edaf24-ex2-G1

Leia estas informações e instruções na totalidade
antes de iniciar a resolução da prova

Critérios de avaliação e cotação

-- As cotações são indicadas nas próprias questões.
-- As respostas às questões devem fazer sentido, ser coerentes e constituídas por palavras próprias do aluno. Não serão aceites transcrições ou traduções de livros e textos, incluindo textos de orientações de respostas de provas anteriores. As respostas que não respeitem estas condições serão classificadas com zero valores ou fortemente desvalorizadas.
-- Exceto indicação contrária, todas as respostas devem ser relativamente desenvolvidas e elaboradas de modo a demonstrar o raciocínio e conhecimento que leva à resposta final. A clareza do texto e da explicação também são levadas em conta na classificação das respostas. À simples indicação do resultado é atribuída a cotação zero.
-- Nas questões de escrita de programas, a sua correção tem em conta critérios de proficiência e compreensibilidade do código tais como: legibilidade, indentação, estrutura, comentários e explicação geral do seu funcionamento.

Normas a respeitar

-- Está a redigir a sua prova na WISEflow. A prova não será de consulta, exceto se existirem materiais ou recursos indicados pelo Professor neste enunciado.
-- O texto de todas as respostas deve ser introduzido pelo Editor de texto, não sendo aceites respostas escritas à mão, digitalizadas e incluídas como imagens, caso em que não serão consideradas.
-- Para expressões ou fórmulas matemáticas mais elaboradas, pode apresentá-las simplificadas desde que compreensíveis ou recorrer a um anexo tipo "Editor de fórmulas".
-- No caso de escrita de código de programas, existem 2 possibilidades: (i) No editor de texto escolher "bloco de código"; (ii) Introdução por um anexo tipo "Código".
-- No caso de figuras e diagramas é obrigatória a sua introdução por um anexo tipo "Desenho".
-- Nota: A introdução de dados via painel lateral com as ferramentas de apoio "Bloco de notas", "Notas adesivas" e "Modo de desenho" são apenas para rascunhos e não constituem parte das respostas, não sendo incluídos na submissão final da prova.
-- É permitido utilizar folhas de rascunho de papel.
-- Se fizer a prova remotamente, deve ter um comportamento em tudo semelhante à realização da prova em contexto presencial num centro de exame.
-- O(a) estudante em avaliação remota deve, durante a prova online realizada através da WISEflow, seguir as seguintes instruções:
 -- Não se pode levantar durante a prova, incluindo ir à casa de banho;
 -- Deve procurar um lugar calmo, onde possa estar sozinho, com as costas viradas para uma parede;
 -- Deve desligar o telemóvel, ou qualquer outro dispositivo informático, com o qual possa aceder à Internet;
 -- No caso de se tratar de uma prova sem consulta, deve retirar todas as folhas, livros ou fotocópias de cima da mesa onde realizará a prova, exceto se autorizado pelo docente;
 -- Durante a prova, não pode conversar com pessoas independentemente, do teor da conversa.
-- Deve reservar tempo suficiente para no final da prova efetuar a revisão das suas respostas e submeter a prova.

Votos de bom trabalho!
Paulo Shirley

Grupo I

1.1 [1] Utilizando a definição, prove que $f(n) = n + \sqrt{n + 3} + 10$ é $O(n)$.

Para n >= 1

raiz (n+3) <= raiz(n) + raiz (3) <= n + 2

f(n) = n + raiz (n+3) + 10 <= n+(n+2) +10 = 2n +12 <= 3n (para n>=12)
Portanto f(n) <= 3n para n >= n0 = 12 => f(n) pertence a O(n) com c=3

43 / 220 Word Limit

1.2.1 [0.5] Para o seguinte par de funções $f(n)$ e $g(n)$, indique (apenas uma opção) se $f(n) = O(g(n))$, $f(n) = \Omega(g(n))$, $f(n) = \Theta(g(n))$ ou nenhum dos casos. Justifique a sua resposta com base apenas na ordem de grandeza relativa dos termos das funções.

