

Lezione 5

Sommario della Lezione

Oggi parliamo di Analisi di Algoritmi

Sommario della Lezione

Oggi parliamo di Analisi di Algoritmi

- ▶ Analisi di Algoritmi = valutazione delle risorse usate da algoritmi per risolvere un dato problema

Sommario della Lezione

Oggi parliamo di Analisi di Algoritmi

- ▶ Analisi di Algoritmi = valutazione delle risorse usate da algoritmi per risolvere un dato problema
- ▶ Risorse =
Tempo impiegato dall'algoritmo

Sommario della Lezione

Oggi parliamo di Analisi di Algoritmi

- ▶ Analisi di Algoritmi = valutazione delle risorse usate da algoritmi per risolvere un dato problema
- ▶ Risorse =
 - Tempo impiegato dall'algoritmo
 - Memoria usata dall'algoritmo

Sommario della Lezione

Oggi parliamo di Analisi di Algoritmi

- ▶ Analisi di Algoritmi = valutazione delle risorse usate da algoritmi per risolvere un dato problema
- ▶ Risorse =
 - Tempo impiegato dall'algoritmo
 - Memoria usata dall'algoritmo
- ▶ Tipicamente analizzeremo gli algoritmi in termini del **tempo di esecuzione** da loro impiegato per produrre l'output richiesto.

In pratica, il tempo di esecuzione dipende (anche)
dall'implementazione

In pratica, il tempo di esecuzione dipende (anche)
dall'implementazione

Schema di Implementazione di Algoritmi

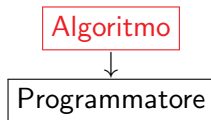
In pratica, il tempo di esecuzione dipende (anche)
dall'implementazione

Schema di Implementazione di Algoritmi

Algoritmo

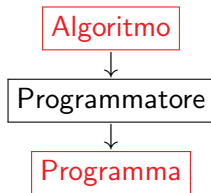
In pratica, il tempo di esecuzione dipende (anche)
dall'implementazione

Schema di Implementazione di Algoritmi



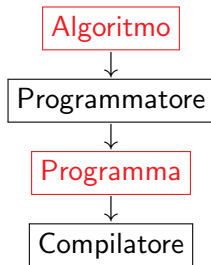
In pratica, il tempo di esecuzione dipende (anche)
dall'implementazione

Schema di Implementazione di Algoritmi



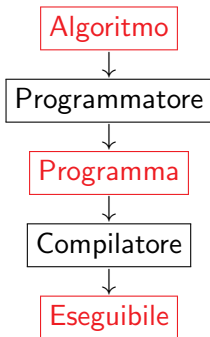
In pratica, il tempo di esecuzione dipende (anche)
dall'implementazione

Schema di Implementazione di Algoritmi



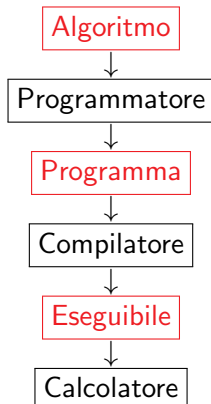
In pratica, il tempo di esecuzione dipende (anche)
dall'implementazione

Schema di Implementazione di Algoritmi



In pratica, il tempo di esecuzione dipende (anche)
dall'implementazione

Schema di Implementazione di Algoritmi



Il tempo di esecuzione dipende quindi...

Il tempo di esecuzione dipende quindi...

- ▶ Dall'abilità del programmatore e dal linguaggio di programmazione usato

Il tempo di esecuzione dipende quindi...

- ▶ Dall'abilità del programmatore e dal linguaggio di programmazione usato
- ▶ Dal linguaggio macchina del computer

Il tempo di esecuzione dipende quindi...

- ▶ Dall'abilità del programmatore e dal linguaggio di programmazione usato
- ▶ Dal linguaggio macchina del computer
- ▶ Dal codice che il compilatore genera

Il tempo di esecuzione dipende quindi...

- ▶ Dall'abilità del programmatore e dal linguaggio di programmazione usato
- ▶ Dal linguaggio macchina del computer
- ▶ Dal codice che il compilatore genera
- ▶ Dalla taglia del problema (=la "dimensione" dell' input)

Il tempo di esecuzione dipende quindi...

- ▶ Dall'abilità del programmatore e dal linguaggio di programmazione usato
- ▶ Dal linguaggio macchina del computer
- ▶ Dal codice che il compilatore genera
- ▶ Dalla taglia del problema (=la "dimensione" dell' input)
- ▶ Da come é fatto l'input (ad es., ordinato/non-ordinato)

Il tempo di esecuzione dipende quindi...

- ▶ Dall'abilità del programmatore e dal linguaggio di programmazione usato
- ▶ Dal linguaggio macchina del computer
- ▶ Dal codice che il compilatore genera
- ▶ Dalla taglia del problema (=la "dimensione" dell' input)
- ▶ Da come é fatto l'input (ad es., ordinato/non-ordinato)
- ▶ Dal numero di operazioni elementari eseguite dall' algoritmo

Quindi...

- ▶ Quantificheremo le risorse usate da un algoritmo **a meno** di una costante moltiplicativa (useremo quindi le notazioni asintotiche O , Θ e Ω).

Quindi...

- ▶ Quantificheremo le risorse usate da un algoritmo **a meno** di una costante moltiplicativa (useremo quindi le notazioni asintotiche O , Θ e Ω).
- ▶ Perché?

Quindi...

- ▶ Quantificheremo le risorse usate da un algoritmo **a meno** di una costante moltiplicativa (useremo quindi le notazioni asintotiche O , Θ e Ω).
- ▶ Perché?
Perché le costanti moltiplicative dipendenti dai primi 3 aspetti prima enunciati non sono a priori quantificabili, in quanto dipendono da aspetti a noi ignoti

Quindi...

- ▶ Quantificheremo le risorse usate da un algoritmo **a meno** di una costante moltiplicativa (useremo quindi le notazioni asintotiche O , Θ e Ω).
- ▶ Perchè?
Perchè le costanti moltiplicative dipendenti dai primi 3 aspetti prima enunciati non sono a priori quantificabili, in quanto dipendono da aspetti a noi ignoti
Inoltre, per input molto grandi (il che corrisponde a situazioni pratiche) l'influenza che tali costanti hanno sul "reale" tempo di esecuzione di un algoritmo è minore

Cosa sono le “Operazioni elementari”

Cosa sono le “Operazioni elementari”

- ▶ Misureremo la complessità di un algoritmo in termini del numero di operazioni elementari usate

Cosa sono le “Operazioni elementari”

- ▶ Misureremo la complessità di un algoritmo in termini del **numero di operazioni elementari** usate
 - ▶ Esempi di operazioni elementari: addizione e moltiplicazione di numeri (se piccoli...), confronto tra elementi, seguire un puntatore...

Cosa sono le “Operazioni elementari”

- ▶ **Misureremo la complessità di un algoritmo in termini del numero di operazioni elementari usate**
 - ▶ Esempi di operazioni elementari: addizione e moltiplicazione di numeri (se piccoli...), confronto tra elementi, seguire un puntatore...
- ▶ Le operazioni elementari richiedano un tempo **costante** per la loro esecuzione (=una unità di tempo)

Cos'è "La taglia dell'input"

Cos'è “La taglia dell'input”

Il tempo di esecuzione di un algoritmo in generale dipende quanto grande è l'input (taglia dell'input).

Cos'è “La taglia dell'input”

Il tempo di esecuzione di un algoritmo in generale dipende quanto grande è l'input (taglia dell'input).

Ma cosa intendiamo esattamente per *taglia dell'input*?

- ▶ Criterio di costo logaritmico: la taglia dell'input è il numero di bits necessari per rappresentarlo

Cos'è “La taglia dell'input”

Il tempo di esecuzione di un algoritmo in generale dipende quanto grande è l'input (taglia dell'input).

Ma cosa intendiamo esattamente per *taglia dell'input*?

- ▶ Criterio di costo logaritmico: la taglia dell'input è il numero di bits necessari per rappresentarlo
- ▶ Criterio di costo uniforme: la taglia dell'input è il numero di "elementi" che lo costituiscono

Cos'è "La taglia dell'input"

Il tempo di esecuzione di un algoritmo in generale dipende quanto grande è l'input (taglia dell'input).

Ma cosa intendiamo esattamente per *taglia dell'input*?

- ▶ Criterio di costo logaritmico: la taglia dell'input è il numero di bits necessari per rappresentarlo
- ▶ Criterio di costo uniforme: la taglia dell'input è il numero di "elementi" che lo costituiscono
- ▶ Nell'ipotesi che ogni istruzione del computer richieda un'unità di tempo e che un numero o "elemento" richieda una parola di memoria, le due misure coincidono a meno di una costante moltiplicativa

Esempi

Problema	taglia dell'input	operazione elementare
Trovare x in una lista	# di elementi della lista	confronto tra elementi

Esempi

Problema	taglia dell'input	operazione elementare
Trovare x in una lista Moltiplicazione di matrici	# di elementi della lista dimensione delle matrici	confronto tra elementi moltiplicazione di due numeri

Esempi

Problema	taglia dell'input	operazione elementare
Trovare x in una lista	# di elementi della lista	confronto tra elementi
Moltiplicazione di matrici	dimensione delle matrici	moltiplicazione di due numeri
Ordinamento di un array	# di elementi dell'array	confronto tra elementi

Esempi

Problema	taglia dell'input	operazione elementare
Trovare x in una lista	# di elementi della lista	confronto tra elementi
Moltiplicazione di matrici	dimensione delle matrici	moltiplicazione di due numeri
Ordinamento di un array	# di elementi dell'array	confronto tra elementi
Visita di un albero/grafico	#numero di nodi/archi	seguire un puntatore

Esempi

Problema	taglia dell'input	operazione elementare
Trovare x in una lista	# di elementi della lista	confronto tra elementi
Moltiplicazione di matrici	dimensione delle matrici	moltiplicazione di due numeri
Ordinamento di un array	# di elementi dell'array	confronto tra elementi
Visita di un albero/grafico	#numero di nodi/archi	seguire un puntatore
Moltiplicazione di due numeri	# bits per rappresentarli	addizione/sottrazione

Esempi

Problema	taglia dell'input	operazione elementare
Trovare x in una lista	# di elementi della lista	confronto tra elementi
Moltiplicazione di matrici	dimensione delle matrici	moltiplicazione di due numeri
Ordinamento di un array	# di elementi dell'array	confronto tra elementi
Visita di un albero/grafico	#numero di nodi/archi	seguire un puntatore
Moltiplicazione di due numeri	# bits per rappresentarli	addizione/sottrazione

Nell'ultimo caso della moltiplicazione di due numeri abbiamo usato il criterio di costo logaritmico per misurare la taglia dell'input, nei primi quattro casi abbiamo usato il criterio di costo uniforme.

Anche la forma dell'input influenza la complessità

- ▶ La complessità di un algoritmo può dipendere anche da come è fatto l'input (ad es., ordinare n elementi può essere fatto più velocemente se essi sono già (quasi) ordinati)

Anche la forma dell'input influenza la complessità

- ▶ La complessità di un algoritmo può dipendere anche da come è fatto l'input (ad es., ordinare n elementi può essere fatto più velocemente se essi sono già (quasi) ordinati)
- ▶ Useremo come strumento di analisi il **caso peggiore**: il tempo usato dall'algoritmo nella situazione più "sfavorevole" in input.

Anche la forma dell'input influenza la complessità

- ▶ La complessità di un algoritmo può dipendere anche da come è fatto l'input (ad es., ordinare n elementi può essere fatto più velocemente se essi sono già (quasi) ordinati)
- ▶ Useremo come strumento di analisi il **caso peggiore**: il tempo usato dall'algoritmo nella situazione più "sfavorevole" in input.
- ▶ La complessità di tempo di un algoritmo \mathcal{A} , su input di taglia n , che denoteremo con $T(n)$ sarà quindi
 $T(n)$ = il massimo tempo richiesto da \mathcal{A} , dove **il massimo** è calcolato su tutti i possibili input di taglia n

Anche la forma dell'input influenza la complessità

- ▶ La complessità di un algoritmo può dipendere anche da come è fatto l'input (ad es., ordinare n elementi può essere fatto più velocemente se essi sono già (quasi) ordinati)
- ▶ Useremo come strumento di analisi il **caso peggiore**: il tempo usato dall'algoritmo nella situazione più "sfavorevole" in input.
- ▶ La complessità di tempo di un algoritmo \mathcal{A} , su input di taglia n , che denoteremo con $T(n)$ sarà quindi
 $T(n)$ = il massimo tempo richiesto da \mathcal{A} , dove **il massimo** è calcolato su tutti i possibili input di taglia n
- ▶ Ciò ci darà un limite superiore alla complessità dell'algoritmo, **su tutti i possibili input** (nel senso che \mathcal{A} non richiederà mai tempo superiore a $T(n)$ su di un input di taglia n).

