Algoritmos e Estruturas de Dados
Exercícios das Aulas Práticas

Departamento de Informática
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa

Ano Letivo 2023/2024

1. Uma instituição bancária necessita de uma nova aplicação de gestão de cartões de crédito. Essa aplicação deve permitir registar electronicamente os movimentos (de débito ou de crédito) de um cartão, fornecer a lista dos movimentos (não saldados) de um cartão, e obter e liquidar o saldo de um cartão.

   Os dados correspondentes a cada *cartão* de crédito são o *número* do cartão e o *nome* do detentor do cartão. É possível registar *movimentos* de cartões. Um *movimento* de um cartão é identificado pela *data do movimento* (até aos segundos) e possui um *tipo* (débito ou crédito) e um *montante* (o valor do débito ou do crédito). Considera-se que o *saldo* de um cartão é a soma dos montantes dos movimentos do cartão presentes no sistema. *Liquidar o saldo* de um cartão é pagar ou receber esse valor, removendo os respectivos movimentos do sistema e anulando o saldo do cartão.

   A aplicação deve permitir efectuar as seguintes operações.

   (a) Obter o número de um novo cartão, com o nome do detentor. Assuma que o número (único) é gerado internamente pelo sistema, por um processo que iremos ignorar neste contexto.

   (b) Obter o nome do detentor de um cartão, dado o seu número.

   (c) Inserir um novo movimento, dados o número do cartão, e a data, o tipo e o montante do movimento.

   (d) Listar a data, o tipo e o montante dos movimentos de um cartão, dado o seu número. A listagem deve estar ordenada cronologicamente por data.

   (e) Obter o saldo de um cartão, dado o seu número.

   (f) Liquidar o saldo de um cartão, dado o seu número.

   Elabore o **Modelo dos Tipos Abstractos de Dados (TAD)** deste problema. Esse modelo deve conter a seguinte informação.

   - Uma definição dos tipos abstractos de dados necessários para resolver o problema, incluindo a lista e descrição de todas as operações públicas a disponibilizar pelos TAD.
   - Uma enumeração dos atributos (e dos respectivos tipos de dados) das entidades cujo tipo foi definido no ponto anterior.

2. Elabore o **Modelo das Estruturas de Dados** do problema introduzido na pergunta 1. Esse modelo deve conter a seguinte informação:

   (a) A definição completa das estruturas de dados, indicando o que são e que informação guardam.

   (b) Para cada uma das operações do enunciado, uma descrição do comportamento do programa em termos das operações efectuadas sobre as estruturas de dados.

   (c) Uma justificação para terem escolhido aquelas estruturas de dados e não outras.

   (d) O estudo das complexidades temporais das operações do enunciado, no melhor caso, no pior caso e no caso esperado.

   (e) O estudo da complexidade espacial da solução proposta.

   Neste exercício, assuma que a única estrutura de dados base disponível (para além dos tipos básicos) é o vector.

3. Pedimos-lhe que considere a implementação de um Projeto de Revista Online, pensando no Sistema de Gestão de conteúdos. Este sistema irá gerir os autores dos textos publicados na revista, assim como a informação associada aos próprios textos. Os textos estarão disponíveis num repositório pelo que, para se ler um determinado texto, utilizamos o seu URL. Assim, o sistema irá fazer a gestão dos Autores dos textos, cuja informação inclui Código de autor, Nome de autor, Email e Número de telefone. Relativamente aos *Textos*, estes têm um Código de Texto, Título, um Autor e um URL. Assuma que não existem nomes de Autores repetidos e que os títulos dos textos de um autor são únicos. O Sistema deverá implementar as seguintes operações:

   (a) *Inserir um Autor*, recebendo a seguinte informação: Código de Autor, Nome de Autor, Email e Número de telefone. A inserção só tem sucesso se o código do autor ainda não existir no sistema;

   (b) *Inserir um Texto*, que é composto de: Código de Texto, Título, Código de Autor do Texto e URL. A Inserção só tem sucesso se o código de autor já existir no sistema e se o código do texto não existir no sistema;

   (c) *Aceder ao URL de um texto*, dando o código do texto. Esta operação só tem sucesso se o código do texto existir no Sistema;

   (d) *Remover Texto*, dando o código do texto. Esta operação só tem sucesso se o código do texto existir no sistema;

   (e) *Listar os Títulos de todos os textos de um determinado autor*, dando o **nome do autor**. A listagem deve ser apresentada por ordem alfabética do título do texto. Esta operação só tem sucesso se o autor em causa fizer parte do sistema.

