INSTITUTO SUPERIOR TÉCNICO

Análise e Síntese de Algoritmos

Ano lectivo de 2018/2019 2º Teste - versão A

RESOLUÇÃO DO 2º TESTE

I. (2,0 + 4,0 + 2,0 = 8,0 val.)

I.a)

i	1	2	3	4	5	6	7	8	9	10	11
S/E	S	E	S	S	S	E	E	S	S	E	E

I.b)

Primeira	Z =-15	x_1 =3	x_2 =0	x_3 =0
Ótima	Z =-5	x_1 =13	x_2 =10	x_3 =0
Dual Ótima	Z' =-5		$y_1 = 2$	$y_2 = 1$

I.c)

	a	b	c	d	e	f
Codificação	1	00	010	0111	01100	01101

Total Bits	148

II. (2,0 + 2,0 = 4,0 val.)

II.a)

`c[2,9]`	`c[3,5]`	`c[3,6]`	`c[3,10]`	`c[4,8]`	`c[4,9]`
5	1	2	2	2	3

II.b) ⟨ XXX ⟩

III. (2,0 + 2,0 = 4,0 val.)

III.a)

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}
	4	7	6	1	1	9	6	9	7	4	7	4	9	5
$T =$	5 6 5	6	7	2	7	3	2	8	2	9	3	9	4	0

III.b)

i	1	2	3	4	5	6	7	8	9	10	11
π	0	0	1	2	0	1	2	3	4	3	4
$P =$	A	B	A	B	C	A	B	A	B	A	B

IV. (2,0 + 2,0 = 4,0 val.)

IV.a)

	a)	b)	c)	d)	e)
Resposta	V	V	V	D	F

IV.b) ⟨ XXX ⟩

I. (2,0 + 4,0 + 2,0 = 8,0 val.)

I.a) Nesta questão pretendemos utilizar o algoritmo greedy que maximiza o número de atividade compatíveis que podem ser selecionadas. Indique, para cada atividade, *S* se a respetiva atividade for selecionada ou *E* se for excluída.

A tabela seguinte descreve um sequência de atividades, em que o valor a_i representa o tempo de inicio e d_i a duração da atividade i.

i	1	2	3	4	5	6	7	8	9	10	11
a_i	5	11	11	12	15	17	24	25	26	27	28
d_i	5	4	1	1	3	4	3	1	3	6	3

I.b) Considere o seguinte programa linear:

$$\begin{array}{lrcl} \text{Maximizar} & -5x_1+6x_2+4x_3 & & \\ \text{Sujeito a} & -\ x_1+\ x_2+\ x_3 & \leq & -3 \\ & -3x_1+4x_2+4x_3 & \leq & 1 \\ & x_1,x_2,x_3 & \geq & 0 \end{array}$$

Indique o valor da função objectivo e o respectivo valor das variáveis x_1, x_2 e x_3 na **primeira solução exequível** encontrada pelo algoritmo Simplex. Em caso de empate em algum critério de aplicação do algoritmo, aplique a regra de Bland. Ou seja, escolha a variável de menor índice.

Indique também o valor da função objectivo e o respectivo valor das variáveis para a solução ótima e para a solução do sistema dual, com a variável y_1 associada à primeira restrição e a variável y_2 associada à segunda restrição.

I.c) Considere o problema de compressão de dados de um ficheiro usando a codificação de Huffman. Indique o código livre de prefixo ótimo para cada caractere num ficheiro com 100 caracteres com o seguinte número de ocorrências: $f(a) = 75, f(b) = 12, f(c) = 6, f(d) = 4, f(e) = 1, f(f) = 2$. Quando constrói a árvore, considere o bit 0 para o nó com menor frequência.

Indique também o total de bits no ficheiro codificado.

II. (2,0 + 2,0 = 4,0 val.)

II.a) Considere o problema de realizar trocos minimizando o número de moedas necessário. Assuma que os valores v_i das moedas são $1 < 2 < 5 < 6$ e que calcula uma tabela de programação dinâmica `c[i,j]`, onde `i` indica que apenas podem ser utilizadas moedas com os primeiros i valores e `j` indica o valor que se pretende trocar. O valor guardado na tabela é o menor número de moedas necessárias para obter os valor `j`, assumindo que existe uma quantidade ilimitada de moedas de cada valor.