Esempio

Input: $a=a[0]a[1]\dots a[n-1]$ ed un numero x .

Esempio

Input: $a=a[0]a[1]\dots a[n-1]$ ed un numero x .

Output: $a[i]$ se $x = a[i]$, "x non c'è", altrimenti.

Esempio

Input: $a=a[0]a[1]\dots a[n-1]$ ed un numero x .

Output: $a[i]$ se $x = a[i]$, "x non c'è", altrimenti.

```
Scopri(a,x)
```

Esempio

Input: $a=a[0]a[1]\dots a[n-1]$ ed un numero x .

Output: $a[i]$ se $x = a[i]$, "x non c'è", altrimenti.

```
Scopri(a,x)
```

```
i=0
```

```
WHILE(i < n && a[i]!=x){
```

```
    i=i+1 }
```


Esempio

Input: $a=a[0]a[1]\dots a[n-1]$ ed un numero x .

Output: $a[i]$ se $x = a[i]$, "x non c'è", altrimenti.

```
Scopri(a,x)
i=0
WHILE(i < n && a[i]!=x){
    i=i+1 }
IF (i=n){
    RETURN ('x non c'è')
```

Esempio

Input: $a=a[0]a[1]\dots a[n-1]$ ed un numero x .

Output: $a[i]$ se $x = a[i]$, "x non c'è", altrimenti.

```
Scopri(a,x)
i=0
WHILE(i < n && a[i]!=x){
    i=i+1 }
IF (i=n){
    RETURN ('x non c'è')
    ELSE RETURN ('x=a[i]') }
```

Esempio

Input: $a=a[0]a[1]\dots a[n-1]$ ed un numero x .

Output: $a[i]$ se $x = a[i]$, "x non c'è", altrimenti.

```
Scopri(a,x)
i=0
WHILE(i < n && a[i]!=x){
    i=i+1 }
IF (i=n){
    RETURN ('x non c'è')
    ELSE RETURN ('x=a[i]') }
```

A seconda del caso che x appaia o meno in a (ed in quale posizione di a esso appaia), l'algoritmo $\text{Scopri}(a,x)$ può eseguire un *qualunque* numero di operazioni comprese tra 1 ed n .

Esempio

Input: $a=a[0]a[1]\dots a[n-1]$ ed un numero x .

Output: $a[i]$ se $x = a[i]$, "x non c'è", altrimenti.

```
Scopri(a,x)
i=0
WHILE(i < n && a[i]!=x){
    i=i+1 }
IF (i=n){
    RETURN ('x non c'è')
    ELSE RETURN ('x=a[i]')} }
```

A seconda del caso che x appaia o meno in a (ed in quale posizione di a esso appaia), l'algoritmo $\text{Scopri}(a,x)$ può eseguire un *qualunque* numero di operazioni comprese tra 1 ed n .

Possiamo dire che la complessità $T(n)$ è (in generale) $O(n)$.

Esempio

Input: $a=a[0]a[1]\dots a[n-1]$ ed un numero x .

Output: $a[i]$ se $x = a[i]$, “ x non c'è”, altrimenti.

```
Scopri(a,x)
i=0
WHILE(i < n && a[i]!=x){
    i=i+1 }
IF (i=n){
    RETURN (“x non c'è”)
    ELSE RETURN (“x=a[i]”) }
```

A seconda del caso che x appaia o meno in a (ed in quale posizione di a esso appaia), l'algoritmo $\text{Scopri}(a,x)$ può eseguire un *qualunque* numero di operazioni comprese tra 1 ed n .

Possiamo dire che la complessità $T(n)$ è (in generale) $O(n)$. Visto che abbiamo convenuto di valutare la complessità di un algoritmo in base al suo caso peggiore, è più corretto dire che complessità $T(n)$ è $\Theta(n)$,

Esempio

Input: $a=a[0]a[1]\dots a[n-1]$ ed un numero x .

Output: $a[i]$ se $x = a[i]$, “ x non c'è”, altrimenti.

```
Scopri(a,x)
i=0
WHILE(i < n && a[i]!=x){
    i=i+1 }
IF (i=n){
    RETURN (“x non c'è”)
    ELSE RETURN (“x=a[i]”) }
```

A seconda del caso che x appaia o meno in a (ed in quale posizione di a esso appaia), l'algoritmo $\text{Scopri}(a,x)$ può eseguire un *qualunque* numero di operazioni comprese tra 1 ed n .

Possiamo dire che la complessità $T(n)$ è (in generale) $O(n)$. Visto che abbiamo convenuto di valutare la complessità di un algoritmo in base al suo caso peggiore, è più corretto dire che complessità $T(n)$ è $\Theta(n)$, in quanto nel caso peggiore (ad esempio nel caso in cui x non appare in a) l'algoritmo $\text{Scopri}(a,x)$ esegue $\Omega(n)$ operazioni.

Quindi:

Se diciamo che la complessità $T(n)$ di un algoritmo \mathcal{A} è $O(n^2)$, vuol dire che, avendo un qualsivoglia input di dimensione n , l'algoritmo \mathcal{A} non richiede mai tempo superiore a cn^2 , per produrre il suo output, per c opportuna e n sufficientemente grande.

Quindi:

Se diciamo che la complessità $T(n)$ di un algoritmo \mathcal{A} è $O(n^2)$, vuol dire che, avendo un qualsivoglia input di dimensione n , l'algoritmo \mathcal{A} non richiede mai tempo superiore a cn^2 , per produrre il suo output, per c opportuna e n sufficientemente grande.

Se diciamo invece che la complessità $T(n)$ di un algoritmo \mathcal{A} è $\Omega(n^2)$, vuol dire che \mathcal{A} richiede almeno tempo cn^2 , per produrre il suo output nel caso peggiore, per c opportuna e n sufficientemente grande.

Quindi:

Se diciamo che la complessità $T(n)$ di un algoritmo \mathcal{A} è $O(n^2)$, vuol dire che, avendo un qualsivoglia input di dimensione n , l'algoritmo \mathcal{A} non richiede mai tempo superiore a cn^2 , per produrre il suo output, per c opportuna e n sufficientemente grande.

Se diciamo invece che la complessità $T(n)$ di un algoritmo \mathcal{A} è $\Omega(n^2)$, vuol dire che \mathcal{A} richiede almeno tempo cn^2 , per produrre il suo output nel caso peggiore, per c opportuna e n sufficientemente grande.

Quindi NON si intende dire che \mathcal{A} impiega tempo cn^2 , su ogni input di taglia n , ma nel secondo caso si intende che ESISTE almeno un input di taglia n (con n sufficientemente grande) su cui \mathcal{A} richiede tempo cn^2

Quindi:

Se diciamo che la complessità $T(n)$ di un algoritmo \mathcal{A} è $O(n^2)$, vuol dire che, avendo un qualsivoglia input di dimensione n , l'algoritmo \mathcal{A} non richiede mai tempo superiore a cn^2 , per produrre il suo output, per c opportuna e n sufficientemente grande.

Se diciamo invece che la complessità $T(n)$ di un algoritmo \mathcal{A} è $\Omega(n^2)$, vuol dire che \mathcal{A} richiede almeno tempo cn^2 , per produrre il suo output nel caso peggiore, per c opportuna e n sufficientemente grande.

Quindi NON si intende dire che \mathcal{A} impiega tempo cn^2 , su ogni input di taglia n , ma nel secondo caso si intende che ESISTE almeno un input di taglia n (con n sufficientemente grande) su cui \mathcal{A} richiede tempo cn^2 (e nel primo caso si intende che NON esistono input per cui l'algoritmo richiede tempo $> cn^2$).

Esempio

```
Alg(n)
1. x=0
2. IF (n è pari){
3.     RETURN 0
4.     ELSE FOR(i=1; i<n+1; i=i+1){
5.         x=x+1
           }
       }
RETURN x
```

Esempio

```
Alg(n)
1. x=0
2. IF (n è pari){
3.     RETURN 0
4.     ELSE FOR(i=1; i<n+1; i=i+1){
5.         x=x+1
           }
       }
RETURN x
```

La complessità dell'algoritmo *nel caso peggiore* è $\Theta(n)$.

Esempio

```
Alg(n)
1. x=0
2. IF (n è pari){
3.     RETURN 0
4.     ELSE FOR(i=1; i<n+1; i=i+1){
5.         x=x+1
           }
       }
RETURN x
```

La complessità dell'algoritmo *nel caso peggiore* è $\Theta(n)$.

Possiamo dire che l'algoritmo esegue $\Theta(n)$ passi *per ogni input*?

Esempio

```
Alg(n)
1. x=0
2. IF (n è pari){
3.     RETURN 0
4.     ELSE FOR(i=1; i<n+1; i=i+1){
5.         x=x+1
           }
       }
RETURN x
```

La complessità dell'algoritmo *nel caso peggiore* è $\Theta(n)$.

Possiamo dire che l'algoritmo esegue $\Theta(n)$ passi *per ogni* input?

No! Infatti esiste un numero infinito di input su cui l'algoritmo esegue *un solo passo*.

Esempio

```
Alg(n)
1. x=0
2. IF (n è pari){
3.     RETURN 0
4.     ELSE FOR(i=1; i<n+1; i=i+1){
5.         x=x+1
           }
       }
RETURN x
```

La complessità dell'algoritmo *nel caso peggiore* è $\Theta(n)$.

Possiamo dire che l'algoritmo esegue $\Theta(n)$ passi *per ogni* input?

No! Infatti esiste un numero infinito di input su cui l'algoritmo esegue *un solo passo*.

Ciò che possiamo dire è che l'algoritmo non esegue mai più di $O(n)$ passi *per ogni possibile input*,

Esempio

```
Alg(n)
1. x=0
2. IF (n è pari){
3.     RETURN 0
4.     ELSE FOR(i=1; i<n+1; i=i+1){
5.         x=x+1
           }
       }
RETURN x
```

La complessità dell'algoritmo *nel caso peggiore* è $\Theta(n)$.

Possiamo dire che l'algoritmo esegue $\Theta(n)$ passi *per ogni* input?

No! Infatti esiste un numero infinito di input su cui l'algoritmo esegue *un solo passo*.

Ciò che possiamo dire è che l'algoritmo non esegue mai più di $O(n)$ passi *per ogni possibile input*, e che esegue almeno n passi *per certi input*.

Effettueremo (quasi) esclusivamente analisi nel caso peggiore, usando le seguenti regole:

- ▶ il costo di istruzioni semplici, quali assegnazione, letture/scrittura di variabili è $O(1)$ (ovvero, una costante che non dipende dall'input)

Effettueremo (quasi) esclusivamente analisi nel caso peggiore, usando le seguenti regole:

- ▶ il costo di istruzioni semplici, quali assegnazione, letture/scrittura di variabili è $O(1)$ (ovvero, una costante che non dipende dall'input)
- ▶ il costo di istruzioni tipo IF ELSE è pari al tempo per effettuare il test (tipicamente $O(1)$) piú $O(\text{costo della alternativa piú costosa})$

Effettueremo (quasi) esclusivamente analisi nel caso peggiore, usando le seguenti regole:

- ▶ il costo di istruzioni semplici, quali assegnazione, letture/scrittura di variabili è $O(1)$ (ovvero, una costante che non dipende dall'input)
- ▶ il costo di istruzioni tipo IF ELSE è pari al tempo per effettuare il test (tipicamente $O(1)$) piú $O(\text{costo della alternativa piú costosa})$
- ▶ il costo di iterazioni (FOR, WHILE, REPEAT) è pari alla somma su tutte le iterazioni del costo di ogni iterazioni

Valutazione di progressioni geometriche $\sum_{i=0}^n \alpha^i$

Valutazione di progressioni geometriche $\sum_{i=0}^n \alpha^i$

Somme finite: Sia $S_n = \sum_{i=0}^n \alpha^i$.

