Universidade Federal Rural de Pernambuco

Departamento de Estatística e Informática

Coordenação de Graduação em Ciência da Computação

Algoritmos e Estruturas de Dados

VA 1: Parte On Line

Aluno
Giuseppe Fiorentino Neto

Professor
Rodrigo Nonamor Pereira Mariano de Souza

Recife
julho–2017

Algoritmos e Estruturas de Dados
BCC 2017/1 VA 1: Parte On Line
Rodrigo de Souza

Prazo: 07/07/2017 (12h00)

1. Descreva um algoritmo recursivo que recebe um inteiro positivo n e calcula o piso de lg n. Estime o número de chamadas recursivas de seu algoritmo em função de n. Você pode apresentar o pseudocódigo de seu algoritmo, ou uma descrição precisa, clara em Português.

```
//O algoritmo recebe o valor n para calcular o seu logaritmo na base 2.
//E retorna o valor do piso desse logaritmo
int loga(int ini, int X , int n){
        int XX=XY=X;                  //1
        int q;                        //2
        if(ini>=n) return X;          //3
        q=(ini+n)/2;                  //4
        int x=loga(ini,++XX,q);       //5
        int y=loga(q+1,++XY,n)        //6;
        if(x<y) return x;             //7
        else return y;                //8
}
```

Em uma árvore a distância entre a raiz e um vértice é denominado nível e a maior distância de uma raiz para um dado vértice é denominado altura.
Em uma árvore de que tem n vértices e altura h tem folhas na altura

$$h \geq \lfloor \log n \rfloor$$

Assim ao calcularmos um logaritmo de base 2 de um dado numero estamos encontrando a altura dessa árvore.

Na linha 3 temos a condição base para uma árvore.
Na linha 4 temos o calculo do meio do número para q possamos dividir a árvore em 2
Nas linhas 5 e 6 temos a construção de uma árvore
Na linha 7 e 8 serve para analisar o piso

2.Escreva um programa que recebe um vetor de inteiros A[1..n] e decide (responde SIM ou NAO) se há duas posições em A contendo o mesmo valor. Calcule a Complexidade do seu algoritmo. Mesmo comentário do exercício anterior.

Versão Iterativa

```
//Recebe um vetor v[0..n] e busca os valores repetidos
//se encontrar pelo menos 1 retorna true(1) senao retorna false(0)
int Repetido(int e,int n,int v[]){
        for(int i=e;i<n;i++){
                for(int j=i;j<n;j++){

                        if(v[i]==v[j]) return 1;
                }

        }
        return 0;
```

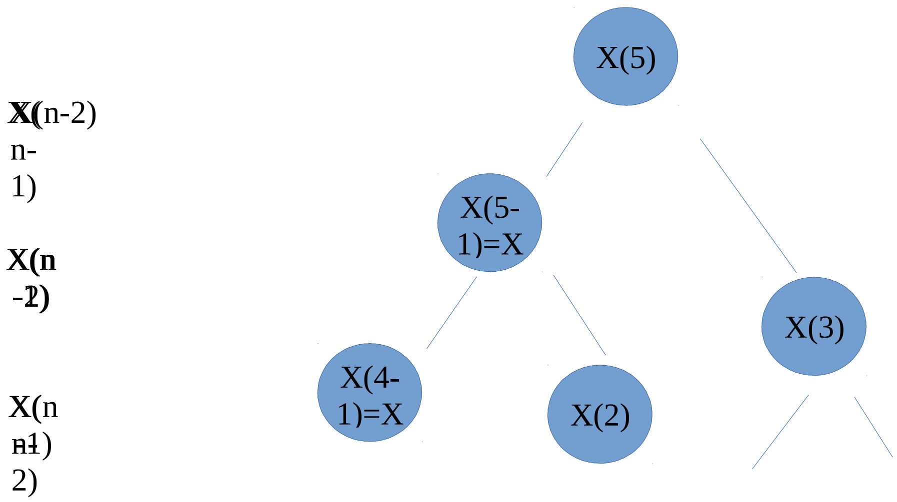
}
Como há um laço aninhado o primeiro laço é executado exatamente n vezes e o segundo laço é executado n-i vezes. Dessa forma o algoritmo faz cerca de n*(n-i) comparações. Assim, o consumo de tempo do algoritmo é sempre proporcional a n2.

