Algoritmos e Estruturas de Dados

Departamento de Informática, Universidade Nova de Lisboa

Segundo teste - 16 de Dezembro de 2014

Atenção: Os Anexos ao teste poderão ser-lhe úteis.

1. Considere a árvore AVL apresentada na Figura 1. Em cada nó está apenas representada a chave do mesmo.

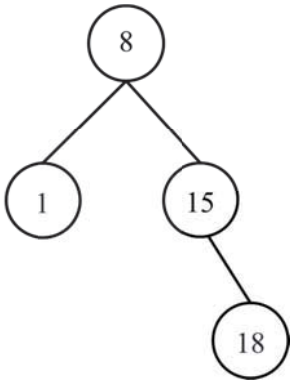

Figura 1

   a) Desenhe a árvore AVL resultante de inserir um elemento com a chave 16 na árvore apresentada;
   b) Desenhe a árvore AVL resultante de remover o elemento com a chave 8 da árvore apresentada na Figura 1. Atenção que a remoção deve ser feita na **árvore original,** representada na Figura.

2. Considere o Tipo Abstrato de Dado `Player` representado pelo interface abaixo. O interface `Player` permite a consulta de informação relativa a um jogador de Basquetebol (nome, ano de nascimento, média de cestos marcados e convocação para o próximo jogo).

```
public interface Player {

   //Devolve o nome do jogador.
   String getName();

   //Devolve o ano de nascimento do jogador.
   int getYearOfBirth();

   //Devolve a média de cestos marcados pelo jogador.
   double getAverage();

   //Devolve um valor boolean que determina se o jogador foi convocado para o
   //próximo jogo.
   boolean getConvened();

}
```

Dado este interface, implemente o iterador ConvenedIterator, um iterador de objetos do tipo Player que recebe, no construtor, um iterador de objetos do mesmo tipo, em qualquer ordem, e que implementa a iteração de todos os jogadores da equipa, iterando primeiros os jogadores convocados para o próximo jogo. A ordem de iteração, dentro dos jogadores convocados e não convocados não é relevante.

Defina as variáveis de instância do iterador e implemente todos métodos auxiliares de que necessitar para completar a classe. Para cada método apresentado calcule (justificando) as suas complexidades temporais, no pior caso. Para este efeito pode considerar que os jogadores associados ao iterador de objetos Player, submetido no construtor do iterador a desenvolver, estão guardados em Lista duplamente ligada (DoublyLinkedList<Player>).

```
public class ConvenedIterator implements Iterator<Player> {

   //Cria um iterador dos jogadores contidos no iterador it. Este iterador
   //deverá iterar primeiro os jogadores convocados para o próximo jogo e
   //só depois os restantes.
   public ConvenedIterator(Iterator<Player> it);

   //Inicializa a iteração.
   public void rewind();

   //Devolve true se a iteração ainda não tiver chegado ao fim.
   public boolean hasNext();

   //Devolve o próximo jogador na iteração, se esta não tiver chegado ao fim.
   //Requires: hasNext()
   public Player next() throws NoSuchElementException;
}
```

3. Pretendemos desenvolver um Sistema de gestão de pagamentos mensais de um Liceu Privado. O Liceu gere os pagamentos com base nas famílias cujos filhos frequentam o liceu. O pagamento é efetuado por família, e quando a família inscreve os seus jovens no liceu, tem direito a desconto de 10% a partir do segundo filho inscrito. O valor base a pagar individualmente por cada aluno pode variar ao longo do ano e cada pagamento é efetuado por família, ou seja, engloba o pagamento das propinas de todos os alunos pertencentes à família. Desta forma, a informação associada à família é: código identificador (único) da família, número de filhos inscritos, nome do encarregado de educação e informação sobre se o pagamento mensal já foi regularizado. Relativamente aos dados do aluno, estes contêm: número de aluno (único no sistema), nome do aluno, ciclo em que se inscreve, valor mensal de base a pagar e família a que o aluno pertence.
   Importante: O sistema só gere os pagamentos de um mês. Quando um novo mês se inicia são arquivados os dados relativos ao mês anterior e o sistema é reinicializado.
   O sistema deverá permitir as seguintes operações:

   a) *Inserir família*, recebendo a seguinte informação: código de família e nome do encarregado de educação. A inserção só tem sucesso se o código de família ainda não existir no sistema. Pode assumir que o nome do encarregado de educação é único;

b) *Inserir um novo Aluno*, recebendo: número de aluno, nome, ciclo, valor mensal base e família a que o aluno pertence. A Inserção só tem sucesso se a família já existe no sistema e se o número de aluno ainda não existe. O sistema deve ainda validar se o nome do aluno não existe ainda na família em causa, ou seja o nome do aluno deve ser único dentro da família;
c) *Consultar valor mensal a pagar,* dando o código da família. Esta operação só terá sucesso se o código da família existir no sistema e deve calcular o valor mensal a pagar pela família, tendo em conta os descontos aplicáveis;
d) *Efetuar pagamento*, dando o código da família. Esta operação só terá sucesso se o código da família existir no sistema, se pelo menos um filho estiver inscrito no liceu e se o pagamento do mês a decorrer não tiver ainda sido regularizado;
e) *Alterar valor mensal de base a pagar,* dando o número do aluno e o código da família. Esta operação só tem sucesso se o número do aluno e o código da família existirem no sistema e se o pagamento mensal da família ainda não tiver sido efetuado;
f) *Remover aluno,* dando o código da família e o número do aluno. Para a operação ter sucesso, o código da família e o número do aluno têm de existir no sistema. Se o aluno for o único filho da família inscrito no liceu, a família deverá também ser removida;
g) *Listar pagamentos não efetuados,* ordenados alfabeticamente pelo nome do encarregado de educação. Esta operação deverá listar, para cada família que ainda não efetuou o pagamento mensal, o nome do encarregado de educação e o código da família.

Espera-se que o número de alunos e de famílias seja da ordem dos milhares. Com base nesta especificação:

- Explicite detalhadamente as estruturas de dados e as variáveis de instância mais adequadas para implementar esta aplicação;
- Descreva sumariamente os algoritmos para efetuar as sete operações (enumeradas de (a) a (g)) e calcule (justificando) as suas complexidades temporais, no caso esperado.

Não deve desenvolver código em java, apenas fazer uma descrição da implementação das operações de acordo com a sua escolha de estruturas de dados.