Valutazione di progressioni geometriche $\sum_{i=0}^n \alpha^i$

Somme finite: Sia $S_n = \sum_{i=0}^n \alpha^i$. Se $\alpha \neq 1$ allora

$$S_n(\alpha - 1) = \alpha \cdot S_n - S_n$$

Valutazione di progressioni geometriche $\sum_{i=0}^n \alpha^i$

Somme finite: Sia $S_n = \sum_{i=0}^n \alpha^i$. Se $\alpha \neq 1$ allora

$$S_n(\alpha - 1) = \alpha \cdot S_n - S_n = \sum_{i=0}^n \alpha^{i+1} - \sum_{i=0}^n \alpha^i$$

Valutazione di progressioni geometriche $\sum_{i=0}^n \alpha^i$

Somme finite: Sia $S_n = \sum_{i=0}^n \alpha^i$. Se $\alpha \neq 1$ allora

$$\begin{aligned} S_n(\alpha - 1) &= \alpha \cdot S_n - S_n = \sum_{i=0}^n \alpha^{i+1} - \sum_{i=0}^n \alpha^i \\ &= \alpha + \alpha^2 + \dots + \alpha^n + \alpha^{n+1} \end{aligned}$$

Valutazione di progressioni geometriche $\sum_{i=0}^n \alpha^i$

Somme finite: Sia $S_n = \sum_{i=0}^n \alpha^i$. Se $\alpha \neq 1$ allora

$$\begin{aligned} S_n(\alpha - 1) &= \alpha \cdot S_n - S_n = \sum_{i=0}^n \alpha^{i+1} - \sum_{i=0}^n \alpha^i \\ &= \alpha + \alpha^2 + \dots + \alpha^n + \alpha^{n+1} - 1 - \alpha - \alpha^2 - \dots - \alpha^n \end{aligned}$$

Valutazione di progressioni geometriche $\sum_{i=0}^n \alpha^i$

Somme finite: Sia $S_n = \sum_{i=0}^n \alpha^i$. Se $\alpha \neq 1$ allora

$$\begin{aligned} S_n(\alpha - 1) &= \alpha \cdot S_n - S_n = \sum_{i=0}^n \alpha^{i+1} - \sum_{i=0}^n \alpha^i \\ &= \alpha + \alpha^2 + \dots + \alpha^n + \alpha^{n+1} - 1 - \alpha - \alpha^2 - \dots - \alpha^n \\ &= \alpha^{n+1} - 1 \end{aligned}$$

Valutazione di progressioni geometriche $\sum_{i=0}^n \alpha^i$

Somme finite: Sia $S_n = \sum_{i=0}^n \alpha^i$. Se $\alpha \neq 1$ allora

$$\begin{aligned} S_n(\alpha - 1) &= \alpha \cdot S_n - S_n = \sum_{i=0}^n \alpha^{i+1} - \sum_{i=0}^n \alpha^i \\ &= \alpha + \alpha^2 + \dots + \alpha^n + \alpha^{n+1} - 1 - \alpha - \alpha^2 - \dots - \alpha^n \\ &= \alpha^{n+1} - 1 \end{aligned}$$

da cui $S_n = (\alpha^{n+1} - 1)/(\alpha - 1)$.

Valutazione di progressioni geometriche $\sum_{i=0}^n \alpha^i$

Somme finite: Sia $S_n = \sum_{i=0}^n \alpha^i$. Se $\alpha \neq 1$ allora

$$\begin{aligned} S_n(\alpha - 1) &= \alpha \cdot S_n - S_n = \sum_{i=0}^n \alpha^{i+1} - \sum_{i=0}^n \alpha^i \\ &= \alpha + \alpha^2 + \dots + \alpha^n + \alpha^{n+1} - 1 - \alpha - \alpha^2 - \dots - \alpha^n \\ &= \alpha^{n+1} - 1 \end{aligned}$$

da cui $S_n = (\alpha^{n+1} - 1)/(\alpha - 1)$.

Esempio:

Valutazione di progressioni geometriche $\sum_{i=0}^n \alpha^i$

Somme finite: Sia $S_n = \sum_{i=0}^n \alpha^i$. Se $\alpha \neq 1$ allora

$$\begin{aligned} S_n(\alpha - 1) &= \alpha \cdot S_n - S_n = \sum_{i=0}^n \alpha^{i+1} - \sum_{i=0}^n \alpha^i \\ &= \alpha + \alpha^2 + \dots + \alpha^n + \alpha^{n+1} - 1 - \alpha - \alpha^2 - \dots - \alpha^n \\ &= \alpha^{n+1} - 1 \end{aligned}$$

da cui $S_n = (\alpha^{n+1} - 1)/(\alpha - 1)$.

Esempio:

$$\sum_{i=0}^n 2^i = 2^{n+1} - 1,$$

Valutazione di progressioni geometriche $\sum_{i=0}^n \alpha^i$

Somme finite: Sia $S_n = \sum_{i=0}^n \alpha^i$. Se $\alpha \neq 1$ allora

$$\begin{aligned} S_n(\alpha - 1) &= \alpha \cdot S_n - S_n = \sum_{i=0}^n \alpha^{i+1} - \sum_{i=0}^n \alpha^i \\ &= \alpha + \alpha^2 + \dots + \alpha^n + \alpha^{n+1} - 1 - \alpha - \alpha^2 - \dots - \alpha^n \\ &= \alpha^{n+1} - 1 \end{aligned}$$

da cui $S_n = (\alpha^{n+1} - 1)/(\alpha - 1)$.

Esempio:

$$\sum_{i=0}^n 2^i = 2^{n+1} - 1, \quad \sum_{i=0}^n 3^i = \frac{3^{n+1} - 1}{2}$$

Valutazione di progressioni geometriche $\sum_{i=0}^n \alpha^i$

Somme finite: Sia $S_n = \sum_{i=0}^n \alpha^i$. Se $\alpha \neq 1$ allora

$$\begin{aligned} S_n(\alpha - 1) &= \alpha \cdot S_n - S_n = \sum_{i=0}^n \alpha^{i+1} - \sum_{i=0}^n \alpha^i \\ &= \alpha + \alpha^2 + \dots + \alpha^n + \alpha^{n+1} - 1 - \alpha - \alpha^2 - \dots - \alpha^n \\ &= \alpha^{n+1} - 1 \end{aligned}$$

da cui $S_n = (\alpha^{n+1} - 1)/(\alpha - 1)$.

Esempio:

$$\sum_{i=0}^n 2^i = 2^{n+1} - 1, \quad \sum_{i=0}^n 3^i = \frac{3^{n+1} - 1}{2}$$

$$\sum_{i=0}^n \left(\frac{1}{2}\right)^i = \frac{(1/2)^{n+1} - 1}{1/2 - 1}$$

Valutazione di progressioni geometriche $\sum_{i=0}^n \alpha^i$

Somme finite: Sia $S_n = \sum_{i=0}^n \alpha^i$. Se $\alpha \neq 1$ allora

$$\begin{aligned} S_n(\alpha - 1) &= \alpha \cdot S_n - S_n = \sum_{i=0}^n \alpha^{i+1} - \sum_{i=0}^n \alpha^i \\ &= \alpha + \alpha^2 + \dots + \alpha^n + \alpha^{n+1} - 1 - \alpha - \alpha^2 - \dots - \alpha^n \\ &= \alpha^{n+1} - 1 \end{aligned}$$

da cui $S_n = (\alpha^{n+1} - 1)/(\alpha - 1)$.

Esempio:

$$\sum_{i=0}^n 2^i = 2^{n+1} - 1, \quad \sum_{i=0}^n 3^i = \frac{3^{n+1} - 1}{2}$$

$$\sum_{i=0}^n \left(\frac{1}{2}\right)^i = \frac{(1/2)^{n+1} - 1}{1/2 - 1} = \frac{2^{n+1} - 1}{2^n}$$

Somme infinite:

Supponiamo $0 < \alpha < 1$ e sia $S = \sum_{i=0}^{\infty} \alpha^i$.

Somme infinite:

Supponiamo $0 < \alpha < 1$ e sia $S = \sum_{i=0}^{\infty} \alpha^i$.

Allora $\alpha S = \sum_{i=1}^{\infty} \alpha^i$,

Somme infinite:

Supponiamo $0 < \alpha < 1$ e sia $S = \sum_{i=0}^{\infty} \alpha^i$.

Allora $\alpha S = \sum_{i=1}^{\infty} \alpha^i$, da cui

$$S(1-\alpha) = S - \alpha S$$

Somme infinite:

Supponiamo $0 < \alpha < 1$ e sia $S = \sum_{i=0}^{\infty} \alpha^i$.

Allora $\alpha S = \sum_{i=1}^{\infty} \alpha^i$, da cui

$$S(1-\alpha) = S - \alpha S = \sum_{i=0}^{\infty} \alpha^i - \sum_{i=1}^{\infty} \alpha^i$$

Somme infinite:

Supponiamo $0 < \alpha < 1$ e sia $S = \sum_{i=0}^{\infty} \alpha^i$.

Allora $\alpha S = \sum_{i=1}^{\infty} \alpha^i$, da cui

$$S(1-\alpha) = S - \alpha S = \sum_{i=0}^{\infty} \alpha^i - \sum_{i=1}^{\infty} \alpha^i = \alpha^0 = 1,$$

Somme infinite:

Supponiamo $0 < \alpha < 1$ e sia $S = \sum_{i=0}^{\infty} \alpha^i$.

Allora $\alpha S = \sum_{i=1}^{\infty} \alpha^i$, da cui

$$S(1-\alpha) = S - \alpha S = \sum_{i=0}^{\infty} \alpha^i - \sum_{i=1}^{\infty} \alpha^i = \alpha^0 = 1, \text{ ovvero } S = 1/(1-\alpha)$$

Somme infinite:

Supponiamo $0 < \alpha < 1$ e sia $S = \sum_{i=0}^{\infty} \alpha^i$.

Allora $\alpha S = \sum_{i=1}^{\infty} \alpha^i$, da cui

$$S(1-\alpha) = S - \alpha S = \sum_{i=0}^{\infty} \alpha^i - \sum_{i=1}^{\infty} \alpha^i = \alpha^0 = 1, \text{ ovvero } S = 1/(1-\alpha)$$

Vediamo un esempio. Calcoliamo $\sum_{i=0}^{\lfloor \log_3 n \rfloor} (1/3)^i$.

Somme infinite:

Supponiamo $0 < \alpha < 1$ e sia $S = \sum_{i=0}^{\infty} \alpha^i$.

Allora $\alpha S = \sum_{i=1}^{\infty} \alpha^i$, da cui

$$S(1-\alpha) = S - \alpha S = \sum_{i=0}^{\infty} \alpha^i - \sum_{i=1}^{\infty} \alpha^i = \alpha^0 = 1, \text{ ovvero } S = 1/(1-\alpha)$$

Vediamo un esempio. Calcoliamo $\sum_{i=0}^{\lfloor \log_3 n \rfloor} (1/3)^i$.

Ovviamente vale che $\sum_{i=0}^{\lfloor \log_3 n \rfloor} (1/3)^i$

Somme infinite:

Supponiamo $0 < \alpha < 1$ e sia $S = \sum_{i=0}^{\infty} \alpha^i$.

Allora $\alpha S = \sum_{i=1}^{\infty} \alpha^i$, da cui

$$S(1-\alpha) = S - \alpha S = \sum_{i=0}^{\infty} \alpha^i - \sum_{i=1}^{\infty} \alpha^i = \alpha^0 = 1, \text{ ovvero } S = 1/(1-\alpha)$$

Vediamo un esempio. Calcoliamo $\sum_{i=0}^{\lfloor \log_3 n \rfloor} (1/3)^i$.

Ovviamente vale che $\sum_{i=0}^{\lfloor \log_3 n \rfloor} (1/3)^i < \sum_{i=0}^{\infty} (1/3)^i$

Somme infinite:

Supponiamo $0 < \alpha < 1$ e sia $S = \sum_{i=0}^{\infty} \alpha^i$.

Allora $\alpha S = \sum_{i=1}^{\infty} \alpha^i$, da cui

$$S(1-\alpha) = S - \alpha S = \sum_{i=0}^{\infty} \alpha^i - \sum_{i=1}^{\infty} \alpha^i = \alpha^0 = 1, \text{ ovvero } S = 1/(1-\alpha)$$

Vediamo un esempio. Calcoliamo $\sum_{i=0}^{\lfloor \log_3 n \rfloor} (1/3)^i$.