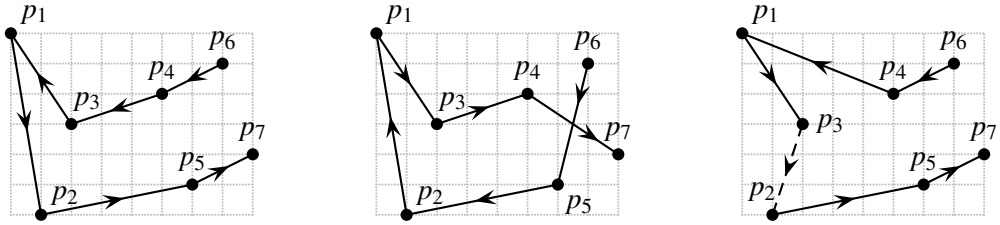
Indique os valores `c[2,9]`, `c[3,5]`, `c[3,6]`, `c[3,10]`, `c[4,8]` e `c[4,9]`.

II.b) Nesta pergunta iremos calcular um caminho bitónico mais curto, utilizando a distância Euclidiana, i.e, $d(p_1,p_2)=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$ para os pontos $p_1=(x_1,y_1)$ e $p_2=(x_2,y_2)$. Vamos assumir que a sequência de pontos $p_1,\ldots,p_n$ está ordenada de forma crescente na coordenada x, i.e., $x_1<x_2<\ldots<x_n$ e que as coordenadas x são distintas.

Os caminhos devem passar por todos os n pontos, pelo que, podem ser representados por uma permutação σ, em que o ponto $p_{\sigma(k)}$ liga ao ponto $p_{\sigma(k+1)}$, para todo $k\in\{1,\ldots,n-1\}$. A seguinte tabela exemplifica cinco caminhos:

k	1	2	3	4	5	6	7
$\sigma(k)$	6	4	3	(1)	2	5	7
$\sigma'(k)$	6	5	2	(1)	3	4	7
$\sigma''(k)$	6	4	(1)	3	2	5	7
$\sigma^+(k)$	(1)	2	3	4	5	6	7
$\sigma^-(k)$	7	6	5	4	3	2	(1)

Os caminhos σ, σ' e σ'' estão representados gráficamente na figura seguinte.



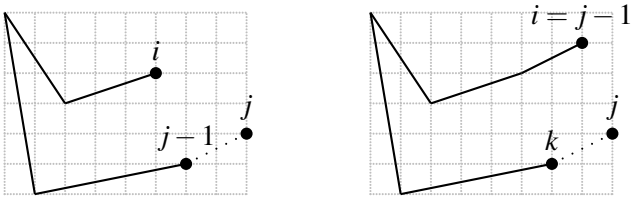
Um caminho é bitónico se a respetiva permutação é descrescente até atingir o valor 1 e a partir desse valor é crescente até ao fim. No nosso exemplo apenas a permutação σ'' não corresponde a um caminho bitónico, porque $\sigma''(4)=3>2=\sigma''(5)$ e $\sigma''(3)=1$. As restantes permutações correspondem a caminhos bitónicos.

Pretendemos determinar, utilizando programação dinâmica, o caminho bitónico mais curto que passa por todos os pontos. No nosso exemplo o caminho que corresponde à permutação σ é mais curto que o caminho que corresponde à permutação σ'. Note que tanto a distância Euclidiana como a definição de caminho bitónico são simétricas, pelo que, por exemplo, as permutações σ^+ e σ^- correspondem essencialmente ao mesmo caminho. Para evitar esta duplicação de representações vamos convencionar que o primeiro ponto do caminho, p_i, com $i=\sigma(1)$, deve ter um índice menor que o último ponto, p_j, com $j=\sigma(n)$, i.e., $i<j$. No caso particular de σ temos $i=6<7=j$. Das permutações acima que correspondem a caminhos bitónicos apenas σ^- não respeita esta propriedade, pelo que esse caminho deveria ser representado por σ^+

