ALEXANDRO DE JESUS SILVA

Análise de complexidade de funções de crescimento temporal

- **Função fatorial**

função fatorial sequencial.

```
#include <stdio.h>
int main(){

    int i, fat;
    scanf(“%d”, &fat);
    for(i = 1; fat > 1; fat = fat - 1){
        i = i * fat;
    }
    printf("\n%d", i);
    return 0;
}
```

parte do estudo para achar o T(n).

/*
T(n) = 1+ 1+n+2(n-1)+2(n-1)
T(n) = 5n -2
o(n)

//n = 1
1+
(1+1+0)

total: 3

//n 2
1+
(1+1+2) -> 2>1
  2
(0+1+0) -> 2>2

total: 9

//n 3
1+
(1+1+2) -> 3>1
  2+

```
(0+1+2)     -> 3>2
   2+
(0+1+0)     -> 3>3

total: 15

//n 4
1+
(1+1+2)     -> 4>1
   2+
(0+1+2)     -> 4>2
   2+
(0+1+2)     -> 4>3
   2
(0+1+0)     -> 4>4

total: 21
*/
```

função fatorial recursiva

```
#include <stdio.h>
int fatorial(int n){
  int fat;
  if ( n <= 1 ){
    return (1);
  }
  else{
    fat = n * fatorial(n - 1);
    return (fat);
  }
}

int main(void){
  int fat;
  scanf(“%d”, &fat)
  printf(" %d",  fatorial(fat));
  return 0;
}
```

parte do estudo para achar o T(n).

/*

T(n)= 1+n + 3(n-1)
T(n)= 4n
o(n)

// n 1
1+
1+
fim_1
total = 2

// n 2
1+
1+
(1+1+1)
fim_1
1+
fim_2

total = 6

// n 3
1+
1+
(1+1+1)
fim_1
1+
(1+1+1)
fim_2
1+

total = 10
*/

Professor, seguir todos os passos que o senhor demonstrou nas aulas, porém em algumas pesquisas que fiz na internet e a função fatorial recursiva estava **o(2^n)**, refiz várias vezes, mas não cheguei nesse resultado e sim a **o(n)** na recursiva e sequencial.

Então, nos dois casos de da função fatorial não há diferença. Por que a complexidade dos dois é o mesmo. Ou seja, o(n).

- **Função Fibonacci**

Função Fibonacci sequência

```
#include<stdio.h>
#include <time.h>
int main(){
        int n, num, a, b, i;
        scanf("%d", &n);                        //      1
        a = 1;                                  //      1
        b = 0;                                  //      1
        for(i = 0 ; i < n; i++){                //      1; n+1, 2*n
            num = a + b;                        //      2*n
            b = a;                              //      n
            a = num;                            //       n
         }
}
```

parte do estudo para achar o T(n).

```
/*
T(n)= 1+1+1+1+n+1+2n+2n+n+n
T(n)= 7n +5
o(n)


// n = 0
1+1+1
(1+1+0)
total: 5


// n = 1
1+1+1+
(1+1+2)       ->   0 < 1
        +2+1+1
(0+1+0)       ->   1< 1
total : 12

//n = 2
```

1+1+1+
(1+1+2) -> 0 < 2
  +2+1+1
(0+1+2) -> 1< 2
  + 2+1+1
(0+1+0) -> 2<2
total : 19

//n = 3
1+1+1+
(1+1+2) -> 0 < 3
  +2+1+1
(0+1+2) -> 1< 3
  + 2+1+1
(0+1+2) -> 2<3
  + 2+1+1
(0+1+0) -> 3<3

total: 26

Função Fibonacci recursiva

```
#include<stdio.h>
int fib(int n){
   if(n == 1 || n == 0){
         return 1;
      }
   else{
         return fib (n - 1) + fib (n - 2);
   }
}

int main (){
  scanf("%d", &n);
  for(int i = 0; i < n; i++){
     fib (i);
     printf("%d, ", fib(i));
  }
  return 0;
}
```

parte do estudo para achar o T(n).

/*

T(n)= 1+1+(n+1)+ n+2+n+n
T(n)= 4n +5
o(n)

//n 0
1+
(1+1+0) -> 0<0

total: 3

//n1
1+
(1+1+2) -> 0<1
  1+1+1
(0+1+0) -> 1<1

total: 8

//n2
1+
(1+1+2) -> 0<2
  1+1+1
(0+1+2) -> 1<2
  1+1+1
(0+1+0) -> 2<2

total: 15

//n3
1+
(1+1+2) -> 0<3
  1+1+1
(0+1+2) -> 1<3
  1+1+1
(0+1+2) -> 2<3
  1+1+1
  1+1+1
(0+1+0) -> 3<3

total: 24

*/

Seguindo o mesmo raciocínio da função anterior de Fatorial.
Então, nos dois casos de da função Fibonacci não há diferença. Por que a complexidade dos dois é a mesma. Ou seja, o(n).