Instituto Superior Técnico

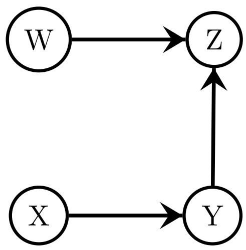
Análise e Síntese de Algoritmos

Ano Lectivo 2021/2022 2º Exame

RESOLUÇÃO

Grupo I: 2 + 2 + 2 + 3 = 9 val.

I.A) Considere o grafo dirigido acíclico (DAG) que se apresenta em baixo:



A sequência de vértices $\langle W, X, Y, Z\rangle$ é uma ordenação topológica do grafo. Indique os tempos de descoberta e de fim para três DFSs distintas que induzem a referida ordenação topológica.

DFS1:

	W	X	Y	Z
$d[i]$	**7**	**5**	**1**	**2**
$f[i]$	**8**	**6**	**4**	**3**

DFS2:

	W	X	Y	Z
$d[i]$	**7**	**1**	**2**	**3**
$f[i]$	**8**	**6**	**5**	**4**

DFS3:

	W	X	Y	Z
$d[i]$	**7**	**5**	**3**	**1**
$f[i]$	**8**	**6**	**4**	**2**

DFS4:

	W	X	Y	Z
$d[i]$	**7**	**3**	**4**	**1**
$f[i]$	**8**	**6**	**5**	**2**

I.B) 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 200 caracteres com a seguinte frequência de ocorrências (dada em percentagem):

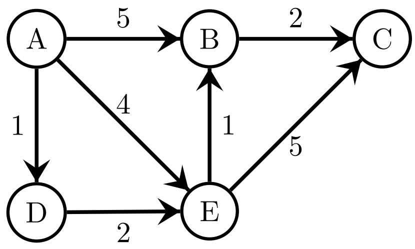
$$f(a) = 51, f(b) = 7, f(c) = 8, f(d) = 10, f(e) = 24$$

Quando constrói a árvore, atribua o bit 0 para o nó com menor frequência. Indique também o total de bits no ficheiro codificado.

	a	b	c	d	e
Codificação	**1**	**0110**	**0111**	**010**	**00**

Total Bits	**378**

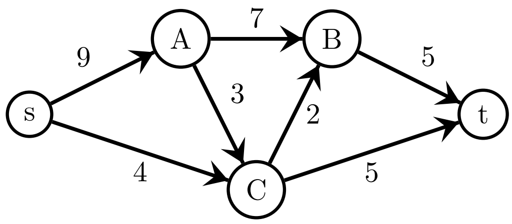
I.C) Considere o grafo da figura:



Indique os valores de d e π para cada vértice quando faltam extrair dois nós da fila de prioridade na execução do algoritmo de Dijkstra a partir do vértice A.

	A	B	C	D	E
$d[i]$	**0**	**4**	**8**	**1**	**3**
$\pi[i]$	**Nil**	**E**	**E**	**A**	**D**

I.D) 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:

- $N[A] = \langle B, C, s \rangle$
- $N[B] = \langle A, C, t \rangle$
- $N[C] = \langle s, B, A, t \rangle$

e que a lista de vértices inicial é $L = \langle A, B, C \rangle$. 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**	**6**	**0**

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

Grupo II: 4 + 3.5 + 3.5 = 11 val.

II.A) Recorde o algoritmo para o cálculo da maior subsequência comum estudado nas aulas que, dadas duas sequências $x[1..n]$ e $y[1..m]$, determina o tamanho da sua maior subsequência comum. Pretende-se generalizar o algoritmo estudado nas aulas para três sequências $x[1..n]$, $y[1..m]$ e $z[1..o]$, devendo o novo algoritmo calcular o tamanho da maior subsequência comum entre as três sequências dadas. Por exemplo, dadas as sequências $\langle 2, 1, 3, 1, 2, 4\rangle$, $\langle 1, 3, 5, 1, 4, 2, 6\rangle$, e $\langle 1, 2, 1, 2, 1\rangle$, o algoritmo deverá retornar 3, correspondendo ao tamanho da subsequência $\langle 1, 1, 2\rangle$ comum às três sequências dadas.

