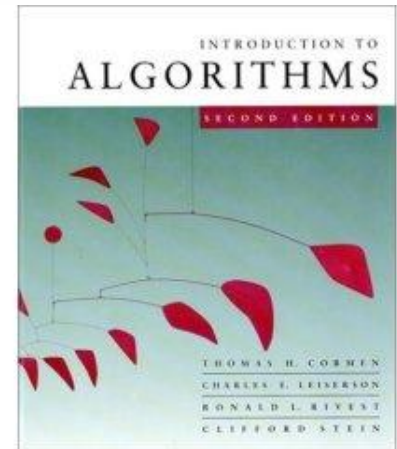




Quicksort

Algoritmi, Strutture Dati e Calcolo Parallelo

- ❑ Questo materiale è tratto dalle trasparenze del corso Introduction to Algorithms (2005-fall-6046) tenuto dal Prof. Leiserson all'MIT (<http://people.csail.mit.edu/cel/>)
- ❑ T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein Introduction to Algorithms, Second Edition, The MIT Press, Cambridge, Massachusetts London, England McGraw-Hill Book Company
- ❑ Queste trasparenze sono disponibili sui siti <http://www.pierlucalanzi.net>
<http://www.slideshare.net/pierluca.lanzi>

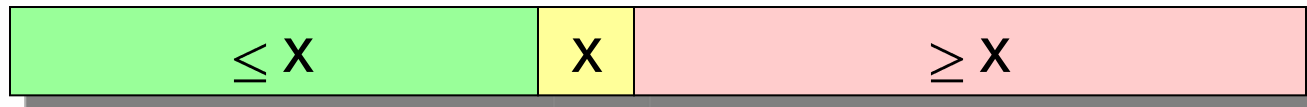


Quicksort

Quicksort

- ❑ Introdotto da Sir Charles Antony Richard Hoare nel 1962.
- ❑ È un approccio Divide-et-Impera
- ❑ Merge-Sort suddivide un vettore in due sottovettori che sono ordinati ricorsivamente.
- ❑ Il quicksort ordina sul posto (come l'insertion-sort) ma è asintoticamente più veloce
- ❑ Molto funzionale, con un tuning opportuno

- ❑ Supponiamo di dover ordinare un vettore di n elementi
- ❑ Divide: Suddivide il vettore in due parti attorno a un **pivot x**
 - ▶ Il sottovettore alla sinistra di x contiene elementi $\leq x$
 - ▶ Il sottovettore alla destra di x contiene elementi $\geq x$



- ❑ Impera: ordina ricorsivamente i due sottovettori
- ❑ Ricombina: ovvio

Punto chiave: il partitioning è lineare

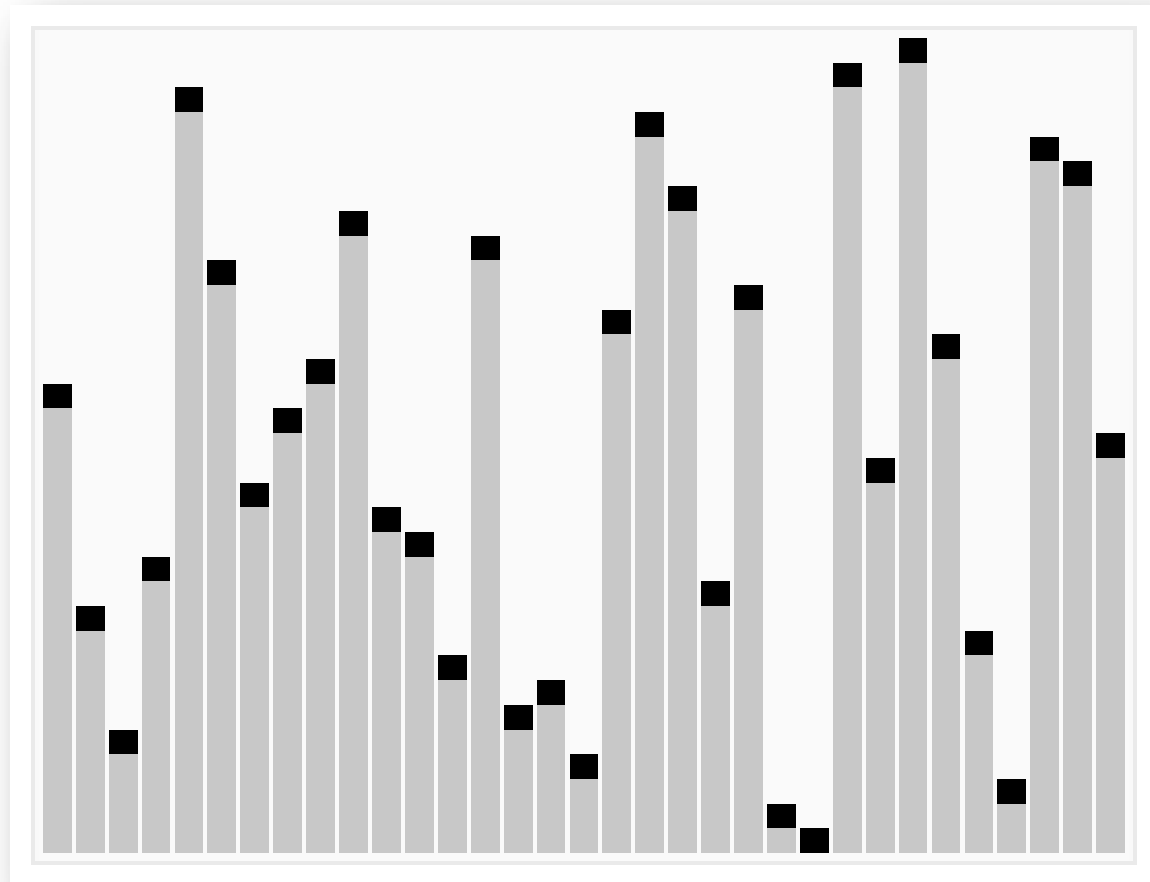
Pseudocodice del Quicksort

- Procedura che dato un vettore A ordina le posizioni comprese fra l'indice iniziale p e l'indice finale r

```
QUICKSORT( $A, p, r$ )  
  if  $p < r$   
    then  $q \leftarrow$  PARTITION( $A, p, r$ )  
         QUICKSORT( $A, p, q-1$ )  
         QUICKSORT( $A, q+1, r$ )
```

- Per ordinare tutto il vettore viene effettuata la chiamata

```
QUICKSORT( $A, 1, n$ )
```



