Progettazione di un metodo ricorsivo.

Ogni metodo ricorsivo deve distinguere tre:

- CASO BASE (la risposta è immediata e non richiede una chiamata ricorsiva)
- CASO RICORSIVO (per fornire la risposta effettua una chiamata ricorsiva con parametri "più semplici")

Per capire se un metodo ricorsivo funziona, bisogna controllare tre cose:

1) Controllare che la ricorsione termini perché prima o poi si arriva di sicuro al caso base.

2) Quando siamo nei casi base, il risultato è corretto.

3) Quando siamo nel caso ricorsivo, dobbiamo controllare che se le chiamate ricorsive sono corrette, anche il valore finale restituito dal metodo è corretto.

Esempio

Voglio verificare che il metodo fattorialeRec è corretto

```
11      /* Versione ricorsiva del fattoriale. */
12      public static int fattorialeRec(int n) {
13          if (n <= 1)
14              // caso base
15              return 1;
16          else
17              // caso ricorsivo
18              return n * fattorialeRec(n - 1);
19      }
```

1) fattorialeRec(n) termina sicuramente?

Sì perché:

se $n \leq 1$, termina subito senza chiamate ricorsive

se $n > 1$, fa chiamate ricorsive ma, poiché il parametro n viene decrementato a ogni chiamata, prima o poi raggiunge il caso base.

2) Il CASO base è corretto?

fattorialeRec(1) = 1!

Sì, perché entrambi sono uguali a 1.

3) Il CASO RICORSIVO è corretto?

Quando $n > 1$, il metodo fattoriale calcola:

n * fattorialeRec(n-1)

Possiamo supporre che le chiamate ricorsive sia corrette, ovvero che fattorialeRec(n-1) = (n-1)! In tal caso noi restituiamo $n * (n-1)! = n!$ quindi il valore di ritorno è corretto.

Esempio. Potenza

$75^2 = 74^2$

$x^n = (x^{n-1})$

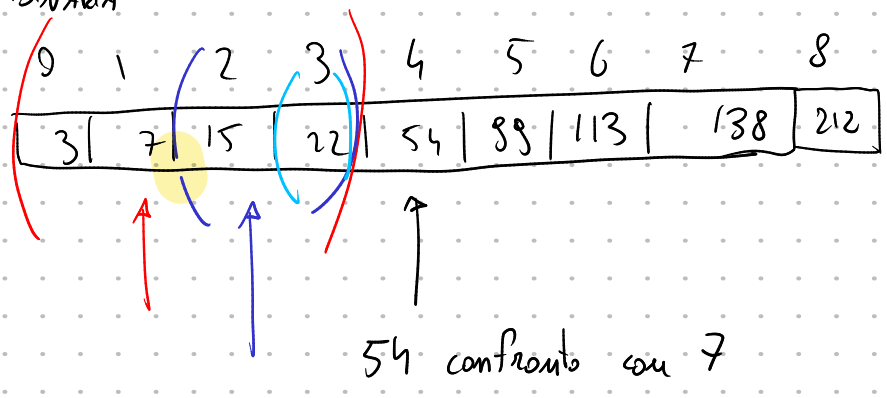
$x^n = x^{n-1} \cdot x$

$x^n = (x-1)^n$?

Ricerca Binaria

Merge Sort

Ricerca Binaria



X = 7 VALORE DA CERCARE

X = 33

Algoritmo di ricerca binaria:

Problema: trovare un elemento x nel vettore a.

