Departamento de Engenharia de Telecomunicações e Informática

Algoritmos e Complexidade

Exame Normal
14 de Junho de 2012

1

1.1 Pergunta

Seja n um número natural, desenvolva um algoritmo eficiente que permita calcular a seguinte expressão

$$S(n) = \begin{cases} 1 & , \quad \text{se } n = 0 \\ n + \sum_{i=0}^{n-1} (-1)^i \times S(i) & , \quad \text{se } n > 0 \end{cases}$$

1.1.1 Resposta

```
1  int calcS(int n)
2     if(n==0)
3                  return 1;
4     else{
5                  int i, soma = n;
6                  for(i = 0; i <= (n-1); i+=2)
7                           soma += calcS(i);
8                  for(i = 1; i <= (n-1); i+=2)
9                           soma -= calcS(i);
10                 return soma;
11    }
```

1.2 Pergunta

Conte o número de operações básicas e diga qual a ordem de complexidade do algoritmo.

1.2.1 Resposta

Seja $A(n)$ o número de adições / subtrações efetuadas para um determinado n.

$$A(n) = \begin{cases} 0 & , \quad \text{se } n = 0 \\ n + \sum_{i=0}^{n-1} A(i) = \frac{2^n - 1}{3} & , \quad \text{se } n > 0 \end{cases}$$

Logo, $\mathcal{O}(2^n)$.

2

Seja um array dado por ordem não-decrescente, com elemento [repetidos | não repetidos]

2.1

Faça um algoritmo que percorra o array apenas uma vez, implementando a estratégia do BubbleSort.

2.1.1

```
int trocarTudo(int[] v, int n){
int i, troca=0;
for(i = 1; i < n; i++) if(troca |= v[i] > v[i-1] ) trocar(v[i], v[i-1]);
return troca;
}
void trocar(int &a, int&b){
a^= b;
b^=a;
a^=b;
}
```

2.2

Calcule o número de comparações e de atribuições no melhor e no pior caso do algoritmo 2.1

2.2.1

TODO...

2.3

Efetue o algoritmo BubbleSort usando a função feita na pergunta 2a

2.3.1

```
void bubbleSort(int [] v, int n){
while(trocarTudo(v,n--));
}

int trocarTudo(int[] v, int n){
int i, troca=0;
for(i = 1; i < n; i++) if(troca |= v[i] > v[i-1] ) trocar(v[i], v[i-1]);
return troca;
}
void trocar(int &a, int&b){
a^= b;
b^=a;
a^=b;
}
```

2.4

Calcule o número de comparações e de atribuições no melhor e no pior caso do algoritmo 2.3

2.4.1

TODO...

3

Árvore binária "em-ordem"crescente.

3.1

Liste todos os nós que sejam inferiores a um dado valor a.

3.1.1

```
PtNo listaNo(PtNo no, int a){
        if(no == NULL)
                return NULL;
        if(listaNo(no->esq, a) == NULL){
                if(no->elem < a)
                        printf("Elemento: %d\n",no->elem);
                else ??????
                        return NULL; ??????
        }
        return listaNo(no->dir, a);
}
```

3.2

Faça uma função que permita inserir um determinado valor/nó ordenadamente e que não esteja na árvore.

3.2.1

```
TODO...
```