

# MAC0323 Algoritmos de Estruturas de Dados II

Primeira Prova – 10 de abril de 2018

Nome: \_\_\_\_\_

Assinatura: \_\_\_\_\_

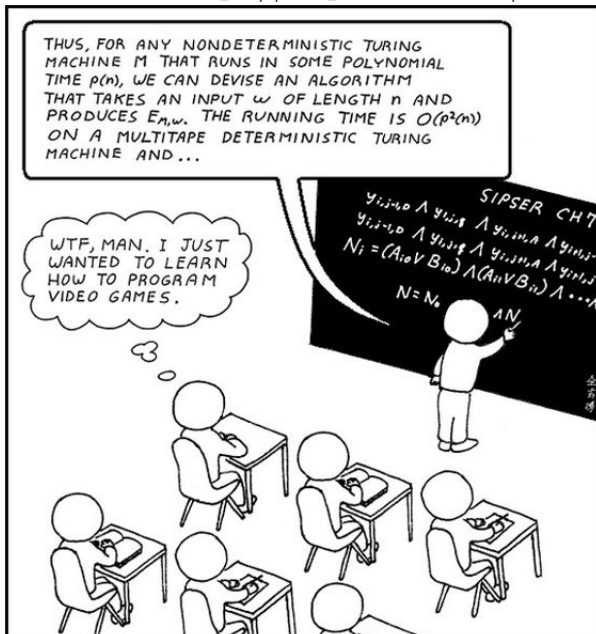
Nº USP: \_\_\_\_\_

## Instruções:

1. Não destaque as folhas deste caderno. A prova pode ser feita a lápis.
2. A prova consta de 6 questões; Verifique antes de começar a prova se o seu caderno está completo.
3. Cuidado com a legibilidade.
4. Não é permitido o uso de folhas avulsas para rascunho, a consulta a livros, apontamentos, colegas ou equipamentos eletrônicos. Desligue o seu celular e qualquer equipamento que possa perturbar o andamento da prova.

**DURAÇÃO DA PROVA: 100 minutos**

Fonte: <https://br.pinterest.com/>



Questão	Valor	Nota
1	2,0	
2	1,0	
3	2,0	
4	2,5	
5	1,5	
6	1,0	
Total	10,0	

1. Union find (vale 2,0 pontos)

Circule as letras correspondente a configuração de vetores `id[]` (`=pai[]`) que **não podem** ocorrer durante a execução de uma sequência de operações `find()` e `union()` que utilizam o algoritmo *weighted quick union*.

	i:	0	1	2	3	4	5	6	7	8	9
A.	id[i]:	9	0	0	0	0	0	9	9	9	9
B.	id[i]:	8	0	4	0	0	4	0	4	2	0
C.	id[i]:	4	1	8	2	1	5	1	1	4	5
D.	id[i]:	3	3	6	9	3	6	3	4	1	9
E.	id[i]:	2	1	1	1	1	1	1	2	1	7

## 2. Análise de algoritmos (vale 1,0 ponto)

Foram implementados três algoritmos: algoritmo A, B e C. As tabelas abaixo mostram o resultado de um estudo experimental dessas implementações. Nas tabelas, para alguns valores do tamanho  $n$  da entrada do problema são indicados os consumos de tempo dos algoritmos, em segundos.

$n$	algoritmo A
256	0.00
512	0.01
1024	0.03
2048	0.09
4096	0.37
8192	1.70
16384	7.08
32768	28.54
65536	113.55
131072	493.08

$n$	algoritmo B
256	0.00
512	0.00
1024	0.00
2048	0.00
4096	0.01
8192	0.01
16384	0.02
32768	0.04
65536	0.10
131072	0.23
262144	0.54
524288	1.31
1048576	3.06
2097152	7.07
4194304	15.84
8388608	35.85

$n$	algoritmo C
1000000	0.06
2000000	0.11
3000000	0.16
6000000	0.32
8000000	0.43
10000000	0.55
15000000	0.82
20000000	1.08
25000000	1.37
30000000	1.70

Apresente funções  $A(n)$ ,  $B(n)$ , e  $C(n)$  para as quais você suspeita que o algoritmos A, B e C consomem tempo proporcional a  $A(n)$ ,  $B(n)$ , e  $C(n)$ , respectivamente. Justifique a sua resposta em no máximo uma ou duas linhas para cada função.

