor ótimo se o caminho só puder ter como vértices intermédios os que já sairam da fila. Quando v é extraído da fila, $d[v] = \delta(s, v)$ e o nó que antecede v no caminho ótimo encontrado é $pai[v]$, e, para os restantes vértices y ainda na fila, $d[y] \geq \delta(s, y) \geq \delta(s, v)$.

c) Aplique o algoritmo que apresentou para obter um caminho ótimo $\gamma^{\star}_{CM}$ de C para M na rede desenhada abaixo. Acrescente informação à rede que permita verificar os passos principais (valores intermédios) e indique a ordem pela qual os nós foram explorados.



Resposta:



Na rede, em cada nó v, o par $(d[v], Pai[v])$ mais abaixo é o valor final e os restantes são os valores que foram sendo substituídos. A sequência de saída dos nós da fila foi: C, O, A, Q, G, I, E, M (uma possibilidade, que seria, de facto, a ordem de saída da heap se, na comparação dos nós, fosse tido em conta o valor de d e, em caso de empate, a ordem alfabética das suas designações).

O caminho ótimo encontrado é $CAQGIEM$ e $\mathcal{P}(CAQGIEM) = 5$.

2. Seja $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ um grafo dirigido. Pretende-se determinar, para cada vértice $v \in \mathcal{V}$, o conjunto dos vértices $w \in \mathcal{V}$ tais que v é acessível de w e w é acessível de v em $\mathcal{G}$.

a) Apresente (em pseudocódigo) um algoritmo para resolver o problema com complexidade temporal $O(|\mathcal{V}| + |\mathcal{A}|)$, sendo $\mathcal{G}$ representado por listas de adjacências. Explique sucintamente a correção do algoritmo e apresente a complexidade dos passos principais e as estruturas de dados usadas.

Resposta:

O conjunto pretendido para cada $v \in \mathcal{V}$ é a componente fortemente conexa a que v pertence. O algoritmo de Kosaraju, dado nas aulas, pode assim ser usado para resolver o problema com a complexidade pretendida.

```
CompFortementeConexas(G)
1 | S ← DFS(G);
2 | G^T ← GrafoTransposto(G);
3 | Para v ∈ G.V fazer cor[v] ← branco;
4 | Enquanto (NotEmptyStack(S)) fazer
      v ← POP(S);
      Se cor[v] = branco então
          escrever("Nova componente");
          DFS_V_Transp(v, G^T);

DFS_V_Transp(v, G)
  | cor[v] ← preto;
  | Para cada w ∈ G.Adjs[v] fazer
      Se cor[w] = branco então
          DFS_V_Transp(w, G);
  | escrever(v);
```

```
DFS(G)
  | S ← MkEmptyStack(|G.V|);
  | Para cada v ∈ G.V fazer cor[v] ← branco;
  | Para cada v ∈ G.V fazer
      Se cor[v] = branco então DFS_V(v, G, S);
  | retornar S;

DFS_V(v, G, S)
  | cor[v] ← preto;
  | Para cada w ∈ G.Adjs[v] fazer
      Se cor[w] = branco então DFS_V(w, G, S);
  | PUSH(S, v);

GrafoTransposto(G)
  | Gt ← CriaNovoGrafo(|G.V|);
  | Para cada v ∈ G.V fazer
      Para cada w ∈ G.Adjs[v] fazer
          Gt.Adjs[w] ← Gt.Adjs[w] ∪ {v};
  | retornar Gt;
```

Supõe-se que $cor[.]$ é uma variável global e que S é passada por referência. Na chamada DFS_V_Transp$(v, \mathcal{G}^T)$ são escritos os nós que formam a componente fortemente conexa de v (que é também a componente dos nós que a constituem). A correção do algoritmo resulta de o grafo definido pelas componentes fortemente conexas ser um DAG e de os grafos $\mathcal{G}$ e $\mathcal{G}^T$ terem as mesmas componentes conexas.

Cada um dos três passos 1, 2 e 4 tem complexidade $O(|\mathcal{V}| + |\mathcal{A}|)$ e o passo 3 tem complexidade $O(|\mathcal{V}|)$.

b) Por aplicação do algoritmo, determine esses conjuntos para o grafo dado em **1c)**, ignorando os valores nos arcos. Para estabelecer a relação com o algoritmo, na resposta deve indicar o contéudo das estruturas de dados em passos cruciais do mesmo.

