Esercitazione 2

Esercizio 6.a

```
 i=1;
while (i <= f(n)) {
                a=a*a;
                i=i+1;}
```

$T(1)=a$

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

Complessità di f

```
int f (int n) {
        if (n == 1) return 1;
                else return n+f(n-1);
    }
```

$R(1)=1$

$R(n) = n + R(n-1)$ $O(n^2)$

Complessità del risultato

Esercizio 6.a

```
i=1;
while (i <= f(n)) {
            a=a*a;
            i=i+1;}

int f (int n) {
      if (n == 1) return 1;
            else return n+f(n-1);
   }
```

$T(1)=a$

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

Complessità di f

$R(1)=1$

$R(n) = n + R(n-1)$ $O(n^2)$

Complessità del risultato

numero iterazioni del while: $O(n^2)$ dipende dalla complessità del risultato di f

complessitá di una iterazione : $C[f(n)] = O(n)$ numero di chiamate ricorsive in f

complessitá del while : $O(n^2)* O(n)= O(n^3)$

Esercizio 6.b

```
i=1;
while (i <= f(n)) {
            a=a*a;
            i=i+1;}

int f (int n) {
            return n*(n+1)/2;
   }
```

$T(n) \in O(1)$ non si hanno chiamate ricorsive

$R(n) \in O(n^2)$

Esercizio 6.b

```
i=1;
while (i <= f(n)) {
            a=a*a;
            i=i+1;}
int f (int n) {
            return n*(n+1)/2;
   }
```

$T(n) \in O(1)$ non si hanno chiamate ricorsive

$R(n) \in O(n^2)$

numero iterazioni del while: $O(n^2)$

complessitá di una iterazione : $C[f(n)] = O(1)$

complessitá del while : $O(n^2)*O(1)= O(n^2)$

Esercizio 7

```
int i=F(n); for (int j=1; j<=i; j++) a+=b;

int F (int x) {if (x==1) return 1; else return 1+2*F(x-1);}
```

$T(1)=a$

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

Numero chiamate ricorsive in F

$R(1)=1$

$R(n) = 1+2R(n-1)$ $O(2^n)$

Complessità del risultato di F

Esercizio 7

```
int i=F(n); for (int j=1; j<=i; j++) a+=b;
int F (int x) {if (x==1) return 1; else return 1+2*F(x-1);}
```

$T(1)=a$

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

Numero chiamate ricorsive in F

$R(1)=1$

$R(n) = 1+2R(n-1)$ $O(2^n)$

Complessità del risultato di F

numero iterazioni del for: $O(2^n)$
complessitá di una iterazione : $O(1)$
complessitá del for : $O(2^n) * O(1) = O(2^n)$

complessitá del frammento di programma
C[i=F(n);] + C[for..]= $O(n) + O(2^n) = O(2^n)$

Esercizio 8

```
for (int j=1; j<=F(n)*F(n); j++)
            a+=F(n);

int F (int x) {
      if (x<=2) return 1;
            else return 3+3*F(x/3);
}
```

$T(0)=T(1)=T(2)=a$

$T(n) = b + T(n/3)$ $O(\log n)$

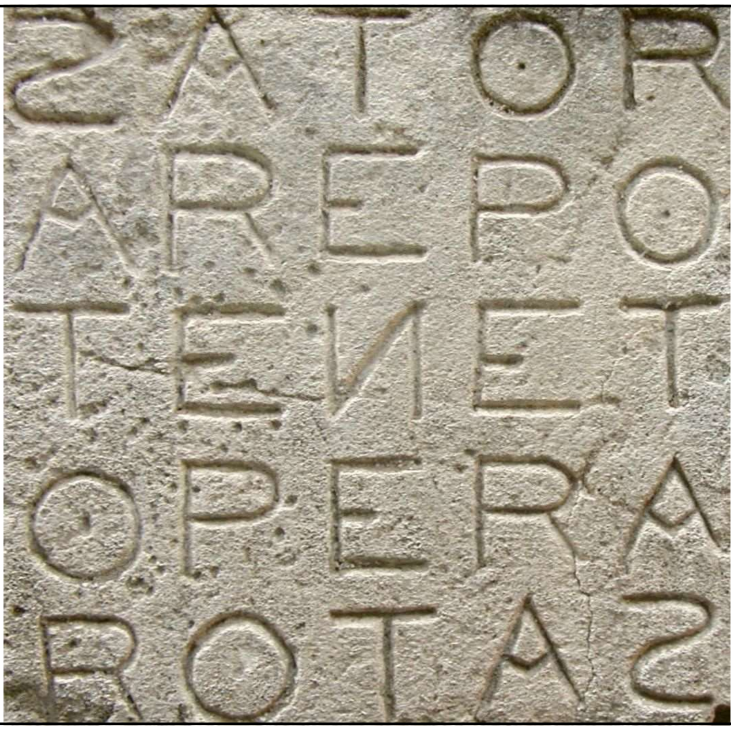
$R(0)=R(1)=R(2)=1$

$R(n) = 3+3*R(n/3)$ $O(n)$

numero iterazioni del for: $O(n^2)$

complessitá di una iterazione : C[F(n)]= $O(\log n)$

complessitá del for : $O(n^2) * O(\log n) = O(n^2 \log n)$



Quadrato magico

Array palindroma

```
int palindroma(int *a, int i=0, int j=n-1){
      if (j<i) return 1;
      if (a[i]==a[j]) return palindroma(a,i+1,j-1);
      return 0;
}
```

- T(1)=k1
- T(n)=K2+T(n-2)

- Palindroma su un’array di n elementi è **O(n)**

Es. 9

```
P(B,1,n)

void P(int  A [], int i, int j) {
        if (i<j) {
                int k=(i+j)/2;
                P(A,i,k-1);
                P(A,k+1,j);
                for (int r=i; r<=j; r++)  A[r]=2*A[r];
                P(A,i,k-1);
        }
}
```

T(1)=a

T(n) =bn+ 3T(n/2)

$O(n^{1.5})$ $\log_2 3 = 1.5$

Es. 10

```
int f(int x) {
        if (x==1) return 1;
                else return 1+ f(x/2);
}

int g(int x) {
        if (x==1) return 1;
                else return 2+2*g(x-1);
}

for (int i=1; i <=g(f(n)); i++;) a++;
```


Es. 10

```
int f(int x) {
        if (x==1) return 1;
                else return 1+ f(x/2);
}
```

Tf(1)=a
Tf(n) =b + Tf(n/2) O(logn)

Rf(1)=a
Rf(n) =b + Rf(n/2) O(logn)

Es. 10

```
int g(int x) {
        if (x==1) return 1;
                else return 2+2*g(x-1);
}
```

Tg(1)=a
Tg(m) =b + Tg(m-1) O(m)

Rg(1)=a
Rg(m) =2+2 Rg(m-1) $O(2^m)$

Es. 10

```
for (int i=1; i <=g(f(n)); i++;) a++;
```

numero iterazioni del for:

$O(\ Rg(Rf(n))\ ) = O(\ Rg(\log_2 n)\ ) = O(2^{\log n}\ ) = O(n)$

complessitá di una iterazione :
C[f(n)] + C[g(Rf(n))] =
C[f(n)] + C[g(log n)] =
O(log n) + O(log n)= O(log n)

complessitá del for : O(n)* O(log n)= O(nlog n)