### 3. Estruturas de dados (vale 2,0 pontos)

Suponha que a biblioteca `java.util.LinkedList` é implementada usando uma lista duplamente ligada e mantém uma referência para o primeiro (`=first`) e último (`=last`) nó da lista, junto com o seu número de itens.

```
public class LinkedList<Item> {
    private Node first; // primeiro nó da lista
    private Node last; // último nó da lista
    private int n;

    private class Node {
        private Item item; // o item
        private Node next; // referência para o próximo nó
        private Node prev; // referência para o nó anterior
    }
    [...]
}
```

(a) Quantos bytes de memória um objeto `Node` usa e quantos bytes um objeto `LinkedList` usa para armazenar `n` itens? Não inclua a memória para os próprios itens, mas inclua a memória usada pelas referências para eles.

- Memória usada por um objeto `Node`:

- Memória usada por um objeto `LinkedList` com `n` itens:

(b) Qual é a ordem de crescimento no *pior caso* para o tempo consumido pelas operações a seguir. Escreva a melhor resposta no espaço fornecido, usando uma das seguintes possibilidades.

1       $\lg n$        $\sqrt{n}$        $n$        $n \lg n$        $n^2$

<code>addFirst(item)</code>	insere <code>item</code> no início da lista	
<code>get(i)</code>	retorna o item da posição <code>i</code> da lista	
<code>set(i, item)</code>	troca o item da posição <code>i</code> da lista por <code>item</code>	
<code>removeLast()</code>	remove e retorna o item no final da lista	
<code>contains(item)</code>	<code>item</code> está na lista?	

#### 4. Heaps ternários (vale 2,5 pontos)

Um **3-heap** é um vetor que representa uma árvore ternária. Um **max-3-heap** é um 3-heap em que cada posição representa um item que possui uma chave de um conjunto comparável e a chave de cada nó é maior ou igual a chave de cada um de seus três possíveis filhos.

(a) Considere o **max-3-heap** a seguir em que os números indicam as chaves do item em cada nó.

–	88	33	77	66	10	30	25	23	60	75	14	21	50	9	7
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Represente no vetor a seguir o resultado da operação `delMax()` no **max-3-heap** acima. Faça um círculo em cada chave que muda de posição.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

(b) Dado um índice  $k$  do vetor, quais são os índices dos possíveis três filhos de  $k$  em um **3-heap**?

(c) Qual o número máximo de comparações de `delMax()` como uma função do número  $n$  de chaves no **max-3-heap**? Faça um círculo em torno da melhor resposta. Se você desejar, pode escrever uma linha para justificar a sua resposta.

$\sim 1$      $\sim \log_2 n$      $\sim \log_3 n$      $\sim 2 \log_3 n$      $\sim 2 \log_2 n$      $\sim 3 \log_3 n$      $\sim n$