Resposta:

A: N, Q	A: C
B: O	B: C
C: A, B, O	C: N
D: M	D: K
E: I, M	E: M, I
F: L, P	F: P, M
G: I, N	G: Q, N
H: J, O	H: O, J
I: E, J	I: J, G, E
J: H, I, K	J: I, H
K: D	K: J
L: P	L: N, F
M: E, F	M: E, D
N: C, G, L	N: G, A
O: H, Q	O: H, C, B
P: F	P: L, F
Q: G	Q: O, A

Deve resolver apenas uma das duas questões 3b) e 6.
Se não resolver 3b), deverá ter em conta a informação que contém.

3. Suponha que v é um vetor de n inteiros e que os elementos de v são indexados de 1 a n. Considere a função FUNC(v, n) apresentada abaixo, ao centro, em pseudocódigo.

Linha	Algoritmo	Tempo
	Func(v, n):	
1	$k \leftarrow 1$;	a_1
2	Enquanto $(k < n)$ fazer	a_2
3	$r \leftarrow k$;	a_3
4	$j \leftarrow k + 1$;	a_4
5	Enquanto $(j \leq n)$ fazer	a_2
6	Se $v[j] \leq v[r]$ então	a_5
7	$r \leftarrow j$;	a_3
8	$j \leftarrow j + 1$;	a_4
10	Se $r \neq k$ então	a_6
11	$aux \leftarrow v[k]$;	a_7
12	$v[k] \leftarrow v[r]$;	a_8
13	$v[r] \leftarrow aux$;	a_9
14	$k \leftarrow k + 1$;	a_4

a) Justifique sucintamente, mas com rigor, que FUNC(v, n) ordena o vetor v por ordem crescente. Comece por descrever, com rigor, o estado das variáveis r, j e v na iteração k, imediatamente antes da execução da instrução que está na linha 10.

Resposta:

A função implementa ordenação por seleção (*selection sort*) do mínimo. Na iteração k, na linha 10, $j = n + 1$, a variável r guarda o índice da posição da última ocorrência do mínimo de $v[k], \ldots, v[n]$, e, se $k > 1$, o vetor v está ordenado por ordem crescente até à posição $k - 1$ (inclusivé) e $v[i] \geq v[k - 1]$ para todo i, com $k \leq i \leq n$.

Nas linhas 10-13, o elemento $v[r]$ é trocado com $v[k]$ se $r \neq k$, pelo que, este invariante é preservado no ciclo externo para o próximo valor de k. Assim, quando o ciclo termina, o vetor está ordenado, pois, na última iteração, também o último elemento ficou na posição correta.

b) (alternativa a 6.) À direita, em cada linha, a_i é uma constante positiva e representa o tempo de execução da instrução que está nessa linha, com excepção das linhas 2, 5, 6 e 10, em que esse tempo engloba a execução do teste da condição e a transferência de controlo. Seja $t_v(n)$ o tempo de execução do algoritmo para a instância (v, n).

1. Deduza a expressão de $t_v(n)$ quando: ***(i)*** todos os elementos de v são iguais, e ***(ii)*** todos são distintos e v está ordenado por ordem crescente.

Resposta:

Linha	Algoritmo	Tempo	Tempo (i)	Tempo (ii)
	Func(v, n):			
1	$k \leftarrow 1$;	a_1	a_1	a_1
2	Enquanto $(k < n)$ fazer	a_2	$a_2 n$	$a_2 n$
3	$r \leftarrow k$;	a_3	$a_3(n-1)$	$a_3(n-1)$
4	$j \leftarrow k + 1$;	a_4	$a_4(n-1)$	$a_4(n-1)$
5	Enquanto $(j \leq n)$ fazer	a_2	$\sum_{k=1}^{n-1} a_2(n-k+1)$	$\sum_{k=1}^{n-1} a_2(n-k+1)$
6	Se $v[j] \leq v[r]$ então	a_5	$\sum_{k=1}^{n-1} a_5(n-k)$	$\sum_{k=1}^{n-1} a_5(n-k)$
7	$r \leftarrow j$;	a_3	$\sum_{k=1}^{n-1} a_3(n-k)$	0
8	$j \leftarrow j + 1$;	a_4	$\sum_{k=1}^{n-1} a_4(n-k)$	$\sum_{k=1}^{n-1} a_4(n-k)$
10	Se $r \neq k$ então	a_6	$a_6(n-1)$	$a_6(n-1)$
11	$aux \leftarrow v[k]$;	a_7	$a_7(n-1)$	0
12	$v[k] \leftarrow v[r]$;	a_8	$a_8(n-1)$	0
13	$v[r] \leftarrow aux$;	a_9	$a_9(n-1)$	0
14	$k \leftarrow k + 1$;	a_4	$a_4(n-1)$	$a_4(n-1)$

