Instituto Superior Técnico

Análise e Síntese de Algoritmos

Ano Lectivo 2020/2021 — 2º Teste - versão A

RESOLUÇÃO

I. (2.5 + 2.5 + 2.5 + 2.5 = 10 val.)

I.a) Considere o padrão $P = baabab$ e construa o autómato finito que emparelhe este padrão numa cadeia de caracteres. Indique o estado resultante das seguintes transições:

$\delta(0,a)$	$\delta(1,b)$	$\delta(2,b)$	$\delta(3,a)$	$\delta(4,b)$	$\delta(5,a)$	$\delta(6,a)$	$\delta(6,b)$
0	**1**	**1**	**0**	**1**	**3**	**2**	**1**

Ilustre a sua aplicação no seguinte texto $T = baabbaabab$.

q	0	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**5**	**6**

I.b) Considere o problema de multiplicar cadeias de matrizes. O objetivo é determinar por que ordem devem ser feitas as multiplicações por forma a minimizar o número total de multiplicações escalares que precisam de ser efetuadas.

Considere uma sequência com 4 matrizes $A(5 \times 2)$, $B(2 \times 4)$, $C(4 \times 3)$, $D(3 \times 5)$, com as respetivas dimensões entre parênteses. Resolva este problema preenchendo a matriz `m[i,j]` que guarda o menor número de multiplicações escalares que precisam de ser efectuadas para obter o produto das matrizes $A \ldots D$.

Indique os valores de `m[1,2]`, `m[2,4]` e `m[1,4]`, sabendo que `m[1,3]=54` para $k = 1$. Indique também a colocação de parênteses que obtém o valor indicado em `m[1,4]`. Em caso de empate associe à esquerda.

`m[1,2]`	`m[2,4]`	`m[1,4]`	Parênteses
40	**54**	**104**	$\mathbf{A} \times ((\mathbf{B} \times \mathbf{C}) \times \mathbf{D})$

I.c) Considere o problema de compressão de dados de um ficheiro usando a codificação de Huffman. Indique o código livre de prefixo óptimo para cada carácter num ficheiro com 1 000 caracteres com a seguinte frequência de ocorrências:
$f(a) = 15, f(b) = 4, f(c) = 13, f(d) = 12, f(e) = 48, f(f) = 8$.

Quando constrói a árvore, atribue o bit 1 para o nó com menor frequência. Em caso de empate, atribua o bit 1 ao nó que inclui o caracter que aparece primeiro por ordem alfabética. Analogamente, em caso de empate na *min-priority queue*, considera-se primeiro o nó que inclui o caracter que aparece primeiro por ordem alfabética.

Indique também o total de bits no ficheiro codificado.

	a	b	c	d	e	f
Codificação	**000**	**0111**	**001**	**010**	**1**	**0110**

Total Bits	**2160**

I.d) Considere o seguinte programa linear:

$$\begin{array}{rrcl} \text{Min} & -2x_1 - x_2 + x_3 + & & 4 \\ \text{s.a} & x_1 + 3x_2 + 2x_3 & \leq & 4 \\ & -2x_1 + x_2 + x_3 & \leq & 8 \\ & -x_1 - 2x_2 - x_3 & \geq & -5 \\ & x_1, x_2, x_3 & \geq & 0 \end{array}$$

Indique o valor da função objectivo e o respectivo valor das variáveis básicas e não-básicas na solução óptima. Em caso de empate na escolha da variável de entrada ou da variável de saída, aplique a regra de Bland (ou seja, escolha a variável de menor índice).

Z	x_1	x_2	x_3	x_4	x_5	x_6
−4	**4**	**0**	**0**	**0**	**16**	**1**

II. (3 + 3 + 2.5 + 1.5 = 10 val.)

II.a) Uma sequência diz-se um *palíndromo* se é simétrica, isto é, se permanece igual quando lida de trás para diante; por exemplo, são palíndromos as sequências: a, aa, $abbba$ e $abbaabba$. Pretende-se desenvolver um algoritmo que, dada uma sequência de caracteres arbitrária, retorne o tamanho do maior palíndromo que esta contém. Por exemplo, dada a sequência $abbaabbabaabc$, o algoritmo deve retornar 8, que corresponde ao tamanho do palíndromo $abbaabba$.

1. Seja $x[1..n]$ a string de texto dada como input e $B(i,j)$ o valor Booleano que indica se a cadeia de caracteres $x[i..j]$ forma um palíndromo. Defina $B(i,j)$ recursivamente completando os campos em baixo:

$$B(i,j) = \begin{cases} \textbf{true} & \text{se } j < i \\ \boxed{} & \text{se } j = i \\ \boxed{} & \text{c.c.} \end{cases}$$

   Admite-se, para simplificar a formulação, que $B(i,j) = \textbf{true}$ quando $j < i$.

2. Complete o template de código em baixo que calcula o tamanho do maior palíndromo contido no array dado como input, $x[1..n]$. Para obter a cotação máxima, o algoritmo deve retornar o valor pretendido assim que encontra o palíndromo de tamanho máximo, não devendo de efectuar o preenchimento completo da matriz $B[1..n, 1..n]$.

