

UNIDADE CURRICULAR: Estruturas de Dados e Algoritmos Fundamentais

CÓDIGO: 21046

DOCENTE: Paulo Shirley

A preencher pelo estudante

NOME: Luís Carlos Crispim Pereira

N.º DE ESTUDANTE: 2300163

CURSO: LEI – Licenciatura em Engenharia Informática

1.1 [1] Utilizando a definição, prove que $f(n) = (n + 3)^2$ é $O(n^2)$.

Definição formal - Diz se que uma função $f(n)$ é $O(g(n))$ se existirem constantes positivas de c e n_0 tais que:

$$n \geq n_0, f(n) \leq c \cdot g(n)$$

$$f(n) = (n+3)^2 = n^2 + 6n + 9$$

$$n^2 + 6n + 9 \leq c \cdot n^2, n \geq n_0$$

$(n^2 + 6n + 9)/n^2 = 1 + 6/n + 9/n^2$ - Tende para 1 quando n tende para infinito logo existe um valor de n_0 onde

$$1 + 6/n + 9/n^2 \leq c$$

testado com $n \geq 1$

$$1 + 6/n + 9/n^2 = 1 + 15 = 16$$

portanto

$$f(n) \leq 16 \cdot n^2, n \geq 1$$

Conclusão:

Escolhendo $c=16$ e $n_0=1$ verifica-se

$$(n+3)^2 \leq 16n^2, n \geq 1$$

logo $f(n)$ é $O(n^2)$

Nota: Por dificuldades de abertura de formulas em editor wiseflow, e tambem por falta de simblos matematicos utilizei o teclado diretamente, acredito que se consiga entender o raciocinio.

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.

$$f(n) = n^2 + 1, g(n) = \sqrt[3]{n^5} + 100$$

$$f(n) = n^2 + 1$$

$$g(n) = \sqrt[3]{n^5} + 100 = n^{5/3} + 100$$

Termos dominantes:

$f(n)$ termo dominante n^2

$g(n)$ termo dominante $n^{5/3}$

Comparando crescimentos

Como $2 > 5/3$ então:

$f(n)$ é $\Omega(g(n))$ pois $f(n)$ cresce mais depressa

não é Θ , pois crescem a ritmos diferentes

nao é O , porque $f(n)$ cresce mais rapidamente que $g(n)$

Nota: Por dificuldades de abertura de formulas em editor wiseflow, e tambem por falta de simblos matematicos utilizei o teclado diretamente, acredito que se consiga entender o raciocinio.

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.

$$f(n) = n^{4 \log n} + 1000, \quad g(n) = n^{\log n^2}$$

$$f(n) = n^{(4 \log n)} + 1000$$

$$g(n) = n^{(\log n^2)} = n^{(2 \log n)}$$

Termos dominantes:

$f(n)$ termo dominante $n^{(4 \log n)}$

$g(n)$ termo dominante $n^{(2 \log n)}$

Comparando crescimentos

Como $n^{(4 \log n)} > n^{(2 \log n)}$ então:

$f(n)$ E omega ($g(n)$) pois $f(n)$ cresce mais depressa

não é \emptyset , pois crescem a ritmos diferentes

nao é O, porque $f(n)$ cresce mais rapidamente que $g(n)$

Nota: Por dificuldades de abertura de formulas em editor wiseflow, e tambem por falta de simblos matematicos utilizei o teclado diretamente, acredito que se consiga entender o raciocinio.

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 $O(\cdot)$.

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

Determinar o numero total de vezes que $a++$ é executado, o laço exterior vai de $i=1$ ate $i=n$ ou seja n iterações, para cada valor de i o laço interior vai de $j=1$ ate $j=i$ ou seja i iterrações sendo assim o numero total de incrementos de a é:

$f(n) = \sum i = (n(n+1))/2$ (nota wiseflow pois não consigo adicionar corretamente as formulas matematicas, o sumatorio é de $i=1$ ate n)

$f(n) = (n(n+1))/2 \in \mathcal{O}(n^2)$ -- $f(n) \in \mathcal{O}(n^2)$

Assim a complexidade de $f(n)$ é:

$f(n) \in \mathcal{O}(n^2)$

Nota: Por dificuldades de abertura de formulas em editor wiseflow, e tambem por falta de simblos matematicos utilizei o teclado diretamente, acredito que se consiga entender o raciocinio.