Resposta (cont.):

No caso (i), pior caso, $t_v(n) = a_1+a_2+(2a_2+a_3+2a_4+a_6+a_7+a_8+a_9)(n-1)+(a_2+a_5+a_3+a_4)\sum_{k=1}^{n-1}(n-k)$, ou seja,

$$t_v(n) = a_1 + a_2 + (2a_2 + a_3 + 2a_4 + a_6 + a_7 + a_8 + a_9)(n-1) + (a_2 + a_5 + a_3 + a_4)\frac{(n-1)n}{2}.$$

No caso (ii), melhor caso, $t_v(n) = a_1 + a_2 + (2a_2 + a_3 + 2a_4 + a_6)(n-1) + (a_2 + a_5 + a_4)\sum_{k=1}^{n-1}(n-k)$, ou seja,

$$t_v(n) = a_1 + a_2 + (2a_2 + a_3 + 2a_4 + a_6)(n-1) + (a_2 + a_5 + a_4)\frac{(n-1)n}{2}.$$

2. Apresente a definição formal de "$t_v(n) \in \Theta(n^2)$" e, seguindo essa definição e a resposta à questão anterior, prove que, qualquer que seja (v, n), se tem $t_v(n) \in \Theta(n^2)$.

Resposta:

Diz-se que $t_v(n) \in \Theta(n^2)$ sse existirem constantes c_1, c_2 e n_0 positivas tais que $c_1n^2 \leq t_v(n) \leq c_2n^2$, para todo $n \geq n_0$.

Os casos (i) e (ii) determinam a complexidade temporal do algoritmo no pior caso e no melhor caso. Assim, de (i) concluímos que, qualquer que seja a instância (v, n), se tem

$$\begin{array}{rcl} t_v(n) & \leq & a_1 + a_2 + (2a_2 + a_3 + 2a_4 + a_6 + a_7 + a_8 + a_9)(n-1) + (a_2 + a_5 + a_3 + a_4)\frac{(n-1)n}{2} \\ & \leq & a_1 + a_2 + (2a_2 + a_3 + 2a_4 + a_6 + a_7 + a_8 + a_9)n + (a_2 + a_5 + a_3 + a_4)n^2 \\ & \leq & (a_1 + a_2)n^2 + (2a_2 + a_3 + 2a_4 + a_6 + a_7 + a_8 + a_9)n^2 + (a_2 + a_5 + a_3 + a_4)n^2 \\ & = & (a_1 + 4a_2 + 2a_3 + 3a_4 + a_5 + a_6 + a_7 + a_8 + a_9)n^2 \end{array}$$

pois $1 \leq n \leq n^2$, para todo $n \geq 1$. Tomamos assim $c_2 = a_1 + 4a_2 + 2a_3 + 3a_4 + a_5 + a_6 + a_7 + a_8 + a_9$.

Por outro lado, de (ii) concluímos que

$$\begin{array}{rcl} t_v(n) & \geq & a_1 + a_2 + (2a_2 + a_3 + 2a_4 + a_6)(n-1) + (a_2 + a_5 + a_4)\frac{(n-1)n}{2} \\ & \geq & (a_2 + a_5 + a_4)\frac{(n-1)n}{2} = \beta(n-1)n \geq c_1n^2 \end{array}$$

para $\beta = (a_2 + a_5 + a_4)/2$, se $n \geq 1/(1 - c_1/\beta)$ e $c_1/\beta < 1$. Se definirmos $c_1 = \beta/2 = (a_2 + a_5 + a_4)/4$, então $t_v(n) \geq c_1n^2$ para $n \geq 1/(1 - 1/2) = 2$, e podemos tomar $n_0 = 2$.

c) Sendo a complexidade do algoritmo dada pelo máximo de $t_v(n)$ para (v, n) qualquer, diga para que valores de $p \in \mathbb{N}$, a complexidade se pode caracterizar como $\Theta(n^p)$, $\Omega(n^p)$ ou $O(n^p)$. Explique.

