

EINF MA CID

Disciplina: Estruturas de Dados e Algoritmos I-2019/2020

Prova: mini-teste 4 (17-12-2019)

Esta prova tem a duração de **1 hora** e é **sem consulta**. Identifique TODAS as folhas de teste.

1. Considere o array:

0	1	2	3	4	5	6	7	8	9	10
3	19	26	20	32	22	21	10	25	23	24

- (a) Insira numa AVL vazia os elementos do array desde o índice 0 ao índice 10 e por esta ordem. Após cada inserção deve indicar se está tudo (OK!), caso em que a árvore está correctamente balanceada, ou se há necessidade de restaurar o equilíbrio, caso em que deve indicar o nó em desequilíbrio, o caso e a(s) rotações(RSD, RSE, RDDE, RDDE) que vai realizar de modo a restaurar o equilíbrio. Por exemplo:  
insere(10)  
OK  
insere(12)  
OK  
insere(11)  
desequilíbrio(10); caso 3; RDDE  
....
- (b) Desenhe somente as árvores obtidas no exercício anterior, antes e depois das rotações. Todas as inserções que não causem desequilíbrio pode ser desenhadas na mesma árvore.

2. Considere o array

0	1	2	3	4	5	6	7	8	9
6	4	18	33	9	17	5	29	3	101

Desenhe numa tabela de hash de tamanho 11 ( $N = 11$ ), os resultados de inserir as chaves do array da figura tomando para função de hash,  $f(x) = (2 * (x \bmod 5)) + 1$  e usando:

- (a) Hasinhg fechado com acesso quadrático  
(b) Hashing fechado com duplo hashing, usando como segunda função de hash,  $g(x) = 7 - (x \bmod 7)$   
(c) Teça considerações sobre a 1ª função de hash,  $f(x)$

3. Considere o array:

0	1	2	3	4	5	6	7	8	9	10	11
4	12	1	33	56	20	17	71	6	19	15	40

- (a) Apresente a maxHeap correspondente ao array da figura
- (b) Apresente as primeiras três(3) iterações do **Heapsort**
- (c) Apresente uma iteração do **Quicksort**, especificando no final da iteração qual o índice da partição; e como fica o array após a iteração. Tome para pivot o elemento na posição 5.
- (d) Implemente o método **static void ordena(Comparable[] arr)**, que recebe um array de Comparables e o ordena, por ordem crescente. Apresente a complexidade do método implementado.