

## Métodos Computacionais

Teste #1 – 19 Dezembro 2020

### Grupo 3

Disponibilizado às 16:45 - Resolução entregues até às 17:30

1. a) (1.5 val) Complete a especificação da função **sublista**, no ficheiro **grupo\_3.py**, que dada uma lista **L** e um valor “alvo” **v**, escolhe arbitrariamente elementos distintos de **L**, até que a escolha de um outro elemento distinto faça a sua soma exceder **v**, retornando o tuplo com a sublista obtida e a soma dos seus elementos.

Por exemplo, para **L1 = [9, 21, 37, 11, 19]** e **v1 = 32** (definidos no ficheiro **g3.py**), a chamada da função

**sublista(L1,v1)**

pode retornar entre outros e em chamadas diferentes, os tuplos **([19,9],28)**, ou **([19,11],30)**, ou **([11,19],30)**, ou **([11,21],32)**, ou ..., dependendo das escolhas aleatórias que forem feitas na escolha dos elementos de **L1**.

- b) (1.5 val) Complete a especificação da função **sublista\_melhor** que repete a chamada da função anterior **n** vezes e retorna a sublista obtida cuja soma tenha sido mais próxima de **v**, bem como o valor da sua soma.

Por exemplo, para os anteriores valores de **L1** e **v1**, e para um valor de **n** “suficientemente grande”, a chamada da função

**sublista\_melhor(L1,v1,n)**

deve retornar, um dos tuplos **([11,21],32)** ou **([11,21],32)**.

2. (2.5 val) Considere ficheiros de texto que contêm uma tabela formatada da seguinte forma. A primeira linha contém pares nome/tipo, separados por “,” (vírgulas), em que cada par identifica os nomes dos campos da tabela, e o seu tipo (“**int**”, “**flt**” ou “**str**”). As linhas seguintes incluem os vários itens da tabela, igualmente separados por vírgulas. Note que as linhas podem conter espaços imediatamente antes e depois das vírgulas de separação. Por exemplo, considere o ficheiro **tabela.txt** (enviado com o enunciado) com o seguinte conteúdo:

```
id/int, nome/str, idade/int, dep/str, salario/flt
314, Rui Santos, 46, CNT, 2167.23
196, Isabel Sá, 37, RH, 1585.35
278, Luís Matos, 31, INF, 3142.50
```

Complete a especificação da função **dicionários** que recebe como argumento o nome de um ficheiro com a codificação indicada e devolve um tuplo **(C, L)** em que **C** é uma lista dos nomes dos campos que funcionam como chaves, e **L** é a lista de dicionários cujas chaves são as obtidas na primeira linha e os campos têm os tipos indicados nessa linha.

Por exemplo, a chamada da função

**dicionários("tabela.txt")**

deve retornar, um tuplo **(C,L)** em que **C=["id","nome","idade","dep","salario"]** é a lista de chaves com os nomes indicados na primeira linha, e a lista **D =**

```
[{"id":314, "nome":"Rui Santos","idade":46 , "dep":"CNT","salario":2167.23},  
 {"id":196, "nome":" Isabel Sá","idade":37 , "dep":"RH","salario": 1585.35},  
 {"id":278, "nome":" Luís Matos","idade":31 , "dep":"INF","salario": 3142.50}]
```

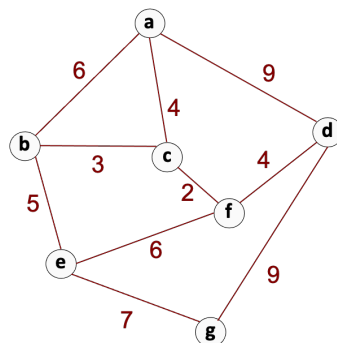
é a lista dos dicionários das linhas seguintes do ficheiro **"tabela.txt"**.