Resposta:

Como $t_v(n) \in \Theta(n^2)$ qualquer que seja a instância (v, n), então, pela definição de $\Theta(n^p)$, $\Omega(n^p)$ e $O(n^p)$, podemos concluir que $T(n) = \max\{t_v(n) \mid v \text{ vetor de } n \text{ elementos}\} \in \Omega(n^p)$ para $p \in \{0, 1, 2\}$, e $T(n) \in O(n^p)$ para $p \in \mathbb{N} \setminus \{0, 1\}$, e $T(n) \in \Theta(n^p)$ apenas para $p = 2$.

d) Designe por FUNC_NOVA(v, n) a função que se obtém quando se substitui, na linha 6, a condição $v[j] \leq v[r]$ por $v[j] < v[r]$. O que contém r na iteração k na linha 10? Conclua que FUNC_NOVA(v, n) também ordena v por ordem crescente e diga, justificando, que relação existe entre a complexidade temporal assintótica de FUNC_NOVA(v, n) e de FUNC(v, n).

Resposta:

O valor de r é o índice da primeira ocorrência do mínimo de $v[k], \ldots, v[n]$, para o estado de v nessa iteração. Assim, a função também implementa ordenação por seleção do mínimo. Se o vetor v não tiver valores repetidos, a sequência de valores de r é igual nos dois casos.

As funções são equivalentes do ponto de vista da complexidade assintótica, sendo $T(n) \in \Theta(n^2)$ em ambas. Se o vetor estiver ordenado (caso (ii)) então as expressões de $t_v(n)$ seriam idênticas para os dois algoritmos. Por outro lado, o pior caso de FUNC_NOVA(v, n) não obrigaria a realizar mais operações do que as que FUNC(v, n) realiza no caso (i).

4. Considere o problema de formar uma certa quantia Q usando moedas de valores $v[1], v[2], \ldots, v[m]$, sendo $v[1] > v[2] > \ldots > v[m]$, tendo disponíveis $c[i]$ moedas de valor $v[i]$ em caixa, sendo $c[i] \in \mathbb{Z}_0^+$, para $1 \leq i \leq m$, só podendo usar essas moedas.

a) Escreva uma recorrência que defina o número de alternativas para a formação da quantia Q nessas condições (só distinguindo quantas moedas de cada tipo são usadas). Explique de que modo se pode usar programação dinâmica com memoização para calcular esse número, dados Q, v, c e m.

Resposta:

Seja $num(x, j)$ o número de alternativas para formação da quantia x quando se podem utilizar apenas as moedas de valores $v[j], \ldots, v[m]$ disponíveis. Então, o valor pretendido é $num(Q, 1)$ e

$$\begin{array}{lcl} num(x,j) & = & 1, \text{ se } x = 0, \text{ para todo } j, \text{ com } 1 \leq j \leq m \\ num(x,m) & = & 1, \text{ se } x \neq 0,\ x/v[m] \leq c[m] \text{ e } x \text{ é múltiplo de } v[m] \\ num(x,m) & = & 0, \text{ se } x \text{ não é múltiplo de } v[m] \text{ ou } x/v[m] > c[m] \\ num(x,j) & = & \displaystyle\sum_{0 \leq k \leq \min\{c[j], \lfloor \frac{x}{v[j]} \rfloor\}} num(x - kv[j], j+1), \text{ se } 1 \leq j < m \text{ e } x \neq 0. \end{array}$$

Na chamada recursiva, tabelaria os valores $num(x, j)$ para $x \neq 0$ que fosse calculando, e só calcularia um novo valor se ainda não estivesse na tabela.