2.1 Considere uma árvore de pesquisa binária (BST) inicialmente vazia.

2.1.1 [1] Insira na árvore as chaves 6, 11, 9, 2, 10, 13, 7, 1, 4, 8 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".



Sempre que se insere um valor x numa BST, compara-se com a raiz se $x <$ raiz vai para a subárvore esquerda se $x >$ que raiz vai para a subárvore direita, repete-se recursivamente este processo até encontrar um lugar vazio.

Assim sendo:

Inserção do 4:

$4 < 6$ esquerda

$4 > 2$ direita

Inserção do 8:

$8 > 6$ direita

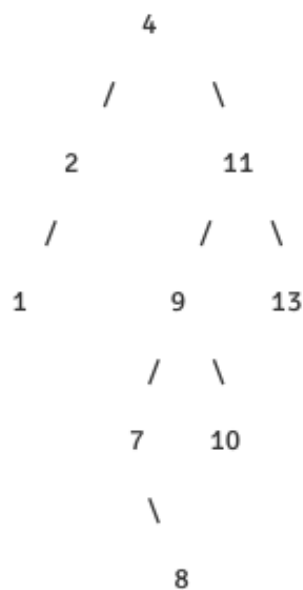
$8 < 11$ esquerda

$8 < 9$ esquerda

$8 > 7$ direita

Nota: Ferramenta de desenho não funciona em todas as caixas de texto

2.1.2 [1] Remova da árvore original a chave 6 utilizando o algoritmo de remoção por cópia (Deletion by Copying). Das várias opções possíveis escolha a que lhe parece melhor. Desenhe a árvore obtida justificando a sua escolha e os passos intermédios / raciocínio.



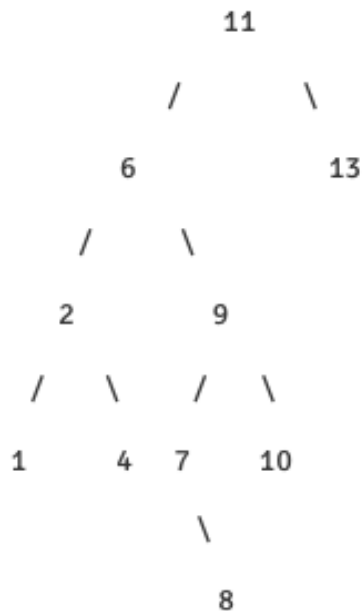
Passos, remoção por cópia com o antecessor (o maior valor na subarvore esquerda de 6)

Maior valor é 4, primeiro pasos substituir o 6 por uma copia do 4, passo seguinte remover antigo 4 que agora esta duplicado, como não tem filho é simplesmente removido.

Nota: Ferramenta de desenho não funciona em todas as caixas de texto

2.1.3 [1] Explique em que consiste a operação de rotação de um nó em torno do seu parente. Efetue na árvore original uma rotação de 11 em torno do seu parente. Desenhe a árvore obtida justificando os passos intermédios / raciocínio.

Uma rotação em árvore binária preserva a BST, mas muda a estrutura da árvore melhorando o equilíbrio, estamos essencialmente a promover o nó para cima e a rebaixar o seu pai.



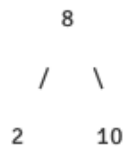
11 sobe para a raiz, 6 torna-se filho esquerdo de 11, o filho esquerdo de 11 que era 9 manteve-se a esquerda mas passa para filho direito de 6, 9 mantém os seus filhos esquerdos e direitos, 2 1 e 4 mantemse e 13 mantem se como filho direito de 11.

Nota: Ferramenta de desenho não funciona nesta caixa de texto.

