u évora
ESCOLA DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

EINF MA CID

Disciplina: Estruturas de Dados e Algoritmos I-2015/2016 Prova: Exame 1 (7-01-2016)

Esta prova tem a duração de **3 horas** e é **sem consulta**.

1. É-lhe dada uma stack que pretende ordenar(o maior valor no topo da stack). As únicas operações que tem permissão para usar são:

   - `boolean empty(Stack<T> s)}{`
     `return s.isEmpty();`
     `}`
   - `void popPush(Stack<T> s1,Stack<T> s2){`
     `s2.push(s1.pop());`
     `}`
   - **boolean LT(Stack<T> s1,Stack<T> s2)** que retorna true se o elemento no topo de s1 é menor que o elemento no topo de s2.

   Apresente um algoritmo de ordenação que utilize exclusivamente stacks e estas 3 operações para manipular as stacks.

2. Implemente o método *entremeada(Queue<T> q1, Queue<T> q2)* que recebe como parâmetros duas `queues` de elementos do mesmo tipo e retorna uma única queue, cujos elementos são a fusão das duas queues mas com os elementos na sequência, 1º elemento da 1º queue, 1º elemento da segunda queue, 2º elemento da 1ª queue, 2º elemento da segunda queue, etc. Não pode usar na resolução deste exercício nenhum tipo de estruturas auxiliares. Por exemplo se $q1 = [2, 17, 8, 21]$ e $q2 = [10, 5, 1, 4]$ então *entremeada(q1,q2)* deve retornar $[2, 10, 17, 5, 8, 1, 21, 4]$

3. Implemente o método `void crescenteParaDecrescente(Node<T> inicio)` que converte a lista simplesmente ligada com início no nó `inicio` e cujos nós se apresentam por ordem crescente e a converte numa lista com os mesmos nós mas agora por ordem decrescente. Não pode criar novos nós nem usar quaisquer estruturas auxiliares.

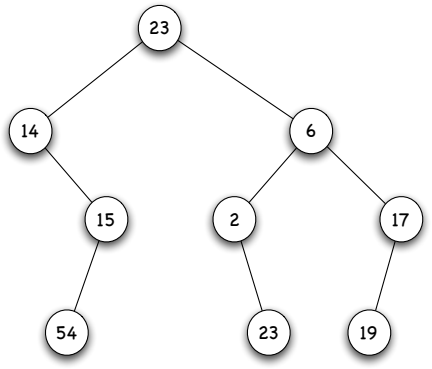

Figure 1: árvore binária

4. Implemente o método `void listaNosNivel( Btree<T> t)` que liste todos os nós de `t` por níveis. Por exemplo o método listaNosNivel para a árvore da figura 1 deverá produzir o seguinte output:

```
(23)
(14,6)
(-,15) (2,17)
(54,-) (-,23) (19,-)
```

5. Apresente as árvores AVL, resultantes de inserir numa árvore inicialmente vazia, a sequência de valores 18;45;20;15;8;17;22;9;10;13;16 Em caso de desequilíbrio indique o nó correspondente, o caso da rotação e se a resolução passa por uma rotação simples ou dupla

6. Desenhe numa tabela de hash de tamanho 11 ($N = 11$), os resultados de inserir as chaves: 7; 66; 12; 40; 19; 18; 14; 20; 4; 23; tomando para função de hash $h(x) = (2x + 1)\ mod\ N$ e usando como revolvedor de colisões um duplo hashing, sendo a 2ª função de hash $h'(x) = 7 - (x\ mod\ 7)$

   (a) Suponha que uma tabela de hash de capacidade 37, tem um factor de carga de 3/4. Suponha que se pretende fazer um "rehasing" e obter um factor de carga abaixo dos 40%. Indique, justificando a capacidade para a nova tabela.

7. Considere o array da figura 2. Ordene-o usando os seguintes métodos:

   (a) *Heapsort*: apresente todas as iterações do método, e as respectivas heaps

   (b) *Mergesort*: apresente todas as iterações do método

   (c) *Quicksort*: Apresente duas iterações do método; a primeira iteração sobre o array todo e até à obtenção da primeira partição usando como pivot o elemento na posição índice 1, a segunda iteração sobre partição esquerda obtida da 1ª iteração e tomando como

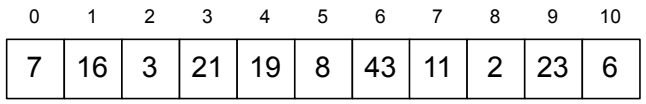

Figure 2: array de valores

pivot um elemento à sua escolha. Indique o índice do elemento escolhido para pivot, e indique claramente as partições obtidas em ambas as iterações.