1. Seja $\mathbf{LCS}(i, j, k)$ o tamanho da maior subsequência comum entre $x[1..i]$, $y[1..j]$ e $z[1..k]$ (com $0 \leq i \leq n$, $0 \leq j \leq m$ e $0 \leq k \leq o$); defina $\mathbf{LCS}(i, j, k)$ recursivamente completando os campos em baixo:

$$\mathbf{LCS}(i,j,k) = \begin{cases} \boxed{\qquad\qquad\qquad\qquad} & \text{se } i = 0 \lor j = 0 \lor k = 0 \\ \boxed{\qquad\qquad\qquad\qquad} & \text{se } \boxed{\qquad\qquad\qquad\qquad} \\ \boxed{\qquad\qquad\qquad\qquad\qquad\qquad} & \text{c.c.} \end{cases}$$

2. Complete o template de código em baixo que, dadas três sequências $x[1..n]$, $y[1..m]$ e $z[1..o]$, calcula o valor $\mathbf{LCS}(n, m, o)$.

`LCS`$(x[1..n], y[1..m], z[1..o])$

Para obter a cotação total o algoritmo deverá ter complexidade espacial: $O(n.m)$.

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

4. Explique como calcular o comprimento da maior subsquência comum *estritamente crescente* entre duas sequências de inteiros $x[1..n]$ e $y[1..m]$ utilizando o algoritmo desenvolvido na alínea anterior e indique, justificando, a complexidade assimptótica da solução proposta.

Solução:

1.

$$\mathbf{LCS}(i,j,k) = \begin{cases} 0 \quad \text{ se } i = 0 \vee j = 0 \vee k = 0 \\ \mathbf{LCS}(i{-}1, j{-}1, k{-}1) + 1 \quad \text{ se } x[i] = y[j] = z[k] \\ \mathbf{max}(\mathbf{LCS}(i{-}1, j, k), \mathbf{LCS}(i, j{-}1, k), \mathbf{LCS}(i, j, k{-}1)) \quad \text{ c.c.} \end{cases}$$

2.

```
LCS(x[1..n], y[1..m], z[1..o])
  let LC_prev be a (n + 1) × (m + 1) matrix initialized to 0
  let LC_cur be a (n + 1) × (m + 1) matrix initialized to 0
  for k = 1 to o do
    for i = 0 to n do
      for j = 0 to m do
        if(i == 0 ∨ j == 0){
          LC_cur[i, j] := 0
        } else{
          if(x[i] == y[j] && x[i] == z[k]){
            LC_cur[i, j] := LC_prev[i − 1, j − 1] + 1
          } else{
            LC_cur[i, j] := max(LC_cur[i − 1, j], LC_cur[i, j − 1], LC_prev[i, j])
          }
        }
      endfor
    endfor
    tmp := LC_prev; LC_prev := LC_cur; LC_cur := tmp;
  endfor
  return LC_prev[n, m]
```

3. Complexidade: $O(n.m.o)$. O algoritmo tem de preencher $o + 1$ matrizes cada uma com dimensão $(n+1) \times (m+1)$. O preenchimento de cada célula faz-se em tempo $O(1)$.

4. Dadas duas sequências de inteiros $x[1..n]$ e $y[1..m]$, o algoritmo procede da seguinte forma:

   - Ordena a sequência $x[1..n]$, obtendo uma nova sequência $z[1..n]$. Complexidade: $O(n\ logn)$.
   - Remove os elementos duplicados de $z[1..n]$, obtendo uma nova sequência $z'[1..o]$ (com $o < n$). Complexidade: $O(n)$.
   - Aplica o algoritmo LCS a $x[1..n]$, $y[1..m]$ e $z'[1..o]$. Complexidade: $O(n.m.o)$.

   Complexidade total: $O(n^2.m)$.

II.B) O gestor de pessoal do Hospital Central de Caracolândia foi encarregue de fazer a calendarização das férias dos médicos do hospital tendo em conta as restrições indicadas em baixo:

- O hospital dispõe de m médicos $\{M_1, ..., M_m\}$.
- O calendário hospitalar é constituído por n dias de trabalho $\{D_1, ..., D_n\}$ distribuídos por k períodos de trabalho não sobrepostos $\{P_1, ..., P_k\}$; $\mathbf{days}(P_i)$ denota o conjunto dos dias de trabalho incluídos no período P_i.
- Cada médico pode usufruir de 22 dias de férias por ano e de, no máximo, 5 dias de férias por cada período de trabalho.
- Em cada dia de trabalho não podem estar mais de 10 médicos de férias.
- Cada médico comunicou ao gestor de pessoal o conjunto de dias de férias que lhe interessaria usufruir; $\mathbf{holidays}(M_j)$ representa o conjunto dos dias de férias seleccionados pelo médico M_j, com $22 \leq |\mathbf{holidays}(M_j)| < n$.
- $k < m < n$ e $m.k = O(n)$.