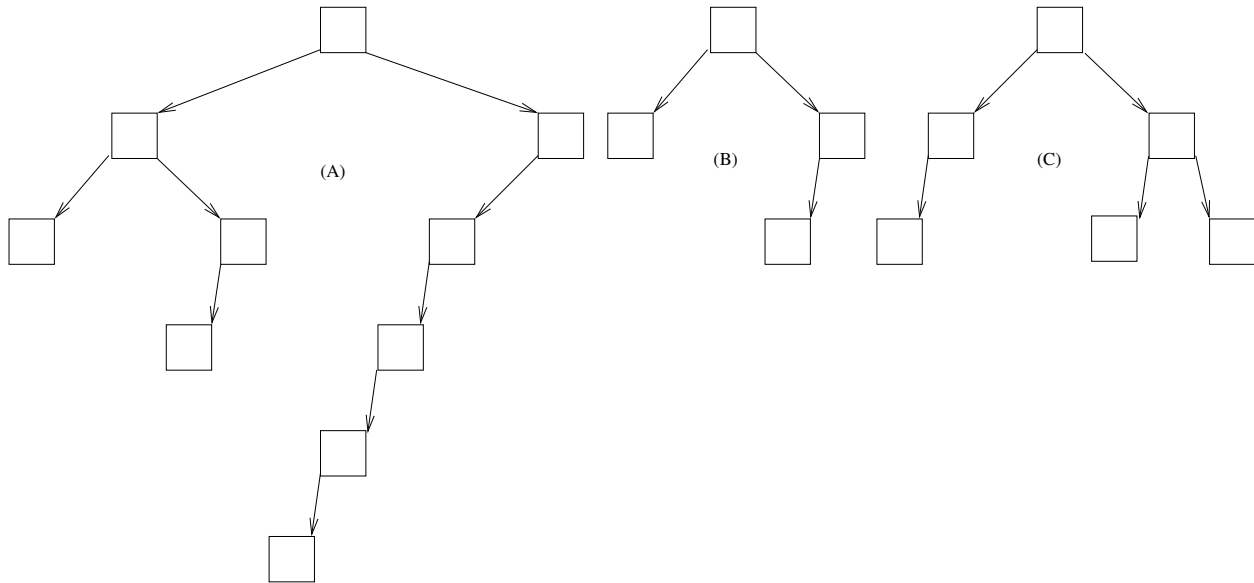
(d) Escreva um método

```
public static void sink(int p, int m, Comparable a[]) { ... }
```

que rearranja um o vetor  $a[1 \dots m]$  de tal forma que o “subvetor” cuja raiz é  $p$  seja um max-3-heap.

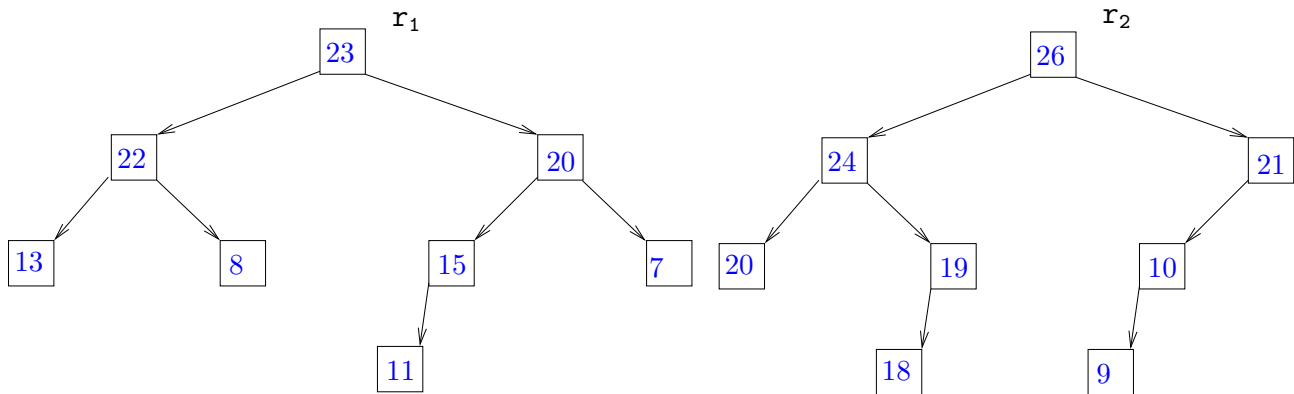
5. Leftist heaps (vale 1,5 pontos)

(a) Quais das árvores abaixo são esquerdista? Justifique a sua resposta. Você pode rabiscar sobre as figura para justificar.



(b) Qual a propriedade estrutural fundamental de árvores esquerdista para a eficiência da união de heaps esquerdistas?

(c) Desenhe o heap esquerdista resultante da **união** do dois heaps esquerdistas abaixo. Os valores nos nós representam as chaves dos itens.



**6. Skip lists** (vale 1,0 ponto)

Desenhe o skip list resultante da inclusão das chaves

9    3    6    7    19    12    17    21    26    25

em um skip list inicialmente vazio. Suponha que a sequência de níveis devolvidos pelas chamadas sucessivas do método `randLevel()` seja

1    4    1    2    1    2    1    1    3    1

Na sua ilustração destaque o caminho feito pela operação `get(21)`.