UNIVERSIDADE DE COIMBRA
FACULDADE DE CIÊNCIAS E TECNOLOGIA

Departamento de Engenharia Informática

Projeto #2
Algoritmos e Estruturas de Dados

2020-2021 – 2º Semestre

Submissão no Mooskak:
até 15 min. após o final da segunda aula P#2

Anotações: *este projeto foi preparado para ser resolvido maioritariamente no espaço das duas sessões práticas. O código deve ser submetido até 15 min. após o final da segunda sessão prática.*

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 da autoria de cada estudante. Procedimentos contrários ao que é dito acima, nomeadamente cópia de código desenvolvido por colegas ou obtido da net é entendido como fraude. Para além de a fraude denotar uma grave falta de ética e constituir um comportamento não admissível num estudante do ensino superior e futuro profissional licenciado, esta prejudica definitivamente o processo de aprendizagem do infrator.

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. 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 de um algoritmo devem ser definidos com rigor;
- ***Input*****:** deve receber zero ou mais dados à entrada;
- ***Output*****:** deve retornar um ou mais resultados à saida;
- ***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: Máximo da soma de dois números num array

Dado um *array*, determinar a soma máxima de quaisquer dois números do mesmo.

Exemplos :

Input : {2, 7, 3, 4, 1, 9}
Output : 16 (7 + 9)

Input : {5, 1, 5, 3}
Output : 10 (5 + 5)

Entrada:

Como entrada o programa recebe uma linha com o número de elementos do *array*, seguida de uma linha com os elementos do array do tipo int contidos na gama [0 .. 1000000] separados por um espaço.

Saída:

Como saída o programa imprime a soma máxima de dois números do array.

Exemplos:

Entrada

```
6
2 7 3 4 1 9
```

Saída

```
16
```

Entrada

```
4
5 1 5 3
```

Saída

```
10
```

Solução exaustiva

Resolver o problema recorrendo a uma solução exaustiva, ou seja, ter dois ciclos para verificar a soma de todas as combinações de dois números.

Solução melhorada 1 - com ordenamento

Resolver o problema começando por ordenar o array de forma decrescente. De seguida, basta escolher os primeiros dois elementos e somá-los.

Solução melhorada 2 - sem ordenamento

Dado que só precisamos dos dois números máximos no array, podemos manter duas variáveis, uma que contém o número mais alto, e outra que contém o segundo número mais alto. Depois basta percorrer o array com um ciclo e manter essas duas variáveis atualizadas.

Tarefa A

Implementar e submeter no Mooshak as diferentes soluções.

- A solução exaustiva, deve ser submetida ao problema A no Mooshak;
- A primeira solução melhorada (com ordenamento), deve ser submetida ao problema B;
- A segunda solução melhorada (sem ordenamento), deve ser submetida ao problema C.

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 arrays de diferentes tamanhos;
- Agregar os resultados das medições numa tabela;
  - É importante definir de forma clara o que representa cada coluna e linha na tabela;
- Para cada solução criar um gráfico com os dados das medições e resultado de uma regressão;
  - É importante definir as escalas (e.g. linear, logarítmica) que mais se adequem;
  - Colocar toda a informação necessária nos gráficos, e.g. nomes nos eixos, unidades de tempo.

Relatório (a submeter no inforesudante):

O relatório a realizar com base no formulário disponibilizado deve ter em conta as recomendações em cima descritas e incluir:

- Tabela concisa com as medições efetuadas;
- 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;