Animazione del Quicksort da Wikipedia

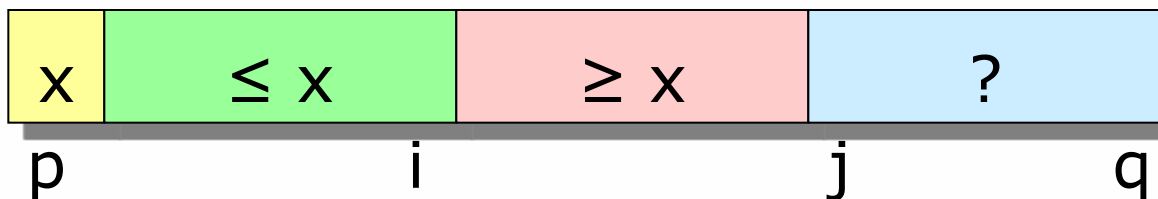
<http://en.wikipedia.org/wiki/Quicksort>

http://en.wikipedia.org/wiki/Image:Sorting_quicksort_anim.gif

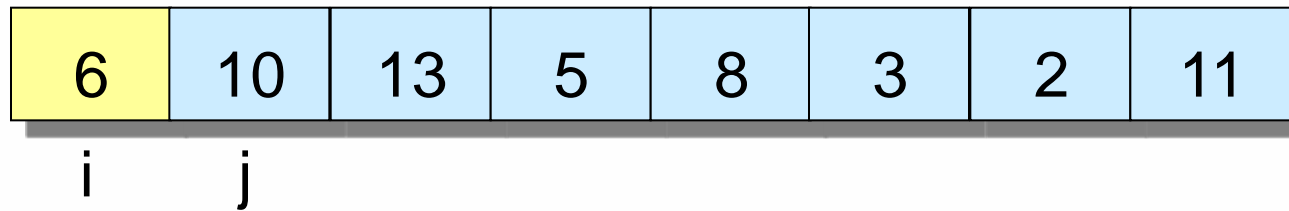
La Procedura di Partitioning

```
PARTITION(A, p, q) ▷ A[p . . q]
  x ← A[p]          ▷ pivot = A[p]
  i ← p
  for j ← p + 1 to q
    do if A[j] ≤ x
      then i ← i + 1
           exchange A[i] ↔ A[j]
  exchange A[p] ↔ A[i]
  return i
```

