**Fattoriale Ricorsivo**

Int fact (int n) {

If(n==0) return 1

Else return n\*fact(n-1);

**Fattoriale Iterativo**

int fattorialeIter(int n){

int i,prod=1;

if(n==0)return 1;

for(i=1;i<=n;i++)

prod=prod\*i;

return prod;

}

**Equazione di Ricorrenza del fattoriale**

T(n)=c

T(n) = T(n-1) +c

Effettua una sola chiamata ricorsiva, rientriamo nel caso 1.a **Lavoro di combinazione costante senza partizione dei dati.**

Fattoriale con ricorsione Tail

Int fact(int n) {

Return factIn(n,1,1)

}

Int factIt(int n, int F, int i) {

If(i<=n) {

F=i\*f;

i=i+1;

return factIt(n,F,i);

}

return F; -> F è l’accumulatore

}

**Fibonacci Ricorsivo**

int fibonacciRic(int n){

if(n==1||n==2)return 1;

return fibonacci(n-1)+fibonacci(n-2);

}

**Equazione di ricorrenza** T(n)= T(1)=c

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

Effettua due chiamate ricorsive siamo nel caso 1.a **Lavoro di combinazione costante senza partizione dei dati**

**//Equazione di ricorrenza della RicercaBinariaRic del tempo di esecuzione T(n)=T(n/2)+c;**

**//Lavoro di combinazione costante con partizionamento dell'input siamo nel caso 1.b**

int RicercaBinariaRic1(int \*arr, int inizio, int fine, int k) {

int centro;

if(inizio > fine)

return -1;

else {

centro = (inizio+fine)/2;

if( k== arr[centro])

return centro;

else if(k > arr[centro])

return RicercaBinariaRic1(arr, centro+1,fine,k);

else

return RicercaBinariaRic1(arr, inizio, centro-1,k);

}

}

//E' la stessa di quella di sopra ma fa da interfaccia a quella di sopra

int RicercaBinariaRic(int \*arr, int size, int k) {

return RicercaBinariaRic1(arr, 0, size-1,k);