Ovviamente vale che $\sum_{i=0}^{\lfloor \log_3 n \rfloor} (1/3)^i < \sum_{i=0}^{\infty} (1/3)^i$

Applicando quindi la formula $S_n = \sum_{i=0}^{\infty} \alpha^i = \frac{1}{1-\alpha}$ alla espressione

$$\sum_{i=0}^{\infty} \left(\frac{1}{3}\right)^i$$

Somme infinite:

Supponiamo $0 < \alpha < 1$ e sia $S = \sum_{i=0}^{\infty} \alpha^i$.

Allora $\alpha S = \sum_{i=1}^{\infty} \alpha^i$, da cui

$$S(1-\alpha) = S - \alpha S = \sum_{i=0}^{\infty} \alpha^i - \sum_{i=1}^{\infty} \alpha^i = \alpha^0 = 1, \text{ ovvero } S = 1/(1-\alpha)$$

Vediamo un esempio. Calcoliamo $\sum_{i=0}^{\lfloor \log_3 n \rfloor} (1/3)^i$.

Ovviamente vale che $\sum_{i=0}^{\lfloor \log_3 n \rfloor} (1/3)^i < \sum_{i=0}^{\infty} (1/3)^i$

Applicando quindi la formula $S_n = \sum_{i=0}^{\infty} \alpha^i = \frac{1}{1-\alpha}$ alla espressione

$$\sum_{i=0}^{\infty} \left(\frac{1}{3}\right)^i$$

otteniamo

$$\sum_{i=0}^{\lfloor \log_3 n \rfloor} \left(\frac{1}{3}\right)^i$$

Somme infinite:

Supponiamo $0 < \alpha < 1$ e sia $S = \sum_{i=0}^{\infty} \alpha^i$.

Allora $\alpha S = \sum_{i=1}^{\infty} \alpha^i$, da cui

$$S(1-\alpha) = S - \alpha S = \sum_{i=0}^{\infty} \alpha^i - \sum_{i=1}^{\infty} \alpha^i = \alpha^0 = 1, \text{ ovvero } S = 1/(1-\alpha)$$

Vediamo un esempio. Calcoliamo $\sum_{i=0}^{\lfloor \log_3 n \rfloor} (1/3)^i$.

Ovviamente vale che $\sum_{i=0}^{\lfloor \log_3 n \rfloor} (1/3)^i < \sum_{i=0}^{\infty} (1/3)^i$

Applicando quindi la formula $S_n = \sum_{i=0}^{\infty} \alpha^i = \frac{1}{1-\alpha}$ alla espressione

$$\sum_{i=0}^{\infty} \left(\frac{1}{3}\right)^i$$

otteniamo

$$\sum_{i=0}^{\lfloor \log_3 n \rfloor} \left(\frac{1}{3}\right)^i < \sum_{i=0}^{\infty} \left(\frac{1}{3}\right)^i =$$

Somme infinite:

Supponiamo $0 < \alpha < 1$ e sia $S = \sum_{i=0}^{\infty} \alpha^i$.

Allora $\alpha S = \sum_{i=1}^{\infty} \alpha^i$, da cui

$$S(1-\alpha) = S - \alpha S = \sum_{i=0}^{\infty} \alpha^i - \sum_{i=1}^{\infty} \alpha^i = \alpha^0 = 1, \text{ ovvero } S = 1/(1-\alpha)$$

Vediamo un esempio. Calcoliamo $\sum_{i=0}^{\lfloor \log_3 n \rfloor} (1/3)^i$.

Ovviamente vale che $\sum_{i=0}^{\lfloor \log_3 n \rfloor} (1/3)^i < \sum_{i=0}^{\infty} (1/3)^i$

Applicando quindi la formula $S_n = \sum_{i=0}^{\infty} \alpha^i = \frac{1}{1-\alpha}$ alla espressione

$$\sum_{i=0}^{\infty} \left(\frac{1}{3}\right)^i$$

otteniamo

$$\sum_{i=0}^{\lfloor \log_3 n \rfloor} \left(\frac{1}{3}\right)^i < \sum_{i=0}^{\infty} \left(\frac{1}{3}\right)^i = \frac{1}{1 - (1/3)} = (3/2).$$

Algoritmi Ricorsivi

Algoritmi Ricorsivi

- ▶ Come esprimere la complessità di algoritmi ricorsivi mediante relazioni di ricorrenza

Algoritmi Ricorsivi

- ▶ Come esprimere la complessità di algoritmi ricorsivi mediante relazioni di ricorrenza
- ▶ Come derivare le relazioni di ricorrenza,

Algoritmi Ricorsivi

- ▶ Come esprimere la complessità di algoritmi ricorsivi mediante relazioni di ricorrenza
- ▶ Come derivare le relazioni di ricorrenza,
- ▶ e Come risolvere le relazioni di ricorrenza.

Gli algoritmi ricorsivi hanno, generalmente, questa struttura:

```
Algoritmo  $\mathcal{A}(I)$ 
IF (  $I$  è ‘‘di taglia piccola’’){
    RETURN output
}ELSE {
    :
     $\mathcal{A}(I_1)$ 
    :
     $\mathcal{A}(I_a)$ 
    :
```

Gli algoritmi ricorsivi hanno, generalmente, questa struttura:

```
Algoritmo  $\mathcal{A}(I)$ 
IF ( $I$  è ‘di taglia piccola’){
    RETURN output
}ELSE {
    ⋮
     $\mathcal{A}(I_1)$ 
    ⋮
     $\mathcal{A}(I_a)$ 
    ⋮
}
```

► I è l'input iniziale all'algoritmo

Gli algoritmi ricorsivi hanno, generalmente, questa struttura:

```
Algoritmo  $\mathcal{A}(I)$ 
IF ( $I$  è "di taglia piccola"){
    RETURN output
}ELSE {
    :
     $\mathcal{A}(I_1)$ 
    :
     $\mathcal{A}(I_a)$ 
    :
```

- ▶ I è l'input iniziale all'algoritmo
- ▶ i puntini " : " sono istruzioni di qualche tipo

Gli algoritmi ricorsivi hanno, generalmente, questa struttura:

```
Algoritmo  $\mathcal{A}(I)$ 
IF ( $I$  è "di taglia piccola"){
    RETURN output
}ELSE {
    :
     $\mathcal{A}(I_1)$ 
    :
     $\mathcal{A}(I_a)$ 
    :
```

- ▶ I è l'input iniziale all'algoritmo
- ▶ i puntini ":" sono istruzioni di qualche tipo
- ▶ $\mathcal{A}(I_1), \dots, \mathcal{A}(I_a)$ sono le a chiamate ricorsive all'algoritmo \mathcal{A} , rispettivamente sugli input I_1, \dots, I_a .

Per derivare le relazioni di ricorrenza che descrivono il tempo di esecuzione $T(n)$ di un algoritmo occorre:

1. Determinare la taglia dell'input n

Per derivare le relazioni di ricorrenza che descrivono il tempo di esecuzione $T(n)$ di un algoritmo occorre:

1. Determinare la taglia dell'input n
2. Determinare quale valore n_0 di n è usato per la base della ricorsione (generalmente, ma non sempre, $n_0 = 1$).

Per derivare le relazioni di ricorrenza che descrivono il tempo di esecuzione $T(n)$ di un algoritmo occorre:

1. Determinare la taglia dell'input n
2. Determinare quale valore n_0 di n è usato per la base della ricorsione (generalmente, ma non sempre, $n_0 = 1$).
3. Determinare il valore $T(n_0)$ del tempo di esecuzione sulla base della ricorsione (spesso basterà essere certi che $T(n_0) = c$, per qualche costante c)

Per derivare le relazioni di ricorrenza che descrivono il tempo di esecuzione $T(n)$ di un algoritmo occorre:

1. Determinare la taglia dell'input n
2. Determinare quale valore n_0 di n è usato per la base della ricorsione (generalmente, ma non sempre, $n_0 = 1$).
3. Determinare il valore $T(n_0)$ del tempo di esecuzione sulla base della ricorsione (spesso basterà essere certi che $T(n_0) = c$, per qualche costante c)

Il valore $T(n)$ sarà generalmente uguale ad una somma del tipo $T(m_1) + \dots + T(m_a)$ (per le chiamate ricorsive),

Per derivare le relazioni di ricorrenza che descrivono il tempo di esecuzione $T(n)$ di un algoritmo occorre:

1. Determinare la taglia dell'input n
2. Determinare quale valore n_0 di n è usato per la base della ricorsione (generalmente, ma non sempre, $n_0 = 1$).
3. Determinare il valore $T(n_0)$ del tempo di esecuzione sulla base della ricorsione (spesso basterà essere certi che $T(n_0) = c$, per qualche costante c)

Il valore $T(n)$ sarà generalmente uguale ad una somma del tipo $T(m_1) + \dots + T(m_a)$ (per le chiamate ricorsive), piú la somma di eventuale altro lavoro fatto.

Per derivare le relazioni di ricorrenza che descrivono il tempo di esecuzione $T(n)$ di un algoritmo occorre:

1. Determinare la taglia dell'input n
2. Determinare quale valore n_0 di n è usato per la base della ricorsione (generalmente, ma non sempre, $n_0 = 1$).
3. Determinare il valore $T(n_0)$ del tempo di esecuzione sulla base della ricorsione (spesso basterà essere certi che $T(n_0) = c$, per qualche costante c)

Il valore $T(n)$ sarà generalmente uguale ad una somma del tipo $T(m_1) + \dots + T(m_a)$ (per le chiamate ricorsive), piú la somma di eventuale altro lavoro fatto. Spesso le a chiamate ricorsive saranno effettuate tutte su sottoproblemi di taglia uguale $f(n)$, dando un termine $aT(f(n))$ nella relazione di ricorrenza.

Una tipica relazione di ricorrenza sarà:

$$T(n) = \begin{cases} c & \text{se } n = n_0 \\ aT(f(n)) + g(n) & \text{altrimenti} \end{cases}$$

Una tipica relazione di ricorrenza sarà:

$$T(n) = \begin{cases} c & \text{se } n = n_0 \\ aT(f(n)) + g(n) & \text{altrimenti} \end{cases}$$

dove

- ▶ n_0 =base ricorsione, c =tempo di esecuzione per la base

Una tipica relazione di ricorrenza sarà:

$$T(n) = \begin{cases} c & \text{se } n = n_0 \\ aT(f(n)) + g(n) & \text{altrimenti} \end{cases}$$

dove

- ▶ n_0 =base ricorsione, c =tempo di esecuzione per la base
- ▶ a =numero di volte che le chiamate ricorsive sono effettuate

Una tipica relazione di ricorrenza sarà:

$$T(n) = \begin{cases} c & \text{se } n = n_0 \\ aT(f(n)) + g(n) & \text{altrimenti} \end{cases}$$

dove

- ▶ n_0 =base ricorsione, c =tempo di esecuzione per la base
- ▶ a =numero di volte che le chiamate ricorsive sono effettuate
- ▶ $f(n)$ =taglia dei problemi risolti nelle chiamate ricorsive

Una tipica relazione di ricorrenza sarà:

$$T(n) = \begin{cases} c & \text{se } n = n_0 \\ aT(f(n)) + g(n) & \text{altrimenti} \end{cases}$$

dove

- ▶ n_0 =base ricorsione, c =tempo di esecuzione per la base
- ▶ a =numero di volte che le chiamate ricorsive sono effettuate
- ▶ $f(n)$ =taglia dei problemi risolti nelle chiamate ricorsive
- ▶ $g(n)$ =tutto il tempo di calcolo non incluso nelle chiamate ricorsive

Esempio.