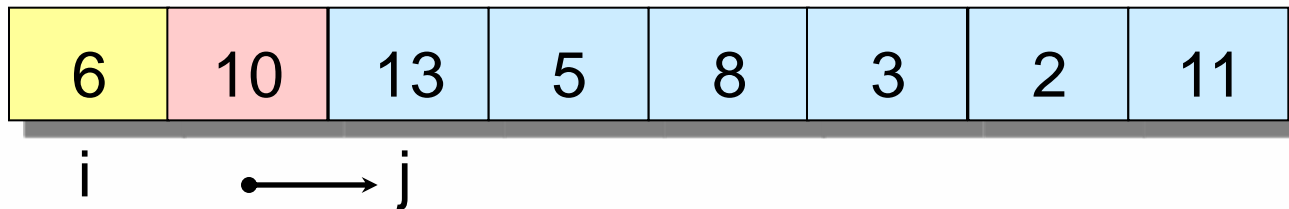
Complessità $\Theta(n)$



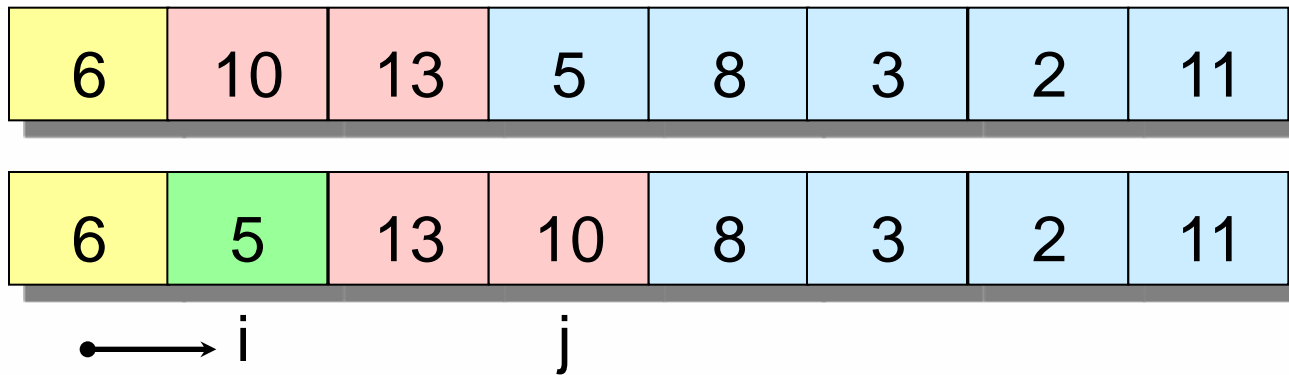
Example of partitioning



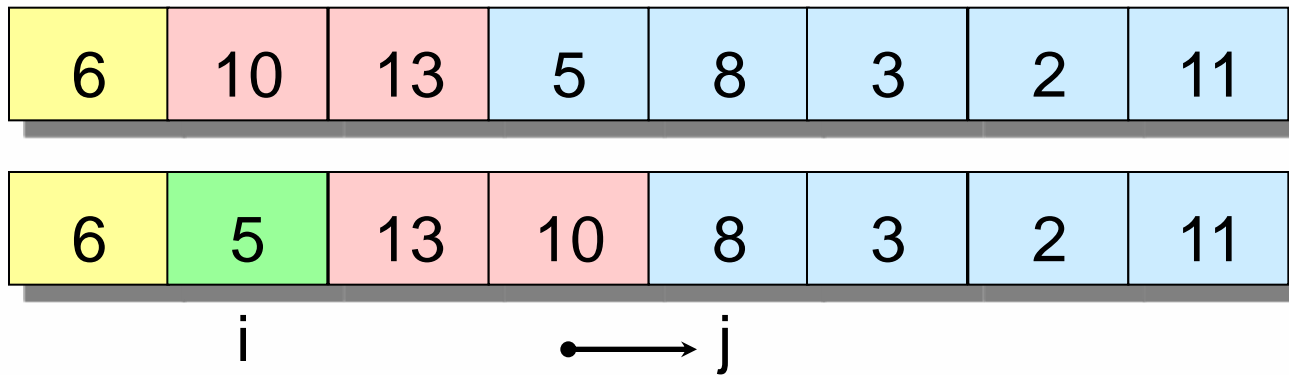
Example of partitioning



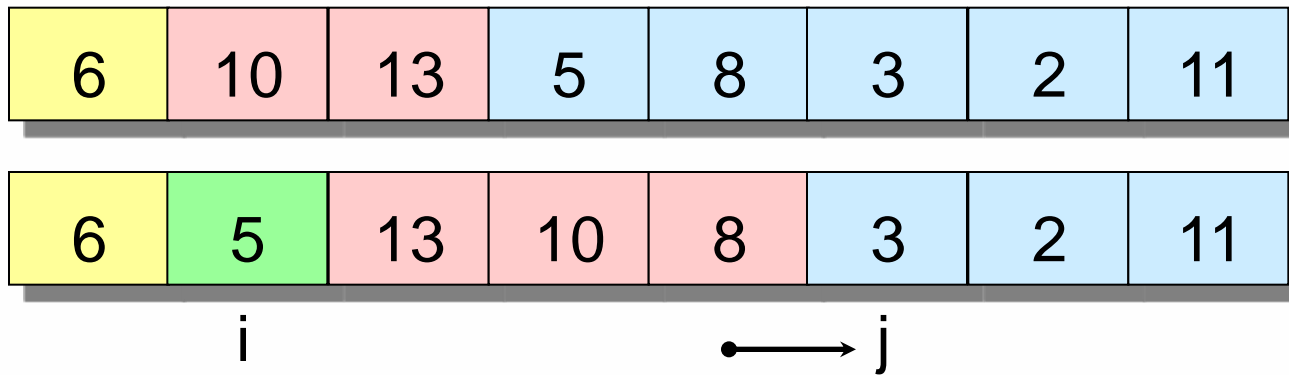
Example of partitioning



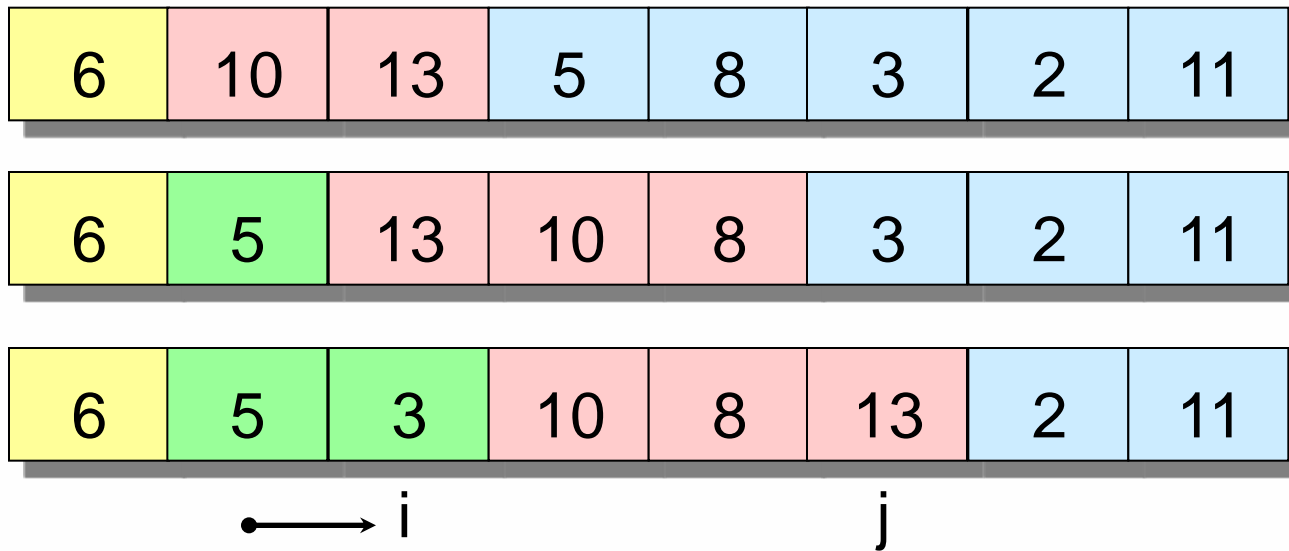
Example of partitioning



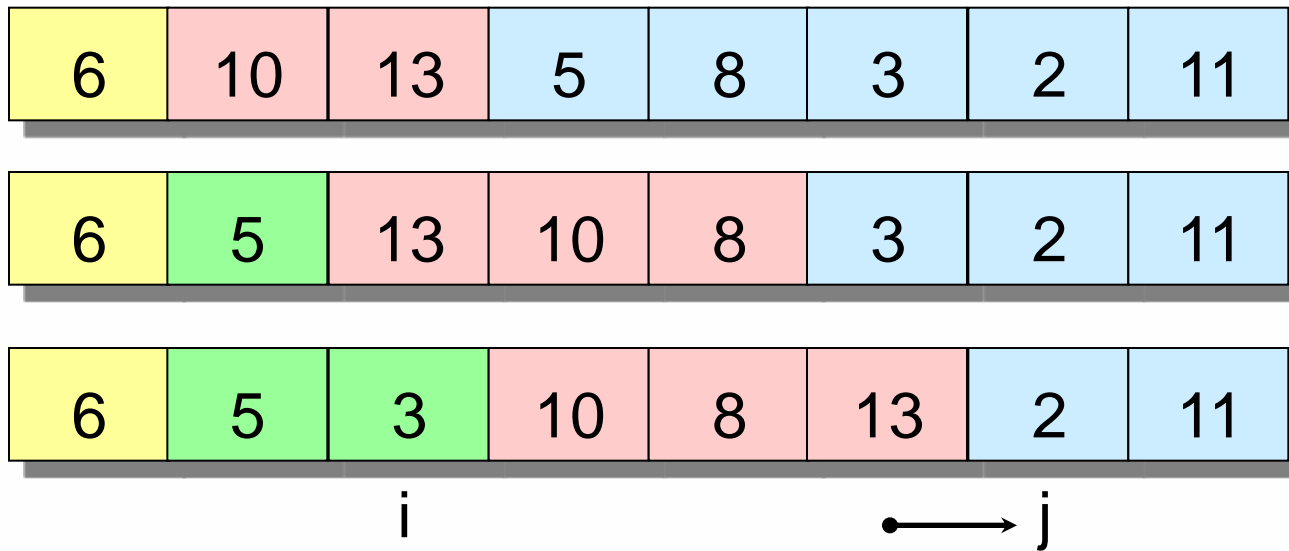
Example of partitioning



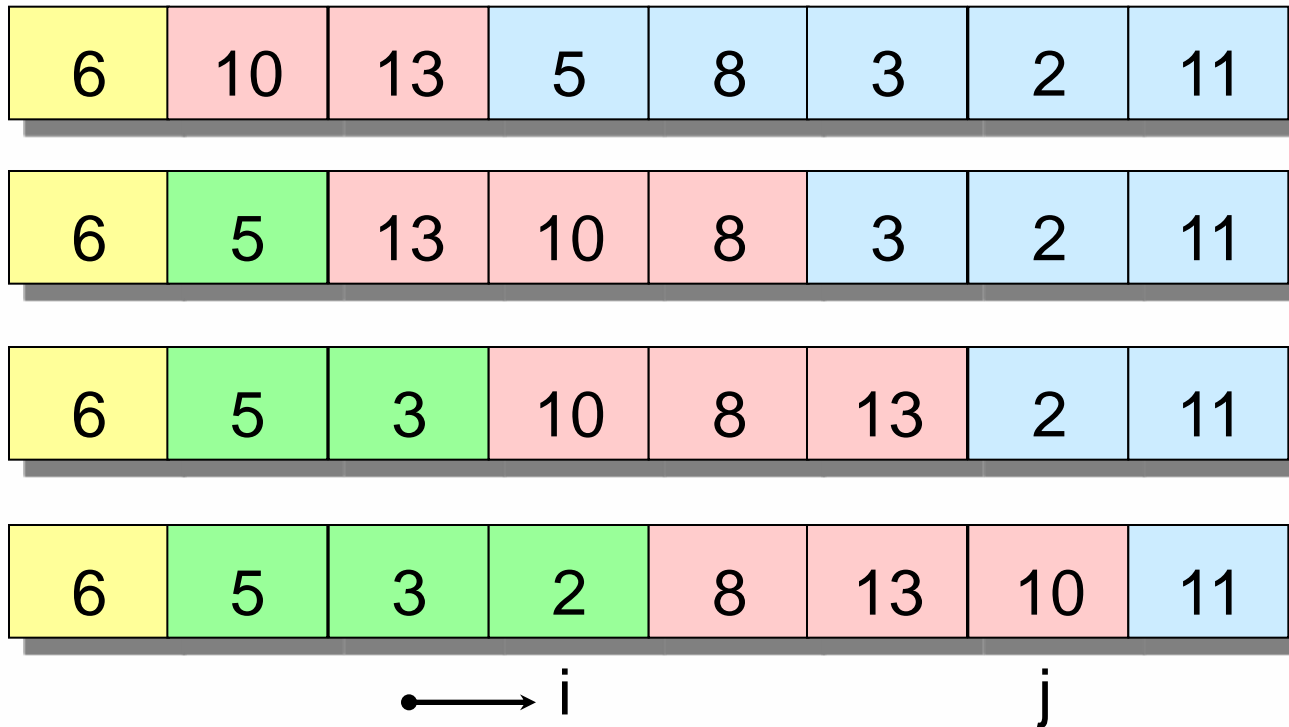
Example of partitioning