   Elabore o **Modelo dos Tipos Abstractos de Dados (TAD)** deste problema. Esse modelo deve conter a seguinte informação:

   - Uma definição dos tipos abstractos de dados necessários para resolver o problema, incluindo a lista e descrição de todas as operações públicas a disponibilizar pelos TAD.
   - Uma enumeração dos atributos (e dos respectivos tipos de dados) das entidades cujo tipo foi definido no ponto anterior.

4. Elabore o **Modelo das Estruturas de Dados** do problema introduzido na pergunta 3. Esse modelo deve conter a seguinte informação.

   (a) A definição completa das estruturas de dados, indicando o que são e que informação guardam.

   (b) Para cada uma das operações do enunciado, uma descrição do comportamento do programa em termos das operações efectuadas sobre as estruturas de dados.

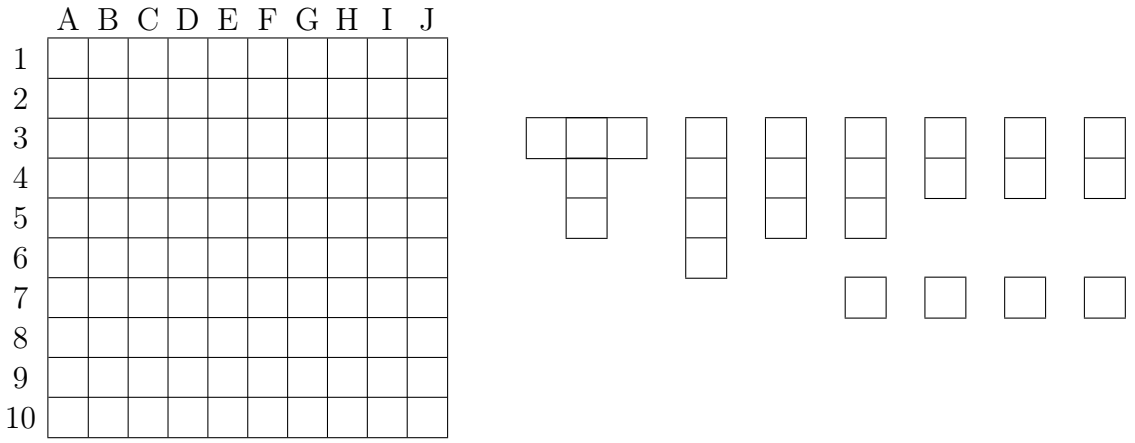
   (c) Uma justificação para terem escolhido aquelas estruturas de dados e não outras.

   (d) O estudo das complexidades temporais das operações do enunciado, no melhor caso, no pior caso e no caso esperado.

   (e) O estudo da complexidade espacial da solução proposta.

   Neste exercício, assuma que a única estrutura de dados base disponível (para além dos tipos básicos) é o vector.

5. O jogo da Batalha Naval simula um ataque a uma frota de barcos. O jogador deverá afundar toda a frota inimiga, que é composta por um porta-aviões, um barco de 4 canos, dois barcos de 3 canos, três barcos de 2 canos e quatro submarinos (barcos de 1 cano). O porta-aviões ocupa "o espaço de um T" e os barcos de n canos ocupam o espaço de n quadrados, em fila ou em coluna (como se ilustra na figura). Os barcos estão posicionados numa grelha quadrada (que, no jogo habitual é de dez por dez) e não se tocam, nem sequer na diagonal.



   O objectivo do *jogo singular* da batalha naval é afundar a frota inimiga, através de uma sequência de rajadas. Cada *rajada* é constituída por três tiros. Após cada rajada, sabe-se que tipos de barcos foram atingidos e (se for o caso) afundados. Um barco diz-se *afundado* quando todas as posições que ocupa foram atingidas por um tiro. Um *tiro* é composto por um par (fila,coluna), que identifica uma posição na grelha. O jogo termina quando todos os barcos forem afundados.