Esempio.

```
Algoritmo1(n)
IF (n==1) {
    fà qualcosa
```

Esempio.

```
Algoritmo1(n)
IF (n==1) {
  fà qualcosa
  ELSE {
    Algoritmo1(n-1)
```

Esempio.

```
Algoritmo1(n)
IF (n==1) {
  fà qualcosa
  ELSE {
    Algoritmo1(n-1)
    Algoritmo1(n-2)
```

Esempio.

```
Algoritmo1(n)
IF (n==1) {
    fà qualcosa
    ELSE {
        Algoritmo1(n-1)
        Algoritmo1(n-2)
    }
}
FOR(i=1; i<n+1;i=i+1){
    fà qualcos'altro
}
```

Esempio.

```
Algoritmo1(n)
IF (n==1) {
  fà qualcosa
  ELSE {
    Algoritmo1(n-1)
    Algoritmo1(n-2)
  }
}
FOR(i=1; i<n+1;i=i+1){
  fà qualcos'altro
}
```

L'equazione di ricorrenza che descrive la complessità $T(n)$ dell'Algoritmo1(n) sarà

$$T(n) = \begin{cases} c & \text{se } n = 1 \\ T(n-1) + T(n-2) + dn & \text{altrimenti} \end{cases}$$

Esempio:

Esempio:

```
Algoritmo2(n)
IF ((n==1)|| (n==2) {
    fà qualcosa
```


Esempio:

```
Algoritmo2(n)
IF ((n==1)|| (n==2) {
    fà qualcosa
    ELSE Algoritmo2(n/2)
```

Esempio:

```
Algoritmo2(n)
IF ((n==1)|| (n==2) {
    fà qualcosa
    ELSE Algoritmo2(n/2)
        Algoritmo2(n/2) }
```

Esempio:

```
Algoritmo2(n)
IF ((n==1)|| (n==2) {
    fà qualcosa
    ELSE Algoritmo2(n/2)
        Algoritmo2(n/2) }
FOR(i=1; i<n+1;i=i+1){
    fà qualcos'altro
}
```

Esempio:

```
Algoritmo2(n)
IF ((n==1)|| (n==2) {
    fà qualcosa
    ELSE Algoritmo2(n/2)
        Algoritmo2(n/2) }
FOR(i=1; i<n+1;i=i+1){
    fà qualcos'altro
}
```

L'equazione di ricorrenza che descrive la complessità $T(n)$ dell'Algoritmo2(n) sarà

$$T(n) = \begin{cases} c & \text{se } n \leq 2 \\ 2T(n/2) + dn & \text{altrimenti} \end{cases}$$

Algoritmo3(n)

```
Algoritmo3(n)
IF (n==1) {
    fà qualcosa
```

```
Algoritmo3(n)
IF (n==1) {
    fà qualcosa
} ELSE {
    IF (n==2) {
        fà qualcos'altro
```

```
Algoritmo3(n)
IF (n==1) {
    fà qualcosa
} ELSE {
    IF (n==2) {
        fà qualcos'altro
    } ELSE {
        FOR(i=1; i<n+1;i=i+1){
            Algoritmo3(n-1)
```



```
Algoritmo3(n)
IF (n==1) {
    fà qualcosa
} ELSE {
    IF (n==2) {
        fà qualcos'altro
    } ELSE {
        FOR(i=1; i<n+1;i=i+1){
            Algoritmo3(n-1)
            fà altro ancora
        }
    }
}
```

```

Algoritmo3(n)
IF (n==1) {
    fà qualcosa
} ELSE {
    IF (n==2) {
        fà qualcos'altro
    } ELSE {
        FOR(i=1; i<n+1;i=i+1){
            Algoritmo3(n-1)
            fà altro ancora
        }
    }
}

```

L'equazione di ricorrenza che descrive la complessità $T(n)$ dell'Algoritmo3(n) sarà

$$T(n) = \begin{cases} c & \text{se } n \leq 2 \\ nT(n-1) + dn & \text{altrimenti} \end{cases}$$

Come risolvere le equazioni di ricorrenza?

Come risolvere le equazioni di ricorrenza?

Risultato: Per arbitrarie costanti c , a , e d la soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn & \text{altrimenti} \end{cases}$$

Come risolvere le equazioni di ricorrenza?

Risultato: Per arbitrarie costanti c , a , e d la soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn & \text{altrimenti} \end{cases}$$

è data da

$$T(n) = \begin{cases} O(n) & \text{se } a < c \end{cases}$$

Come risolvere le equazioni di ricorrenza?

Risultato: Per arbitrarie costanti c , a , e d la soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn & \text{altrimenti} \end{cases}$$

è data da

$$T(n) = \begin{cases} O(n) & \text{se } a < c \\ O(n \log n) & \text{se } a = c \end{cases}$$

Come risolvere le equazioni di ricorrenza?

Risultato: Per arbitrarie costanti c , a , e d la soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn & \text{altrimenti} \end{cases}$$

è data da

$$T(n) = \begin{cases} O(n) & \text{se } a < c \\ O(n \log n) & \text{se } a = c \\ O(n^{\log_c a}) & \text{se } a > c \end{cases}$$

Come risolvere le equazioni di ricorrenza?

Risultato: Per arbitrarie costanti c , a , e d la soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn & \text{altrimenti} \end{cases}$$

è data da

$$T(n) = \begin{cases} O(n) & \text{se } a < c \\ O(n \log n) & \text{se } a = c \\ O(n^{\log_c a}) & \text{se } a > c \end{cases}$$

Esempi di applicazione del Teorema:

Come risolvere le equazioni di ricorrenza?

Risultato: Per arbitrarie costanti c , a , e d la soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn & \text{altrimenti} \end{cases}$$

è data da

$$T(n) = \begin{cases} O(n) & \text{se } a < c \\ O(n \log n) & \text{se } a = c \\ O(n^{\log_c a}) & \text{se } a > c \end{cases}$$

Esempi di applicazione del Teorema:

- Se $T(n) = 2T(n/3) + dn$, allora $T(n) = O(n)$

Come risolvere le equazioni di ricorrenza?

Risultato: Per arbitrarie costanti c , a , e d la soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn & \text{altrimenti} \end{cases}$$

è data da

$$T(n) = \begin{cases} O(n) & \text{se } a < c \\ O(n \log n) & \text{se } a = c \\ O(n^{\log_c a}) & \text{se } a > c \end{cases}$$

Esempi di applicazione del Teorema:

- Se $T(n) = 2T(n/3) + dn$, allora $T(n) = O(n)$
- Se $T(n) = 2T(n/2) + dn$, allora $T(n) = O(n \log n)$

Come risolvere le equazioni di ricorrenza?

Risultato: Per arbitrarie costanti c , a , e d la soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn & \text{altrimenti} \end{cases}$$

è data da

$$T(n) = \begin{cases} O(n) & \text{se } a < c \\ O(n \log n) & \text{se } a = c \\ O(n^{\log_c a}) & \text{se } a > c \end{cases}$$

Esempi di applicazione del Teorema:

- Se $T(n) = 2T(n/3) + dn$, allora $T(n) = O(n)$
- Se $T(n) = 2T(n/2) + dn$, allora $T(n) = O(n \log n)$
- Se $T(n) = 4T(n/2) + dn$, allora $T(n) = O(n^2)$

Diamo una bozza di dimostrazione del Teorema quando $n = c^i$ per qualche intero i .

Diamo una bozza di dimostrazione del Teorema quando $n = c^i$ per qualche intero i . Si ha

$$T(n) = a \cdot T(n/c) + bn \text{ (per ipotesi)}$$

Diamo una bozza di dimostrazione del Teorema quando $n = c^i$ per qualche intero i . Si ha

$$\begin{aligned} T(n) &= a \cdot T(n/c) + bn \text{ (per ipotesi)} \\ &= a(a \cdot T(n/c^2) + bn/c) + bn \text{ (in quanto } T(n/c) = aT(n/c^2) + bn/c) \end{aligned}$$

Diamo una bozza di dimostrazione del Teorema quando $n = c^i$ per qualche intero i . Si ha

$$\begin{aligned} T(n) &= a \cdot T(n/c) + bn \text{ (per ipotesi)} \\ &= a(a \cdot T(n/c^2) + bn/c) + bn \text{ (in quanto } T(n/c) = aT(n/c^2) + bn/c) \\ &= a^2 \cdot T(n/c^2) + abn/c + bn \end{aligned}$$

Diamo una bozza di dimostrazione del Teorema quando $n = c^i$ per qualche intero i . Si ha

$$\begin{aligned}T(n) &= a \cdot T(n/c) + bn \text{ (per ipotesi)} \\&= a(a \cdot T(n/c^2) + bn/c) + bn \text{ (in quanto } T(n/c) = aT(n/c^2) + bn/c) \\&= a^2 \cdot T(n/c^2) + abn/c + bn \\&= a^2(a \cdot T(n/c^3) + bn/c^2) + abn/c + bn\end{aligned}$$

Diamo una bozza di dimostrazione del Teorema quando $n = c^i$ per qualche intero i . Si ha

$$\begin{aligned}T(n) &= a \cdot T(n/c) + bn \text{ (per ipotesi)} \\&= a(a \cdot T(n/c^2) + bn/c) + bn \text{ (in quanto } T(n/c) = aT(n/c^2) + bn/c) \\&= a^2 \cdot T(n/c^2) + abn/c + bn \\&= a^2(a \cdot T(n/c^3) + bn/c^2) + abn/c + bn \\&= a^3 \cdot T(n/c^3) + a^2bn/c^2 + abn/c + bn\end{aligned}$$

Diamo una bozza di dimostrazione del Teorema quando $n = c^i$ per qualche intero i . Si ha

$$\begin{aligned}T(n) &= a \cdot T(n/c) + bn \text{ (per ipotesi)} \\&= a(a \cdot T(n/c^2) + bn/c) + bn \text{ (in quanto } T(n/c) = aT(n/c^2) + bn/c) \\&= a^2 \cdot T(n/c^2) + abn/c + bn \\&= a^2(a \cdot T(n/c^3) + bn/c^2) + abn/c + bn \\&= a^3 \cdot T(n/c^3) + a^2bn/c^2 + abn/c + bn \\&\dots\end{aligned}$$

Diamo una bozza di dimostrazione del Teorema quando $n = c^i$ per qualche intero i . Si ha

$$\begin{aligned}T(n) &= a \cdot T(n/c) + bn \text{ (per ipotesi)} \\&= a(a \cdot T(n/c^2) + bn/c) + bn \text{ (in quanto } T(n/c) = aT(n/c^2) + bn/c) \\&= a^2 \cdot T(n/c^2) + abn/c + bn \\&= a^2(a \cdot T(n/c^3) + bn/c^2) + abn/c + bn \\&= a^3 \cdot T(n/c^3) + a^2bn/c^2 + abn/c + bn \\&\dots \\&= a^i T(n/c^i) + bn \sum_{j=0}^{i-1} (a/c)^j\end{aligned}$$

Diamo una bozza di dimostrazione del Teorema quando $n = c^i$ per qualche intero i . Si ha

$$\begin{aligned}T(n) &= a \cdot T(n/c) + bn \text{ (per ipotesi)} \\&= a(a \cdot T(n/c^2) + bn/c) + bn \text{ (in quanto } T(n/c) = aT(n/c^2) + bn/c) \\&= a^2 \cdot T(n/c^2) + abn/c + bn \\&= a^2(a \cdot T(n/c^3) + bn/c^2) + abn/c + bn \\&= a^3 \cdot T(n/c^3) + a^2bn/c^2 + abn/c + bn \\&\dots \\&= a^i T(n/c^i) + bn \sum_{j=0}^{i-1} (a/c)^j \\&= a^{\log_c n} T(1) + bn \sum_{j=0}^{\log_c n - 1} (a/c)^j \text{ (poichè } i = \log_c n)\end{aligned}$$

Diamo una bozza di dimostrazione del Teorema quando $n = c^i$ per qualche intero i . Si ha

$$\begin{aligned}T(n) &= a \cdot T(n/c) + bn \text{ (per ipotesi)} \\&= a(a \cdot T(n/c^2) + bn/c) + bn \text{ (in quanto } T(n/c) = aT(n/c^2) + bn/c) \\&= a^2 \cdot T(n/c^2) + abn/c + bn \\&= a^2(a \cdot T(n/c^3) + bn/c^2) + abn/c + bn \\&= a^3 \cdot T(n/c^3) + a^2bn/c^2 + abn/c + bn\end{aligned}$$

...

$$\begin{aligned}&= a^i T(n/c^i) + bn \sum_{j=0}^{i-1} (a/c)^j \\&= a^{\log_c n} T(1) + bn \sum_{j=0}^{\log_c n - 1} (a/c)^j \text{ (poichè } i = \log_c n) \\&= d \cdot n^{\log_c a} + bn \sum_{j=0}^{\log_c n - 1} (a/c)^j\end{aligned}$$

avendo usato il fatto che $T(1) = d$ e $a^{\log_c n} = n^{\log_c a}$.