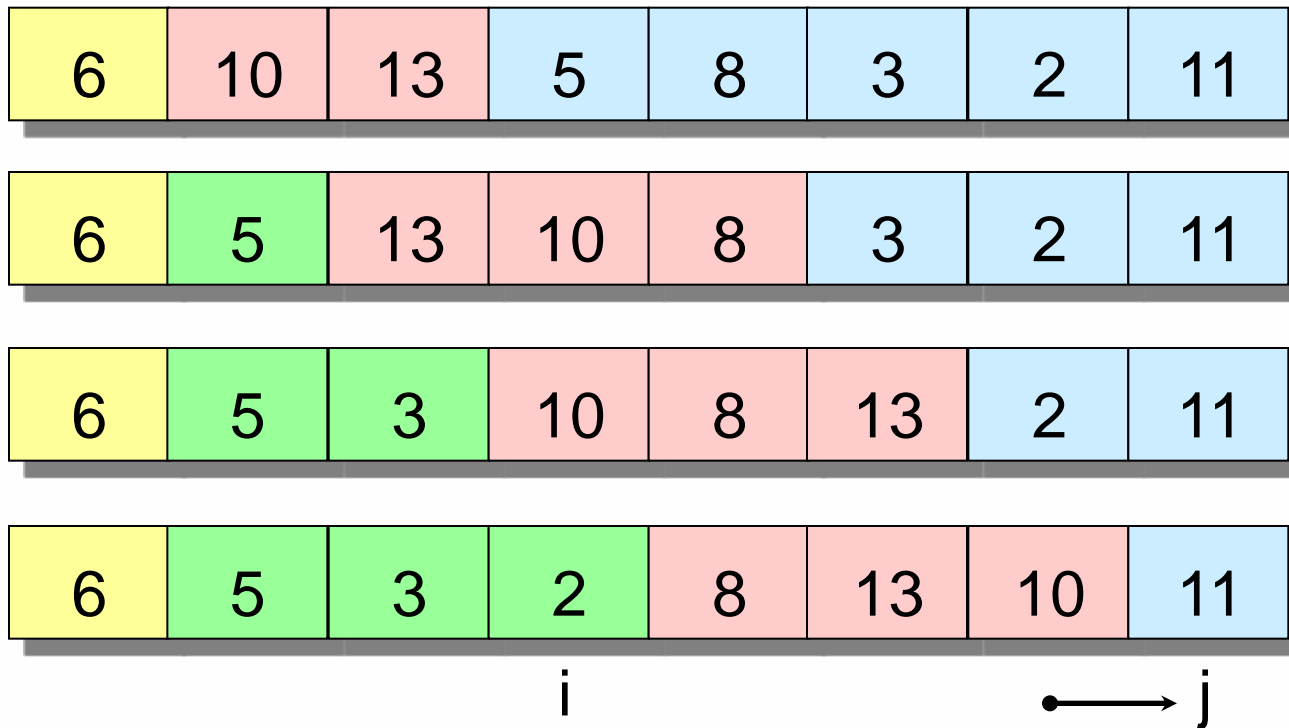
Example of partitioning



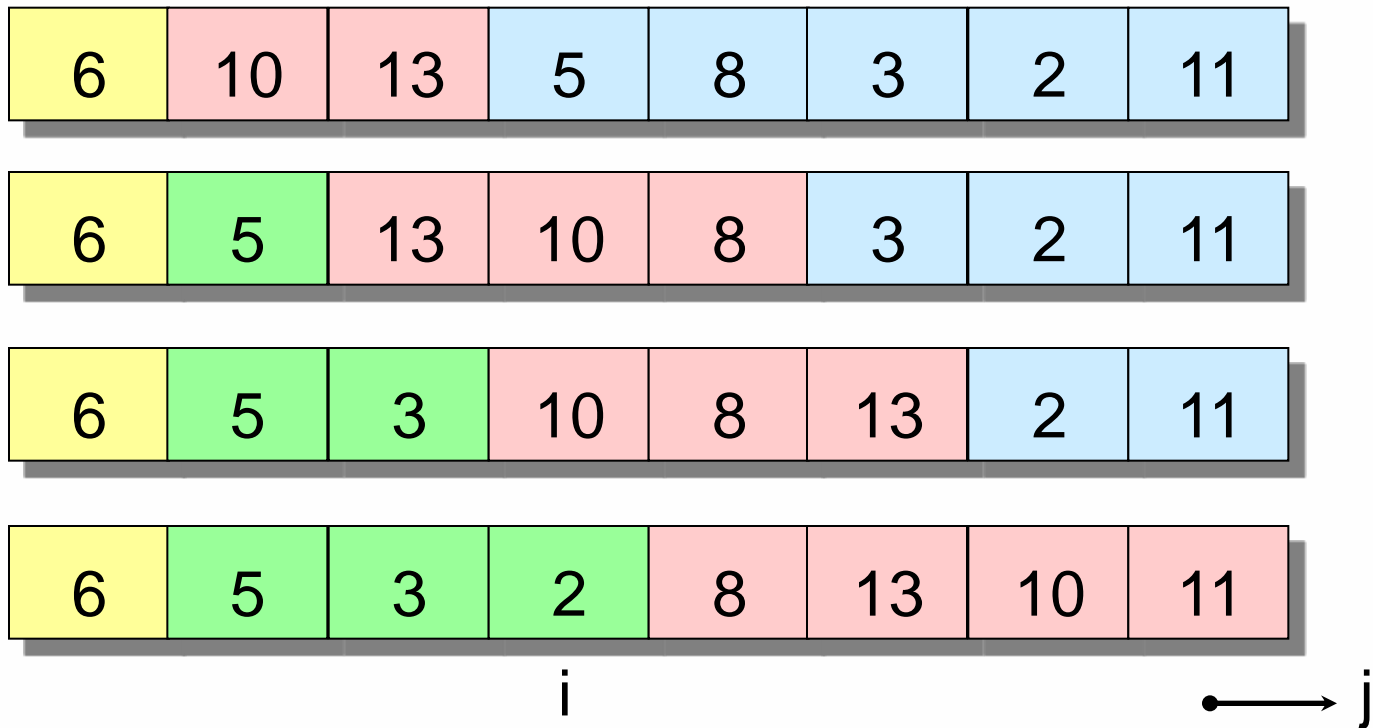
Example of partitioning



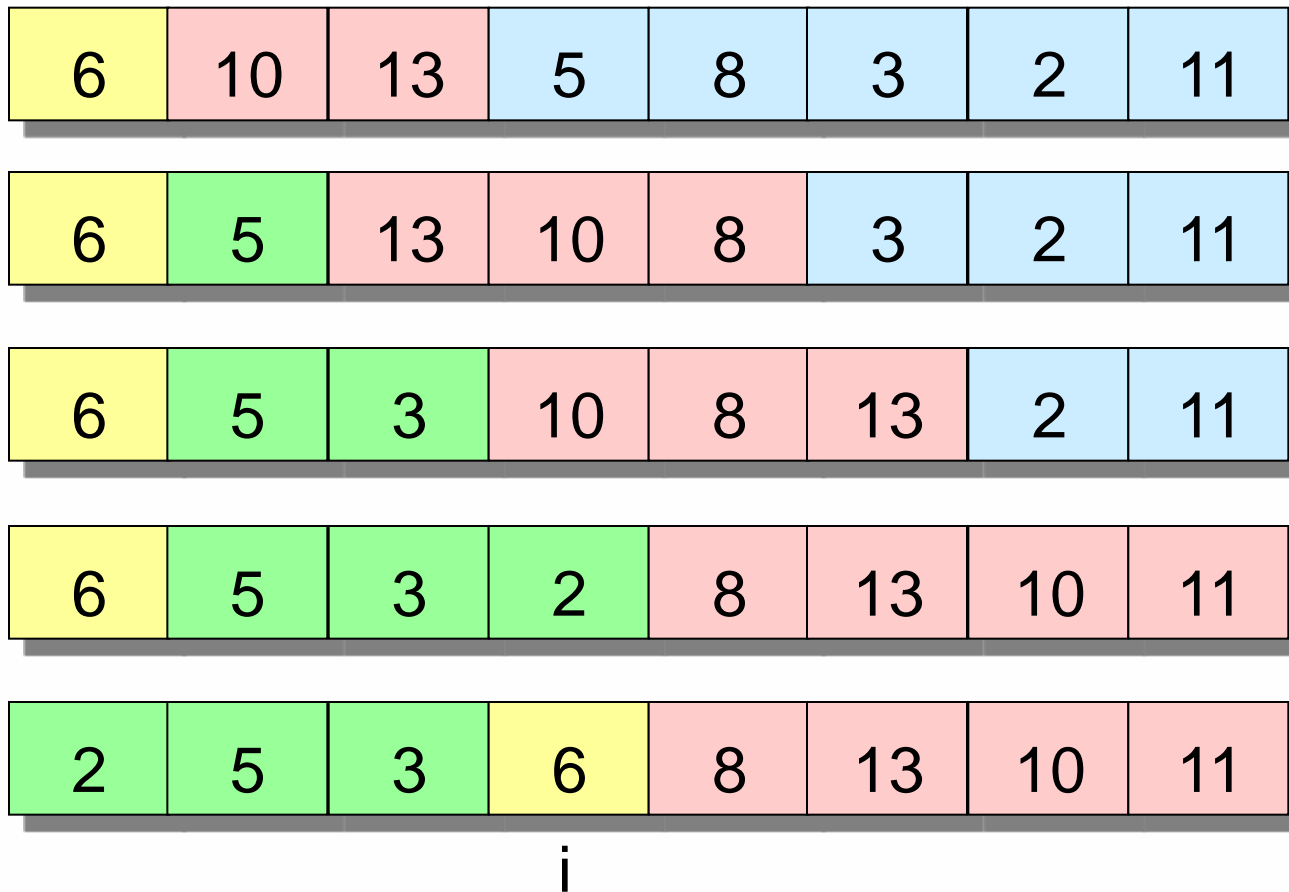
Example of partitioning



Example of partitioning



Example of partitioning



```
QUICKSORT(A, p, r)
  if p < r
    then q ← PARTITION(A, p, r)
        QUICKSORT(A, p, q-1)
        QUICKSORT(A, q+1, r)
```

- ❑ Assumiamo che tutti gli input siano distinti (esistono algoritmi di partitioning migliori se esistono duplicati)
- ❑ Calcoliamo $T(n)$ come il tempo di esecuzione per un vettore di n elementi nel caso peggiore

- ❑ Gli elementi sono ordinati (in ordine crescente o decrescente)
- ❑ La procedura di partizione viene effettuata sempre intorno al minimo o al massimo
- ❑ Una delle due partizioni è sempre vuota

$$\begin{aligned}T(n) &= T(0) + T(n - 1) + \Theta(n) \\ &= \Theta(1) + T(n - 1) + \Theta(n) \\ &= T(n - 1) + \Theta(n) \\ &= \Theta(n^2)\end{aligned}$$