$$f(n) = n^3 + 1,\ g(n) = 3^n + n$$

Temos:
f(n) = n^3 + 1 e g(n) = 3^n + n
o termo dominante de f(n) é n^3 (polinomial)
o termo dominante de g(n) é 3^n(exponencial)

Sabemos que funções exponenciais crescem muito mais do que as polinomiais quando n tende para o infinito. Assim 3^n é maior do que n^3
Logo f(n) cresce mais lentamente que g(n) e conclui-se que:

f(n) pertence a O (g(n))

62 / 220 Word Limit

1.2.2 [0.5] Para o seguinte par de funções $f(n)$ e $g(n)$, indique (apenas uma opção) se $f(n) = O(g(n))$, $f(n) = \Omega(g(n))$, $f(n) = \Theta(g(n))$ ou nenhum dos casos. Justifique a sua resposta com base apenas na ordem de grandeza relativa dos termos das funções.

$$f(n) = n + 2^n + 1024,\ g(n) = 2^n + 10$$

Temos f(n) = n + 2^n +1024 e g(n) = 2^n + 10. O termo dominante em ambas é 2^n, pois em análise assintótica descarta-se o efeito de termos lineares ou constantes. Assim, f e g crescem á mesma taxa exponencial.

Para n >= que 10 : n<= 2^n e 1024 <= 2^n logo :

f(n) = 2^n +n + 1024 <= 3 x 2^n <= 3(2^n+10) = 3g(n), o que mostra que f(n) é igual O(g(n)).

Para n >= 4: 10 <= 2^n , assim:

g(n) =2^n +10 <= 2 x 2^n <= 2f(n) , isto é , 1/2 g(n) <= f(n), logo f(n)= OMEGA (g(n)).

Como se verifica simultaneamente f(n) = O (g(n)) e f(n)= OMEGA (g(n)) , conclui-se que f(n) = BIGTETHA(g(n)) , ou seja ambas as funções têm a mesma ordem de crescimento assintótico (exponencial 2^n).

124 / 220 Word Limit

1.3 [1] Considere a complexidade do seguinte segmento de código em termos do nº $f(n)$ de operações aritméticas realizadas na variável a. Determine a expressão de $f(n)$ e indique a sua complexidade na notação $O(.)$.

```
for(a=0,i=1; i<=n; i*=2)
  for(j=i; j<=n; ++j)
    a++;
```

O ciclo externo começa em i = 1 , e em cada iteração duplica o valor de i , logo repete-se cerca de log2n vezes. Para cada valor de i , o ciclo interno inicia em j=i e vai até n, executando aproximandamente n -i operações. Assim, quando o i é pequeno , o ciclo interno faz quase n iterações e mesmo quando i é grande o número de iterações continua da mesma ordem. No total somam-se cerca de n operações em cada uma das logn iterações do ciclo externo.

Portanto, o número total de incrementos da váriavel 'a' é da ordem de nlogn.

Concluí-se que f(n) = O(nlogn)

