UNIVERSIDADE DE ÉVORA

# 1º Trabalho - Cod Fishing

Estrutura de Dados e Algoritmos II

**Professor:** Vasco Pedro
**Grupo:** G107
**Realizado por:** Filipe Alfaiate (43315), Miguel de Carvalho (43108)

22 de abril de 2021

## 1 Algoritmo

O nosso algoritmo consiste na utilização de uma **matriz** para conseguirmos processar os dados do problema. A `matriz` é construída com o objetivo de guardar todas as melhores atribuições de todos os **locais de pesca** com todos os **barcos**, obtendo na última posição da `matriz` o melhor caso que solucione o problema.

## 2 Complexidade

### 2.1 Temporal

Começámos por proceder à leitura de uma `string` com dois números e ambos os números foram associados a variavéis diferentes, `nBarcos` e `nPeixes`. Este acesso é direto pois na primeiro posição da string é o **número de barcos** e na segunda posiçõ é o **número de peixes**, o que origina `O(1)`.

Em seguida, procedeu-se à leitura e organização dos dados dos **barcos** consoante o `nBarcos`. Por isso estamos perante uma complexidade `O(n)`, onde `n` é o **número de barcos**.

Seguidamente ordenámos o `array` de **barcos**, este tem um complexidade `O(n)` em que `n` é a **quantidade de elementos do array barcos**.

Depois procedeu-se exatamente ao mesmo processo, mas desta vez para os peixes, tendo uma complexidade `O(m)` onde `m` é a **quantidade de locais de peixe/quantidade de elementos do array peixes**.

Além disso, criámos uma matriz que tem custo constante `O(1)`.

Finalmente, executámos o nosso **algoritmo**. Durante a execução estamos perante dois **ciclos**, um interior e outro exterior. No ciclo interior existe **duas condições** de custo constante `O(1)`, mas o ciclo tem um custo linear de `O(n)` onde `n` é a **quantidade de barcos**. No ciclo exterior é apenas realizado o ciclo interior, o que para cada iteração temos uma complexidade `O(m)`, então o ciclo vai ter um complexidade O(m x n)=O($n^2$) onde `n` é o **número de peixes**.

Logo, a **complexidade do programa** será de

$$\mathbf{O(1) + O(n) + O(n) + O(m) + O(m) + ((O(1) \text{ x } O(1)) \text{ x } O(n)) \text{ x } O(m) = O(}n^2\mathbf{)}$$

### 2.2 Espacial

Durante a inicialização dos `arrays` para guardar os **barcos** e os **locais de pesca** é necessário saber quantos **barcos** e **locais de pesca** existem inicialmente, por outras palavras ocupam em memória um número linear, que depende do **número de barcos** e do **número de peixes**, originando assim uma complexidade `O(n)`.

Na inicialização da `matriz` é fundamental existir o número de **locais de pesca** e de **barcos** para determinar o espaço que a `matriz` deverá ocupar em **memória**. Neste caso iremos ocupar em memória um número exponencial, pois tem que se multiplicar o **número de locais de pesca** (m) pelo **número de barcos** (n), originando assim O(m x n) = O($n^2$).

Podemos dizer então que a **complexidade espacial** do programa será de

$$\mathbf{O(n)} + \mathbf{O}(n^2) = \mathbf{O}(n^2)$$

## 3 Comentários Adicionais

Apesar deste trabalho apresentar um **algoritmo** que funciona de **maneira iterative**, inicialmente começámos por densenvolver com um **pensamento recursivo** o que originou alguns **Timeouts** (excedeu o tempo limite estipulado pelo professor) e diversos **Wrong Answers**.

Devido ao facto de obtermos muitas respostas erradas e termos-nos reunido com o Professor, recomeçámos a pensar numa abordagem diferente e seguindo as indicações do professor, iniciámos pelo caso do mesmo **número de barcos** com o mesmo **número de peixes** e como era algo simples decidimos implementar com um ciclo. Como esta abordagem nunca tinha sido testada por nós, decidimos alterar a **forma recursiva** para uma **forma iterativa**, solucionando assim o problema.