Definimos uma tabela `B[i,j]` em que $i<j$ e que guarda o comprimento do menor caminho bitónico que começa no ponto p_i e termina no ponto p_j e inclui todos os pontos de p_1 a p_j. Complete a fórmula da recursão para a resolução deste problema.

$$\texttt{B[i,j]} = \begin{cases} \boxed{} & \text{, se } i=1 \text{ e } j=2 \\ \boxed{} & \text{, se } i<j-1 \\ \min_{1\le k<j-1}\left\{\boxed{}\right\} & \text{, se } i=j-1 \end{cases}$$

As figuras abaixo ilustram os dois últimos casos a considerar.



Indique a complexidade temporal do algoritmo resultante. Caso não tenha preenchido a recursão assuma que as operações em falta requerem tempo $O(1)$.

Solução:

$$\texttt{B[i,j]} = \begin{cases} d(p_1, p_2) & \text{, se } i = 1 \text{ e } j = 2 \\ B[i, j-1] + d(p_{j-1}, p_j) & \text{, se } i < j-1 \\ \min_{1 \le k < j-1} \{ B[k, i] + d(p_k, p_j) \} & \text{, se } i = j-1 \end{cases}$$

A complexidade do algoritmo é $O(n^2)$, porque há este numero de entradas com $i < j-1$, para as quais a computação demora $O(1)$ para cada. Existem $O(n)$ entradas com $i = j-1$, para as quais a computação demora $O(n)$ para cada. Em ambos os casos a computação total é $O(n^2)$, pelo que esse é o tempo total.

III. (2,0 + 2,0 = 4,0 val.)

III.a) Considere o algoritmo de Rabin-Karp, com $P = 146$, $T = 5656727328293940$ e $q = 11$. Indique a sequência de valores t_i obtidos a processar T. Note que este números são um resto modulo q, pelo que são valores entre 0 e 10.

Recorde que é possível calcular o resto de uma divisão por 11 somando e subtraíndo os algorismos do número, por exemplo temos $146 \equiv 1 - 4 + 6 \equiv 3 (\mod 11)$. O que é uma consequência das seguintes relações:

$$\begin{aligned} 1 &\equiv 1 (\mod 11) \\ 10 &\equiv -1 (\mod 11) \\ 100 &\equiv 1 (\mod 11) \end{aligned}$$

III.b) Calcule a função de prefixo do algoritmo de Knuth-Morris-Pratt para o padrão $P = ABABCABABAB$.

IV. (2,0 + 2,0 = 4,0 val.)

IV.a) O Professor Carlos é um investigador perspicaz, que estuda o problema 3CNF-SAT. Contudo até hoje não foi capaz de encontrar um algoritmo polinomial para resolver este problema. Considerando que o Professor Carlos é bastante competente classifique as seguintes afirmações como verdadeira (**V**), falsa (**F**) ou se não se sabe (**D**).

a. VERTEX-COVER $\in$ NP
b. 2-COLOR $\in$ NP
c. P $\subseteq$ NP
d. HORN-SAT $\notin$ NP-HARD
e. HORN-SAT $\notin$ P

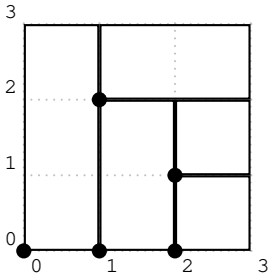
IV.b) Nesta questão vamos considerar o problema da pavimentação com retângulos (R-TILING). Uma instância do problema consiste num saco S de retângulos e um retângulo p que representa o pavimento desejado. Recorde que um saco, ou *multiset*, pode conter repetições.

Considere por exemplo a instância com $S = [(1,3),(2,2),(1,2),(1,2),(2,1),(1,1),(1,1)]$ e $p = (3,3)$, onde cada retângulo é representado por um par de valores que correspondem à dimensão na coordenada x e à dimensão na coordenada y.

O problema em questão consiste em determinar se é possivel pavimentar p utilizando, alguns, dos retângulos de S. Por pavimentar entende-se cobrir toda a área de p sem que haja sobreposição de retângulos. Os retângulos **não** podem ser rodados, apenas podem ser posicionados.