Soluzione:

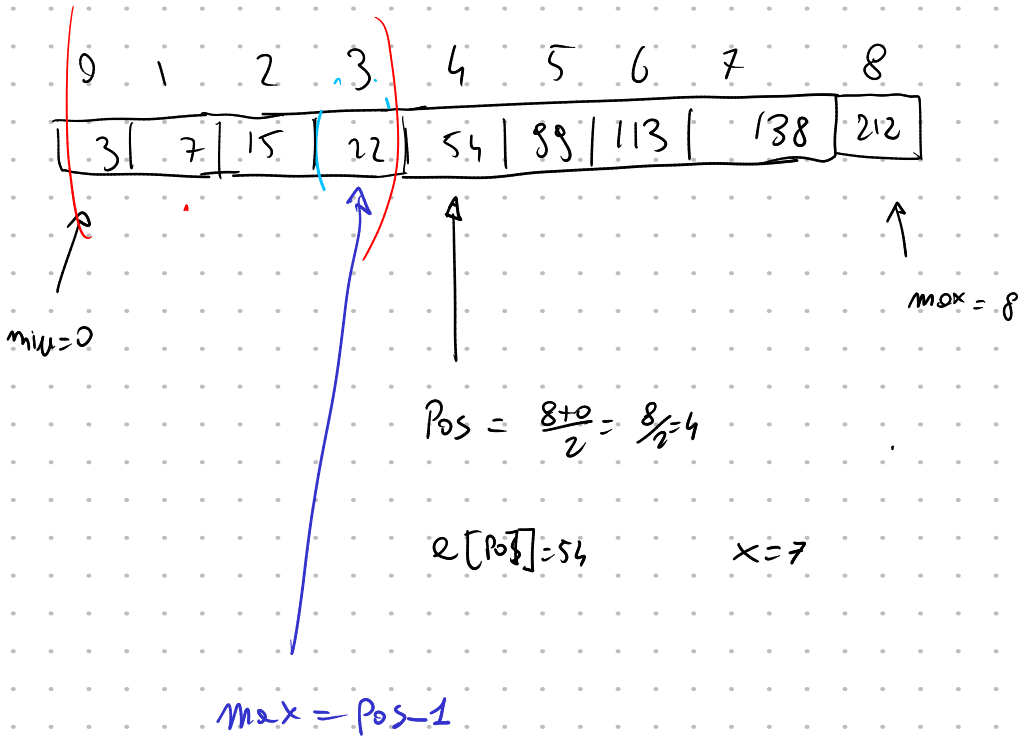
Prendo l'elemento centrale di a. (lo chiamo centro)

Se x = centro, ho trovato x!

Se x < centro, riparto da capo ma guardando solo gli elementi prima di quello centrale

Se x > centro, riparto da capo, ma guardo solo gli elementi di a dopo quello centrale.

Mi fermo se non ho nulla tra cui cercare.

