Formule di ricorrenza

- equazioni o disequazioni che descrivono una funzione in termini del suo valore su input + piccoli
  → prevede casi base e casi induttivi

$$T(n) = \begin{cases} a & n \neq 0 \quad \text{(caso base)} \\ T(n-1) + g(n) & n > 0 \end{cases}$$

*sostituisco

- Le soluzioni di queste formule non sono sempre facili da trovare
- Se si usano per lo studio di complessità asintotiche i casi base sono omessi:
  - Se $T(n)$ è il tempo di esecuzione di un algoritmo, $T(n) = \Theta(1)$ per n piccolo

L'equazione ammette come soluzione: $T(n) = a + \sum_{k=1}^{n} g(k)$ *

$$\begin{cases} a + \sum_{k=1}^{0} g(k) = a + 0 = \boxed{a} & n \neq 0 \end{cases}$$

$$T(n-1) = a + \sum_{k=1}^{n-1} g(k)$$

$$T(n) = T(n-1) + g(n) \Rightarrow T(n) = a + \sum_{k=1}^{n-1} g(k) + g(n)$$

← porto nello sommatorio

$$= a + \sum_{k=1}^{n} g(k)$$

$$T(n) = a + \sum_{k=1}^{n} g(k)$$

$$a + \sum_{k=1}^{n} g(k) = a + \sum_{k=1}^{n} g(k)$$

verificato

Se applico la formula di ricorrenza Fact_Ric(n) ottengo:

$$T(n) = \begin{cases} \Theta(1) & n=0 \\ T(n-1)+\Theta(1) & n>0 \end{cases}$$

complessità fattoriale

So che:

$$T(n) = \begin{cases} a & n=0 \\ T(n-1)+g(n) & n>0 \end{cases}$$

ha soluzione $T(n) = a + \sum_{k=1}^{n} g(k)$

Quindi complessità fattoriale è: $\Theta(1) + \sum_{k=1}^{n} \Theta(1) = \Theta(n)$

Selection sort Ricorsivo

```
Selection (A)
  Selection_Ric (A,0)

Selection_Ric (A,i)
if i ≤ lenght -1
 min = i
 for j = i+1 to A. lenght -1
  if A[j] < A[min]
     min = j
  temp = A[i]
  A[i] = A[min]
  A[min] = temp
   Selection_Fact (A, i+1)
```

Complessità $= \Theta(1) + \sum_{k=1}^{n} \Theta(k) = \Theta(n^2)$

tecnica divide et impera:

- Suddivido il problema originale in diversi sottoproblemi:
  - hanno dimensione + piccola
  - possono essere a loro volta divisi
  - se raggiungono la minima dimensione, la loro soluzione è banale

Merge Sort: (stabile non in loco)

- divido la sequenza in input in sottosequenze più piccole, le ordino tramite merge sort e poi le fondo insieme

Tempo esecuzione merge sort

caso base: $\Theta(1)$

divide: calcolo $n/2$ $\Rightarrow$ costo $D(n) = \Theta(1)$

impera: · ogni sottoproblema ha dim. $n/2$
· i sottoproblemi sono 2 $\Rightarrow 2T \cdot n/2$

combina: $C(n) = \Theta(n)$

Complessivamente

$$T(n) = \begin{cases} \Theta(1) & n = 0 \text{ o } n = 1 \\ 2 \cdot T(n/2) + D(n) + C(n) & n > 1 \end{cases}$$

(divisione, merge)

- poiché $D(n) + C(n) = \Theta(1) + \Theta(n) = \Theta(n)$

$$T(n) = \begin{cases} \Theta(1) & n = 0 \quad n = 1 \\ 2T(n/2) + \Theta(n) & n > 1 \end{cases}$$

ammette soluzione $\underline{\Theta(n \log n)}$ caso peggiore

Master theorem (teorema dell'esperto)

- problemi in cui $T(n)$ dipende dal tempo di + me calcolato su una porzione di n asse n/b

  $T(n) = T(n/b) \quad b \geq 1$

- il tempo è poi moltiplicato per una costante $a \neq b \geq 1$, e sommato con tempo polinomiale $O(n^k)$

$$T(n) = \begin{cases} \Theta(1) & n = 0 \\ a \cdot T(n/b) + O(n^k) & n > 0 \end{cases}$$

! Le equazioni di ricorrenza ammette soluzione:

1) se $a < b^k$ allora $T(n) = \Theta(n^k)$
2) se $a = b^k$ allora $T(n) = \Theta(n^k \log n)$
3) se $a > b^k$ allora $T(n) = \Theta(n^{\log_b a})$

- se $b^k > a \Rightarrow \Theta(n^k) \gg a \cdot T(n/b) \Rightarrow$ trascuro $a \cdot T(n/b)$
- se $a > b^k \Rightarrow a \cdot T(n/b) \gg \Theta(n^k) \Rightarrow$ trascuro $\Theta(n^k)$
- se $a = b^k \Rightarrow$ non posso trascurare nulla

1 caso $a < b^k$ → sorgente

La somma del costo di tutti i livelli è:

$$T(n) = \sum_{i=0}^{h} a^i \left(\frac{n}{b^i}\right)^k = \sum_{i=0}^{h} a^i \frac{n^k}{b^{ik}} = n^k \sum_{i=0}^{h} \left(\frac{a}{b^k}\right)^i$$

$\sum_{i=0}^{h} \left(\frac{a}{b^k}\right)^i$ è una serie geometrica con ragione $r = \frac{a}{b^k}$ che converge a $1/(1-r)$ con $r<1$ e $a<b^k$

- dunque $T(n) = O(n^k)$

2 caso $a > b^k$

$$T(n) = n^k \sum_{i=0}^{h} \left(\frac{a}{b^k}\right)^i$$

- Se $a > b^k$, ossia $r>1$

$$\frac{1-r^n}{1-r} = \frac{r^n - 1}{r-1} \Rightarrow \in O(r^n)$$

$$r^n = \left(\frac{a}{b^k}\right)^{\log_b n} = \frac{a^{\log_b n}}{b^{k \log_b n}} = \frac{a^{\log_b n}}{(b^{\log_b n})^k} = \frac{a^{\log_b n}}{n^k}$$

$b^{\log_b n} = n$

Dunque:

$$T(n) = O(n^k) \cdot O(r^n) = O(n^k) \cdot O\left(\frac{a^{\log_b n}}{n^k}\right)$$

ossia:

$$T(n) = O(a^{\log_b n}) = O(n^{\log_b a})$$

Esempi

$T(n) = 9T(n/3) + n \Rightarrow a=9 \quad b=3 \quad p = n^k = n \quad k=1$

$\Rightarrow a > b^k$

$$T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_3 9}) = \Theta(n^2)$$