GUIÃO 03 – ANÁLISE DA COMPLEXIDADE DE ALGORITMOS

A documentação da linguagem C pode ser consultada em https://en.cppreference.com/w/c

1. Analise o código do ficheiro **`functions_iterations_count.c`** que permite realizar testes computacionais sobre um conjunto de algoritmos (simples) com diferentes características, contando o número de iterações efetuadas e apresentando tabelas com resultados.

A análise dessas tabelas permite determinar a **ordem de complexidade** de cada um dos algoritmos e uma **expressão para o número de iterações** que efetuam.

a) Compile o ficheiro e execute o programa. Guarde o output num ficheiro de texto.

b) **Análise experimental:** para cada tabela, verifique como crescem os valores de cada coluna, determine a ordem de complexidade do correspondente algoritmo e tente obter uma expressão para o número de iterações efetuadas.

c) **Análise formal:** para cada um dos algoritmos implementados, estabeleça uma expressão, em função de n, para o número de iterações que efetuam e determine a sua ordem de complexidade.

d) Compare os resultados da análise formal e da análise experimental dos algoritmos.

e) Analisando a última coluna de cada tabela, consegue identificar o padrão associado (i.e., a "assinatura") de cada uma das ordens de complexidade?

2. Analise o código do ficheiro **`functions_timing.c`** que permite realizar testes computacionais sobre um conjunto de algoritmos (simples) com diferentes características, registando o seu tempo de execução e apresentando tabelas de resultados.

A análise dessas tabelas permite determinar a **ordem de complexidade** de cada um dos algoritmos.

Atenção: os algoritmos (funções) são os mesmos do exercício anterior.

a) Compile o ficheiro e execute **várias vezes** o programa. Guarde o output em ficheiros de texto.

b) Há diferenças significativas entre os resultados de diferentes execuções do programa?

c) **Análise experimental:** para um dos ficheiros de resultados, verifique como crescem os valores das colunas de cada tabela e determine a ordem de complexidade do correspondente algoritmo.

d) Analisando a última coluna de cada tabela, consegue identificar o padrão associado (i.e., a "assinatura") de cada uma das ordens de complexidade?

e) Compare as tabelas e os resultados que obteve com as tabelas e os resultados do exercício anterior.

f) No seu computador, consegue obter **resultados**, em tempo útil, para **maiores valores de n?**

3. Um **factorião**, para uma dada base, é um número inteiro positivo n que é igual à soma do **factorial** de cada um dos seus **algarismos**.

Escreva um programa eficiente que lhe permita listar, para a base 10, todos os **factoriões menores que 10^6**.

ATENÇÃO: 0! = 1

SUGESTÃO: Armazene num *array* os factoriais dos sucessivos algarismos.

Escolha uma **operação** que seja **determinante** para o desempenho computacional do seu algoritmo.

Verifique experimentalmente o **número de vezes que essa operação é executada**.

Confirme os seus resultados consultando a sequência A014080 na OEIS https://oeis.org/A014080

4. Um **número de Armstrong**, para uma dada base, é um número inteiro positivo de **n algarismos** que é igual à **soma** de cada um dos seus **algarismos** levantado à **n-ésima potência**.

Escreva um programa eficiente que lhe permita listar, para a base 10, todos os **números de Armstrong de 3 algarismos**.

SUGESTÃO: Armazene num *array* as potências dos sucessivos algarismos.

OPCIONAL: Generalize o seu algoritmo, de modo a poder listar números de Armstrong com um maior número de algarismos.

Escolha uma **operação** que seja **determinante** para o desempenho computacional do seu algoritmo.

Verifique experimentalmente o **número de vezes que essa operação é executada**.

Confirme os seus resultados consultando a sequência A005188 na OEIS https://oeis.org/A005188

**** Exercício Extra ****

5. O código do ficheiro **`examples.c`** permite realizar testes computacionais sobre conjuntos de algoritmos com diferentes ordens de complexidade e apresenta tabelas de resultados para cada um desses algoritmos.

Analise o modo como são efetuados os **testes computacionais**, gerando **instâncias** de teste **aleatórias**, e construídas as **tabelas de resultados**.

Escolha alguns algoritmos computacionalmente mais exigentes e efetue alguns testes computacionais.

Quais são as **maiores instâncias** de teste que consegue resolver, **em tempo útil**, no seu computador?