



UNIVERSIDADE DE COIMBRA

Faculty of Science and Technology

Department of Informatics Engineering

## Laboratório de Programação Avançada

Written Test – June 13 2018

Name: \_\_\_\_\_ Student ID: \_\_\_\_\_

8 grade points in total, 2h 30m, closed books.

1. Consider the following recurrence relation. Let  $T_1, \dots, T_n$  be a sequence of  $n$  positive integers. For a given  $n$  and  $c$ , we define  $M(i, j)$ ,  $0 \leq i \leq n$ ,  $0 \leq j \leq c$ , as follows

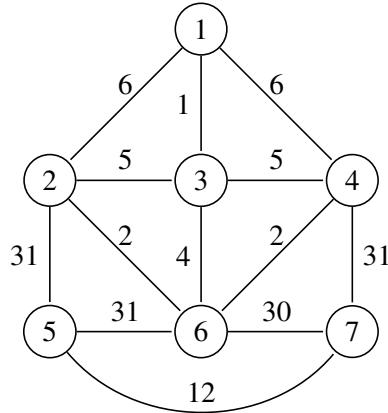
$$M(i, j) = \begin{cases} \text{true} & \text{if } j = 0 \text{ and } i \geq 0 \\ \text{false} & \text{if } j > 0 \text{ and } i = 0 \\ M(i - 1, j) \vee M(i - 1, j - T_i) & \text{if } j > 0, i > 0 \text{ and } j - T_i \geq 0 \\ M(i - 1, j) & \text{if } j > 0, i > 0 \text{ and } j - T_i < 0 \end{cases}$$

where  $\vee$  is the *logical or*. Give the pseudo-code of a bottom-up dynamic programming algorithm that explores the recurrence above to find the value for  $M(n, c)$ , for a given  $n$  and  $c$ , and discuss its time complexity. (1.5 g.p.)

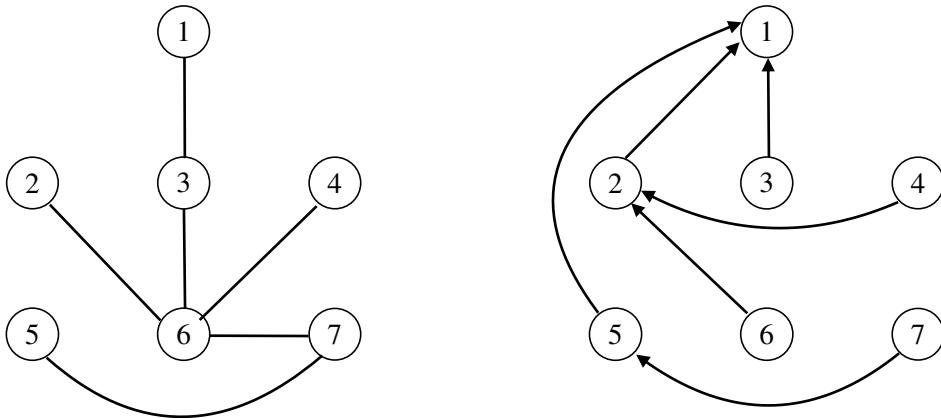
```
define M(n, c):
    for j = 0 to c:
        Matrix[0][j] = false
    for i = 0 to n:
        Matrix[i][0] = true
    for i = 1 to n:
        for j = 1 to c:
            if (j - T[i] >= 0):
                Matrix[i][j] = Matrix[i-1][j] or Matrix[i-1][j-T[i]]
            else:
                Matrix[i][j] = Matrix[i-1][j]
    return Matrix[n][c]
```

The algorithm presents a time complexity of  $O(n^2)$  given that it has a nested loop where it fills the result matrix.

2. Consider the following graph.



Draw its minimum spanning tree (left) as well as the graph of the union-find data structure (right), without path compression, using Kruskal algorithm. Always connect the root of the tree with the smallest height to the root of the tree with largest height and, in case of a tie, choose as root the node with the smallest label. (1.5 g.p.)



3. Write the tail recursive version of the algorithm below as well as its first call for an arbitrary non-negative number  $n$ . (1 g.p.)

```
Function real( $n$ )
  if  $n = 0$  then
    return 0.0
  else
    return 1.0 + real( $n - 1$ )
```

```
define real_tail( $n, r$ ):
  if ( $n = 0$ ):
    return  $r$ 
  else
    return real_tail( $n-1, r+1.0$ )
```

```
real_tail(2, 0)
  ↘ real_tail(1, 1.0)
    ↘ real_tail(0, 2.0)
      ↘ return 2.0
```

4. Consider the following problem: Given a tree  $T = (V, E)$ , where  $V$  is the set of vertices and  $E$  is the set of edges, and a root  $r \in V$ , color as many vertices in  $T$  as possible without coloring any pair of adjacent vertices (but two adjacent vertices are allowed to have no color). In order to solve this problem, we consider two subproblems at each vertex  $v \in V$ :

- $A(v)$  is the maximum number of colored vertices in the subtree rooted at  $v$  if  $v$  is not colored
- $B(v)$  is the maximum number of colored vertices in the subtree rooted at  $v$  if  $v$  is colored.

