eglobal de recurso 2024/2025

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 $f(n) = \sqrt{n+3} + 10$ é $O(n)$.

Resposta:

A função f(n) = O(g(n)) se, e só se, existirem constantes positivas c e N, tal que

f(n)≤c*g(n), para todo n≥ N,

Ou seja,

n+√(n+3) +10 ≤c*n, para todo n ≥N

o que é equivalente a

n/n + (√(n+3) /n)+ 10/n ≤c, para todo n ≥N

o que é equivalente a

1+ (√(n+3) /n)+ 10/n≤ c, para todo n ≥N.

Por exemplo para N=1, temos:

1+ (√(1+3) /1)+ 10/1= 1+2+10=13

Por exemplo para c=13 e N=1 temos que

n+ √(n+3) + 10≤ 13*n, para todo n ≥1.

Desta forma, fica provado que f(n) = n+ √(n+3) + 10 é O(n), pois existe um c>0 e um N tal que n+ √(n+3) + 10≤ c*n para todo n ≥N. A condição é válida para c=13 e N=1, sendo que existem inúmero pares {c,N} que satisfazem essa condição.

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.

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

Resposta:

Termos de f(n):

n^3 é um termo polinomial de ordem O(n^3).

1 é uma constante então é de ordem O(1).

Termos de g(n):

3^n é um termo exponencial em base 3, e é de ordem O(3^n).

n é um termo linear de ordem O(n)

Podemos verificar que o termo dominante em f(n) é n^3 , pois cresce mais rapidamente que a constante. No caso de g(n) o termo dominante é 3^n, pois cresce muito mais rapidamente que n. Comparando os dois termos, observamos que 3^n cresce muito mais rapidamente que n^3 à medida que n tende para o infinito, pois funções exponenciais dominam funções polinomiais. Portanto, f(n) é limitada superiormente por g(n) multiplicada por uma constante para n suficientemente grande.

Assim, a relação entre f(n) e g(n) é f(n)=O(g(n))

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.

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

Resposta:

Termos de f(n):

2^n é um termo exponencial em base 2, e é de ordem O(2^n).

n é um termo linear, de ordem O(n).

1024 é uma constante, então é de ordem O(1).

Termos de g(n):

2^n é um termo exponencial em base 2, e é de ordem O(2^n).

10 é uma constante então é de ordem O(1).

Podemos verificar que o termo dominante em f(n) é 2^n, pois cresce mais rapidamente que os outros termos. No caso de g(n) o termo dominante é 2^n, pois cresce mais rapidamente que a constante. Comparando os dois termos dominantes, ambos são 2^n , o que implica que f(n) e g(n) crescem à mesma taxa assintoticamente. Portanto, a relação entre f(n) e g(n) é

f(n)) = Theta(g(n))