Tendo em conta as restrições enunciadas:

1. Modele o problema descrito em cima como um problema de fluxo máximo. A resposta deve incluir o procedimento utilizado para decidir se existe uma calendarização que satisfaça as restrições do problema e, se esta existir, quais os dias de férias a atribuir a cada médico.
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 médicos m, número de dias trabalho n e número de períodos k. De entre os algoritmos de fluxo estudados nas aulas deve escolher aquele que garanta a complexidade assimptótica mais baixa para o problema em questão.
   Nota: A resposta deverá necessariamente incluir as expressões que definem o número de vértices e de arcos da rede de fluxo proposta ($|V|$ e $|E|$, respectivamente) em função dos parâmetros do problema, bem como um upper-bound para o valor do fluxo máximo.

Solução:

1. *Construção da rede de fluxo:* $G = (V, E, c, s, t)$. Na construção da rede de fluxo consideramos um vértice por dia de trabalho, $k + 1$ vértices por médico, e dois vértices adicionais s e t, respectivamente a fonte e o sumidouro. Associamos cada médico M_j a $k + 1$ vértices, respectivamente designados por: M_j^0, M_j^1, ..., M_j^k. Intuitivamente, o vértice M_j^0 serve para seleccionar os dias de férias atribuídos ao médico M_j, enquanto os vértices M_j^i, com $1 \leq i \leq k$, servem para seleccionar os dias de férias atribuídos ao médico M_j no período i. Formalmente:

   - $V = \{s, t\} \cup \{M_j^i \mid 1 \leq j \leq m \ \wedge \ 0 \leq i \leq k\} \cup \{D_i \mid 1 \leq i \leq n\}$
   -

$$
\begin{aligned}
E = \ & \{(s, M_j^0, 22) \mid 1 \leq j \leq m\} \\
& \qquad M_j \text{ tem direito a 22 dias de férias} \\
& \cup \{(M_j^0, M_j^i, 5) \mid 1 \leq j \leq m \ \wedge \ 1 \leq i \leq k \ \wedge \mathbf{days}(P_i) \cap \mathbf{holidays}(M_j) \neq \emptyset\} \\
& \qquad \text{Cada médico pode ter no máximo 5 dias de férias em cada período} \\
& \cup \{(M_j^i, D_l, 1) \mid D_l \in \mathbf{days}(P_i) \cap \mathbf{holidays}(M_j)\} \\
& \qquad D_l \text{ pertence ao período } P_i \text{ e } M_j \text{ está interessado em } D_l \\
& \cup \{(D_l, t, 10) \mid 1 \leq l \leq n\} \\
& \qquad \text{só podemos ter 10 médicos de férias por dia}
\end{aligned}
$$

Existe uma calendarização que satisfaz as restrições do problema se $|f^*| = 22.m$. O conjunto de dias de férias do médico M_j é dado por: $\{D_l \mid \exists i. f^*(M_j^i, D_l) = 1\}$.

2. *Complexidade:*

   - $|V| \leq 2 + m.(k+1) + n = O(m.k + n) = O(n)$
   - $|E| \leq m + m.k + m.k.n + n = O(m.k.n) = O(n^2)$
   - $|f^*| \leq 22.m = O(m)$
   - Edmonds Karp (upper bound de FF): $O(|f^*|.E) = O(m.n^2)$
   - Edmonds Karp (upper bound EK): $O(E^2.V) = O(n^5)$
   - Relabel-To-Front: $O(n^3)$

   O limite mais apertado é obtido pelo upper bound do método de FF, pelo que podemos utilizar qualquer implementação do método Ford-Fulkerson.

II.C) Dado um conjunto de inteiros positivos $X = \{x_1, ..., x_n\}$, o problema da bi-partição de conjunto, **SetBiPartition**, consiste em determinar se existem dois subconjuntos de X, Y e Z tais que: **(1)** $Y \cap Z = \emptyset$, **(2)** $Y \cup Z = X$, e **(3)** $\sum_{y \in Y} y = \sum_{z \in Z} z$. Por exemplo, o conjunto $\{3, 4, 6, 9, 10\}$ pode ser dividido nos subconjuntos: $\{3, 4, 9\}$ e $\{6, 10\}$, ambos com soma 16.

