Ciências ULisboa | Informática

Programação II
Exame Tipo
1 junho 2022

Material junto de si: *Deverá ter consigo* apenas *canetas, o cartão de estudante e possivelmente uma garrafa de água. Os casacos, os sacos e as mochilas com o restante material, incluindo telemóveis, deverão ser deixados junto ao quadro. A violação destas regras implica a anulação do exame.*
Sobre a escrita do exame: *Inicie cada novo grupo numa* página *separada.*
Duração: *Duas horas e meia.*

> **Nota:** Este exame tipo contém um grupo sobre cada tópico abordado na disciplina. Cada um destes grupos poderia ter sido tirado de uma das fichas de exercícios que foram disponibilizadas ao longo do semestre. O exame de primeira e de segunda época deverão ser mais curtos e possivelmente endereçarão mais de que um tópico no mesmo exercício (por exemplo, programação de sistema com expressões regulares).

Grupo 1. [Complexidade algorítmica, 2 valores]
Considere o seguinte programa.

```
def f (l1, l2):
  resultado = 0
  for x in l1:
    if x % 2 == 0:
      resultado = resultado + (1 if sum(l2)==0 else 2)
  return resultado
```

Considere que as duas listas `l1` e `l2` têm comprimentos na mesma ordem de grandeza. Pretendemos analisar a complexidade de `f` em função de uma variável n.

a) Qual o significado do valor n que faz sentido para este caso?

b) Qual o número de operações básicas que a função `f` efetua, em função de n? Justifique.

c) Qual a complexidade algorítmica da função quando expressa na notação $\mathcal{O}$? Justifique.

Grupo 2. [Algoritmos de pesquisa & ordenação, 3 valores]
Nota: Para este grupo, adicione documentação para todas as funções que escrever em formato `pydoc`, tal como estudado nas aulas.

Considere uma lista com as existências num supermercado: cada entrada na lista é um triplo contendo o nome do produto, a quantidade disponível e o preço.

```
produtos = [('Grão', 78, 2.94), ('Seitan', 7, 3.14),
            ('Couve-roxa', 78, 1.79), ...]
```

a) Utilize a função **`sorted`** para obter, a partir da lista original, uma outra lista em que os elementos se encontram por ordem crescente das quantidades disponíveis.

b) Mesma questão, mas desta vez, em caso de empate, as entradas devem aparecer por ordem alfabética dos nomes. Para a lista acima, o triplo contendo a `Couve-roxa` deverá aparecer antes daquele que contém `Grão`.

c) Considere o algoritmo de ordenação por seleção descrito do seguinte modo:

> O algoritmo corre em tantas iterações quantos os elementos na lista. Na iteração i calcula-se o índice do menor valor da sublista que começa em i. O valor no índice assim obtido é trocado com o valor no índice i.

Mais considere dada uma função `indice_minimo(lista)` que dada uma `lista` *não* vazia devolve o índice do menor valor na lista. Escreva uma função `ordenacao_selecao` que recebe uma lista e ordena-a pelo método de seleção. Note que a própria lista deve ser alterada, pelo que a função `ordenacao_selecao` não devolve nada.

Grupo 3. [Testes, 3 valores]

Considere uma função `ordena` que ordena uma lista. A lista é alterada pela função, a função não devolve nada. A função de ordenação poderia ser a função do grupo anterior.

a) Proponha pelo menos 3 características para analisar o comportamento da função.

b) Para cada característica identifique blocos adequados.

c) Combine todos os blocos, eliminando os pares inviáveis. Apresente os resultados na forma de uma tabela.

d) Apresente os testes em formato `doctest`.

Grupo 4. [Ordem superior, 2 valores]

Nota: Para este grupo, adicione documentação para todas as funções que escrever em formato `pydoc`, tal como estudado nas aulas.

Considere uma função de inteiros para inteiros e um intervalo de números inteiros $\{n, n+1, \ldots, m\}$ dado por um par `(n, m)`.

a) Escreva a função `max_em_intervalo(funcao, intervalo)` utilizando *uma linha* de código Python (para além da linha do **`def`**).