Let  $C(v)$  denote the children of vertex  $v \in V$  when traversing  $T$  from the root  $r$  to the leaves. The problem can be solved by the two following recurrence relations for any vertex  $v \in V$ :

$$A(v) = \sum_{u \in C(v)} \max\{A(u), B(u)\} \quad B(v) = 1 + \sum_{u \in C(v)} A(u)$$

The optimal value is given at the root  $r$  with  $\max\{A(r), B(r)\}$ .

- (a) Show that the calculation of  $B(v)$  for any  $v \in V$  is correct by using an optimal substructure argument. (1 g.p.)

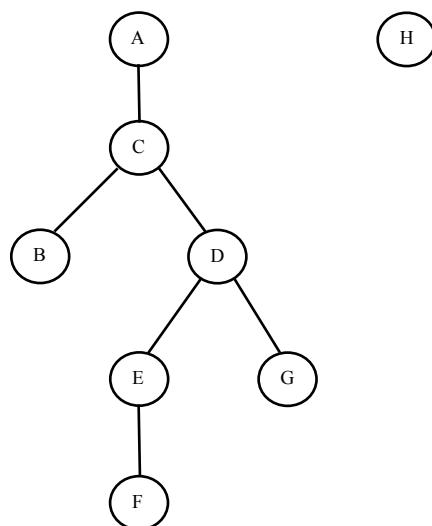
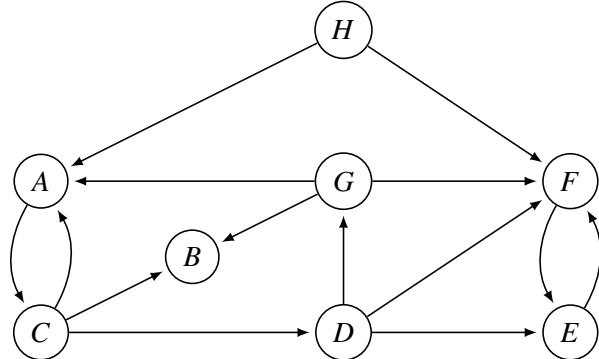
Optimal Substructure:  $\text{MaxB}(u) \setminus \{v\} = \cup \text{MaxA}(u), u \in C(v)$

- Every  $u \in C(v)$  cannot be colored.
- Every subtree rooted in  $u \in C(v)$  is independent.
- Every subtree rooted in  $u \in C(v)$  is required.
- If  $\exists w < A(u), u \in C(v) \Rightarrow B(v)$  would not be optimal.
- If  $\exists w > A(u), u \in C(v) \Rightarrow A(u)$  would not be optimal and  $B(v)$  would not be optimal.

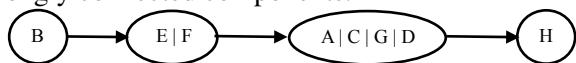
- (b) Write the pseudo-code of an algorithm that explores the two recursions above. Assume that the algorithm starts from the root  $r$  and traverses the tree in a depth-first search fashion. Give the first call of the algorithm for a tree  $T$  and a root  $r$ . In addition, discuss the time complexity of your approach. (1.5 g.p.)

```
Function Color(T, v):
    A = 0
    B = 1
    for u ∈ C(v):
        (a, b) = Color(T, u)
        A = A + max(a, b)
        B = B + a
    return (A, B)
```

5. Find the strongly connected components of the following graph using Tarjan algorithm. Report the DFS tree(s) starting from vertex  $A$  and traversing the graph following the alphabetic order of the vertices labels. In addition, report the strongly connected components on the box below, ordered by the time they are found in the Tarjan algorithm. (1.5 g.p.)



Strongly connected components:





UNIVERSIDADE DE COIMBRA

Faculdade de Ciências e Tecnologia  
Departamento de Engenharia Informática

Laboratório de Programação Avançada  
Primeiro teste escrito – 13 de junho de 2018

Nome: \_\_\_\_\_ N° de estudante: \_\_\_\_\_

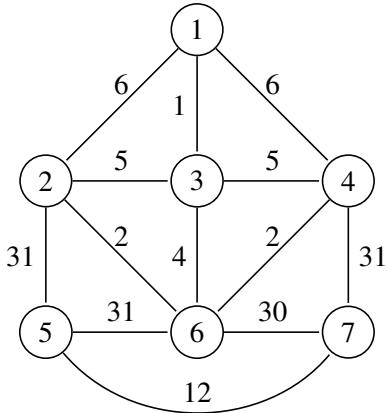
8 pontos no total, 2 horas e 30 minutos, sem consulta.

