Instituto Superior Técnico

Análise e Síntese de Algoritmos

Ano Lectivo 2021/2022 Exame de Época Especial

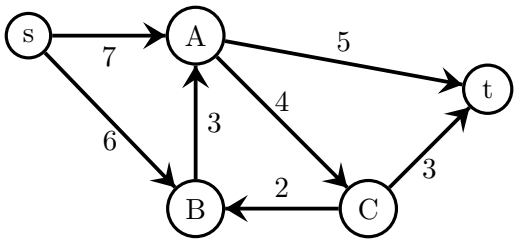
RESOLUÇÃO

I. (3 + 3 + 3 + 3 = 12 val.)

I.a) Considere a maior sub-sequência comum entre as strings $ABBACBA$ e $ABCBCABA$ e calcule a respectiva matriz de programação dinâmica $c[i, j]$ para este problema, em que o índice i está associado à string $ABBACBA$. Indique os seguintes valores: $c[1, 3]$, $c[2, 6]$, $c[3, 3]$, $c[4, 4]$, $c[5, 6]$, $c[6, 3]$, $c[7, 8]$.

$c[1,3]$	$c[2,6]$	$c[3,3]$	$c[4,4]$	$c[5,6]$	$c[6,3]$	$c[7,8]$
1	**2**	**2**	**3**	**4**	**3**	**6**

I.b) Considere a rede de fluxo da figura:



Aplique o algoritmo Relabel-to-Front à rede de fluxo da figura. Considere que as listas de vizinhos dos vértices intermédios são as seguintes:

- $N[A] =< C, B, s, t >$
- $N[B] =< A, C, s >$
- $N[C] =< B, t, A >$

e que a lista de vértices inicial é $L =< A, B, C >$. Preencha a tabela abaixo com as alturas finais dos vértices e a sequência de diferentes configurações da lista L.

	s	A	B	C	t
$h()$	**5**	**6**	**6**	**7**	**0**

	1°	2°	3°	4°	5°	6°	7°	8°
	$\langle A, B, C \rangle$	$\langle B, A, C \rangle$	$\langle A, B, C \rangle$	$\langle C, A, B \rangle$				

I.c) Considere a função recursiva:

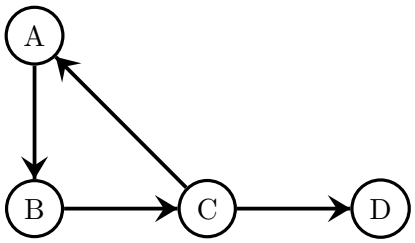
```
int f(int n) {
  int i, j, r = 0, k = 0;

  for (i = n; i > 0; i /= 2) {
    for (j = i; j > i/2; j--) { // Loop 1
      k = k+2*j;
    }
  }

  while (j++ < 4)
    r += 2*f(n/2);

  while (k-- > 0) { // Loop 2
    r = r + 1;
  }

  return r;
}
```

1. Determine um upper bound medido em função do parâmetro n para o número de iterações dos loops **1** e **2** da função f.
2. Determine o menor majorante assimptótico da função f em termos do número n utilizando os métodos que conhece.

Solução:

1. Analisamos cada loop separadamente:
   - *Loop 1:* j assume os valores $n,\ n-1,\ \ldots,\ 1$, logo o número de iterações é $n = O(n)$.
   - *Loop 2:* Observando que após a execução do loop 1, $k = \sum_{j=1}^{n} 2j = n(n+1)$, e que o número de iterações do loop 2 é igual ao valor de k, temos que o número de iterações do loop 2 é dado por $n(n+1) = O(n^2)$.
2. Observamos que:
   $$T(n) = 4\,T(n/2) + O(n^2)$$
   Podemos aplicar o Teorema Mestre na forma simplificada notando que $a = 4$, $b = 2$, $d = 2$ e $log_b\ a = 2 = d$. Concluímos que $T(n) = O(n^2 \lg n)$.

I.d) Considere o grafo dirigido que se apresenta em baixo:



Aplique o algoritmo que utiliza duas travessias em profundidade primeiro (Kosaraju-Sharir) para encontrar os componentes fortemente ligados do grafo. Considere que na primeira DFS os vértices são considerados por ordem alfabética (ou seja, A, B, C, ...). Em ambas as DFS, os vértices adjacentes são sempre considerados também por ordem alfabética.

- Indique os tempos de descoberta d e de fim f de cada vértice após a **segunda DFS**.
- Indique os componentes fortemente ligados **pela ordem em que são descobertos**.

	A	B	C	D
d[i]	**1**	**3**	**2**	**7**
f[i]	**6**	**4**	**5**	**8**
SCCs	$\{\mathbf{A}, \mathbf{B}, \mathbf{C}\}, \{\mathbf{D}\}$			

II. (3 + 3 + 2 = 8 val.)

II.a) Dada uma sequência de inteiros positivos $\langle x_1, ..., x_n \rangle$, pretende desenvolver-se um algoritmo que determina o maior valor suceptível de ser obtido a partir da expressão $x_1/x_2/x_3/.../x_n$, determinando a ordem pela qual as divisões devem ser efectuadas. Por exemplo, dada a sequência $\langle 16, 8, 4, 2 \rangle$, a parentização que resulta no maior valor final é: $(16/((8/4)/2)) = 16$.

1. Seja $M[i, j]$ o maior valor que é possível obter a partir da expressão $x_i/x_{i+1}/.../x_j$ e $m[i, j]$ o menor valor. Por exemplo, dada a sequência $\langle 16, 8, 4, 2 \rangle$, $M[1, 4] = 16$ e $m[1, 4] = 0.25$. Admitindo que a sequência dada como input é $\langle x_1, ..., x_n \rangle$, defina $M[i, j]$ e $m[i, j]$ recursivamente completando os campos em baixo:

$$M(i,j) = \begin{cases} \boxed{} & \text{se } i = j \\ \boxed{} & \text{se } j > i \end{cases}$$

$$m(i,j) = \begin{cases} \boxed{} & \text{se } j = i \\ \boxed{} & \text{se } j > i \end{cases}$$

2. Complete o template de código em baixo que, dada uma sequência de inteiros $\langle x_1, ..., x_n \rangle$, calcula $m[1, n]$ e $M[1, n]$.

```
GreatestValue(x[1..n])
  let M[1..n, 1..n] be a new matrix of size n × n
  let m[1..n, 1..n] be a new matrix of size n × n
  for i = 1 to n do
    M[i, i] := [          ]
    m[i, i] := [          ]
  endfor
  for s = 1 to n−1 do
    for i = 1 to n−s do
      [                   ]
      [                   ]
      [                   ]
    endfor
  endfor
  return M[1, n]
```

3. Determine a complexidade assimptótica do algoritmo proposto na alínea anterior.

Solução:

1.
$$M(i,j) = \begin{cases} x[i] & \text{se } i = j \\ \textbf{max}\{M[i,k]/m[k+1,j] \mid i \leq k < j\} & \text{se } j > i \end{cases}$$
$$m(i,j) = \begin{cases} x[i] & \text{se } j = i \\ \textbf{min}\{m[i,k]/M[k+1,j] \mid i \leq k < j\} & \text{se } j > i \end{cases}$$

2.
```
GreatestValue(x[1..n])
  let M[1..n, 1..n] be a new matrix of size n × n
  let m[1..n, 1..n] be a new matrix of size n × n
  for i = 1 to n do
    M[i, i] := x[i]
    m[i, i] := x[i]
  endfor
  for s = 1 to n−1 do
    for i = 1 to n−s do
      let j = i + s
      let M[i, j] = −∞
      let m[i, j] = +∞
      for k = i to j−1 do
        M[i, j] := max(M[i, j], M[i, k]/m[k + 1, j])
        m[i, j] := min(m[i, j], m[i, k]/M[k + 1, j])
      endfor
    endfor
  endfor
  return M[1, n]
```

3. Complexidade: $O(n^3)$. O algoritmo tem de preencher a metade diagonal superior das matrizes $M[1..n, 1..n]$ e $m[1..n, 1..n]$, sendo que para cada posição da matriz o algoritmo pode percorrer s posições. Formalmente:

$$\begin{aligned} &\textstyle\sum_{s=1}^{n-1} \sum_{i=1}^{n-s} \sum_{k=i}^{j-1} O(1) \\ &\quad = \textstyle\sum_{s=1}^{n-1} \sum_{i=1}^{n-s} \sum_{k=i}^{i+s-1} O(1) \\ &\quad = \textstyle\sum_{s=1}^{n-1} \sum_{i=1}^{n-s} O(1).(i+s-1-i+1) \\ &\quad = \textstyle\sum_{s=1}^{n-1} \sum_{i=1}^{n-s} O(s) \\ &\quad = \textstyle\sum_{s=1}^{n-1} O(s).(n-s) \\ &\quad = O(\textstyle\sum_{s=1}^{n-1} n.s - s^3) \\ &\quad \leq O(n. \textstyle\sum_{s=1}^{n-1} s) \\ &\quad \leq O(n^3) \end{aligned}$$

II.b) O Departamento de Informática da Universidade Técnica de Caracolândia decidiu implementar uma nova aplicação para determinar automaticamente a composição dos júris de dissertações de mestrado. No semestre em consideração existem n estudantes que pretendem defender as suas dissertações, $\{S_1, ..., S_n\}$, e m professores disponíveis para participar em júris, $\{P_1, ..., P_m\}$. O problema da constituição dos júris deve respeitar as seguintes restrições:

- Cada professor P_j, com $1 \leq j \leq m$, pode participar em no máximo l_j júris;
- Cada júri deve ser constituído por três professores.

Admita que o problema tem solução, isto é, que, dadas as disponibilidades dos professores, é possível constituir o júri de todos os alunos. Pretende-se agora calcular uma atribuição de professores a júris.

1. Modele o problema da constituição de júris como um problema de fluxo máximo. A resposta deve incluir o procedimento utilizado para determinar a constituição dos júris a partir do fluxo calculado.

2. Indique o algoritmo que utilizaria para a calcular o fluxo máximo, bem como a respectiva complexidade assimptótica medida em função dos parâmetros do problema (número de alunos, n, e número de professores, m).
   Nota: De entre os algoritmos de fluxo estudados nas aulas deve escolher aquele que garanta a complexidade assimptótica mais baixa para o problema em questão.

Solução:

1. *Construção da rede de fluxo* $G = (V, E, w, s, t)$. Na construção da rede de fluxo consideramos um vértice por professor, um vértice por estudantes e dois vértices adicionais s e t, respectivamente a fonte e o sumidouro. Formalmente:

   - $V = \{S_i \mid 1 \leq i \leq n\} \cup \{P_j \mid 1 \leq i \leq m\} \cup \{s, t\}$
   -

   $$\begin{array}{ll} E = \{(s, P_j, l_j) \mid 1 \leq j \leq m\} & P_j \text{ só pode participar em } l_j \text{ defesas} \\ \quad \cup \{(P_j, S_i, 1) \mid j \in \texttt{SP}(i)\} & P_j \text{ pode participar no júri do estudante } S_i \\ \quad \cup \{(S_i, t, 3) \mid 1 \leq i \leq n\} & 3 \text{ membros por júri} \end{array}$$

   Como sabemos que é possível formar todos os jurís, concluímos que o fluxo máximo é $3.n$. Uma vez calculado o fluxo máximo, f^*, o júri do aluno S_i é constituído pelos professores P_{j_1}, P_{j_2} e P_{j_3} tais que: $f^*(P_{j_1}, S_i) = 1$, $f^*(P_{j_2}, S_i) = 1$, e $f^*(P_{j_3}, S_i) = 1$.

2. *Complexidade.*

   - $|V| = n + m + 2 \in O(n + m)$
   - $|E| = m + m.n + n \in O(n.m)$
   - $|f^*| \leq 3n \in O(n)$
   - Ford-Fulkerson: $O(n^2.m)$
   - RF: $O((n + m)^3$

   A melhor solução consiste em usar um algoritmo baseado no método de Ford Fulkerson.

II.c) Uma matriz de incompatibilidades é uma matriz quadrada cujas células guardam valores decimais entre 0 e 1. Intuitivamente, dada uma matriz de incompatibilidades M, $n \times n$, a célula M_{ij} guarda a incompatibilidade entre os índices i e j; $M_{ij} = 0$ se i e j são completamente compatíveis e $M_{ij} = 1$ se i e j são completamente incompatíveis. Dado um sub-conjunto de índices $I \subseteq \{1, ..., n\}$, o nível de incompatibilidade do conjunto é dado por: $\sum_{i,j \in I} M_{ij}$. O problema das incompatibilidades define-se formalmente da seguinte maneira:

$$\textbf{Incompat} = \{\langle M, k, v\rangle \mid M \text{ contém um sub-conjunto de índices de tamanho } k \text{ e incompatibilidade igual ou inferior a } v\}$$

1. Mostre que o problema **Incompat** está em **NP**.

2. Mostre que o problema **Incompat** é NP-difícil por redução a partir do problema **ISet**, que é sabido tratar-se de um problema NP-completo e que se define em baixo. Não é necessário provar formalmente a equivalência entre os dois problemas; é suficiente indicar a redução e a respectiva complexidade.
   Pista: Dado um grafo G indique como construir uma matriz de incompatibilidades cujos índices correspondem aos vértices de G tendo em conta o problema **ISet**.

Problema ISet: Seja $G = (V, E)$ um grafo não dirigido; dizemos que $V' \subseteq V$ é um conjunto de vértices independentes em G se e apenas se $\forall u, v \in V'.(u, v) \notin E$. O problema **ISet** define-se formalmente da seguinte maneira:

$$\textbf{ISet} = \{\langle G, k\rangle \mid G \text{ contém um conjunto de vértices independentes de tamanho } k\}$$

Solução:

1. O algoritmo de verificação recebe como input uma possível instância $\langle M, k, v\rangle$ e um conjunto de índices I (o certificado). O algoritmo tem de verificar que $|I| = k$ e que $\sum_{i,j \in I} M_{ij} \leq v$. Observamos os certificados têm tamanho $O(n)$ e que a verificação se faz em tempo $O(n^2)$, o tempo de calcular o somatório.

2. Dada uma possível instância $\langle G, k\rangle$ do problema **ISet**, começamos por construir uma matriz de incompatibilidades M_G cujos índices correspondem aos vértices de G. Para tal, admitimos que $|V| = n$ e que os vértices de V estão numerados de 1 a n, sendo v_i o i-ésimo vértice. Assim sendo, definimos a matriz M_G como se segue:
   $$(M_G)_{ij} = \begin{cases} 1 & \text{se } (v_i, v_j) \in E \\ 0 & \text{caso contrário} \end{cases}$$
   Uma vez estabelecida a matriz M_G, a redução é definida da seguinte maneira:
   $$f(\langle G, k\rangle) = \langle M_G, k, 0\rangle$$
   - Equivalência a estabelecer: $\langle G, k\rangle \in \textbf{ISet} \iff \langle M_G, k, 0\rangle \in \textbf{Incompat}$
   - Complexidade da redução: $O(|V|^2)$.