   Pretende-se implementar o jogo singular da batalha naval, quando a configuração da frota é obtida por um processo que iremos ignorar neste contexto.

   O programa interpreta os seguintes comandos.

   - **r x1 y1 x2 y2 x3 y3**: envio de uma rajada constituída por três tiros (o primeiro tiro é dado pelo par (x1,y1) e assim por diante).

     O programa fornece a seguinte informação, respeitante aos **tiros da rajada**.

1. Número de tiros *inválidos*.
   Um tiro é *válido* se corresponder a uma posição da grelha.
2. Número de tiros (válidos e) repetidos.
   Por exemplo, esse número é 1 na rajada (4,5), (2,3), (4,5).
3. Número de tiros (válidos, distintos e) *não originais*.
   Um tiro é *original* se atinge uma posição que ainda não tinha sido atingida.
4. Número de tiros (originais) que atingiram submarinos.
5. Número de tiros (originais) que atingiram barcos de 2 canos.
6. Número de tiros (originais) que atingiram barcos de 3 canos.
7. Número de tiros (originais) que atingiram o barco de 4 canos.
8. Número de tiros (originais) que atingiram o porta-aviões.
9. Número de submarinos afundados (nesta rajada).
10. Número de barcos de 2 canos afundados (nesta rajada).
11. Número de barcos de 3 canos afundados (nesta rajada).
12. Número de barcos de 4 canos afundados (nesta rajada).
13. Número de porta-aviões afundados (nesta rajada).
14. Indicação de que o jogo continua (porque ainda há barcos por afundar) ou terminou.
15. Número total de rajadas enviadas durante o jogo (que permite calcular a pontuação obtida pelo jogador), apenas no caso do jogo ter terminado.

- **d**: o jogo termina por desistência do jogador.
- **g**: apresenta-se a grelha, assinalando as posições atingidas pelos tiros dados até ao momento.
- **?**: afixa-se um texto explicando os comandos relacionados com o jogo.

Elabore o **Modelo dos Tipos Abstractos de Dados** deste problema.

6. Considere o tipo abstracto de dados *Fila Concatenável* de elementos do tipo E, caracterizado pela interface *ConcatenableQueue*.

```
public interface ConcatenableQueue<E> extends Queue<E>
{
        // Removes all of the elements from the specified queue and
        // inserts them at the end of the queue (in proper order).
        void append( ConcatenableQueue<E> queue );
}
```

Por exemplo, se q_1 e q_2 forem filas concatenáveis com os elementos

$$5,\ 44,\ 17,\ 10 \quad \text{e} \quad 11,\ 17,\ 50,$$

após $q_1.append(q_2)$, q_1 fica com 5, 44, 17, 10, 11, 17, 50 e q_2 fica vazia.

(a) Implemente este TAD.

(b) Calcule as complexidades temporais das operações, no melhor caso, no pior caso e no caso esperado.

7. Considere o tipo abstracto de dados *Fila Invertível* de elementos do tipo E, caracterizado pela interface *InvertibleQueue*.

```
public interface InvertibleQueue<E> extends Queue<E>
{
    // Puts all elements in the queue in the opposite order.
    void invert( );
}

public interface Queue<E>
{
    // Returns true iff the queue contains no elements.
    boolean isEmpty( );

    // Returns the number of elements in the queue.
    int size( );

    // Inserts the specified element at the rear of the queue.
    void enqueue( E element );

    // Removes and returns the element at the front of the queue.
    E dequeue( ) throws EmptyQueueException;
}
```

Por exemplo, se q for uma fila invertível com os elementos

$$5,\ 44,\ 17,\ 10,\ 11,\ 17,\ 50,$$

após $q.invert()$ e $q.enqueue(3)$, q fica com 50, 17, 11, 10, 17, 44, 5, 3.

(a) Implemente este TAD.

(b) Calcule as complexidades temporais das operações, no melhor caso, no pior caso e no caso esperado.