UNIVERSIDADE DO ALGARVE
FACULDADE DE CIÊNCIAS E TECNOLOGIA

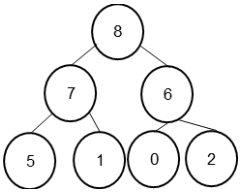
Algoritmos e Estruturas de Dados

A • Exame Época Repescagem • 14 de fevereiro de 2022 • 16:00 – 18:00

- Para perguntas com resposta de escolha múltipla, respostas erradas com cotação c e n respostas possíveis descontam −c / (n − 1).
- As respostas às perguntas de desenvolvimento deverão ser as mais detalhadas possíveis, indicando os vários passos intermédios da resolução.
- Na mesa em que está a fazer o exame deve ter apenas lápis/caneta, identificação, este enunciado, e folha de teste.
- Identifique cada folha de teste com o seu nome e número, em letra bem legível.

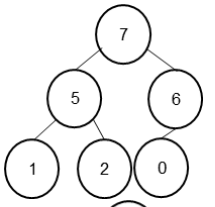
Grupo I: Perguntas de escolha múltipla

1. (0.5) Considere o seguinte *heap*:

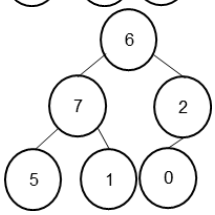


Qual dos seguintes *heaps* corresponde ao resultado de se remover o elemento 8 do *heap*?

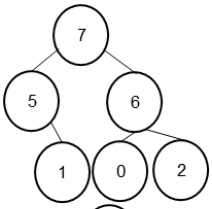
a)



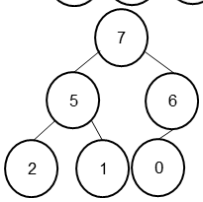
b)



c)



d)



e) Nenhum dos *heaps* apresentados corresponde ao resultado de se remover o elemento 8.

2. (0.5) Considere o algoritmo de ordenação *bubblesort* aplicado ao *array* [4, 2, 5, 0, 1, 7, 3, 8, 9, 6]. Indique qual das seguintes sequências corresponde a uma sequência válida e pela ordem correta do estado do *array* para as primeiras 3 iterações do algoritmo.

Selecione uma opção de resposta:

a)[2, 4, 0, 1, 5, 3, 7, 8, 6, 9]
[2, 0, 1, 4, 3, 5, 7, 6, 8, 9]
[0, 2, 1, 3, 4, 5, 6, 7, 8, 9]

b) [2, 4, 0, 1, 5, 3, 7, 8, 6, 9]
[2, 0, 1, 4, 3, 5, 7, 6, 8, 9]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

c)[2, 4, 0, 1, 5, 3, 6, 7, 8, 9]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

d)[2, 4, 0, 5, 1, 7, 3, 8, 6, 9]
[0, 2, 4, 5, 1, 3, 7, 8, 6, 9]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

3. (0.5) 2. (0.5) Considere o seguinte algoritmo recursivo:

```
public void xpto(int[] a, int n)
{
    if(n < 1) return;

    for(int i = 0; i < n; i++)
    {
        a[i] = func(a,n);
    }
    xpto(a,n-1);
}
```

Admita que a função func(a,n) está definida e tem complexidade O(n). Indique a complexidade da função xpto(). Selecione uma opção de resposta:

a) $O(n^3)$
b) O(n log n)
c) O(log n)
d) O(n)
e) $O(n^2)$

4. (0.5) Indique a aproximação tilde mais correta para a seguinte expressão:

$$\frac{\lg(2n)}{\lg(n)} + \frac{2n \lg(n)}{\lg(n)} + \frac{3n^2}{n}$$

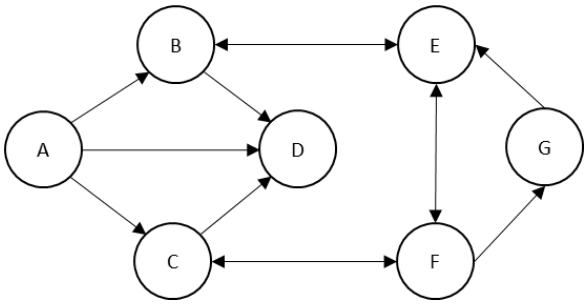
Selecione uma opção de resposta:

a) ~2
b) $\sim 2n$
c) $\sim 2n \log n$.
d) $\sim 3n$
e) $\sim 5n$

5. (0.5) Qual dos *arrays* seguintes é um *max-heap* válido? Assuma que o *heap* começa na posição 1 do *array*. Selecione uma opção de resposta:

a) [null, 7, 4, 3, 2, 5, 0, 1]
b) [null, 0, 1, 3, 2, 5, 4, 7]
c) [null, 7, 5, 3, 2, 0, 4, 1]
d) [null, 7, 4, 5, 2, 1, 3, 0]
e) nenhum dos *arrays* apresentado é um *max-heap* válido.

6. (0.5) Considere o grafo dirigido representado abaixo. Indique o número de componentes fortemente ligados do grafo.



Selecione uma opção de resposta:

a) 0
b) 1
c) 2
d) 3
e) 4

Grupo II: Perguntas de desenvolvimento

7. (2.0) Escreva um método estático chamado elementosMenores em linguagem Java, que dado um inteiro n, e um array de inteiros a, retorna um array que contem os elementos do array a inferiores a n. O array retornado deve ter o tamanho exatamente necessário para guardar apenas todos os elementos menores.