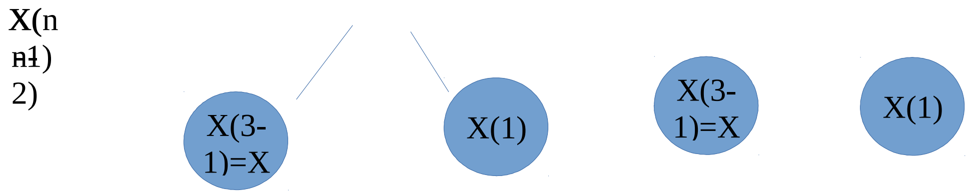
$$T=O(n^2)$$

3. Qual o valor de X(5) se X é dada pelo seguinte código? Justifique desenhando a árvore das chamadas recursivas.

```
int X (int n) {
      if (n == 1 || n == 2) return n;
     else return X (n-1) + n * X (n-2);
}
```

O valor do X(5) é 38





4. Se preciso de t segundos para fazer uma busca binária em um vetor com n elementos, de quanto tempo preciso para fazer uma busca em n^2 elementos? Justifique.
Preciso exatamente de 2t para calcular uma busca binária em um vetor com n^2 elementos. Pois a complexidade de uma busca binária T=O(logn). Sendo assim:

Se

$$\log n = t$$

Calculando

$$\log n^2$$

Pela propriedade dos produtos de um logaritmo em q log(a*b)=log(a+b) temos:

$$\log(n*n)=\log n+\log n$$

Assim:

$$2*\log n$$

Como :

$$\log n = t$$

Ficamos com :

$$\log n^2=2t$$

5.A seguinte versão de busaBinaria está correta? Caso negativo, apresente uma instância onde o algoritmo não funciona como esperado.

```
e = -1;  d = n-1;

while (e < d) {

        m = (e + d)/2;

        if (v[m] < x) e = m;

        else d = m-1;

 }

return d+1;
```

Não o algoritmo não está. Pois se pegarmos uma instância com um valor maior q todos os valores no vetor a função entrara em loop devido ao fato de que e nunca será maior ou igual a d. Isso ocorre, pois o algoritmo fica preso no if por exemplo:
usando o vetor {1,6,9}
X=50

E	D	M
-1	2	(-1+2)/2=0
1	2	(1+2)/2=1
1	2	1
...	...	...

6. Submeta um vetor indexado por 1..4 à função mergesort. Teremos a seguinte sequência de invocações da função:

mergesort (1,5,v)
mergesort (1,3,v)
    mergesort (1,2,v)
    mergesort (2,3,v)
mergesort (3,5,v)
    mergesort (3,4,v)
    mergesort (4,5,v)

(observe a indentação). Repita o exercício com um vetor indexado por 1..5.

Analisando o padrão que possui os dados obtemos o seguinte:
Usando ini para início

mergesort (ini,n+1,v)
mergesort (ini,n-1,v)
    mergesort (ini,(n+ini)/2,v)
    mergesort ((n+ini)/2,n-1,v)
mergesort (n-1,n+1,v)
    mergesort (ini,(n+ini)/2,v)
    mergesort ((n+ini)/2,n-1,v)

Logo os valores para o vetor indexado por 1..5 será

mergesort (1,6,v)
mergesort (1,4,v)
    mergesort (1,2,v)
    mergesort (2,4,v)
mergesort (4,6,v)
    mergesort (4,5,v)
    mergesort (5,6,v)