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++;
```

Resposta:

Laço externo: O loop externo i começa em 1 e é multiplicado por 2 a cada iteração até que i seja maior que n. O número de iterações deste loop é aproximadamente log_2(n)+1

Laço interno:

Para cad valor de i, o loop interno j executa de j=i até j=n, resultando em n-i+1 iterações.

A operação a++ é realizada a cada iteração do loop interno. Portanto, o número total de operações aritéticas f(n) é a soma sobre todos os valores de i (potências de 2) do número de iterações do loop interno.

Ou seja f(n)= ao somatório de k=0 até log_2(n) (n-2^m+1) = ([log_2n]+1)(n+1)-(2^([log_2n]+1)-1)

Assim sendo, podemos concluir que para valores grandes de n, o termo dominante na expressão é n log_2 n. Portanto, na notação assintótica, constantes e termos de moenor ordem são desprezados, resultando em:

f(n)=O(n log n).

Grupo II

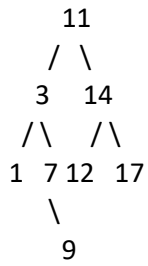
Considere uma árvore de promoção (Splay Tree) inicialmente vazia.

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

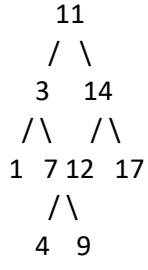
2.1.1 [1] Insira na árvore as chaves 11, 14, 12, 3, 17, 1, 7, 9, 4, 2 pela ordem indicada. Desenhe a árvore obtida após efetuadas todas as inserções (total de 1 desenho). Justifique os passos intermédios / raciocínio apenas para as duas últimas inserções.

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

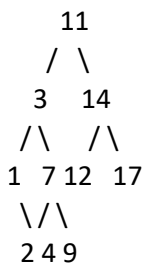
Resposta:
Numa splay tree as operações básicas (inserção, remoção) são realizadas inicialmente com a mesma lógica de uma árvore de pesquisa binária (BST):
os valores menores de um nó vão para a subárvore esquerda; os maiores, para a subárvore direita.
Inserindo as chaves por ordem quando chegamos à penúltima inserção (chave 4) temos a seguinte árvore:



Como 4<11 vamos pela subárvore esquerda. Como 4>3 vamos pela subárvore direita. Como 4<7 vamos pelo ramo da esquerda, como 7 não tem filho esquerdo
inserimos aí a chave 4. Deste modo obtemos a seguinte árvore.



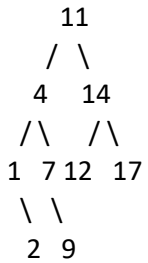
Como 2<11 vamos pela subárvore esquerda. Como 2<3 vamos pela subárvore esquerda. Como 2>1 vamos pelo ramo da direita, como 1 não tem filho direito
inserimos aí a chave 2. Deste modo obtemos a árvore final.



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.

Resposta:

O algoritmo de remoção por cópia remove um nó de uma árvore binária de pesquisa substituindo o seu valor pelo do seu predessecor imediato (o maior valor da subárvore esquerda) ou, de forma análoga, pelo do seu sucessor imediato (o menor valor da subárvore direita), caso que vai ser utilizado. Depois dessa cópia, elimina-se o nó que continha o sucessor (chave 4), que será sempre uma folha ou terá apenas um filho, neste caso é folha, convertendo assim a remoção de um nó com dois filhos num caso simples sem violar a propriedade de pesquisa binária. Para removermos a chave 3 identificamos a sucessora que é a chave 4 (folha), substituímos o valor do nó 3 pelo valor 4, depois eliminamos o nó 4, ficando o 7 apenas com o filho direito 9. Deste modo, obtemos a seguinte árvore:



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.

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.

Estamos perante uma B (B-Tree) de ordem m=3, onde cada nó suporta até 2 chaves (m-1) e 3(m) filhos. Exceto a raiz, todos os nós devem conter no mínimo 1 chave. A Árvore mantém o seu equilíbrio através de splits durante inserções (quando um nó excede 2 chaves) e fusões ou redistribuições durante remoções (quando um nó viola o número mínimo de chaves.

Inicialmente, temos:

[11|]

I 5: Ao inserirmos 5, coamo temos espaço na raiz, colocamos aí a chave deste modo obtemos:

[5|11]

I 2: Ao inserirmos 2 na folha [5|11], obtemos [2|5|11] pelo que ocorre um overflow, deste modo para o resolvermos fazemos um split, onde a mediana
5 é promovida para nova raiz passando a ter dois filhos com uma chave cada um:

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

I 8: Como 8>5 seguimos para o filho direito, como existe espaço na folha colocamos lá a chave 8.

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

I 12: Como 12>5 seguimos pela ramo da direita, mas como a inserção de 12 nessa folha provoca um overflow [8|11|12], resolvemos tal como na inserção 2
com um split. Ou seja, 11 é promovida a pai, como a raiz ainda tem espaço é colocada diretamente lá passando a ter dois filhos além da folha esquerda [2|].

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

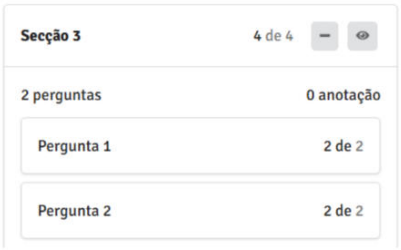
I 13: Como 13>11 vamos pelo ramo à direita, como ainda temos espaço colocamos 13 juntamente na folha com o 12.

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

R 2: Após removermos 2 a folha fica vazia o que provoca um underflow, pelo que verificamos se o irmão adjacente à direita pode emprestar, o que não é possível
visto que o irmão direito só tem 1 chave, pelo que é necessário fazer uma fusão, ou seja, passamos a ter [5|8]. No pai (raiz) removemos a chave promotora 5, ficando
apenas [11|]. Deste modo, obtemos a árvore final:

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

Grupo III



3.1 [2] Considere uma tabela T de dispersão (hash) com dimensão 11 e função de hash $h(x) = x \ mod \ 11$. 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.

Resposta:

I 14 - h(14) = 14 mod 11 = 3 - A posição 3 está vaga pelo que colocamos lá a chave 14

I 4 - h(4) =4 mod 11 = 4 - A posição 4 está vazia pelo que colocamos lá a chave 4

I 8 -h(8) = 8 mod 11 = 8 - A posição 8 está vazia pelo que colocamos lá a chave 8

I 15 - h(15)= 15 mod 11 =4 - Como a posição 4 está ocupada é necessário resolvermos por recurso a sondagem linear.

Tentativa 1 - h+1=16 16 mod11=5 - Como a posição 5 está vazia colocamos lá a chave 15

I 3 - h(3) = 3 mod 11 = 3 - Como a posição 3 está ocupada é necessário resolvermos por recurso a sondagem linear.

Tentativa 1 - h+1= 4 4mod 11 =4 (ocupado por 4)

Tentativa 2 - h+2 =5 5mod 11 = 5 (ocupado por 15)

Tentativa 3 - h+3 = 6 6mod 11 = 6 - A posição está vaga pelo que colocamos lá a chave 3.

I 6- h(6) = 6 mod 11 =6 - Como a posição 6 está ocupada é necessário resolvermos por recurso a sondagem linear.

Tentativa 1 - h+1= 7 7mod 11 =7 -A posição está vaga pelo que colocamos lá a chave 6.

I 0 - h(0) = 0 mod 11 = 0 - A posição está vaga pelo que colocamos lá a chave 0.

Posição	0	1	2	3	4	5	6	7	8	9	10
Conteúdo	0	-	-	14	4	15	3	6	8	-	-

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.

O Merge Sort é um algoritmo de ordenação baseado no paradigma dividir para conquistar, que opera dividindo recursivamente o array em duas metades até que cada subarray contenha apenas um elemento. Em seguida, os subarrays são intercalados (ou fundidos) de forma ordenada, utilizando um array auxiliar para manter os elementos temporariamente. A intercalação é feita comparando os menores elementos de cada subarray e copiando-os para a posição correta. O algoritmo continua esse processo até reconstruir o array original de forma ordenada.