Per calcolare

$$\sum_{j=0}^{\log_c n - 1} (a/c)^j$$

occorre ricordare ciò che abbiamo già visto sulle Progressioni Geometriche,

Per calcolare

$$\sum_{j=0}^{\log_c n - 1} (a/c)^j$$

occorre ricordare ciò che abbiamo già visto sulle Progressioni Geometriche, ovvero:

Somme finite: Se $\alpha \neq 1$ allora

$$\sum_{i=0}^n \alpha^i = \frac{\alpha^{n+1} - 1}{\alpha - 1} \quad (1)$$

Per calcolare

$$\sum_{j=0}^{\log_c n - 1} (a/c)^j$$

occorre ricordare ciò che abbiamo già visto sulle Progressioni Geometriche, ovvero:

Somme finite: Se $\alpha \neq 1$ allora

$$\sum_{i=0}^n \alpha^i = \frac{\alpha^{n+1} - 1}{\alpha - 1} \quad (1)$$

Somme infinite: Se $0 < \alpha < 1$. Allora

$$\sum_{i=0}^{\infty} \alpha^i = \frac{1}{1 - \alpha} \quad (2)$$

Caso 1: $a < c$ (cioè $(a/c) < 1$)

Caso 1: $a < c$ (cioè $(a/c) < 1$)

$$\sum_{j=0}^{\log_c n - 1} (a/c)^j$$

Caso 1: $a < c$ (cioè $(a/c) < 1$)

$$\sum_{j=0}^{\log_c n - 1} (a/c)^j < \sum_{j=0}^{\infty} (a/c)^j$$

Caso 1: $a < c$ (cioè $(a/c) < 1$)

$$\sum_{j=0}^{\log_c n - 1} (a/c)^j < \sum_{j=0}^{\infty} (a/c)^j = \frac{1}{1 - a/c}$$

Caso 1: $a < c$ (cioè $(a/c) < 1$)

$$\sum_{j=0}^{\log_c n - 1} (a/c)^j < \sum_{j=0}^{\infty} (a/c)^j = \frac{1}{1 - a/c} = \frac{c}{c - a}$$

dalla formula

$$\sum_{i=0}^{\infty} \alpha^i = \frac{1}{1 - \alpha} \quad (2)$$

Caso 1: $a < c$ (cioè $(a/c) < 1$)

$$\sum_{j=0}^{\log_c n - 1} (a/c)^j < \sum_{j=0}^{\infty} (a/c)^j = \frac{1}{1 - a/c} = \frac{c}{c - a}$$

dalla formula

$$\sum_{i=0}^{\infty} \alpha^i = \frac{1}{1 - \alpha} \quad (2)$$

Pertanto

$$T(n) < d \cdot n^{\log_c a} + bcn/(c - a) = O(n)$$

Caso 1: $a < c$ (cioè $(a/c) < 1$)

$$\sum_{j=0}^{\log_c n - 1} (a/c)^j < \sum_{j=0}^{\infty} (a/c)^j = \frac{1}{1 - a/c} = \frac{c}{c - a}$$

dalla formula

$$\sum_{i=0}^{\infty} \alpha^i = \frac{1}{1 - \alpha} \quad (2)$$

Pertanto

$$T(n) < d \cdot n^{\log_c a} + bcn/(c - a) = O(n)$$

(Infatti, $a < c \Rightarrow \log_c a < 1$ e quindi il termine $d \cdot n^{\log_c a}$ è $O(n)$).

Caso 2: $a = c$

Caso 2: $a = c$

$$T(n) = d \cdot n^{\log_c a} + bn \sum_{j=0}^{\log_c n - 1} (a/c)^j$$

Caso 2: $a = c$

$$\begin{aligned} T(n) &= d \cdot n^{\log_c a} + bn \sum_{j=0}^{\log_c n - 1} (a/c)^j \\ &= d \cdot n^{\log_c a} + bn \sum_{j=0}^{\log_c n - 1} (1)^j \end{aligned}$$

Caso 2: $a = c$

$$\begin{aligned}T(n) &= d \cdot n^{\log_c a} + bn \sum_{j=0}^{\log_c n - 1} (a/c)^j \\&= d \cdot n^{\log_c a} + bn \sum_{j=0}^{\log_c n - 1} (1)^j \\&= dn + bn \log n\end{aligned}$$

Caso 2: $a = c$

$$\begin{aligned}T(n) &= d \cdot n^{\log_c a} + bn \sum_{j=0}^{\log_c n - 1} (a/c)^j \\&= d \cdot n^{\log_c a} + bn \sum_{j=0}^{\log_c n - 1} (1)^j \\&= dn + bn \log n \\&= O(n \log n)\end{aligned}$$

Caso 3: $a > c$ (il che implica $(a/c) \neq 1$)

Caso 3: $a > c$ (il che implica $(a/c) \neq 1$)

Dalla formula

$$\sum_{i=0}^n \alpha^i = \frac{\alpha^{n+1} - 1}{\alpha - 1} \quad (1)$$

Caso 3: $a > c$ (il che implica $(a/c) \neq 1$)

Dalla formula

$$\sum_{i=0}^n \alpha^i = \frac{\alpha^{n+1} - 1}{\alpha - 1} \quad (1)$$

otteniamo

$$\sum_{j=0}^{\log_c n - 1} (a/c)^j = \frac{(a/c)^{\log_c n} - 1}{(a/c) - 1}$$

Caso 3: $a > c$ (il che implica $(a/c) \neq 1$)

Dalla formula

$$\sum_{i=0}^n \alpha^i = \frac{\alpha^{n+1} - 1}{\alpha - 1} \quad (1)$$

otteniamo

$$\begin{aligned} \sum_{j=0}^{\log_c n - 1} (a/c)^j &= \frac{(a/c)^{\log_c n} - 1}{(a/c) - 1} \\ &= \frac{n^{\log_c a - 1} - 1}{(a/c) - 1} \quad (\text{poichè } a^{\log_c n} = n^{\log_c a} \text{ e } (1/c)^{\log_c n} = 1/n) \end{aligned}$$

Caso 3: $a > c$ (il che implica $(a/c) \neq 1$)

Dalla formula

$$\sum_{i=0}^n \alpha^i = \frac{\alpha^{n+1} - 1}{\alpha - 1} \quad (1)$$

otteniamo

$$\begin{aligned} \sum_{j=0}^{\log_c n - 1} (a/c)^j &= \frac{(a/c)^{\log_c n} - 1}{(a/c) - 1} \\ &= \frac{n^{\log_c a - 1} - 1}{(a/c) - 1} && \text{(poichè } a^{\log_c n} = n^{\log_c a} \text{ e } (1/c)^{\log_c n} = 1/n) \\ &= O(n^{\log_c a - 1}) \end{aligned}$$

Caso 3: $a > c$ (il che implica $(a/c) \neq 1$)

Dalla formula

$$\sum_{i=0}^n \alpha^i = \frac{\alpha^{n+1} - 1}{\alpha - 1} \quad (1)$$

otteniamo

$$\begin{aligned} \sum_{j=0}^{\log_c n - 1} (a/c)^j &= \frac{(a/c)^{\log_c n} - 1}{(a/c) - 1} \\ &= \frac{n^{\log_c a - 1} - 1}{(a/c) - 1} && \text{(poichè } a^{\log_c n} = n^{\log_c a} \text{ e } (1/c)^{\log_c n} = 1/n) \\ &= O(n^{\log_c a - 1}) \end{aligned}$$

Pertanto,

$$T(n) = d \cdot n^{\log_c a} + bn \sum_{j=0}^{\log_c n - 1} (a/c)^j$$

Caso 3: $a > c$ (il che implica $(a/c) \neq 1$)

Dalla formula

$$\sum_{i=0}^n \alpha^i = \frac{\alpha^{n+1} - 1}{\alpha - 1} \quad (1)$$

otteniamo

$$\begin{aligned} \sum_{j=0}^{\log_c n - 1} (a/c)^j &= \frac{(a/c)^{\log_c n} - 1}{(a/c) - 1} \\ &= \frac{n^{\log_c a - 1} - 1}{(a/c) - 1} && \text{(poichè } a^{\log_c n} = n^{\log_c a} \text{ e } (1/c)^{\log_c n} = 1/n) \\ &= O(n^{\log_c a - 1}) \end{aligned}$$

Pertanto,

$$T(n) = d \cdot n^{\log_c a} + bn \sum_{j=0}^{\log_c n - 1} (a/c)^j = O(n^{\log_c a})$$

Cosa accade se n NON è potenza di c ?

Cosa accade se n NON è potenza di c ?

Osserviamo che per $k = \lfloor \log_c n \rfloor$ vale che

$$c^k \leq n < c^{k+1} \quad \text{e} \quad cn \geq c^{k+1} \quad (3)$$

Cosa accade se n NON è potenza di c ?

Osserviamo che per $k = \lfloor \log_c n \rfloor$ vale che

$$c^k \leq n < c^{k+1} \quad \text{e} \quad cn \geq c^{k+1} \quad (3)$$

Immaginiamo quindi di "aumentare" l'input di taglia n , fino a farlo diventare di taglia c^{k+1} .

Cosa accade se n NON è potenza di c ?
Osserviamo che per $k = \lfloor \log_c n \rfloor$ vale che

$$c^k \leq n < c^{k+1} \quad \text{e} \quad cn \geq c^{k+1} \quad (3)$$

Immaginiamo quindi di "aumentare" l'input di taglia n , fino a farlo diventare di taglia c^{k+1} . É ovvio che $T(n) < T(c^{k+1})$.

Cosa accade se n NON è potenza di c ?
Osserviamo che per $k = \lfloor \log_c n \rfloor$ vale che

$$c^k \leq n < c^{k+1} \quad \text{e} \quad cn \geq c^{k+1} \quad (3)$$

Immaginiamo quindi di "aumentare" l'input di taglia n , fino a farlo diventare di taglia c^{k+1} . É ovvio che $T(n) < T(c^{k+1})$.
Applicando ora il teorema a $T(c^{k+1})$, otteniamo

$T(n)$

Cosa accade se n NON è potenza di c ?
Osserviamo che per $k = \lfloor \log_c n \rfloor$ vale che

$$c^k \leq n < c^{k+1} \quad \text{e} \quad cn \geq c^{k+1} \quad (3)$$

Immaginiamo quindi di "aumentare" l'input di taglia n , fino a farlo diventare di taglia c^{k+1} . È ovvio che $T(n) < T(c^{k+1})$.
Applicando ora il teorema a $T(c^{k+1})$, otteniamo

$$T(n) < T(c^{k+1}) = \begin{cases} O(c^{k+1}) = O(cn) \end{cases}$$

Cosa accade se n NON è potenza di c ?

Osserviamo che per $k = \lfloor \log_c n \rfloor$ vale che

$$c^k \leq n < c^{k+1} \quad \text{e} \quad cn \geq c^{k+1} \quad (3)$$

Immaginiamo quindi di "aumentare" l'input di taglia n , fino a farlo diventare di taglia c^{k+1} . È ovvio che $T(n) < T(c^{k+1})$.

Applicando ora il teorema a $T(c^{k+1})$, otteniamo

$$T(n) < T(c^{k+1}) = \begin{cases} O(c^{k+1}) = O(cn) = O(n) & \text{se } a < c \end{cases}$$

Cosa accade se n NON è potenza di c ?
Osserviamo che per $k = \lfloor \log_c n \rfloor$ vale che

$$c^k \leq n < c^{k+1} \quad \text{e} \quad cn \geq c^{k+1} \quad (3)$$

Immaginiamo quindi di "aumentare" l'input di taglia n , fino a farlo diventare di taglia c^{k+1} . É ovvio che $T(n) < T(c^{k+1})$.

Applicando ora il teorema a $T(c^{k+1})$, otteniamo

$$T(n) < T(c^{k+1}) = \begin{cases} O(c^{k+1}) = O(cn) = O(n) \\ O(c^{k+1} \log c^{k+1}) = O(cn \log(cn)) \end{cases} \quad \text{se } a < c$$

Cosa accade se n NON è potenza di c ?

Osserviamo che per $k = \lfloor \log_c n \rfloor$ vale che

$$c^k \leq n < c^{k+1} \quad \text{e} \quad cn \geq c^{k+1} \quad (3)$$

Immaginiamo quindi di "aumentare" l'input di taglia n , fino a farlo diventare di taglia c^{k+1} . È ovvio che $T(n) < T(c^{k+1})$.

