- Studiamo dunque il $T(n)$ tempo di calcolo di un'implementazione di algoritmo nel suo caso peggiore.
- il $T(n)$ si deduce dallo pseudo-codice imponendo che ogni operazione elimentare abbia costo unitario

2 strategie di calcolo $T(n)$

- la prima e più complicata è quella di assegnare ad ogni riga di Pseudo-codice valore pari ad una costante c_n e poi moltiplicarla per il numero di volte che quella riga viene eseguita dal completore

esempio

```
Due_occorrenze(A)
  output = False
    for i = 0 to A_lenght - 2
      for j = i+1 to A_lenght - 1
        if (A[j] == A[i])
            output = True
        Return output
```

COSTO	N° RIPETIZIONI
c_1	(1)
c_2	n
c_3	$0.5n^2+1.5n+1$
c_4	
c_5	
c_6	(1)

- il return e l'assegnazione di output sono eseguiti una sola volta
- i cicli for invece: l'esterno esegue l'operazione n volte perché si ferma a n-1 e poi l'ultima, ossia a n, esegue la verifica. L'interno si ripete n-1 volte ma parte da $i+1 \rightarrow n-1+i$

tramite la somma dei contributi di tutte le righe otteniamo che la sua complessità è pari a $0.5n^2+1.5n$

per $n \gg 0 \longrightarrow n^2 \longrightarrow \Theta(n^2)$

- la seconda e più semplice consiste nell'esaminare codice assegnando ad ogni riga un costo asintotico ricordando alcune fz:

- istruzioni semplice: $\Theta(1)$
- sequenze di istruzioni semplici: $\Theta(1)$
- sequenze istruzioni generiche: somma Θ di ogni singola operazione

ISTRUZIONI CONDIZIONALI

```
IF <condizione> then
  <parte then>
else
  <parte else>
```

- Calcolare $T(n)$ di questa funzione significa:
  -> sapere se la condizione si verifica o meno
  -> Calcoliamo il limite superiore di questo tempo:

- Somma di costi:
- somma O-grande condizione e O grande maggiore tra parte then e parte else.
- somma Ω condizione e Ω + piccolo fra parte then e else.

ISTRUZIONI RIPETITIVE

```
FACT (n)
f = 1
 for K = 2 to n
   f - f * k
return K
```

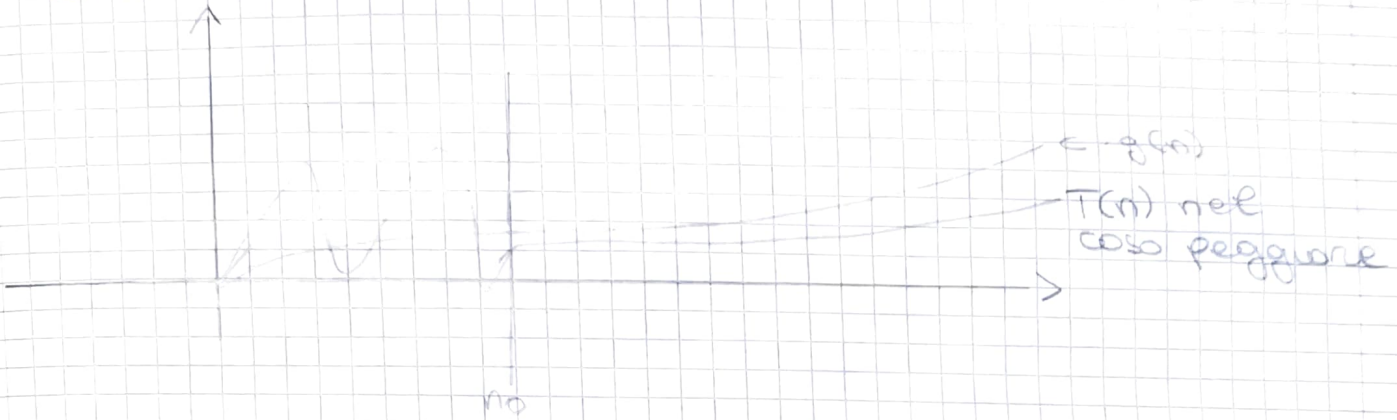
- voluto il costo del ciclo for $\sim \Theta(n)$
- calcolo il costo delle istruzioni del blocco ripetuto $\sim \Theta(1)$
- Costo totale: prodotto tra i costi $\sim \Theta(n \cdot 1) = \Theta(n)$

Algoritmi e complessità $O(f(n))$

- Sia $T(n)$ il tempo di esecuzione di un algoritmo nel caso peggiore.

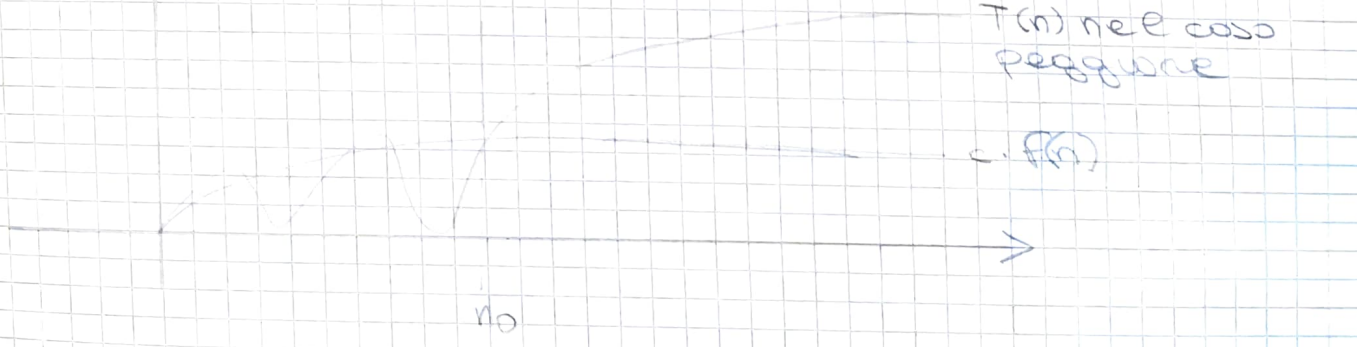
Ha complessità temporale $\Theta(f(n))$ se $T(n) = O(f(n))$

- ossia $f(n)$ ha un limite superiore $\Rightarrow$ al massimo impiega T tempo a eseguire tutte le istruzioni



Algoritmi di complessità $\Omega(f(n))$

Ha complessità temporale $\Omega(f(n))$ se $T(n) = \Omega(f(n))$



Algoritmi di complessità $\Theta(f(n))$

Ha complessità temporale $\Theta(f(n))$ se $T(n)$ è $\Theta(f(n))$