POLITECNICO
DI TORINO

Dipartimento
di Automatica e Informatica

Gli Algoritmi di Ordinamento iterativi linearitmici

Paolo Camurati

Gli ordinamenti linearitmici

- Gli algoritmi di ordinamento basati sul confronto che sono $\Omega(n \lg n)$ sono **OTTIMI**
- In generale sono ricorsivi (argomento del Corso del II anno): Merge sort, Quick sort, Heap sort
- Esiste una versione iterativa del Merge sort: Bottom-up Merge sort

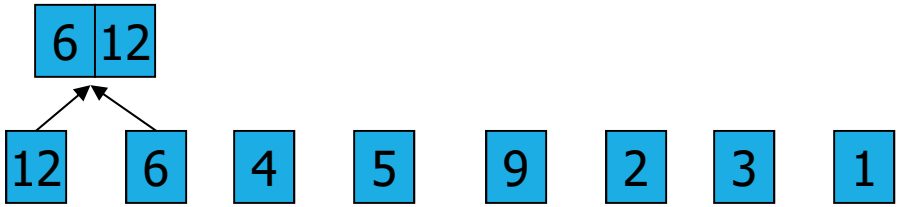
Bottom-up Merge sort

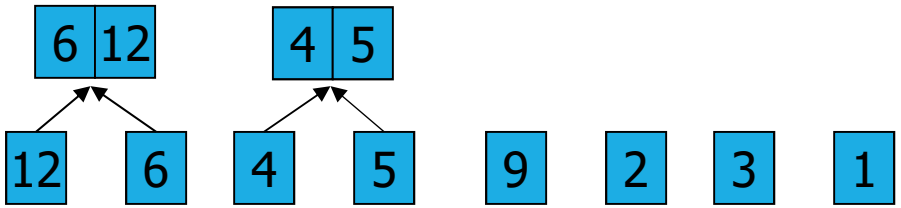
- Un vettore di un solo elemento è per definizione ordinato
- Iterazione:
  - fondere due sottovettori ordinati in un vettore ordinato la cui dimensione è la somma delle dimensioni dei 2 sottovettori di partenza
  - fino a raggiungere la dimensione N del vettore da ordinare.

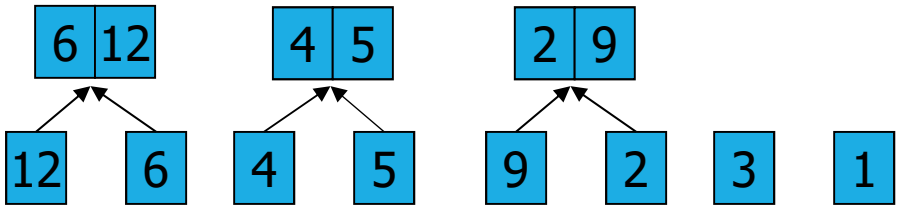
Esempio

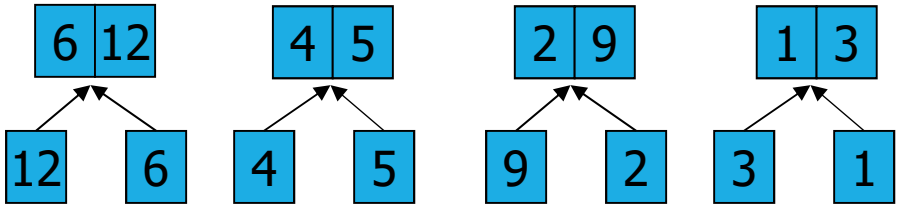
- Ipotesi: la dimensione del vettore da ordinare è una potenza di 2
  $N = 2^k$
  - partendo da sottovettori di lunghezza 1 (quindi ordinati), si applica Merge per ottenere a ogni passo vettori ordinati di lunghezza m doppia
  - serve un vettore di appoggio di dimensione N per contenere il risultato di Merge
  - terminazione: il vettore ha dimensione pari a quello di partenza.