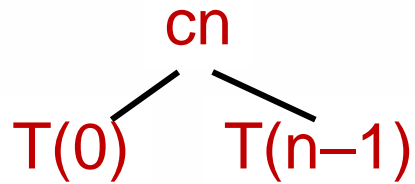
$$T(n) = T(0) + T(n-1) + cn$$

$$T(n) = T(0) + T(n-1) + cn$$

$T(n)$

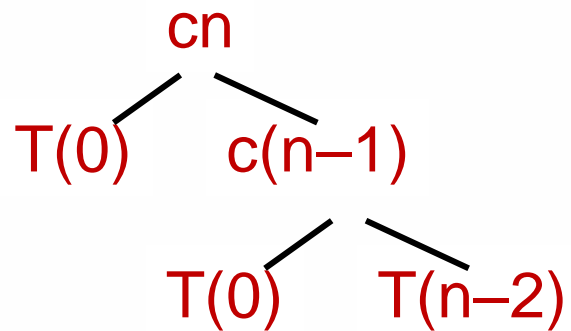
Albero di Ricorsione: worst-case

$$T(n) = T(0) + T(n-1) + cn$$



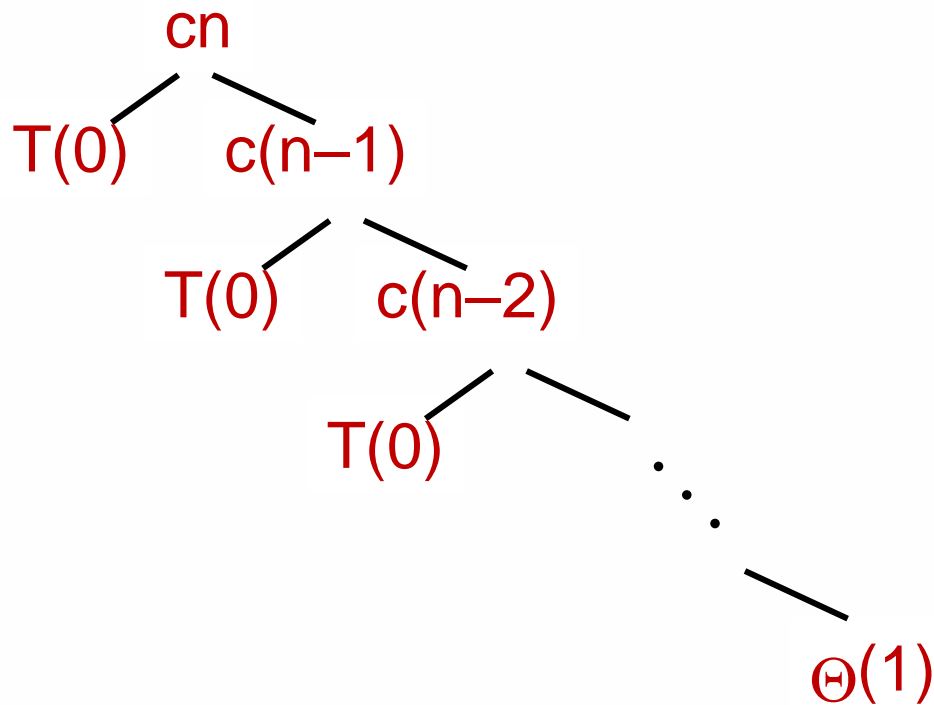
Albero di Ricorsione: worst-case

$$T(n) = T(0) + T(n-1) + cn$$



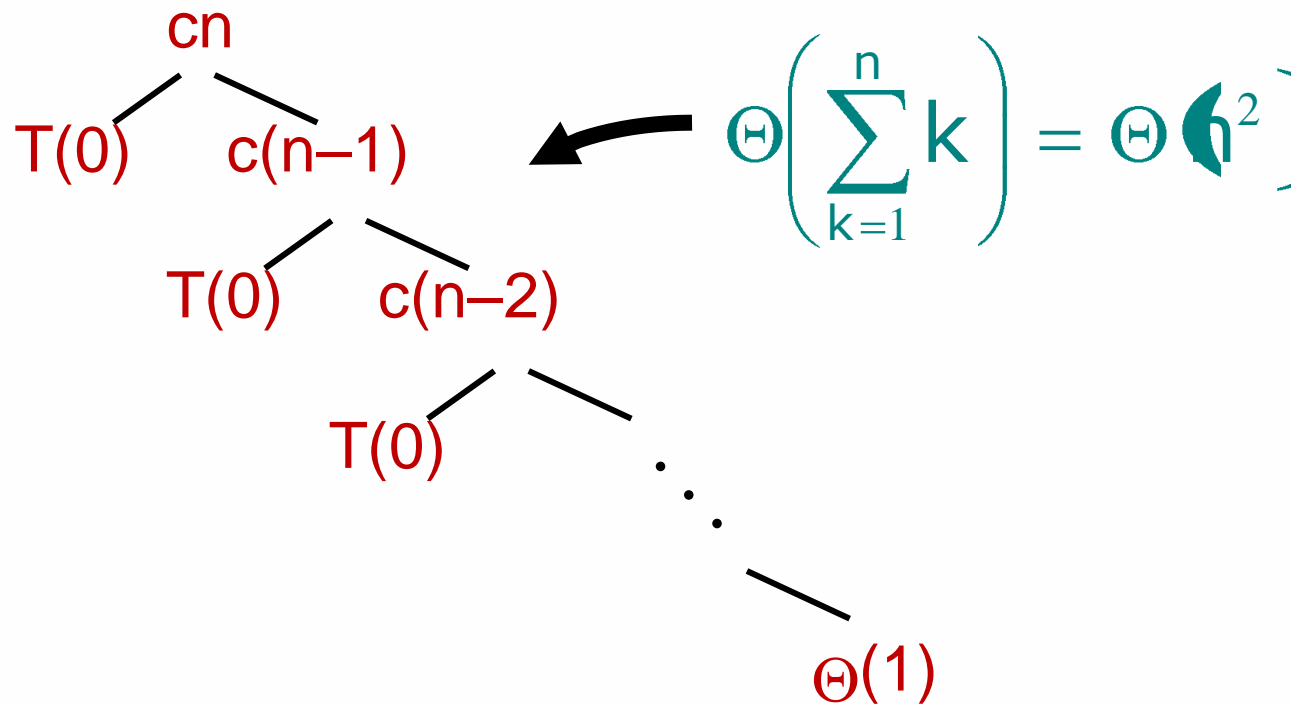
Albero di Ricorsione: worst-case

$$T(n) = T(0) + T(n-1) + cn$$



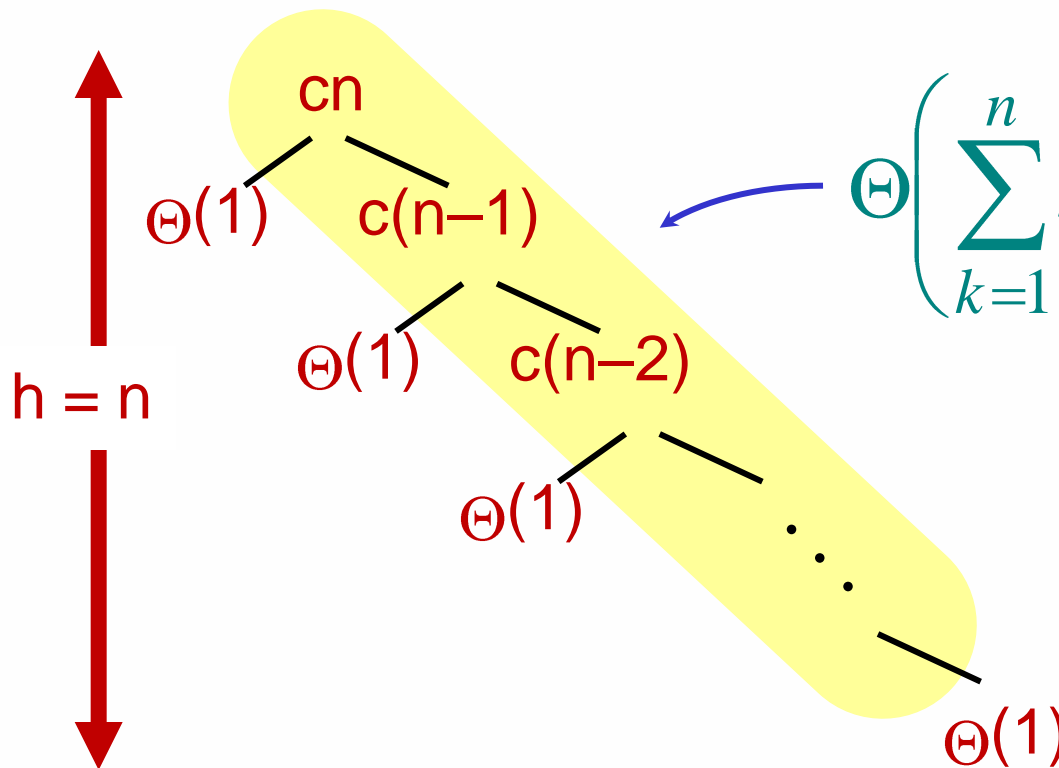
Albero di Ricorsione: worst-case

$$T(n) = T(0) + T(n-1) + cn$$



Albero di Ricorsione: worst-case

$$T(n) = T(0) + T(n-1) + cn$$