1. Considere a seguinte relação de recorrência. Seja  $T_1, \dots, T_n$  uma sequência de  $n$  inteiros positivos. Para um dado  $n$  e  $c$ , seja  $M(i, j)$ ,  $0 \leq i \leq n$ ,  $0 \leq j \leq c$ , definido da seguinte forma:

$$M(i, j) = \begin{cases} \text{verdadeiro} & \text{se } j = 0 \text{ e } i \geq 0 \\ \text{falso} & \text{se } j > 0 \text{ e } i = 0 \\ M(i - 1, j) \vee M(i - 1, j - T_i) & \text{se } j > 0, i > 0 \text{ e } j - T_i \geq 0 \\ M(i - 1, j) & \text{se } j > 0, i > 0 \text{ e } j - T_i < 0 \end{cases}$$

em que  $\vee$  corresponde ao operador lógico *ou*. Descreva, em pseudo-código, um algoritmo de programação dinâmica ascendente que explore a recorrência acima para encontrar o valor de  $M(n, c)$  para um dado  $n$  e  $c$  e discuta a sua complexidade temporal. (1.5 pontos)

2. Considere o seguinte grafo.



Encontre a árvore geradora mínima na rede acima recorrendo ao algoritmo de Kruskal. Desenhe a árvore geradora mínima à sua esquerda e o grafo da estrutura de dados *union-find* (sem o passo de compressão de caminho) à sua direita. Quando necessário, ligue a raiz da árvore com menor altura à raiz da árvore com maior altura e, em caso de empate, escolha, como raiz, o nó que apresentar a etiqueta com o menor valor. (1.5 pontos)

(1)

(1)

(2)

(3)

(4)

(2)

(3)

(4)

(5)

(6)

(7)

(5)

(6)

(7)

3. Escreva a versão com recursão de cauda da seguinte função recursiva. Apresente também a primeira chamada da sua função para um inteiro não-negativo  $n$ . (1 ponto)

```
Function real(n)
  if n = 0 then
    return 0.0
  else
    return 1.0 + real(n - 1)
```

4. Considere o seguinte problema. Dada uma árvore  $T = (V, E)$ , em que  $V$  e  $E$  correspondem ao conjunto de vértices e arestas em  $T$ , respectivamente, e uma raiz  $r \in V$ , atribua uma cor ao maior número possível de vértices em  $T$  sem que dois vértices adjacentes fiquem coloridos (mas é permitido que dois vértices adjacentes não tenham cor). Para resolver este problema, considere os dois subproblemas seguintes a cada vértice  $v \in V$ :

- $A(v)$  é o número máximo de vértices coloridos na sub-árvore com raiz em  $v$  se  $v$  não estiver colorido.
- $B(v)$  é o número máximo de vértices coloridos na sub-árvore com raiz em  $v$  se  $v$  estiver colorido.

Seja  $C(v)$  o número de descendentes do vértice  $v \in V$  quando se efetua uma travessia em  $T$  da raiz para as folhas. O problema pode ser resolvido através das duas relações de recorrência para qualquer vértice  $v \in V$ :

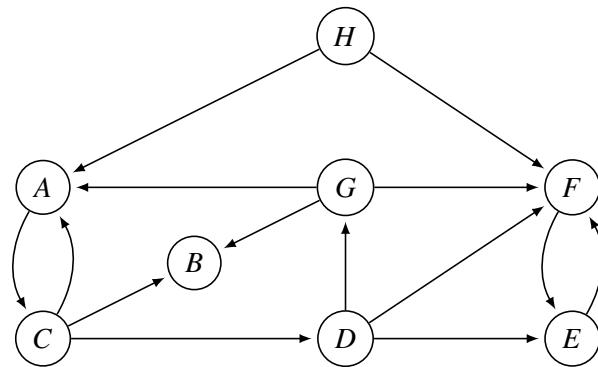
$$A(v) = \sum_{u \in C(v)} \max\{A(u), B(u)\} \quad B(v) = 1 + \sum_{u \in C(v)} A(u)$$

O valor ótimo é obtido na raiz  $r$  com  $\max\{A(r), B(r)\}$ .

- (a) Demonstre, recorrendo ao argumento de sub-estrutura ótima, que o cálculo de  $B(v)$  para qualquer vértice  $v \in V$  está correto. (1 ponto)

- (b) Escreva o pseudo-código de um algoritmo que explore as duas relações de recorrência. Assuma que o algoritmo tem início na raiz  $r$  e que atravessa a árvore em profundidade. Escreva igualmente a primeira chamada do algoritmo para uma árvore  $T$  e uma raiz  $r$ . Discuta igualmente a complexidade temporal da sua abordagem. (1.5 pontos)

5. Encontre as componentes fortemente conexas do grafo seguinte recorrendo ao algoritmo de Tarjan. Reporte a árvore de DFS a partir do vértice  $A$  e escolha os vértices para a travessia de acordo com a ordem alfabética das etiquetas nos vértices. Indique igualmente as componentes fortemente conexas que encontrou, ordenadas pelo momento em que foram encontradas pelo algoritmo de Tarjan (na caixa). (1.5 pontos)



Componentes fortemente conexas:

