Algoritmos e Complexidade
LEI/LCC (2º ano)

4ª Ficha Prática

O objectivo desta ficha é a análise informal da complexidade de programas muito simples.

Considere o seguinte *header file* para **buffers** de inteiros.

```
typedef struct buffer *Buffer;

Buffer init (void); // inicia e aloca espaço
int empty (Buffer); // testa se está vazio
int add (Buffer,int); // acrescenta elemento
int next (Buffer, *int); // próximo a saír
int remove (Buffer, *int); // remove próximo
```

1. Uma instanciação deste conceito de *buffer* são *stacks*. Neste caso, o próximo elemento a sair é o último que foi acrescentado (*Last In First Out*).

   Apresente uma implementação de *Stacks* em que todas as operações executem em tempo constante (i.e., independente do número de elementos que estão na *stack*).

2. Uma outra instanciação do conceito de *buffer* são *queues*. Neste caso, o próximo elemento a sair é o primeiro que foi acrescentado (*First In First Out*).

   Apresente uma implementação de *Queues* em que todas as operações executem em tempo constante (i.e., independente do número de elementos que estão na *queue*).

3. Considere agora uma instanciação de *buffer* em que o próximo elemento a sair é o menor elemento que se encontra no *buffer*. E para este caso vamos considerar 3 implementações possíveis:

   A os elementos do *buffer* são armazenados sequencialmente por ordem crescente;

   B os elementos do *buffer* são armazenados por ordem de chegada;

   C os elementos do *buffer* são armazenados numa *heap*;

   Para cada uma das implementações sugeridas

   (a) analise (informalmente) a complexidade das funções de inserção e remoção no melhor e pior casos (identifique esses casos).

   (b) Considere agora uma sequência de `N` instruções de inserção e remoção que, partindo do *buffer* vazio acabam com o *buffer* também vazio (e por isso mesmo a sequência tem de ter tantas remoções como inserções). Identifique a melhor e a pior destas sequências e calcule, para cada uma destas, o custo da sequência.