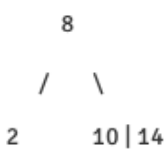
2.2 [2] Considere uma árvore B (B-Tree) de ordem 3 inicialmente contendo apenas o nó raiz com as chaves 2 e 10. Desenhe a árvore após cada uma das seguintes operações de inserção (I) e remoção (R) pela ordem indicada: I 8, 14, 12, 4, 6; R 10; (total de 6 desenhos). Justifique os passos intermédios / raciocínio para cada operação.

Inserção de 8 ha espaço na chave que pode ter 2 chaves e. tres filhos quando ha tres chaves promove-se a chave do meio para pai

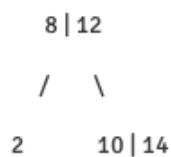
sendo assim [2, 8, 10]



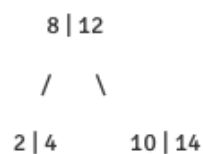
Inserção do 14 onde é maior que 8 insere no 10 sem overflow



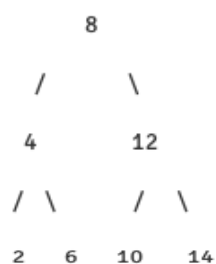
inserção do 12 maior que 8 insere-se no 10 ha overflow promove-se para pai



Inserção do 4 menor que 8 insere no 2 sem overflow



Inserção do 5 menor que 8 insere no 2 4 que faz overflow promove para pai



3. Considere o vetor [8 6 9 2 1 3 5 4 7]. Ordene o vetor utilizando o algoritmo de ordenação indicado. Explique de um modo geral o funcionamento do algoritmo. Indique justificando a sequência de vetores obtida correspondente às iterações principais do algoritmo.

3.1 [2] Algoritmo de ordenação por inserção (Insertion Sort).

Vetor ordenado [1, 2, 3, 4, 5, 6, 7, 8, 9]

Passo 1: insere 8 fica [8]

Passo 2 insere 6 [8,6] que é menor troca [6,8]

Passo 3 insere 9 [6,8,9] é maior que 8 mantém

Passo 4 insere 2 [[6,8,9,2] é menor que todos troca 1 a 1 [2,6,8,9]

Passo 5 insere 1 [2,6,8,9,1]] é menor que todos troca 1 a 1 [1,2,6,8,9]

Paso 6 insere 3 [1,2,6,8,9,3] é menor que 9 8 e 6 troca 1 a 1 [1,2,3,6,8,9]

Passo 7 insere 5 [1,2,3,6,8,9,5] é menor que 9,8,6 troca 1 a 1 [1,2,3,5,6,8,9]

Passo 8 insere 4 [1,2,3,5,6,8,9,4] é menor que 9,8,6,5 troca 1 a 1 [1,2,3,4,5,6,8,9]

Passo 9 insere 7 [1,2,3,4,5,6,8,9,7] é menor que 9,8 troca 1 a 1 [1,2,3,4,5,6,7,8,9]

Fica entao: [1, 2, 3, 4, 5, 6, 7, 8, 9]

3.2 [2] Algoritmo de ordenação rápida (Quick Sort).

Quick sorte faz a verificação do ultimo valor do vetor sendo esse o pivo a mover, apos isso compara o pivo com todos os seus valores menores e maiores e define o seu lugar, sendo que valores menores que pivo ficam a esquerda e maiores a direita

Vetor inicial [8 6 9 2 1 3 5 4 7]

Passo 1 pivo 7 elementos menores [6,2,1,3,5,4] posição definida para o pivo 6 [6,2,1,3,5,4,7,9,8]

Passo 2 pivo 4 elementos menores [2,1,3] posição definida para o pivo 3 [2,1,3,4,5,6,7,9,8]

Passo 3 pivo 3 elementos menores [2,1] posição definida para o pivo 2 [2,1,3,4,5,6,7,9,8]

Passo 4 pivo 1 elementos menores nao tem posição definida para o pivo 0 [1,2,3,4,5,6,7,9,8]

Passo 5 pivo 6 já esta no local nada a fazer.

Passo 6 oivo 8 elementos menores 9 troca de lugar [6,2,1,3,5,4,7,8,9]

Resultado final ordenado [6,2,1,3,5,4,7,8,9]