Programação II

Trabalho 2

Complexidade e busca em listas duplas

Entrega a 22 de março de 2021

A. Complexidade assintótica Para a primeira parte deste trabalho deverá fazer a análise da complexidade assintótica das funções abaixo.

A.1. A função `f1` recebe um inteiro positivo `n` e uma lista `v`. Pode assumir que `v` tem tamanho superior a `n`.

```
def f1(n, v):
    b = n * n
    s = 0
    while b > 1:
        s += v[n]
        s += b
        b -= 2
    return s
```

A.2. A função `f2` recebe um dicionário `d` e uma lista `l`.

```
def f2(d,l):
    r = []
    for x in l:
        if x not in d:
            r.append(x)
    return r
```

As respostas deverão ser dadas em notação O-grande, $\mathcal{O}$, e funções de uma ou mais variáveis n, m.

- Indique claramente o significado das variáveis escolhidas.
- Para cada uma das duas funções apresente os cálculos que conduzem ao resultado (pode utilizar os números de linha das funções para o efeito)

B. Busca em listas duplas Para a segunda parte deste trabalho pretende-se a implementação de uma função de procura em listas duplas ordenadas. Uma lista dupla é uma lista de listas, como por exemplo

```
lista_dupla = [[2, 4, 4, 6], [7, 11, 12, 13],
         [13, 13, 13, 13], [15, 19, 42, 100]]
```

Uma lista dupla está ordenada se todas as suas sublistas estão ordenadas (de forma crescente) e o último elemento de cada sublista é menor ou igual ao primeiro elemento da sublista seguinte. O exemplo `lista_dupla` dado acima constitui uma lista dupla ordenada. Mas o seguinte exemplo não constitui uma lista dupla ordenada.

```
lista_nao_ordenada = [[3, 5, 5, 10], [17, 19, 23, 30],
                      [11, 12, 13, 14], [32, 42, 42, 42]]
```

Para este trabalho deverá escrever o predicado `busca_lista_dupla(lista,x)` que recebe uma lista dupla ordenada `lista` e um elemento `x` e verifica se `x` ocorre em `lista`, isto é, se é um elemento de alguma sublista de `lista`.

Exemplos de execução:

```
>>> print(busca_lista_dupla(lista_dupla,4))
True
>>> print(busca_lista_dupla(lista_dupla,5))
False
>>> print(busca_lista_dupla(lista_dupla,14))
False
>>> print(busca_lista_dupla(lista_dupla,42))
True
```

Pode fazer as seguintes assunções, indicando-as na descrição `docstring` da função:

- os elementos que ocorrem nas sublistas são valores inteiros;

- nenhuma sublista de `lista` é vazia (no entanto, a própria `lista` poderá ser vazia).

Tome em especial atenção os seguintes pontos.

- Deverá ter em conta o que aprendeu sobre algoritmos eficientes de busca. A sua solução deverá tomar partido do facto de a lista estar ordenada.
- As regras de boas práticas de desenvolvimento de software apontam para um número máximo de cerca de 10 linhas por função. Identifique as abstrações relevantes e implemente cada uma numa função separada.
- Pode utilizar e adaptar funções que tenham sido apresentadas nas aulas.
- Cada função que escrever (a obrigatória `busca_lista_dupla`, bem como aquelas que porventura achar relevantes) deve estar equipada com uma descrição em formato `docstring`, tal como sugerido nas aulas.
- Não se esqueça de incluir o seu nome e número de estudante no início do ficheiro: `__author__ = Maria Lopes, 45638`.
- O vosso código será testado por um processo automatizado. É indispensável que este contenha uma função com o nome `busca_lista_dupla`, que a função espere exatamente o número e o tipo dos parâmetros e que devolva exatamente um valor booleano.
- Este é um trabalho de resolução individual. Os trabalhos devem ser entregues no Moodle até às 23:59 do dia 22 de março de 2021.
- Os trabalhos de todos os alunos serão comparados por uma aplicação de deteção de plágio em programas. Recorde o seguinte texto na secção Integridade Académica da Sinopse:

  "Alunos detetados em situação de fraude ou plágio (plagiadores e plagiados) em alguma prova ficam reprovados à disciplina e serão alvo de processo disciplinar, o que levará a um registo dessa incidência no processo de aluno, podendo conduzir à suspensão letiva."