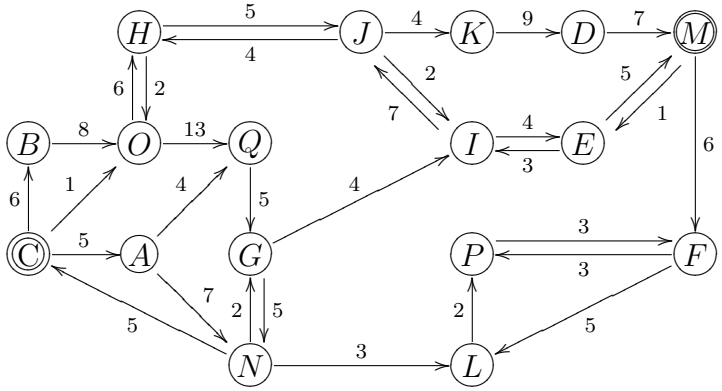
b) Imagine que se pretende formar a quantia Q com o número **mínimo** de moedas possível e que *se usa sempre a moeda mais alta que se puder (aplicando-se a mesma abordagem à quantia restante).* Explique em que sentido tal estratégia é *greedy* e indique se é correta. Justifique.

Resposta:

A estratégia é greedy porque a opção tomada é apenas localmente ótima pois, "cega pela avidez", antes de tomar a decisão não verifica se é ou não possível formar a quantia restante com as moedas disponíveis.

Assim, como no problema "Não lhes dês troco", é óbvio que esta estratégia ávida não é correta. Suponhamos que $m = 6$ e que as moedas eram de 2 euros, 1 euro, e de 50, 20, 10 e 5 cêntimos, mas, que só se tinha em caixa moedas de 50 e de 20 cêntimos, tendo pelo menos uma de 50 cêntimos e três de 20 cêntimos. Então, se se aplicasse a estratégia indicada, não se conseguiria perfazer 60 cêntimos, embora seja claro que se podia perfazer 60 cêntimos se se usasse três moedas 20 cêntimos (e outra estratégia).

5. Suponha que a rede representada em **1b)** é uma rede de fluxo, com $s = C$ e $t = M$, e que $p(x, y)$ indica a capacidade do arco (x, y), para cada $(x, y) \in \mathcal{A}$.



a) Indique um fluxo f de C para M tal que $f(G, I) = 2$, $f(H, J) = 5$, $f(C, B) = 6$ e $f(J, I) = 2$.

b) Determine a capacidade residual associada a f para cada par $(x, y) \in \mathcal{V} \times \mathcal{V}$, com $x \neq y$. Apresente os cálculos que efetuar, omitindo os casos em que $f(x, y) = 0$.

c) Partindo de f, aplique o algoritmo de Edmonds-Karp para obter um fluxo máximo $f^\star$. Descreva sucintamente os passos efetuados pelo algoritmo.

6. (alternativa a 3b) Seja $G_A = (V, A)$ um grafo dirigido acíclico e $G_E = (V, E)$ o grafo não dirigido que

resulta de G_A por substituição de cada arco $(u, v) \in A$ por um ramo não dirigido $\{u, v\}$. 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. Seja $G_\Gamma = (V, A_\Gamma)$ o grafo dirigido formado por V e pelos arcos que constituem os caminhos de Γ.

a) Sabemos que nada se pode concluir sobre a orientação de um ramo $\{u, v\}$ de E no grafo G_A se nem v for acessível de u em G_Γ nem u for acessível de v em G_Γ. Justifique agora que:

1. O grafo G_Γ é acíclico (i.e., um DAG).
2. 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 $(v, u) \in A$).

b) Assuma que os vértices estão numerados de 1 a $|V|$, que $|V|$ é conhecido, que Γ é lido da entrada padrão e que G_E se encontra dado por uma matriz de adjacências simétrica M tal que $M[i, j] = M[j, i] = 1$ se $\{i, j\} \in E$, e $M[i, j] = M[j, i] = 0$ se $\{i, j\} \notin E$.

Baseando-se em **6a)**, escreva um algoritmo para resolver o problema da reconstrução de G_A em tempo $O(|\Gamma||V| + |V|^3)$. O algoritmo deve produzir informação sobre a parte de G_A que se consegue reconstruir e sobre os ramos sobrantes, se existirem. Use matrizes de adjacências para representar os grafos G_T e G_A. Comece por apresentar as ideias principais do algoritmo que delineou e por justificar a sua correção e complexidade.

(FIM)