$$\Theta\left(\sum_{k=1}^n k\right) = \Theta(n^2)$$

$$T(n) = \Theta(n) + \Theta(n^2) \\ = \Theta(n^2)$$

- Intuitivamente: in media, la procedura di partitioning suddividerà il vettore a metà, avremo quindi,

$$\begin{aligned}T(n) &= 2T(n/2) + \Theta(n) \\ &= \Theta(n \lg n)\end{aligned}$$

come per il merge sort

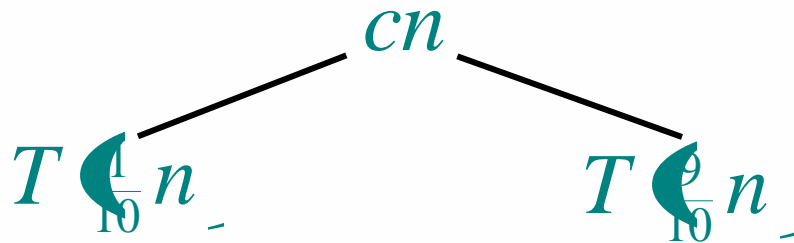
- Cosa succede se il partitioning genera una suddivisione 1/10-9/10?

$$T(n) = T(0.1n) + T(0.9n) + \Theta(n)$$

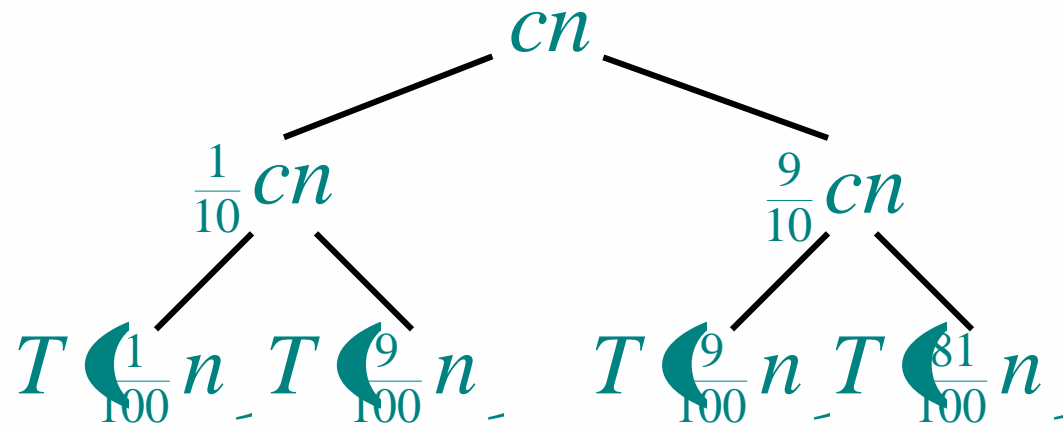
Come si risolve questa equazione alle ricorrenze?

$$T(n)$$

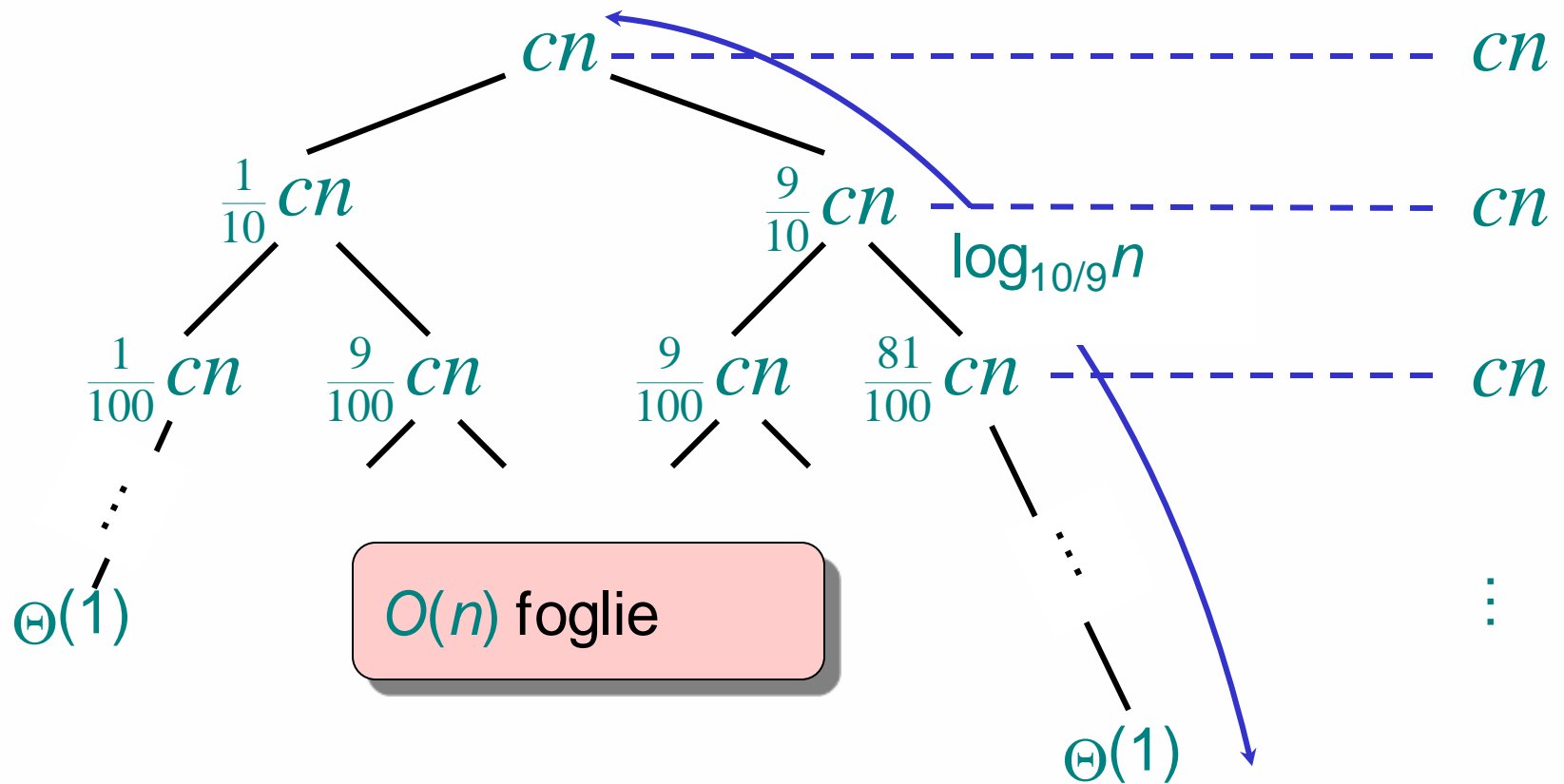
Analisi del caso "quasi-migliore"