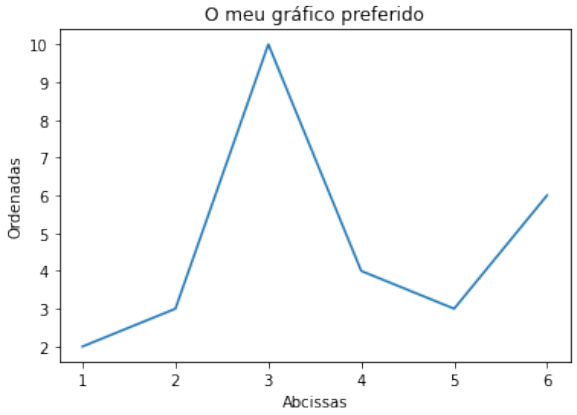
```
>>> max_em_intervalo(lambda x: x**2, (1, 20))
400
```

b) Este exercício é semelhante ao anterior, mas desta vez pretendemos não o valor da função mas o ponto do domínio onde a função toma o valor máximo. Para resolver este exercício deve utilizar uma função de ordem superior e não deve utilizar nem iteração (ciclos) nem recursão.

```
>>> indice_max_em_intervalo(lambda x: x**2, (1,20))
20
```

Grupo 5. [Visualização com `matplotlib`, 2 valores]

Usando módulo `matplotlib` escreva uma função `grafico` que gere o seguinte gráfico.



Grupo 6. [Valores separados por vírgulas, 2 valores]

Considere um ficheiro lista com as existências num supermercado em formato de valores separados por vírgulas. O ficheiro tem uma linha cabeçalho.

```
Produto, Stock, Preço
Grão, 78, 2.94
Seitan, 7,  3.1
Couve-roxa, 78, 1.79
```

Escreva uma função que dado um ficheiro de valores separados por vírgulas, devolva a linha (em forma de triplo) com o produto com menor

stock. Eis um exemplo, considerando `supermercado.csv` um ficheiro com os valores constantes na tabela acima:

```
>>> menor_stock(supermercado.csv)
(Seitan, 7, 3.1)
```

Grupo 7. [Programação de sistema, 2 valores]

O comando Unix `find` percorre diretorias (e sub diretorias) procurando ficheiros com certas características. Na versão extremamente simplificada em que estamos interessados procura ficheiros com uma dada terminação e afixa os caminhos completos dos ficheiros no terminal. Escreva um programa que lê da linha de comandos uma diretoria e uma extensão, e que imprime o caminho completo de todos os ficheiros com a extensão dada.

```
$ python find.py src csv
src/praticas/pauta.csv
src/praticas/temperaturas.csv
src/teoricas/medias.csv
src/teoricas/pessoas.csv
src/teoricas/all_month.csv
```

Para determinar se um ficheiro termina com uma dada extensão pode utilizar o método `endswith(terminação)` da classe **`str`**.

Grupo 8. [Expressões regulares, 2 valores]

Um endereço IPv4 tem 32 bits. Quando apresentado para ser lido por humanos é geralmente apresentado em formato decimal separado por pontos, como por exemplo `192.0.2.1`. Embora os números estejam necessariamente contidos entre `0` e `255`, para efeitos deste exercício vamos imaginar que cada um dos números contém entre um e três algarismos.

a) Defina uma expressão regular que reconheça endereços IPv4 válidos.

b) Os encaminhadores domésticos utilizam automaticamente endereços começados por `192`. Escreva uma função `domestico` que, dada uma frase (uma `string`) devolva `True` se a) a frase contém pelo menos uma ocorrência de um endereço IPv4 e b) o endereço começa por `192`.

```
>>> domestico("O meu endereço IP é 192.0.2.1, e o teu?")
True
>>> domestico("O meu é 255.0.255.0... que estranho!")
False
```