1. Modele o problema **SetBiPartition** como um problema de decisão e mostre que está em **NP**.
2. Mostre que o problema **SetBiPartition** é **NP**-difícil por redução a partir do problema **Subset-Sum** que se sabe ser **NP**-difícil 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.
   Sugestão: A solução passa por determinar que elemento(s) devem ser acrescentados ao conjunto dado como input do problema de partida.

Problema Subset-Sum: Dado um conjunto de inteiros positivos $X = \{x_1, ..., x_n\}$ e um inteiro v, o problema **Subset-Sum** consiste em determinar se existe um subconjunto $Z \subseteq X$ tal que: $\sum_{z \in Z} z = v$. Formalmente, o problema **Subset-Sum** define-se da seguinte maneira:

$$\textbf{Subset-Sum} = \left\{ \langle X, v \rangle \mid \exists Z \subseteq X \,.\, \sum_{z \in Z} z = v \right\}$$

Solução:

1. •

$$\textbf{SetBiPartition} = \left\{ \langle X \rangle \mid \exists Y, Z \subseteq X \,.\, \sum_{y \in Y} y = \sum_{z \in Z} z \ \wedge \ Y \cap Z = \emptyset \ \wedge \ Y \cup Z = X \right\}$$

   - *Certificado:* par de conjuntos (Y, Z)
   - *Tamanho do Certificado:* $O(n)$, com $n = |X|$.
   - *Algoritmo de verificação:* Verificar se:
     - $Y \cap Z = \emptyset$. Utilizar um array *arr* de tamanho $|X|$. Percorrer os elementos de Y, colocando as posições correspondentes do array *arr* a 1. Percorrer os elementos de Z, retornando *false* se alguma das posições correspondentes do array *arr* estiver a 1.
       Complexidade: $O(n)$.
     - $Y \cup Z = X$. Utilizar um array *arr* de tamanho $|X|$. Percorrer os elementos de Y e Z, colocando as posições correspondentes do array *arr* a 1. Percorrer os elementos de X, retornando *false* se alguma das posições correspondentes do array *arr* estiver a 0.
       Complexidade: $O(n)$.
     - $\sum_{y \in Y} y = \sum_{z \in Z} z$.
       Complexidade: $O(n)$.
   - *Complexidade do algoritmo de verificação:* $O(n)$.

2. Há que mostrar que **Subset-Sum** $\leq_P$ **SetBiPartition**.

- *Redução:* Dado um conjunto X e um valor v, há que gerar um conjunto X' tal que $\langle X, v\rangle \in$ **Subset-Sum** $\Leftrightarrow \langle X'\rangle \in$ **SetBiPartition**. O conjunto X' é definido da seguinte forma:

$$X' \triangleq \begin{cases} X & \text{se } v = t/2 \\ X \cup \{2t - v, t + v\} & \text{c.c.} \end{cases}$$

  onde $t = \sum_{x\in X} x$.
- *Complexidade da redução:* $O(n)$.
- Prova da redução (extra): no caso $v = t/2$ não há nada a mostrar. O outro caso mostra-se em baixo.
  - *Prova* $\langle X, v\rangle \in$ **Subset-Sum** $\Rightarrow \langle X'\rangle \in$ **SetBiPartition**. Suponhamos que $\langle X, v\rangle \in$ **Subset-Sum**. Então existe um conjunto $Z \subseteq X$ tal que $\sum Z = v$. Seja $Y = X\backslash Z$, temos que $\sum Y = t-v$. Sejam $Z' = Z \cup \{2t-v\}$ e $Y' = t + v$, segue que $\sum Z' = \sum Y' = 2t$, $Z' \cup Y' = X'$ e $Z' \cap Y' = \emptyset$.
  - *Prova* $\langle X'\rangle \in$ **SetBiPartition** $\Rightarrow \langle X, v\rangle \in$ **Subset-Sum**. Suponhamos que $\langle X'\rangle \in$ **SetBiPartition**. Então existem dois conjuntos $Y', Z' \subseteq X'$ tais que: $\sum Z' = \sum Y' = 2t$, $Z' \cup Y' = X'$ e $Z' \cap Y' = \emptyset$. Como $2t - v > t$ e $t - v > t$, concluímos que estes dois elementos não podem pertencer ao mesmo subconjunto. Suponhamos, sem perda de generalidade, que $2t - v \in Z'$, concluímos que os restantes elementos de Z' têm soma v, pelo que basta escolher o conjunto $Z = Z'\backslash\{2t - v\}$.