Estruturas de Dados / Programação 2 - ECOM008/COMP208
Exame Escrito

Márcio Ribeiro
Universidade Federal de Alagoas - UFAL
Instituto de Computação - IC

14 de Setembro de 2022

Questão 1. Descreva (com riqueza de detalhes) a estimativa de execução do segmento de código abaixo. Em seguida, descreva sua resposta na notação *Big-O*. Considere n o tamanho do array. **(1.0 ponto)**

```
for (i = 1; i <= n; i++) {
   j = n;
   while (j > 1) {
      j = j / 2;
   }
}
```

(n+1)

log n

n.log(n)

O n.log n

Desconsidera a constante

Questão 2. Prove que $2n^2 + 2n = O(n^2)$ utilizando a (1) Definição (detalhando a constante c e o n_0); e (2) Limites, do Cálculo Diferencial e Integral. **(1.0 ponto)**

Questão 3. Escreva a equação de recorrência da Busca Sequencial e resolva-a usando o método da expansão; faça o mesmo para a Busca Binária, só que agora resolvendo utilizando o método da árvore de recursão. **(1.0 ponto)**

Questão 4. O código abaixo possui uma chamada a uma função `print` responsável por imprimir o array a. Nesta questão, você deve escrever o que é impresso na tela a cada passo da execução do algoritmo. **(1.0 ponto)**

$t(n-1) = 1 + t(n-1)$

```
void quick_sort(int a[], int left_index, int right_index)
{
  int left, right, pivot;
  if (left_index >= right_index) return;

  left = left_index;
  right = right_index;

  pivot = a[(left_index + right_index) / 2];

  while (left <= right)
  {
    while (a[left] < pivot) left++;
    while (a[right] > pivot) right--;

    if (left <= right)
    {
      swap(a, left, right);
      left++; right--;
    }

    print(a);
  }

  quick_sort(a, left_index, right);
  quick_sort(a, left, right_index);
}

int main()
{
  int a[SIZE]={1, 12, 5, 26, 7, 14, 3, 7, 2};
  quick_sort(a, 0, SIZE - 1);
}
```

11

n	i	j
8	1	8
8	2	4
		2
		1
		8

Primeira questão

Questão 5. Escreva uma função recursiva em Haskell para, dada uma lista e um inteiro positivo, retornar o n-ésimo elemento da lista. Exemplo: nesimo 4 [3,5,7,9,12] = 9 **(1.0 ponto)**

Questão 6. BÔNUS. Prove que, se $f_1(n) = O(g_1(n))$ e $f_2(n) = O(g_2(n))$, então $f_1(n)+f_2(n) = O(max g_1(n), g_2(n))$ **(1.0 ponto)**

2)

a) $2n^2 + n$ é $O(n^2)$

usando a definição

$$2n^2 + n \le n^2 \cdot c$$

$$\frac{2\overset{1}{\cancel{n^2}}}{\cancel{n^2}} + \frac{\overset{1/n}{\cancel{n}}}{n^2} \le \frac{\overset{1}{\cancel{n^2}}}{\cancel{n^2}} \cdot c$$

$$2 + \frac{1}{n} \le c$$

Se adotamos $n_0 = 1$ e $c = 3$, provamos que $2n^2 + n$ é big O de $O(n^2)$. Podemos colocar quaisquer valores maiores do que $n_0 = 1$ e c maior que 3 que essa afirmação será verdadeira

b)

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = c \text{ ou } 0$$

$$\lim_{n \to \infty} \frac{2n^2 + 2n}{n^2} = \frac{4n + 2}{2n} = \frac{4}{2} = 2$$

Como o resultado deu uma constante, então $2n^2 + 2n$ é big O de n^2

3)

$T(n) = (n-1)+1$

$T(n) = (n-2)+2$

$T(n) = (n-3)+3$

$\vdots$

$T(n) = \underbrace{(n-k)}+k$

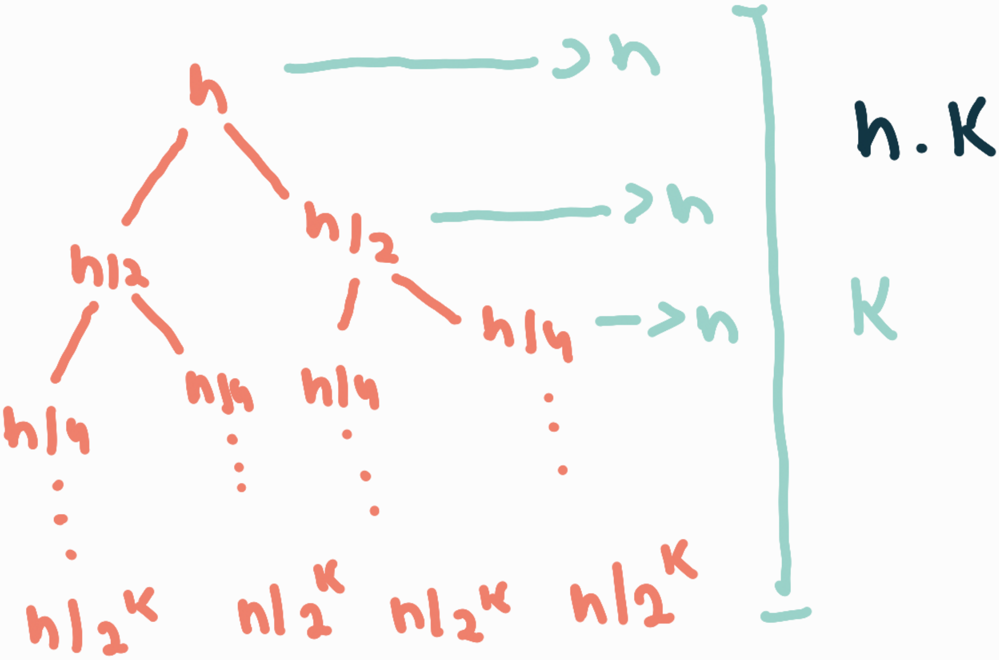
$\rightarrow n-k=0 \rightarrow n=k$

$T(n) = (n-\cancel{k})+k$

$T(n) = n$

$O(n)$

Busca binária



Assume

$n/2^k = 1$

$k = \log n$

$n.k$

$n.\log n$

$O(n.\log.n)$

5)

```haskell
nesimo :: Int -> [Int] -> Int
nesimo 1 (a:as)  = a
nesimo n (a:as) = nesimo (n-1) as
```

A explicação é que você vai recursivamente removendo 1 de N e removendo o primeiro item da lista
vai chegar uma hora que N vai ser 1 e você achou o item que queria
ex:

4 [3, 5, 7, 9, 12]
3 [5, 7, 9, 12]
2 [7, 9, 12]
1 [9, 12] -> entra na condição e retorna o 9