Assuma que a representação de uma solução consiste em associar a cada retângulo de S as coordenadas do canto inferior esquerdo, ou a letra O, para omitido.

Para o exemplo dado a resposta é afirmativa. Uma possível pavimentação e respetiva representação são as seguintes:



(1,3) : (0,0)
(2,2) : O
(1,2) : (1,0)
(1,2) : O
(2,1) : (1,2)
(1,1) : (2,0)
(1,1) : (2,1)

Dado um conjunto de números inteiros S e um número inteiro t, o problema **SUBSET-SUM** consiste em verificar se existe um subconjunto S' tal que $S' \subseteq S$ e a soma dos elementos de S' seja t. Sabendo que o problema **SUBSET-SUM** é NP-Completo, prove que o problema **R-TILING** é NP-Completo. Prove primeiro que **R-TILING** $\in$ NP.

Solução:

Em primeiro lugar é necessário provar que **R-TILING** $\in$ NP. Primeiro calculamos a área de p, representada por k. Seguidamente calculamos a soma t das áreas de todos os retângulos que não estão indicados como omissos na solução. Verificamos se $t = k$, caso contrário o algoritmo de verificação rejeita a solução.

Caso a solução passe neste primeiro teste é preciso verificar que não existe sobreposição de retângulos. Para tal consideram-se todos os pares de retângulos não omissos na solução. Por exemplo vamos considerar o retângulo $r_i = (d_x, d_y)$ com canto inferior esquerdo em (x_i, y_i) e o retângulo $r_j = (d'_x, d'_y)$ com canto inferior esquerdo em (x_j, y_j). Há sobreposição de retângulos caso ambas as seguintes condições se verifiquem:

$$\begin{aligned} \max(x_i, x_j) &< \min(x_i + d_x, x_j + d'_x) \\ \max(y_i, y_j) &< \min(y_i + d_y, y_j + d'_y) \end{aligned}$$

Caso algum par de retângulos indique sobreposição o algoritmo verificação retorna falso, caso contrário a solução é aceite.

Caso o número de retângulos em S seja n o tempo que o algoritmo de verificação necessita é $O(n^2)$, pelo que é polinomial no tamanho do input.

Em segundo lugar vamos provar que **R-TILING** $\in$ NP-HARD, fazendo uma redução a partir do **SUBSET-SUM**. Dado um conjunto C de números e um valor objetivo k construimos o pavimento $p = (1,k)$ e o conjunto de retângulos com dimensões $(1,c)$ para cada numero $c \in C$.

Para verificarmos que esta redução está correcta temos que provar que a instância do **SUBSET-SUM** tem solução se e só se a instância do **R-TILING** gerada tem solução.

Se a instância do **SUBSET-SUM** tem solução $C' \subseteq C$ então selecionamos os retângulos $(1,c')$ com $c' \in C'$. As coordenada do vértice inferior esquerdo serão $(0,a)$ onde a variável a é a soma dos valores de C' já processados, i.e., após atribuir um retângulo fazemos $a = a + c'$. No início temos $a = 0$.

Queremos verificar que esta configuração é uma pavimentação de p. Note que a área de p é k e vamos utilizar t para representar a soma das áreas dos retângulos da pavimentação, que por construção é igual à soma dos elementos de C'. Para confirmar que é uma pavimentação de p basta notar que por construção os retângulos não tem sobreposição e que $k = t$ porque C' era solução do **SUBSET-SUM** e portanto os retângulos selecionados cobrem toda a área de p.

No outro sentido queremos verificar que se a instância gerada do **R-TILING** tem solução então a instância do **SUBSET-SUM** original também tem solução. Escolhemos para C' os numeros c' tais que os respetivos retângulos $(1, c')$ não estão omissos na pavimentação de p. Novamente temos que a área de p é k e iremos utilizar t para representar a soma da área dos retângulos da pavimentação, que devido á forma do retângulos é igual á soma dos números em C'. Como não há sobreposição de retângulos temos $t \leq k$ e como os retângulos têm de cobrir a área de p temos $t \geq k$, pelo que $t = k$ e C' é solução da instância do **SUBSET-SUM**.