Entregar 1 arquivo .pdf

1) Implemente uma função que calcule o *n*-ésimo termo da sequência de Fibonacci com complexidade de tempo linear. Construa um gráfico comparando a execução da versão linear com a versão recursiva definida abaixo:

```
unsigned int fib (unsigned int n)
{
   if (n < 2)
     return n;
    return fib (n-2) + fib (n-1);
}
```

2) Para a função ***potencia*** definida abaixo: Mostre qual a relação de recorrência que descreve o tempo de execução da função. Resolva essa relação de recorrência. Calcule a complexidade de tempo e a complexidade espaço para essa função.

```
unsigned int potencia (unsigned int b, unsigned int e)
{
   unsigned int r;
   if (e == 0)
            return 1;
   r = potencia(b, e/2);
   if (e % 2 == 0)
          return r*r;
   else
         return r*r*b;
}
```

3) Resolva as relações de recorrência:

a) $T(n) = T(n/2) + n$
$T(1) = 1$

b) $T(n) = 2T(n - 1) + n$
$T(1) = 1$

c) $T(n) = 4T(n/2) + n$
$T(1) = 1$

d) $T(n) = T(n/2) + \log_2 n$
$T(1) = 1$

4) Usando indução matemática prove que a solução da relação de recorrência:

$T(n) = 4T(n/2) + n$

$T(1) = 1$

é

$T(n) = 2n^2 – n$, para $n = 2^k$ e $k \geq 1$.

5) Considerando a estrutura de dados ***Lista*** que representa uma lista encadeada, implemente uma função que insira no início da lista um elemento inteiro passado como parâmetro e uma função que insira o elemento ao final da lista. Qual a complexidade de tempo de cada versão? Explique sua resposta.

```
struct Lista
{
    int elem;
    struct Lista *ptr;
};
```

6) Considerando uma árvore binária de pesquisa balanceada representada pela estrutura de dados declarada abaixo, mostre a relação de recorrência que descreve a complexidade de tempo para a função ***imprimir***. Resolva essa relação de recorrência. Mostre a complexidade de tempo e espaço para essa função.

```
struct Arvore
{
    int elem;
    struct Arvore *esq, *dir;
};

void imprimir(struct Arvore *r)
{
    if (r != NULL)
    {
            imprimir (r->esq);
            printf("%d ", r->elem);
            imprimir (r->dir);
    }
}
```

Reescreva a função ***imprimir*** sem usar recursividade.