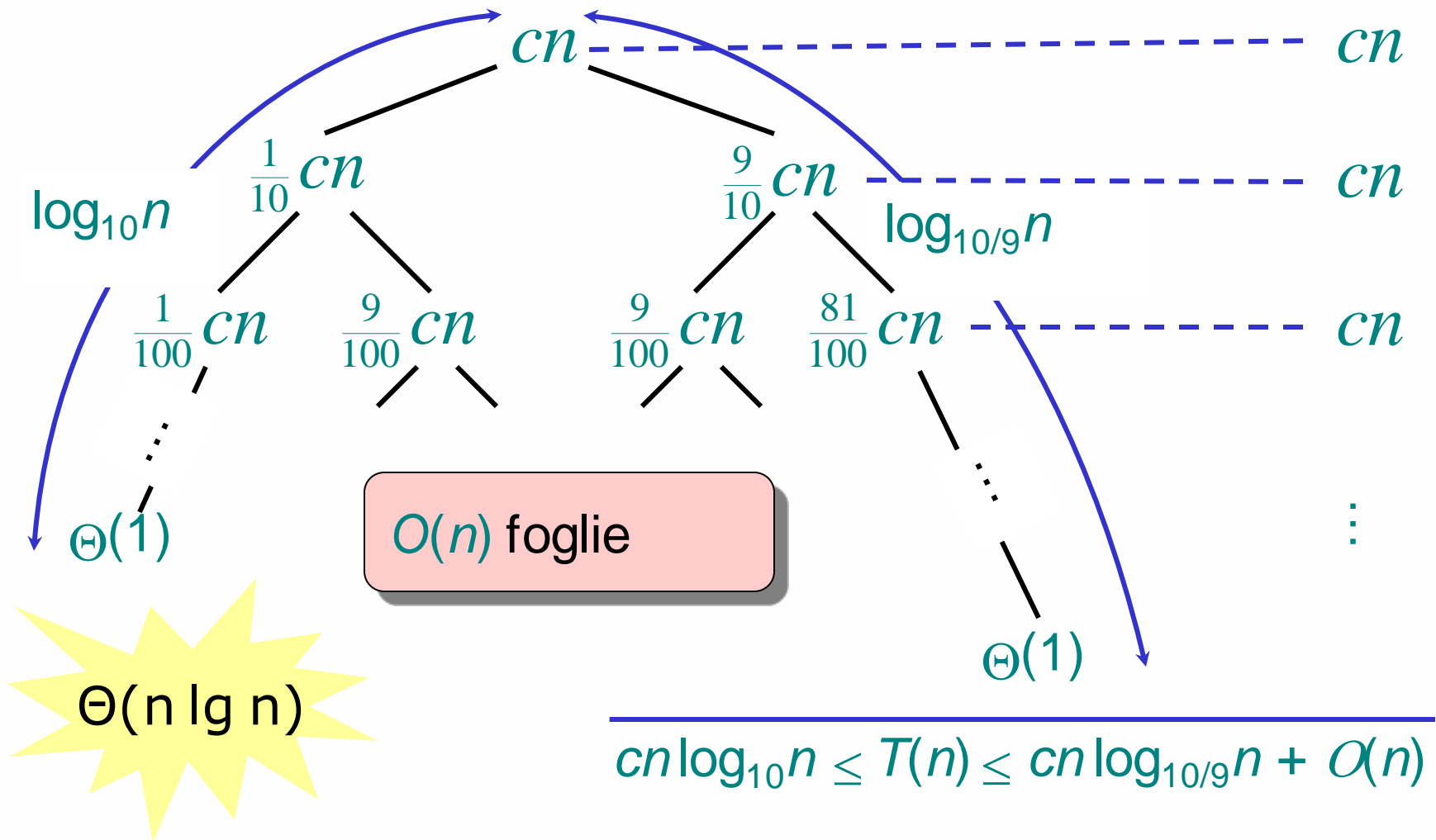
Analisi del caso "quasi-migliore"



Analisi del caso "quasi-migliore"



Analisi del caso "quasi-migliore"



Analisi di Complessità: Cosa Succede con Fortuna Alternata?

- Supponiamo di alternare un partitioning fortunato con uno sfortunato, cosa succede?

$$\begin{array}{ll} L(n) &= 2U(n/2) + \Theta(n) && \text{fortunato} \\ U(n) &= L(n-1) + \Theta(n) && \text{sfortunato} \end{array}$$

- Se lo risolviamo otteniamo,

$$\begin{aligned} L(n) &= 2(L(n/2 - 1) + \Theta(n/2)) + \Theta(n) \\ &= 2L(n/2 - 1) + \Theta(n) \\ &= \Theta(n \lg n) \end{aligned}$$



$\Theta(n \log n)$

Possiamo fare in modo di essere solitamente fortunati?

Quicksort Randomizzato

- ❑ Partizionare scegliendo un elemento a caso
- ❑ Il tempo di esecuzione è indipendente dall'ordine degli input
- ❑ Nessuna ipotesi sulla distribuzione degli input
- ❑ Nessuno specifico input genera il caso peggiore che invece dipende dal generatore di numeri casuali

Sommario

- ❑ Quicksort è un algoritmo estremamente generale
- ❑ È tipicamente due volte più veloce del merge sort
- ❑ Può beneficiare sostanzialmente dall'ottimizzazione del codice
- ❑ Buone performance anche in presenza di caching e memoria virtuale