Applicando ora il teorema a $T(c^{k+1})$, otteniamo

$$T(n) < T(c^{k+1}) = \begin{cases} O(c^{k+1}) = O(cn) = O(n) & \text{se } a < c \\ O(c^{k+1} \log c^{k+1}) = O(cn \log(cn)) = O(n \log n) & \text{se } a = c \end{cases}$$

Cosa accade se n NON è potenza di c ?
Osserviamo che per $k = \lfloor \log_c n \rfloor$ vale che

$$c^k \leq n < c^{k+1} \quad \text{e} \quad cn \geq c^{k+1} \quad (3)$$

Immaginiamo quindi di "aumentare" l'input di taglia n , fino a farlo diventare di taglia c^{k+1} . È ovvio che $T(n) < T(c^{k+1})$.

Applicando ora il teorema a $T(c^{k+1})$, otteniamo

$$T(n) < T(c^{k+1}) = \begin{cases} O(c^{k+1}) = O(cn) = O(n) & \text{se } a < c \\ O(c^{k+1} \log c^{k+1}) = O(cn \log(cn)) = O(n \log n) & \text{se } a = c \\ O((c^{k+1})^{\log_c a}) & \end{cases}$$

Cosa accade se n NON è potenza di c ?

Osserviamo che per $k = \lfloor \log_c n \rfloor$ vale che

$$c^k \leq n < c^{k+1} \quad \text{e} \quad cn \geq c^{k+1} \quad (3)$$

Immaginiamo quindi di "aumentare" l'input di taglia n , fino a farlo diventare di taglia c^{k+1} . È ovvio che $T(n) < T(c^{k+1})$.

Applicando ora il teorema a $T(c^{k+1})$, otteniamo

$$T(n) < T(c^{k+1}) = \begin{cases} O(c^{k+1}) = O(cn) = O(n) & \text{se } a < c \\ O(c^{k+1} \log c^{k+1}) = O(cn \log(cn)) = O(n \log n) & \text{se } a = c \\ O((c^{k+1})^{\log_c a}) = O((cn)^{\log_c a}) & \text{se } a > c \end{cases}$$

Cosa accade se n NON è potenza di c ?

Osserviamo che per $k = \lfloor \log_c n \rfloor$ vale che

$$c^k \leq n < c^{k+1} \quad \text{e} \quad cn \geq c^{k+1} \quad (3)$$

Immaginiamo quindi di "aumentare" l'input di taglia n , fino a farlo diventare di taglia c^{k+1} . É ovvio che $T(n) < T(c^{k+1})$.

Applicando ora il teorema a $T(c^{k+1})$, otteniamo

$$T(n) < T(c^{k+1}) = \begin{cases} O(c^{k+1}) = O(cn) = O(n) & \text{se } a < c \\ O(c^{k+1} \log c^{k+1}) = O(cn \log(cn)) = O(n \log n) & \text{se } a = c \\ O((c^{k+1})^{\log_c a}) = O((cn)^{\log_c a}) = O(n^{\log_c a}) & \text{se } a > c \end{cases}$$

Ricordiamo: Per arbitrarie costanti c , a , e d la soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn & \text{altrimenti} \end{cases}$$

è data da

$$T(n) = \begin{cases} O(n) & \text{se } a < c \\ O(n \log n) & \text{se } a = c \\ O(n^{\log_c a}) & \text{se } a > c \end{cases}$$

Ricordiamo: Per arbitrarie costanti c , a , e d la soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn & \text{altrimenti} \end{cases}$$

è data da

$$T(n) = \begin{cases} O(n) & \text{se } a < c \\ O(n \log n) & \text{se } a = c \\ O(n^{\log_c a}) & \text{se } a > c \end{cases}$$

Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + 6n$

Ricordiamo: Per arbitrarie costanti c , a , e d la soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn & \text{altrimenti} \end{cases}$$

è data da

$$T(n) = \begin{cases} O(n) & \text{se } a < c \\ O(n \log n) & \text{se } a = c \\ O(n^{\log_c a}) & \text{se } a > c \end{cases}$$

Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + 6n \quad T(n) = O(n \log n)$

Ricordiamo: Per arbitrarie costanti c , a , e d la soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn & \text{altrimenti} \end{cases}$$

è data da

$$T(n) = \begin{cases} O(n) & \text{se } a < c \\ O(n \log n) & \text{se } a = c \\ O(n^{\log_c a}) & \text{se } a > c \end{cases}$$

Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + 6n \quad T(n) = O(n \log n)$
- $T(n) = 3T(n/3) + 6n - 9$

Ricordiamo: Per arbitrarie costanti c , a , e d la soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn & \text{altrimenti} \end{cases}$$

è data da

$$T(n) = \begin{cases} O(n) & \text{se } a < c \\ O(n \log n) & \text{se } a = c \\ O(n^{\log_c a}) & \text{se } a > c \end{cases}$$

Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + 6n$ $T(n) = O(n \log n)$
- $T(n) = 3T(n/3) + 6n - 9$ $T(n) = O(n \log n)$

Ricordiamo: Per arbitrarie costanti c , a , e d la soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn & \text{altrimenti} \end{cases}$$

è data da

$$T(n) = \begin{cases} O(n) & \text{se } a < c \\ O(n \log n) & \text{se } a = c \\ O(n^{\log_c a}) & \text{se } a > c \end{cases}$$

Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + 6n$ $T(n) = O(n \log n)$
- $T(n) = 3T(n/3) + 6n - 9$ $T(n) = O(n \log n)$
- $T(n) = 2T(n/3) + 5n$

Ricordiamo: Per arbitrarie costanti c , a , e d la soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn & \text{altrimenti} \end{cases}$$

è data da

$$T(n) = \begin{cases} O(n) & \text{se } a < c \\ O(n \log n) & \text{se } a = c \\ O(n^{\log_c a}) & \text{se } a > c \end{cases}$$

Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + 6n$ $T(n) = O(n \log n)$
- $T(n) = 3T(n/3) + 6n - 9$ $T(n) = O(n \log n)$
- $T(n) = 2T(n/3) + 5n$ $T(n) = O(n)$

Ricordiamo: Per arbitrarie costanti c , a , e d la soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn & \text{altrimenti} \end{cases}$$

è data da

$$T(n) = \begin{cases} O(n) & \text{se } a < c \\ O(n \log n) & \text{se } a = c \\ O(n^{\log_c a}) & \text{se } a > c \end{cases}$$

Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + 6n \quad T(n) = O(n \log n)$
- $T(n) = 3T(n/3) + 6n - 9 \quad T(n) = O(n \log n)$
- $T(n) = 2T(n/3) + 5n \quad T(n) = O(n)$
- $T(n) = 2T(n/3) + 12n + 16$

Ricordiamo: Per arbitrarie costanti c , a , e d la soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn & \text{altrimenti} \end{cases}$$

è data da

$$T(n) = \begin{cases} O(n) & \text{se } a < c \\ O(n \log n) & \text{se } a = c \\ O(n^{\log_c a}) & \text{se } a > c \end{cases}$$

Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + 6n$ $T(n) = O(n \log n)$
- $T(n) = 3T(n/3) + 6n - 9$ $T(n) = O(n \log n)$
- $T(n) = 2T(n/3) + 5n$ $T(n) = O(n)$
- $T(n) = 2T(n/3) + 12n + 16$ $T(n) = O(n)$

Ricordiamo: Per arbitrarie costanti c , a , e d la soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn & \text{altrimenti} \end{cases}$$

è data da

$$T(n) = \begin{cases} O(n) & \text{se } a < c \\ O(n \log n) & \text{se } a = c \\ O(n^{\log_c a}) & \text{se } a > c \end{cases}$$

Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + 6n$ $T(n) = O(n \log n)$
- $T(n) = 3T(n/3) + 6n - 9$ $T(n) = O(n \log n)$
- $T(n) = 2T(n/3) + 5n$ $T(n) = O(n)$
- $T(n) = 2T(n/3) + 12n + 16$ $T(n) = O(n)$
- $T(n) = 4T(n/2) + n$

Ricordiamo: Per arbitrarie costanti c , a , e d la soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn & \text{altrimenti} \end{cases}$$

è data da

$$T(n) = \begin{cases} O(n) & \text{se } a < c \\ O(n \log n) & \text{se } a = c \\ O(n^{\log_c a}) & \text{se } a > c \end{cases}$$

Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + 6n$ $T(n) = O(n \log n)$
- $T(n) = 3T(n/3) + 6n - 9$ $T(n) = O(n \log n)$
- $T(n) = 2T(n/3) + 5n$ $T(n) = O(n)$
- $T(n) = 2T(n/3) + 12n + 16$ $T(n) = O(n)$
- $T(n) = 4T(n/2) + n$ $T(n) = O(n^{\log_2 4}) = O(n^2)$

Ricordiamo: Per arbitrarie costanti c , a , e d la soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn & \text{altrimenti} \end{cases}$$

è data da

$$T(n) = \begin{cases} O(n) & \text{se } a < c \\ O(n \log n) & \text{se } a = c \\ O(n^{\log_c a}) & \text{se } a > c \end{cases}$$

Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + 6n$ $T(n) = O(n \log n)$
- $T(n) = 3T(n/3) + 6n - 9$ $T(n) = O(n \log n)$
- $T(n) = 2T(n/3) + 5n$ $T(n) = O(n)$
- $T(n) = 2T(n/3) + 12n + 16$ $T(n) = O(n)$
- $T(n) = 4T(n/2) + n$ $T(n) = O(n^{\log_2 4}) = O(n^2)$
- $T(n) = 3T(n/2) + 9n$

Ricordiamo: Per arbitrarie costanti c , a , e d la soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn & \text{altrimenti} \end{cases}$$

è data da

$$T(n) = \begin{cases} O(n) & \text{se } a < c \\ O(n \log n) & \text{se } a = c \\ O(n^{\log_c a}) & \text{se } a > c \end{cases}$$

Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + 6n$ $T(n) = O(n \log n)$
- $T(n) = 3T(n/3) + 6n - 9$ $T(n) = O(n \log n)$
- $T(n) = 2T(n/3) + 5n$ $T(n) = O(n)$
- $T(n) = 2T(n/3) + 12n + 16$ $T(n) = O(n)$
- $T(n) = 4T(n/2) + n$ $T(n) = O(n^{\log_2 4}) = O(n^2)$
- $T(n) = 3T(n/2) + 9n$ $T(n) = O(n^{\log_2 3}) = O(n^{1.584\dots})$

Con esattamente la stessa tecnica è possibile dimostrare una forma un pò più generale del Teorema,

Con esattamente la stessa tecnica è possibile dimostrare una forma un pò più generale del Teorema, ad esempio la seguente:

Teorema: La soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn^k & \text{altrimenti} \end{cases}$$

per a, c, b, k costanti,

Con esattamente la stessa tecnica è possibile dimostrare una forma un pò più generale del Teorema, ad esempio la seguente:

Teorema: La soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn^k & \text{altrimenti} \end{cases}$$

per a, c, b, k costanti, è

$$T(n) = \begin{cases} O(n^k) & \text{se } a < c^k \end{cases}$$

Con esattamente la stessa tecnica è possibile dimostrare una forma un pò più generale del Teorema, ad esempio la seguente:

Teorema: La soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn^k & \text{altrimenti} \end{cases}$$

per a, c, b, k costanti, è

$$T(n) = \begin{cases} O(n^k) & \text{se } a < c^k \\ O(n^k \log n) & \text{se } a = c^k \end{cases}$$

Con esattamente la stessa tecnica è possibile dimostrare una forma un pò più generale del Teorema, ad esempio la seguente:

Teorema: La soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn^k & \text{altrimenti} \end{cases}$$

per a, c, b, k costanti, è

$$T(n) = \begin{cases} O(n^k) & \text{se } a < c^k \\ O(n^k \log n) & \text{se } a = c^k \\ O(n^{\log_c a}) & \text{se } a > c^k \end{cases}$$

Con esattamente la stessa tecnica è possibile dimostrare una forma un pò più generale del Teorema, ad esempio la seguente:

Teorema: La soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn^k & \text{altrimenti} \end{cases}$$

per a, c, b, k costanti, è

$$T(n) = \begin{cases} O(n^k) & \text{se } a < c^k \\ O(n^k \log n) & \text{se } a = c^k \\ O(n^{\log_c a}) & \text{se } a > c^k \end{cases}$$

Esempi. Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + n^3$

Con esattamente la stessa tecnica è possibile dimostrare una forma un pò più generale del Teorema, ad esempio la seguente:

Teorema: La soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn^k & \text{altrimenti} \end{cases}$$

per a, c, b, k costanti, è

$$T(n) = \begin{cases} O(n^k) & \text{se } a < c^k \\ O(n^k \log n) & \text{se } a = c^k \\ O(n^{\log_c a}) & \text{se } a > c^k \end{cases}$$

Esempi. Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + n^3 \quad T(n) = O(n^3)$

Con esattamente la stessa tecnica è possibile dimostrare una forma un pò più generale del Teorema, ad esempio la seguente:

Teorema: La soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn^k & \text{altrimenti} \end{cases}$$

per a, c, b, k costanti, è

$$T(n) = \begin{cases} O(n^k) & \text{se } a < c^k \\ O(n^k \log n) & \text{se } a = c^k \\ O(n^{\log_c a}) & \text{se } a > c^k \end{cases}$$

Esempi. Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + n^3 \quad T(n) = O(n^3)$
- $T(n) = T(9n/10) + n$

Con esattamente la stessa tecnica è possibile dimostrare una forma un pò più generale del Teorema, ad esempio la seguente:

Teorema: La soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn^k & \text{altrimenti} \end{cases}$$

per a, c, b, k costanti, è

$$T(n) = \begin{cases} O(n^k) & \text{se } a < c^k \\ O(n^k \log n) & \text{se } a = c^k \\ O(n^{\log_c a}) & \text{se } a > c^k \end{cases}$$

Esempi. Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + n^3 \quad T(n) = O(n^3)$
- $T(n) = T(9n/10) + n \quad T(n) = O(n)$

Con esattamente la stessa tecnica è possibile dimostrare una forma un pò più generale del Teorema, ad esempio la seguente:

Teorema: La soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn^k & \text{altrimenti} \end{cases}$$

per a, c, b, k costanti, è

$$T(n) = \begin{cases} O(n^k) & \text{se } a < c^k \\ O(n^k \log n) & \text{se } a = c^k \\ O(n^{\log_c a}) & \text{se } a > c^k \end{cases}$$

Esempi. Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + n^3 \quad T(n) = O(n^3)$
- $T(n) = T(9n/10) + n \quad T(n) = O(n)$
- $T(n) = 16T(n/4) + n^2$

Con esattamente la stessa tecnica è possibile dimostrare una forma un pò più generale del Teorema, ad esempio la seguente:

Teorema: La soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn^k & \text{altrimenti} \end{cases}$$

per a, c, b, k costanti, è

$$T(n) = \begin{cases} O(n^k) & \text{se } a < c^k \\ O(n^k \log n) & \text{se } a = c^k \\ O(n^{\log_c a}) & \text{se } a > c^k \end{cases}$$

Esempi. Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + n^3$ $T(n) = O(n^3)$
- $T(n) = T(9n/10) + n$ $T(n) = O(n)$
- $T(n) = 16T(n/4) + n^2$ $T(n) = O(n^2 \log n)$

Con esattamente la stessa tecnica è possibile dimostrare una forma un pò più generale del Teorema, ad esempio la seguente:

Teorema: La soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn^k & \text{altrimenti} \end{cases}$$

per a, c, b, k costanti, è

$$T(n) = \begin{cases} O(n^k) & \text{se } a < c^k \\ O(n^k \log n) & \text{se } a = c^k \\ O(n^{\log_c a}) & \text{se } a > c^k \end{cases}$$

Esempi. Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + n^3$ $T(n) = O(n^3)$
- $T(n) = T(9n/10) + n$ $T(n) = O(n)$
- $T(n) = 16T(n/4) + n^2$ $T(n) = O(n^2 \log n)$
- $T(n) = 7T(n/3) + n^2$

Con esattamente la stessa tecnica è possibile dimostrare una forma un pò più generale del Teorema, ad esempio la seguente:

Teorema: La soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn^k & \text{altrimenti} \end{cases}$$

per a, c, b, k costanti, è

$$T(n) = \begin{cases} O(n^k) & \text{se } a < c^k \\ O(n^k \log n) & \text{se } a = c^k \\ O(n^{\log_c a}) & \text{se } a > c^k \end{cases}$$

Esempi. Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + n^3$ $T(n) = O(n^3)$
- $T(n) = T(9n/10) + n$ $T(n) = O(n)$
- $T(n) = 16T(n/4) + n^2$ $T(n) = O(n^2 \log n)$
- $T(n) = 7T(n/3) + n^2$ $T(n) = O(n^2)$

Con esattamente la stessa tecnica è possibile dimostrare una forma un pò più generale del Teorema, ad esempio la seguente:

Teorema: La soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn^k & \text{altrimenti} \end{cases}$$

per a, c, b, k costanti, è

$$T(n) = \begin{cases} O(n^k) & \text{se } a < c^k \\ O(n^k \log n) & \text{se } a = c^k \\ O(n^{\log_c a}) & \text{se } a > c^k \end{cases}$$

Esempi. Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + n^3$ $T(n) = O(n^3)$
- $T(n) = T(9n/10) + n$ $T(n) = O(n)$
- $T(n) = 16T(n/4) + n^2$ $T(n) = O(n^2 \log n)$
- $T(n) = 7T(n/3) + n^2$ $T(n) = O(n^2)$
- $T(n) = 7T(n/2) + n^2$

Con esattamente la stessa tecnica è possibile dimostrare una forma un pò più generale del Teorema, ad esempio la seguente:

Teorema: La soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn^k & \text{altrimenti} \end{cases}$$

per a, c, b, k costanti, è

$$T(n) = \begin{cases} O(n^k) & \text{se } a < c^k \\ O(n^k \log n) & \text{se } a = c^k \\ O(n^{\log_c a}) & \text{se } a > c^k \end{cases}$$

Esempi. Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + n^3$ $T(n) = O(n^3)$
- $T(n) = T(9n/10) + n$ $T(n) = O(n)$
- $T(n) = 16T(n/4) + n^2$ $T(n) = O(n^2 \log n)$
- $T(n) = 7T(n/3) + n^2$ $T(n) = O(n^2)$
- $T(n) = 7T(n/2) + n^2$ $T(n) = O(n^{\log_2 7})$

Con esattamente la stessa tecnica è possibile dimostrare una forma un pò più generale del Teorema, ad esempio la seguente:

Teorema: La soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn^k & \text{altrimenti} \end{cases}$$

per a, c, b, k costanti, è

$$T(n) = \begin{cases} O(n^k) & \text{se } a < c^k \\ O(n^k \log n) & \text{se } a = c^k \\ O(n^{\log_c a}) & \text{se } a > c^k \end{cases}$$

Esempi. Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + n^3$ $T(n) = O(n^3)$
- $T(n) = T(9n/10) + n$ $T(n) = O(n)$
- $T(n) = 16T(n/4) + n^2$ $T(n) = O(n^2 \log n)$
- $T(n) = 7T(n/3) + n^2$ $T(n) = O(n^2)$
- $T(n) = 7T(n/2) + n^2$ $T(n) = O(n^{\log_2 7})$
- $T(n) = 2T(n/3) + \sqrt{n}$

Con esattamente la stessa tecnica è possibile dimostrare una forma un pò più generale del Teorema, ad esempio la seguente:

Teorema: La soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn^k & \text{altrimenti} \end{cases}$$

per a, c, b, k costanti, è

$$T(n) = \begin{cases} O(n^k) & \text{se } a < c^k \\ O(n^k \log n) & \text{se } a = c^k \\ O(n^{\log_c a}) & \text{se } a > c^k \end{cases}$$

Esempi. Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + n^3$ $T(n) = O(n^3)$
- $T(n) = T(9n/10) + n$ $T(n) = O(n)$
- $T(n) = 16T(n/4) + n^2$ $T(n) = O(n^2 \log n)$
- $T(n) = 7T(n/3) + n^2$ $T(n) = O(n^2)$
- $T(n) = 7T(n/2) + n^2$ $T(n) = O(n^{\log_2 7})$
- $T(n) = 2T(n/3) + \sqrt{n}$ $T(n) = O(n^{\log_3 2})$

Con esattamente la stessa tecnica è possibile dimostrare una forma un pò più generale del Teorema, ad esempio la seguente:

Teorema: La soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn^k & \text{altrimenti} \end{cases}$$

per a, c, b, k costanti, è

$$T(n) = \begin{cases} O(n^k) & \text{se } a < c^k \\ O(n^k \log n) & \text{se } a = c^k \\ O(n^{\log_c a}) & \text{se } a > c^k \end{cases}$$

Esempi. Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + n^3$ $T(n) = O(n^3)$
- $T(n) = T(9n/10) + n$ $T(n) = O(n)$
- $T(n) = 16T(n/4) + n^2$ $T(n) = O(n^2 \log n)$
- $T(n) = 7T(n/3) + n^2$ $T(n) = O(n^2)$
- $T(n) = 7T(n/2) + n^2$ $T(n) = O(n^{\log_2 7})$
- $T(n) = 2T(n/3) + \sqrt{n}$ $T(n) = O(n^{\log_3 2})$
- $T(n) = T(n-1) + n$

Con esattamente la stessa tecnica è possibile dimostrare una forma un pò più generale del Teorema, ad esempio la seguente:

Teorema: La soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn^k & \text{altrimenti} \end{cases}$$

per a, c, b, k costanti, è

$$T(n) = \begin{cases} O(n^k) & \text{se } a < c^k \\ O(n^k \log n) & \text{se } a = c^k \\ O(n^{\log_c a}) & \text{se } a > c^k \end{cases}$$

Esempi. Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + n^3$ $T(n) = O(n^3)$
- $T(n) = T(9n/10) + n$ $T(n) = O(n)$
- $T(n) = 16T(n/4) + n^2$ $T(n) = O(n^2 \log n)$
- $T(n) = 7T(n/3) + n^2$ $T(n) = O(n^2)$
- $T(n) = 7T(n/2) + n^2$ $T(n) = O(n^{\log_2 7})$
- $T(n) = 2T(n/3) + \sqrt{n}$ $T(n) = O(n^{\log_3 2})$
- $T(n) = T(n-1) + n$ $T(n) = O(?)$

Con esattamente la stessa tecnica è possibile dimostrare una forma un pò più generale del Teorema, ad esempio la seguente:

Teorema: La soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn^k & \text{altrimenti} \end{cases}$$

per a, c, b, k costanti, è

$$T(n) = \begin{cases} O(n^k) & \text{se } a < c^k \\ O(n^k \log n) & \text{se } a = c^k \\ O(n^{\log_c a}) & \text{se } a > c^k \end{cases}$$

Esempi. Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + n^3$ $T(n) = O(n^3)$
- $T(n) = T(9n/10) + n$ $T(n) = O(n)$
- $T(n) = 16T(n/4) + n^2$ $T(n) = O(n^2 \log n)$
- $T(n) = 7T(n/3) + n^2$ $T(n) = O(n^2)$
- $T(n) = 7T(n/2) + n^2$ $T(n) = O(n^{\log_2 7})$
- $T(n) = 2T(n/3) + \sqrt{n}$ $T(n) = O(n^{\log_3 2})$
- $T(n) = T(n-1) + n$ $T(n) = O(?)$
- $T(n) = T(\sqrt{n}) + 1$

Con esattamente la stessa tecnica è possibile dimostrare una forma un pò più generale del Teorema, ad esempio la seguente:

Teorema: La soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn^k & \text{altrimenti} \end{cases}$$

per a, c, b, k costanti, è

$$T(n) = \begin{cases} O(n^k) & \text{se } a < c^k \\ O(n^k \log n) & \text{se } a = c^k \\ O(n^{\log_c a}) & \text{se } a > c^k \end{cases}$$

Esempi. Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + n^3$ $T(n) = O(n^3)$
- $T(n) = T(9n/10) + n$ $T(n) = O(n)$
- $T(n) = 16T(n/4) + n^2$ $T(n) = O(n^2 \log n)$
- $T(n) = 7T(n/3) + n^2$ $T(n) = O(n^2)$
- $T(n) = 7T(n/2) + n^2$ $T(n) = O(n^{\log_2 7})$
- $T(n) = 2T(n/3) + \sqrt{n}$ $T(n) = O(n^{\log_3 2})$
- $T(n) = T(n-1) + n$ $T(n) = O(?)$
- $T(n) = T(\sqrt{n}) + 1$ $T(n) = O(?)$