103 / 220 Word Limit

edaf24-ex2-G2

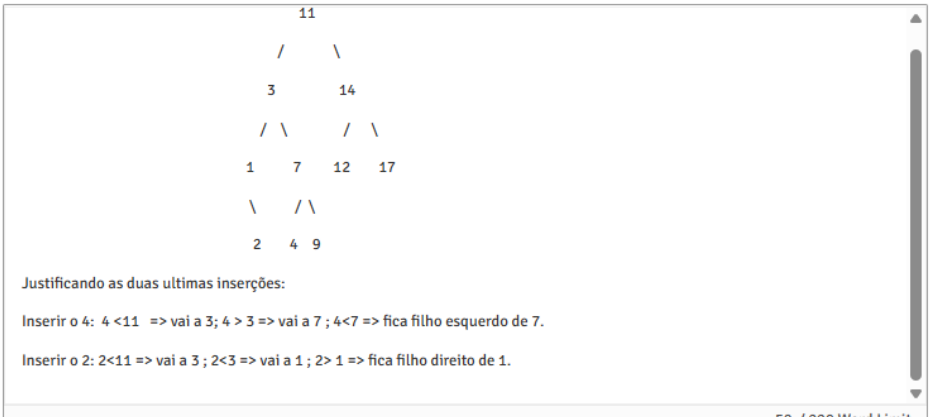
Grupo II

2.1 Considere uma árvore de promoção (Splay Tree) inicialmente vazia.

Nota: A inserção/remoção numa árvore de promoção é considerada igual à efetuada numa árvore de pesquisa binária (BST) regular.

2.1.1 [1] Insira na árvore as chaves 11, 14, 12, 3, 17, 1, 7, 9, 4, 2 pela ordem indicada. Desenhe a árvore obtida após efetuadas todas as inserções (total de 1 desenho). Justifique os passos intermédios / raciocínio apenas para as duas últimas inserções.

Nas alíneas seguintes considere a árvore obtida como a árvore "original".



Justificando as duas ultimas inserções:

Inserir o 4: 4 <11 => vai a 3; 4 > 3 => vai a 7 ; 4<7 => fica filho esquerdo de 7.

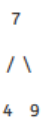
Inserir o 2: 2<11 => vai a 3 ; 2<3 => vai a 1 ; 2> 1 => fica filho direito de 1.

59 / 220 Word Limit

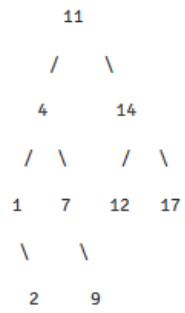
2.1.2 [1] Remova da árvore original a chave 3 utilizando o algoritmo de remoção por cópia (Deletion by Copying) com a chave sucessora. Desenhe a árvore obtida justificando os passos intermédios / raciocínio.

O nó 3 tem 2 filhos. Pelo algoritmo substituímos o seu valor pelo sucessor in-order (menor valor de subárvore direita de 3).

Sabemos que a subárvore direita de 3 é:



O menor valor é 4. Por isso copimos o valor 4 para o lugar de 3 e depois removemos o nó original que continha o 4, ficamos assim com:



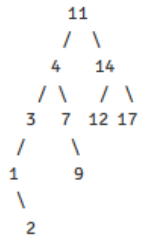
O 3 foi substituido pelo seu sucessor 4. O nó 4 original (filho esquerdo de 7) é eliminado e por isso a estrutura da árvore permanece consistente com as propriedades de BST.

2.1.3 [1] Considere que na árvore original foi efetuado um acesso à chave 4. Desenhe a árvore obtida após o acesso justificando os passos intermédios / raciocínio.

Ao aceder 4 numa splay tree, faz se splay do nó acedido até á raiz por rotações zig zag conforme o caso.

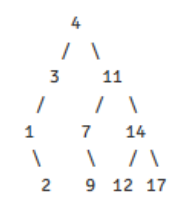
1. zigzag(4 é filho esquerdo de 7 e 7 é filho direito de 3):

   rotação á direita em 7; depois á esquerda em 3



2. zig (4 é filho esquerdo de 11) :

   rotação á direita em 11:



Está é a nova árvore após o acesso, onde o 4 se torna raíz e a propriedade BST se mantem em todos os nós.

2.2 [2] Considere uma árvore B (B-Tree) de ordem 3 inicialmente contendo apenas o nó raiz com a chave 11. Desenhe a árvore após cada uma das seguintes operações de inserção (I) e remoção (R) pela ordem indicada: I 5, 2, 8, 12, 13; R 2; (total de 6 desenhos). Justifique os passos intermédios / raciocínio para cada operação.

Ordem 3 => máximo 2 chaves por nó ; mínimo 1 excepto a raiz. Em overflow , promove-se a mediana; em underflow , redistribui-se ou funde-se com o irmão, movendo o separador do pai.