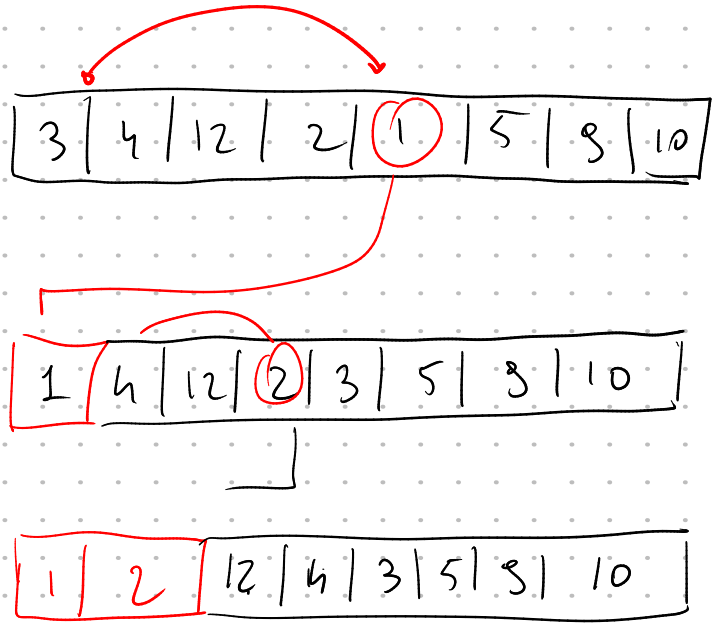


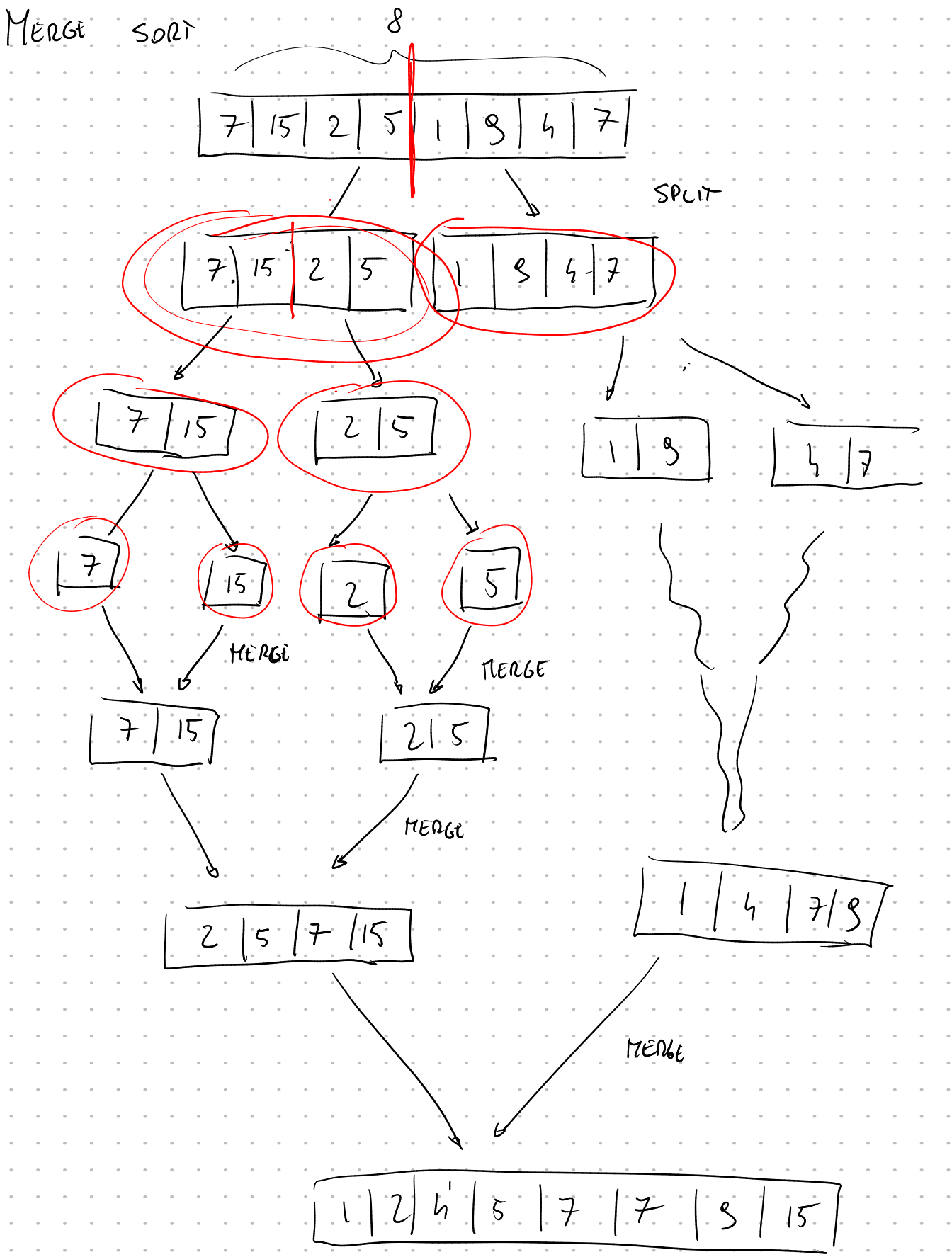
MERGE SORT

INPUT: array a

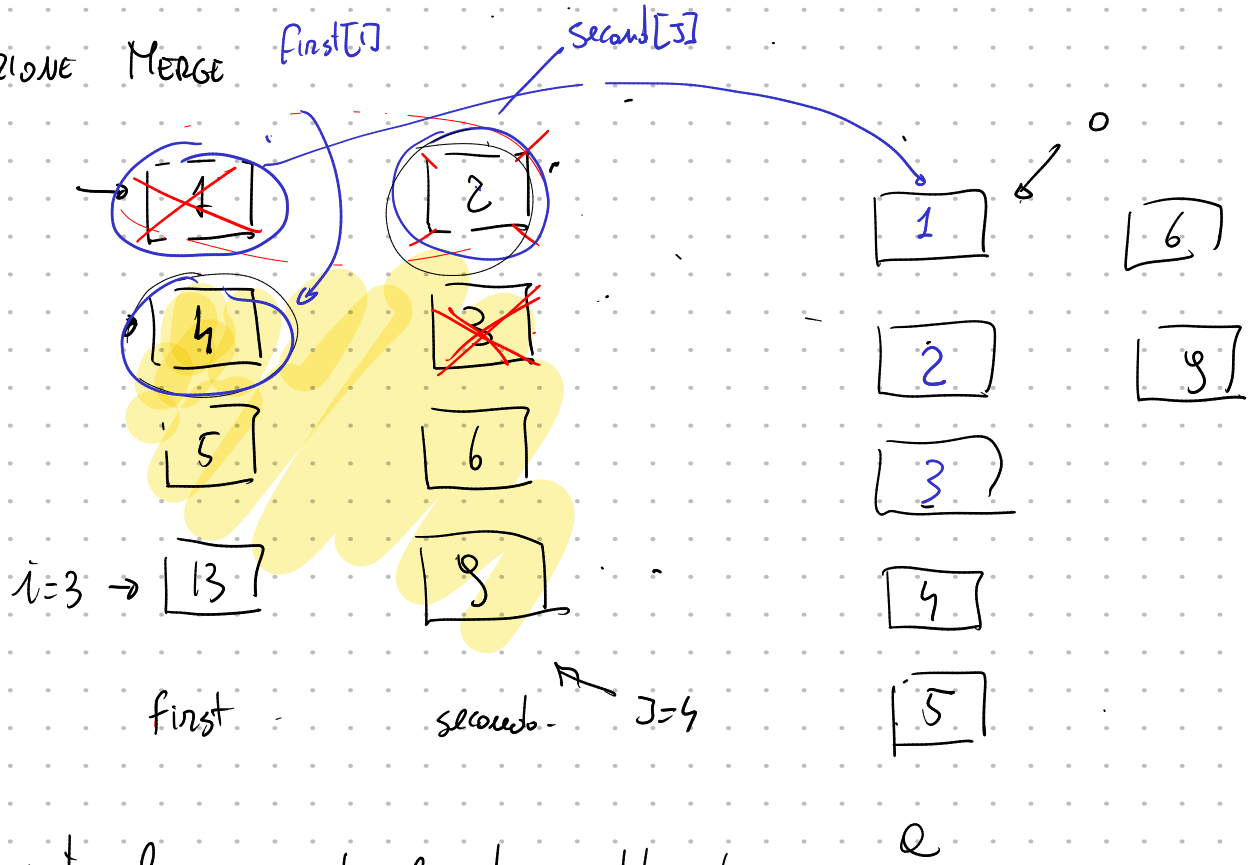
Output: array a ordinato dal più piccolo al più grande

Abbiamo già visto il SELECTION SORT.




MERGE SORT
8
7 15 2 5 1 9 4 7
SPLIT
7 15 2 5
1 9 4 7
7 15
2 5
1 9
4 7
7
15
2
5
MERGE
MERGE
7 15
2 5
MERGE
1 4 7 9
2 5 7 15
MERGE
1 2 4 5 7 7 9 15

Operazione Merge



i conta il numero di elementi spostati da first ad e

j conta il numero di elementi spostati da second ad e

quindi i+j è il numero totale di elementi spostati, in e

ovvero la posizione dove inserire il prossimo elemento in e