Por exemplo, se n=3, a=[1,2,3,4,5], então a função devolve [1,2]. Se n=5, e a=[8,6,5,4,3,2,1] a função devolve [4,3,2,1].

8. (2.0) Considere uma pilha (*stack*) e uma fila (*queue*) inicialmente vazias, sobre as quais é executada a seguinte sequência de instruções:

```
StackList<Integer> s = new StackList<Integer>();
QueueList<Integer> q = new QueueList<Integer>();
int j;

q.enqueue(3);
q.enqueue(5);
q.enqueue(1);


for(int i = 0; i<3; i++)
{
    s.push(q.dequeue());
    s.push(i);
    s.push(q.dequeue());
    q.enqueue(s.pop());
    q.enqueue(s.pop());
}
while(!s.isEmpty()) System.out.println(s.pop() + "+");
```

Represente a fila q e a pilha s, com os elementos introduzidos e removidos ao longo da execução do código e indique a sequência impressa no ecrã quando o código é executado.

9. (3.0) Considere a estrutura de dados *treap*, implementada no 3.º Projeto da cadeira. Um *treap* combina o funcionamento de uma árvore binária de pesquisa com um *heap*. Para um novo nó <c,p,v> a inserir, onde c é a chave do nó, v o valor do nó, e p a prioridade, um *treap* vai usar o valor de c para determinar a posição de inserção ou seleção (de acordo com as propriedades de uma árvore binária de pesquisa) , e vai usar a prioridade para garantir as propriedades de um *max-heap*, aplicando operações de rotação.

a) (2.0) Desenhe a representação em árvore dos vários estados de um *treap,* obtido quando se insere os seguintes elementos num *treap* inicialmente vazio: $<4,8,v_1>$; $<9,4,v_2>$; $<1,3,v_3>$; $<0,4,v_4>$; $<7,6,v_5>$; $<3,5,v_6>$. Desenhe pelo menos uma árvore por cada operação de rotação. Sempre que for efetuada uma operação de rotação, indique se é uma rotação para a esquerda, ou rotação para a direita.

b) (1.0) Desenha a representação em árvore dos vários estados de um *treap* obtido quando se remove o seguinte elemento do *treap* obtido na alínea anterior: <4,8,v1>. Desenhe pelo menos uma árvore por cada operação de rotação. Sempre que for efetuada uma operação de rotação, indique se é uma rotação para a esquerda, ou rotação para a direita.

10. (1.0) Explique qual a principal vantagem de uma árvore *red-black* quando comparada com uma árvore binária de pesquisa tradicional, e indique qual a propriedade de uma árvore *red-black* que faz com que esta vantagem aconteça.

11. (2.0) Considere o algoritmo de ordenação *mergesort* (na sua variante recursiva) para a ordenação do seguinte array:

a = [5, 7, 2, 10, 8, 4, 9, 3, 1, 8, 6, 1, 1, 2, 3,0]

Indique os vários passos do método *sort*, indicando como é que o algoritmo vai decompondo o problema em *sorts* recursivos cada vez mais pequenos, e indique os resultados das várias operações de *merge*.

12. (1.0) Indique, justificando, aproximadamente quantas comparações faz o algoritmo *quicksort tripartido* para ordenar um array de tamanho n em que todos os elementos são iguais?

13. (3.0) Considere uma tabela de dispersão de dimensão M = 7, com resolução de colisões por exploração linear, e função de dispersão h(k) = k mod M.

a) (2.0) Sabendo que que inicialmente a tabela se encontra vazia, indique quais os elementos presentes em cada posição do array de chaves e de valores, apos a inserção da sequência de pares <chave , valor> abaixo. **Indique o valor de h** calculado para cada chave. Indique uma posição vazia com ▬.

<7 , “a”>, <14 , “b”>, <7 , “c”>, < 8, “d”>, <9 , “e”> ,<2,”f”>, <3,”g”>

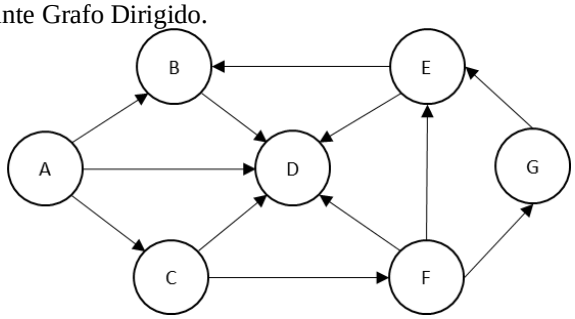
i	0	1	2	3	4	5	6
chaves							
valores							

b) (1.0) Considere agora que o conteúdo da tabela de dispersão é a seguinte:

i	0	1	2	3	4	5	6
chaves	21		2		4	11	8
valores	a		b		c	d	e

Indique o que muda nos *arrays* de chaves e valores após a remoção da chave 4 (recorde-se de que a exploração linear utiliza remoção imediata). Apenas precisa de assinalar o que é diferente. Indique uma posição vazia com ▬

14. (3.0) Considere o seguinte Grafo Dirigido.



a) (2.0) Faça uma procura em profundidade primeiro sobre o grafo, considerando que a procura se inicia a partir do vértice A. Considere que os vértices adjacentes de um vértice são visitados (ou testados) por ordem alfabética. Indique a pré-ordem, a pós-ordem e a pós-ordem invertida dos vértices calculada pela procura em profundidade primeiro.

b) (1.0) Desenhe o grafo invertido que se obtém a partir do grafo apresentado na figura.