1. insercão 5

cabe na raiz (a raiz pode ter até duas chaves) :

[5 11]

2. inserção 2

a raiz ficaria com 3 chaves => overflow. divide-se e promove-se a mediana 5 para nova raiz:

```
    [5]
   /   \
 [2]   [11]
```

3. inserção 8

vai ao filho direito de 5 . insere em [11] => [8 11] , sem overflow:

```
    [5]
   /   \
 [2]   [8 11]
```

4. inserção 12

entra em [8 11] => [8 11 12], overflow . promove-se o 11 , isto é split do nó em [8] e [12] :

```
     [5 11]
    /  |  \
  [2] [8] [12]
```

5. inserção 13

segue para o filho direito de 11: [12] => [12 13] sem overflow

```
     [5 11]
    /  |   \
  [2] [8] [12 13]
```

6. remoção 2

está em folha [2] . remover => folha fica com 0 chaves ou seja underflow. o irmão adjacente [8] tem só 1 (não pode emprestar) => merge de folha com o irmão, descendo o separador 5 do pai. Pai perde o 5 e fica [11]

```
    [11]
   /   \
 [5 8]  [12 13]
```

edaf24-ex2-G3

Grupo III

3.1 [2] Considere uma tabela T de dispersão (hash) com dimensão 11 e função de hash $h(x) = x \bmod 11$. As colisões são resolvidas com sondagem (probing) linear. Considerando a tabela inicialmente vazia, determine o conteúdo final da tabela após a inserção das chaves 14, 4, 8, 15, 3, 6, 0 pela ordem apresentada. Justifique os cálculos efetuados para cada inserção.

- 14 => 14 mod 11 = 3 => entra em 3
- 4 => 4 mod 11 = 4 => entra em 4
- 8 = > 8 mod 11 = 8 => entra em 8
- 15 => 15 mod 11 = 4 => colisão => o próximo livre é 5 por isso entra em 5
- 3 => 3 mod 11 = 3 => colisão => 4 e 5 já estão ocupados => o próximo livre é 6 e por isso entra em 6
- 6 => 6 mod 11 = 6 => colisão => próximo livre é o 7 por isso entra em 7
- 0 => 0 mod 11 = 0 => entra em 0

Fica-se assim com a seguinte tabela

índice	0	1	2	3	4	5	6	7	8	9	10
t	0	-	-	14	4	15	3	6	8	-	-

132 / 220 Word Limit

3.2 [2] Considere o vetor [7 3 2 8 5 9 4 6 1]. Ordene o vetor utilizando o algoritmo de ordenação por fusão (Merge Sort). Explique de um modo geral o funcionamento do algoritmo. Indique justificando a sequência de vetores obtida para todas as iterações principais do algoritmo.

Como a ideia geral do algorítmo, o merge sort é um algorítimo de ordenação 'dividir - para - conquistar':

1. divide o vector em duas metades até obter subvetores de tamanho 1
2. ordena recursivamente cada metade
3. funde (merge) as duas metades ordenadas num único vetor ordenado.

Considerando o nosso vetor inicial [7 3 2 8 5 9 4 6 1] :

1.dividir em duas metades:

[7 3 2 8] e [5 9 4 6 1]
2. dividir [7 3 2 8] => [7 3] e [2 8]
[7 3] => [7] , [3] => merge => [3 7]
[2 8] => [2] , [8] => merge => [2 8]
merge [3 7] e [2 8] => [2 3 7 8]
3. dividir [5 9 4 6 1] => [5 9] e [4 6 1]
[5 9] => [5] , [9] => merge => [5 9]
[4 6 1] => [4] , [6 1] => [6] , [1] => merge => [1 6] => merge com [4] => [1 4 6]
merge [5 9] e [1 4 6] => [1 4 5 6 9]
4. fundir as duas grandes metades:
[2 3 7 8] e [1 4 5 6 9] => [1 2 3 4 5 6 7 8 9]

200 / 220 Word Limit