3. (2.5 val) Relembre as funções **prim**, **floyd** e **path** estudadas nas aulas, e que são incluídas no ficheiro **g3.py**:

- **prim(A)** – Retorna a árvore de cobertura mínima de um grafo dado pela sua matriz de adjacências **A**;
- **floyd(A)** – Retorna uma matriz **D** de distâncias mínimas entre quaisquer dois nós do grafo dado pela sua matriz de adjacências **A**, bem como uma matriz **N** em que **N [i][j]** indica qual o nó que se deve seguir a partir do nó **i**, se se pretender obter o caminho mais curto entre os nós **i** e **j**.
- **path(u,v,N)** – Determina a sequência de nós que correspondem ao caminho mais curto entre os nós **u** e **v**. A matriz **N** é a retornada pela função **floyd**.

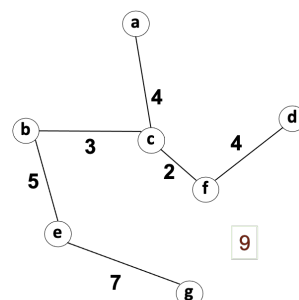
O grafo da figura pode ser codificado pela seguinte matriz de adjacências (com **x = inf**),

```
      a b c d e f g  
A3 = [[0,6,4,9,x,x,x], #a  
      [6,0,3,x,5,x,x], #b  
      [4,3,0,x,x,2,x], #c  
      [9,x,x,0,x,4,9], #d  
      [x,5,x,x,0,6,7], #e  
      [x,x,2,4,6,0,x], #f  
      [x,x,x,9,7,x,0]] #g
```



e a sua árvore de cobertura mínima pode ser igualmente representada pela seguinte matriz de adjacências (com **z = -1**)

```
      a b c d e f g  
T3 = [[0,z,4,z,z,z,z], #a  
      [z,0,3,z,5,z,z], #b  
      [4,3,0,z,z,2,z], #c  
      [z,z,z,0,z,4,z], #d  
      [z,5,z,z,0,z,7], #e  
      [z,z,2,4,z,0,z], #f  
      [z,z,z,z,7,z,0]] #g
```



Assuma que a árvore de cobertura mínima corresponde a uma rede de canalizações de água que se pretende instalar numa aldeia, em que os nós do grafo representam casas e os arcos a distância entre elas através das ruas existentes. Assuma ainda que se pretende construir uma rede de distribuição de água através de (algumas de) as ruas existentes, devendo as canalizações ter capacidade suficiente para os fluxos de água necessários, e sendo o ponto de distribuição inicial junto ao primeiro nó. Para esse efeito são contratados os caudais máximos para alimentar cada uma das casas, numa lista C.

Complete a especificação da função **capacidades**, que dada uma matriz de adjacências de um grafo, e nas condições descritas, determina a capacidade mínima de cada arco do gráfico utilizado de forma a satisfazer as necessidades de águas das diferentes casas. A função deve retornar a matriz D, semelhante a uma matriz de adjacências correspondente a uma árvore de cobertura mínima, mas em que os elementos não nulos da matriz correspondem à capacidade da canalização dos arcos em que são instaladas canalizações (i.e.  $D[i][j]$  representa a capacidade da canalização entre os nós i e j).

Por exemplo, dado a lista de caudais **C3 = [23, 34, 12, 25, 67, 89, 56]**, a chamada  
**capacidades (A3,C3)**

deve retornar a matriz D

```

#   a   b   c   d   e   f   g
D = [[ 0 , 0 ,283, 0 , 0 , 0 , 0 ], #a
      [ 0 , 0 ,157, 0 ,123, 0 , 0 ], #b
      [283,157, 0 , 0 , 0 ,114, 0 ], #c
      [ 0 , 0 , 0 , 0 , 0 , 25, 0 ], #d
      [ 0 ,123, 0 , 0 , 0 , 0 , 56], #e
      [ 0 , 0 ,114, 25, 0 , 0 , 0 ], #f
      [ 0 , 0 , 0 , 0 , 56, 0 , 0 ]] #g

```