   `BiggestPalindromeSize`$(x[1..n])$

   let $B[1..n, 1..n]$ be a new matrix of size $n \times n$ with all cells initialised to true

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

1. $$B(i,j) = \begin{cases} \textbf{true} & \text{se } j \leq i \\ B(i+1, j-1) \ \wedge \ (x[i] == x[j]) & \text{c.c.} \end{cases}$$

2.

```
BiggestPalindromeSize(x[1..n])
  let B[1..n, 1..n] be a new matrix of size n × n with all cells initialised to true
  let not_found = 0
  for s = 1 to n−1 do
    let found = false
    for i = 1 to n−s do
      B[i, i+s] := B[i + 1, (i+s)−1] && (x[i] == x[i+s])
      found := found || B[i, i+s]
    endfor
    if(not found){
      not_found := not_found+1
      if(not_found == 2) return s
    } else not_found := 0
  endfor
  if(not_found == 1) return n−1
  else return n
```

3. Complexidade: $O(n^2)$. No pior caso o algoritmo terá de preencher a metade diagonal superior da matriz B.

II.b) Considere o seguinte programa linear:

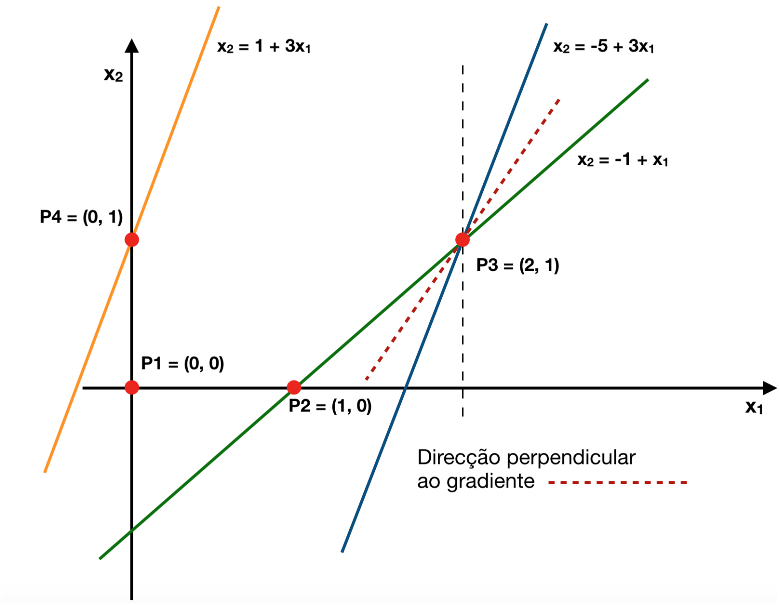
$$\begin{array}{rrrcl} \max & 2x_1 & -x_2 & & \\ \text{s.a} & -3x_1 & +x_2 & \leq & 1 \\ & x_1 & -x_2 & \leq & 1 \\ & 3x_1 & -x_2 & \leq & 5 \\ & & x_1, x_2 & \geq & 0 \end{array}$$

1. Desenhe o conjunto exequível e resolva geometricamente o programa linear. A resposta deve incluir: o valor máximo, as coordenadas onde esse valor é atingido e as equações das rectas que delimitam a região exequível.

2. Formule o programa linear dual e calcule a respectiva solução a partir da solução do programa primal. Indique tanto o valor mínimo como as coordenadas onde esse valor é atingido.

Solução:

1. Representamos a região exequível no diagrama em baixo.



O Teorema Fundamental da Programação Linear estabelece que o valor óptimo da função objectivo, a existir, ocorre num vértice da região exequível. O vector gradiente da função objectivo é: $(2, -1)$. Representamos a tracejado vermelho a recta perpendicular ao gradiente (declive 2). Observamos que no vértice P_3 não existem direcções de subida exequíveis, pelo que concluímos que o valor óptimo é 3 e ocorre no ponto $P_3 = (2, 1)$.

2. O programa linear dual é definido em baixo:

$$\begin{array}{lrrrcr} \min & 1y_1 & +1y_2 & +5y_3 & & \\ \text{s.a} & -3y_1 & +y_2 & +3y_3 & \geq & 2 \\ & y_1 & -y_2 & -y_3 & \geq & -1 \\ & & y_1, y_2, y_3 & & \geq & 0 \end{array}$$

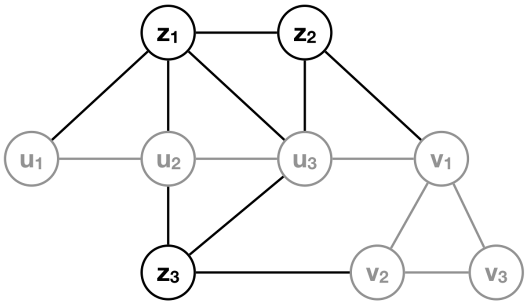
Do Teorema da Dualidade Forte concluímos que o valor mínimo do programa dual coincide com o valor máximo do programa primal, 3. Da inspecção da geometria do programa primal, concluímos que as restrições activas no vértice da solução correspondem às variáveis y_2 e y_3 do problema dual. Segue, por isso, que $y_1 = 0$ no ponto óptimo do problema dual. Resolvendo o sistema:

$$\begin{cases} y_2 + 3y_3 = 2 \\ -y_2 - y_3 = -1 \end{cases}$$

concluímos que o valor mínimo do programa dual se encontra no ponto $(0, 1/2, 1/2)$.

II.c) Um grafo diz-se um *kite*[1] de grau n se é constituído por $2.n$ vértices, tais que n vértices formam um clique e os restantes n vértices formam uma cauda ligada a um dos vértices do clique.

Dado um grafo $G = (V, E)$ e um inteiro k, o problema **Kite** consiste em determinar se G contém um *kite* de grau k. Por exemplo, o grafo em baixo contém vários kites de grau 3, um dos quais está identificado em cinzento.



Formalmente, o problema **Kite** pode ser modelado através do seguinte problema de decisão:

$$\textbf{Kite} = \{\langle G, k\rangle \mid G \text{ contém um } \textit{kite} \text{ de grau } k\}$$

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

2. Mostre que o problema **Kite** é NP-difícil por redução a partir do problema **Clique** estudado nas aulas. Não é necessário provar formalmente a equivalência entre os dois problemas; é suficiente indicar a redução e a respectiva complexidade.

Solução:

1. O algoritmo de verificação recebe como input uma possível instância $\langle G = (V, E), k\rangle$ e um certificado na forma de um triplo $\langle V_1, V_2, u, v\rangle$ tal que:

   - *Restrição 1:* $V_1 \cap V_2 = \emptyset$, $|V_1| = |V_2| = k$;
   - *Restrição 2:* $u \in V_1$, $v \in V_2$, $(u, v) \in E$;
   - *Restrição 3:* $G_1 = (V_1, E_1)$, com $E_1 = \{(w, z) \mid (w, z) \in E \ \wedge \ w, z \in V_1\}$, forma uma linha com k elementos;
   - *Restrição 4:* $G_2 = (V_2, E_2)$, com $E_2 = \{(w, z) \mid (w, z) \in E \ \wedge \ w, z \in V_2\}$, forma um clique de tamanho k.

   Em primeiro lugar, observamos que o certificado tem tamanho $O(V)$. O algoritmo de verificação tem de verificar que as restrições enunciadas em cima são verificadas. Analisamos cada restrição separadamente:

   - *Restrição 1:* $O(V)$;
   - *Restrição 2:* $O(1)$;
   - *Restrição 3:* $O(V)$ (encontrar o vértice sem nós incidentes em V_1 e efectuar uma DFS modificada que não explora arcos em V_2);
   - *Restrição 4:* $O(V^2)$ (verificar se cada vértice em V_2 está ligado a todos os outros vértices em V_2).

[1] Em português *kite* diz-se papagaio.

2. Dada uma instância $\langle G = (V, E), k\rangle$ do problema **Clique** temos de construir uma instância $\langle G' = (V', E'), k\rangle$ do problema **Kite** tal que:

$$\langle G, k\rangle \in \textbf{Clique} \;\Leftrightarrow \langle G', k\rangle \in \textbf{Kite}$$

Intuitivamente definimos o grafo G' acrescentando a cada vértice $v \in V$ uma cauda com k vértices. Admitindo que os vértices de G se encontram numerados: $V = \{v_1, ..., v_n\}$, definimos formalmente o grafo $G' = (V', E')$ como se segue:

- $V' = V \cup \{u_1^1, ..., u_1^k\} \cup \{u_2^1, ..., u_2^k\} \cup ... \cup \{u_n^1, ..., u_n^k\}$
- $E' = E \cup \{(u_i^j, u_i^{j+1}) \mid 1 \leq i \leq n \;\wedge\; 1 \leq j \leq k{-}1\} \cup \{(u_i^k, v_i) \mid 1 \leq i \leq n\}$

Complexidade da redução: $O(V^2)$

II.d) Um padrão P de tamanho n diz-se *livre de sobreposição* se satisfaz a seguinte implicação:

$$\forall_{0 \leq k,j \leq n}\ P_k \sqsupset P_j \implies k = 0 \ \vee \ k = j$$

Recorde que P_i denota o prefixo de P de tamanho i e que a notação $P_k \sqsupset P_j$ se usa para denotar que P_k é sufixo de P_j.

1. Indique todos os padrões livres de sobreposição de tamanho 3 sobre o alfabeto $\Sigma = \{a, b\}$.

2. Indique o número de padrões livres de sobreposição de tamanho n sobre o alfabeto $\Sigma = \{a, b\}$.

3. Descreva a função de prefixo de um padrão livre de sobreposição.

Solução:

1. Padrões: *abb* e *baa*.

2. Resposta: 2

3. Recorde a definição da função de prefixo:

$$\pi[j] = max\{i \mid i < j \ \wedge \ P_i \sqsupset P_j\}$$

   Como P é livre de sobreposição, concluímos que $\pi[j] = 0$ para todo o $1 \leq j \leq n$, onde $n = |P|$.