UNIVERSIDADE DE COIMBRA
FACULDADE DE CIÊNCIAS E TECNOLOGIA

Departamento de Engenharia Informática

Projeto #1
Algoritmos e Estruturas de Dados

2024-2025 – 2º Semestre

Submissão de relatório e código (InforEstudante): 23 de fevereiro 23:59

Anotações: *Em anexo template do relatório a ser entregue no InforEstudante.*

É incentivado que os alunos discutam ideias e questões relativas ao trabalho proposto, mas é entendido que quer a reflexão final sobre os resultados obtidos, quer o código desenvolvido, são próprios.

Objetivos:

Com o desenvolvimento deste projeto pretende-se que o aluno consolide os conhecimentos sobre a importância da análise de complexidade O-Grande de um algoritmo e a viabilidade das implementações propostas. Na análise de complexidade vamo-nos concentrar no fator tempo**.**

Tarefas:

A. Implementação de três algoritmos para o mesmo problema com complexidades diferentes.

B. Análise empírica de complexidade dos três algoritmos.

Conceitos: Algoritmos

Um algoritmo deve ter as seguintes propriedades (Donald E. Knuth, The Art of Computer Programming, vol. 1, 3rd edition, 2016):

- ***Finiteness*****:** deve terminar num número finito de passos.
- ***Definiteness:*** todos os passos devem ser definidos com rigor.
- ***Input*****:** deve receber zero ou mais dados à entrada.
- ***Output*****:** deve retornar um ou mais resultados à saída.
- ***Effectiveness*****:** todos os passos devem ser possíveis de resolver sem ambiguidade em tempo finito com os recursos disponíveis.

Conceitos: Análise de Complexidade

Análise teórica: análise da complexidade de um algoritmo *a priori* sem ser necessária a sua implementação ou execução. Vamos falar sobre isto nas sessões teóricas.

Análise empírica: análise do comportamento empírico de uma implementação do algoritmo *a posteriori* com base na recolha de dados.

Problema a usar neste trabalho: Encontrar o número ausente numa sequência não ordenada de números inteiros.

Dada uma sequência $A = [a_1, a_2, \ldots, a_n]$ composta por n números inteiros, o objetivo deste problema é identificar o valor em falta para completar a sequência. A sequência pode conter números negativos e positivos, e pode não estar ordenada.

Exemplos:

Input: [−2, 6, 1, −4, 7, 0, −5, 4, 2, −1, 5]

Output: 3

Input: [22, 19, 25, 21, 24, 26, 23]

Output: 20

Entrada:

Como entrada o programa deve receber uma lista de números inteiros que representam os elementos da sequência. Os números podem ser negativos ou positivos, e a sequência pode não estar ordenada.

Saída:

Como saída o programa deve devolver o valor que falta numa sequência de números.

Algoritmo 1: Solução exaustiva

Resolver o problema recorrendo a uma solução exaustiva, ou seja, ter dois ciclos para verificar qual o número em falta na sequência.

Algoritmo 2: Solução c/ ordenamento

Implementar uma solução que envolva ordenar a sequência antes de procurar o número ausente.

Nota: *utilizar uma função de ordenamento disponível na linguagem de programação utilizada (C:* `qsort`*, C++:* `sort`*, Java:* `Collections.sort`*, Python:* `list.sort` *ou* `sorted`*).*
Não esquecer que a complexidade temporal deste algoritmo também depende da complexidade do algoritmo de ordenamento utilizado.

Algoritmo 3: Solução elaborada

Implementar uma solução que apresente uma complexidade temporal inferior, utilizando uma abordagem mais eficiente. Uma possibilidade é usar uma fórmula matemática para calcular a soma expectável da sequência, comparando com a soma dos elementos fornecidos, permitindo assim identificar o número ausente de forma otimizada.

Tarefa A

Implementar as diferentes soluções.

Tarefa B

Análise empírica das soluções implementadas (ver exemplo nos slides da aula teórica sobre análise de complexidade). Recomendações a ter em conta:

- Medir o tempo de execução das soluções para sequências de diferentes tamanhos (ex. 100000, 200000, ... 1000000).
- Organizar os resultados das medições num gráfico (identificar os eixos do gráfico e as unidades utilizadas).
- No gráfico mostrar o resultado da regressão escolhida para sumariar os valores obtidos.

Relatório

O relatório deve seguir o *template* disponibilizado, incluindo:

- Gráficos com as medições e resultado da regressão para cada solução desenvolvida.
- Reflexão crítica sobre o resultado da regressão e possíveis *outliers.*
- Análise de complexidade com base nos resultados empíricos obtidos.