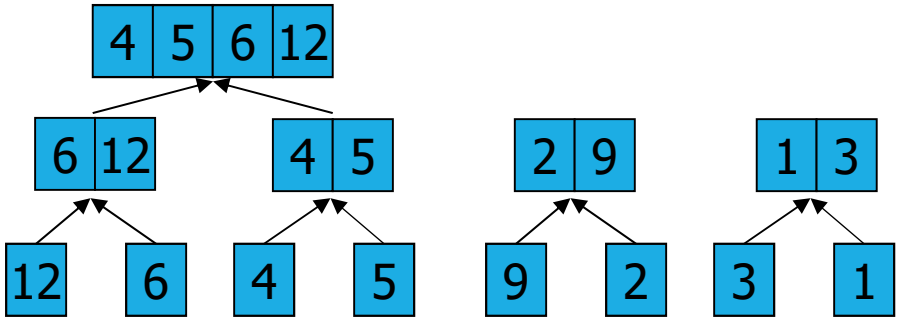

12 6 4 5 9 2 3 1

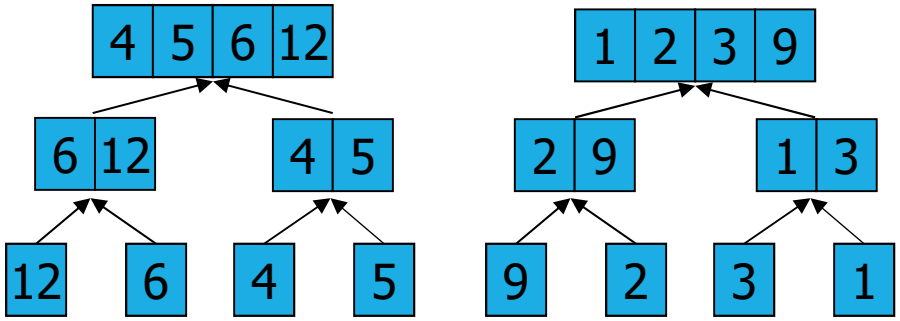

6 12
12 6 4 5 9 2 3 1

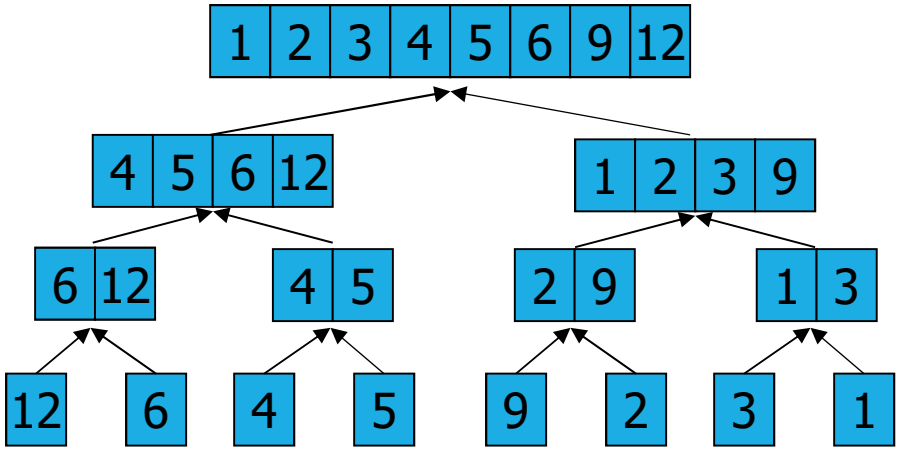

6 12
4 5
12
6
4
5
9
2
3
1


6 12
4 5
2 9
12
6
4
5
9
2
3
1


6 12
4 5
2 9
1 3
12 6
4 5
9 2
3 1

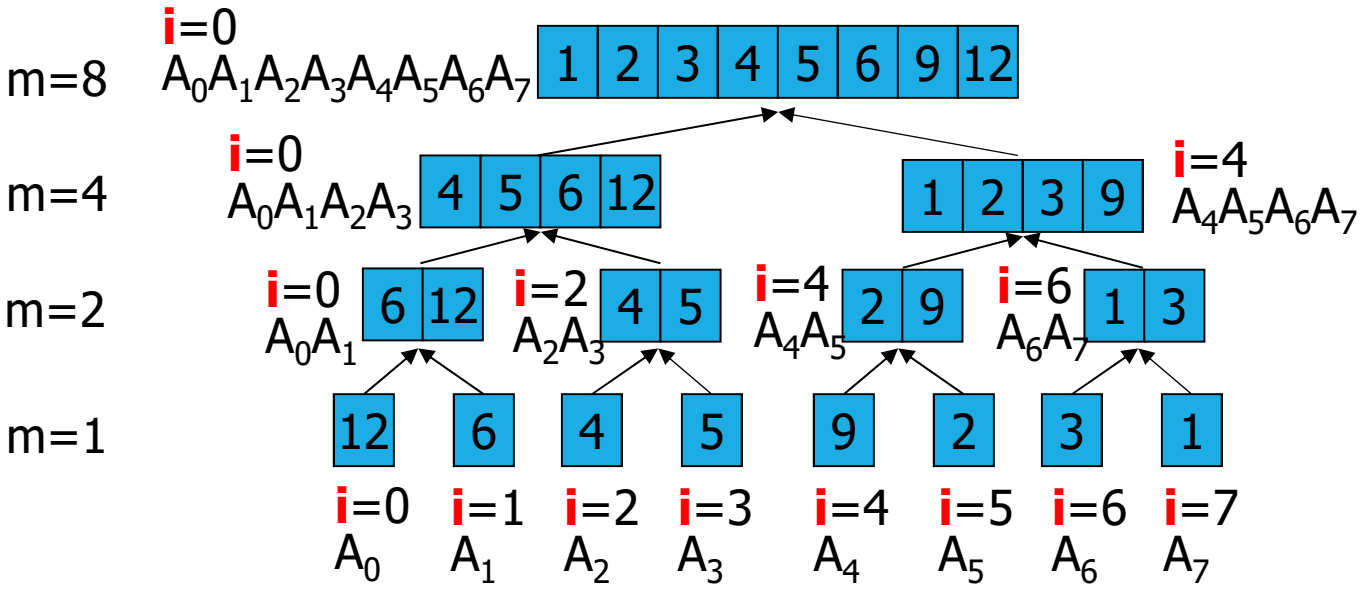

4 5 6 12
6 12
4 5
2 9
1 3
12
6
4
5
9
2
3
1


4 5 6 12
6 12
4 5
12
6
4
5
1 2 3 9
2 9
1 3
9
2
3
1

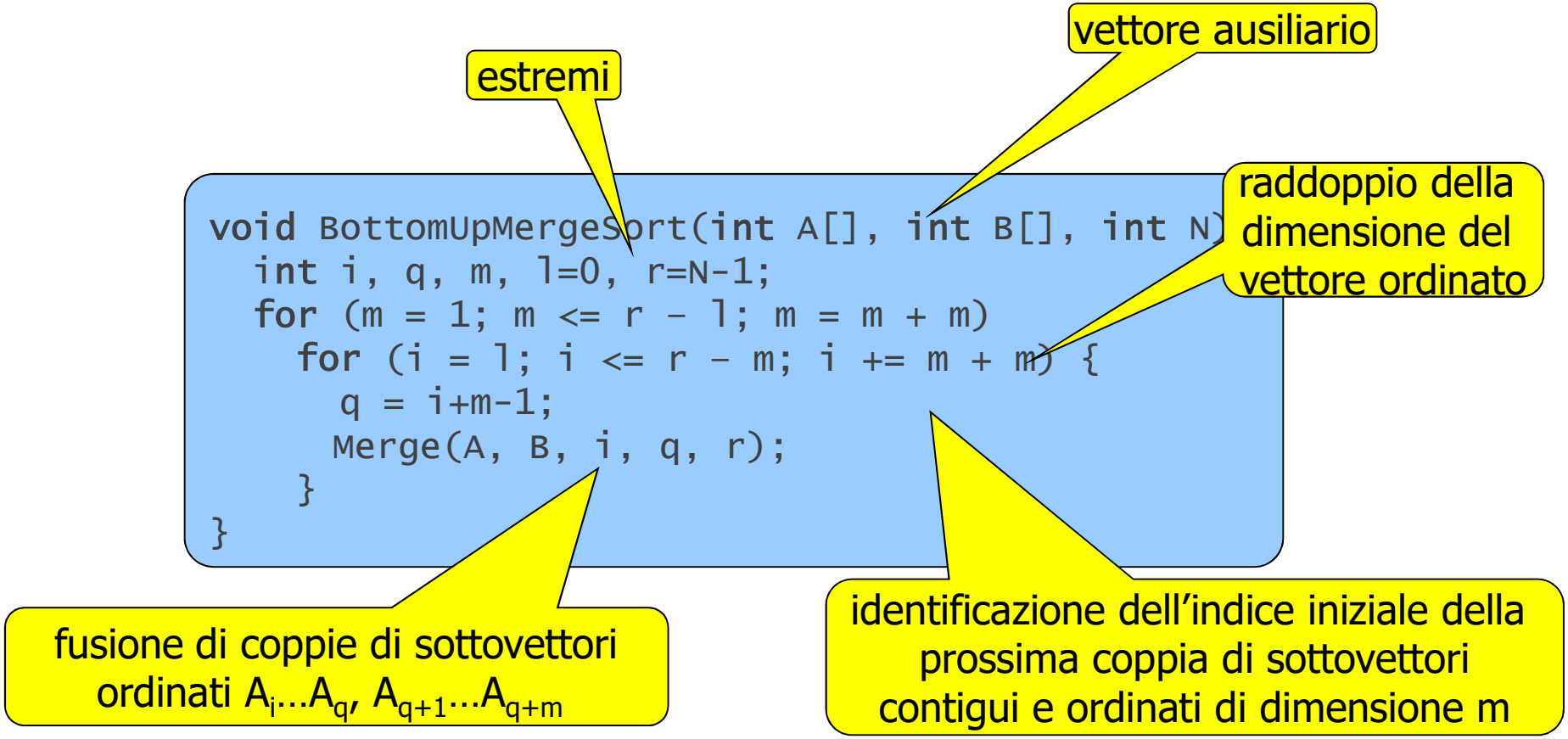

1 2 3 4 5 6 9 12
4 5 6 12
1 2 3 9
6 12
4 5
2 9
1 3
12
6
4
5
9
2
3
1

- ciclo esterno: m inizialmente vale 1, raddoppia ad ogni passo sino a diventare pari a N
- ciclo interno: ad ogni coppia di sottovettori ordinati e adiacenti di dimensione m si applica Merge per ottenere un sottovettore ordinato di lunghezza 2m.

- Identificazione dei sottovettori ordinati e adiacenti:



nel ciclo interno:
$q = i + m - 1$
sottovettore sinistro
$A_i \ldots A_q$
sottovettore destro
$A_{q+1} \ldots A_{q+m}$


estremi
vettore ausiliario
raddoppio della dimensione del vettore ordinato
void BottomUpMergeSort(int A[], int B[], int N)
int i, q, m, l=0, r=N-1;
for (m = 1; m <= r - l; m = m + m)
for (i = l; i <= r - m; i += m + m) {
q = i+m-1;
Merge(A, B, i, q, r);
}
}
fusione di coppie di sottovettori ordinati $A_i...A_q$, $A_{q+1}...A_{q+m}$
identificazione dell'indice iniziale della prossima coppia di sottovettori contigui e ordinati di dimensione m

2-way Merge

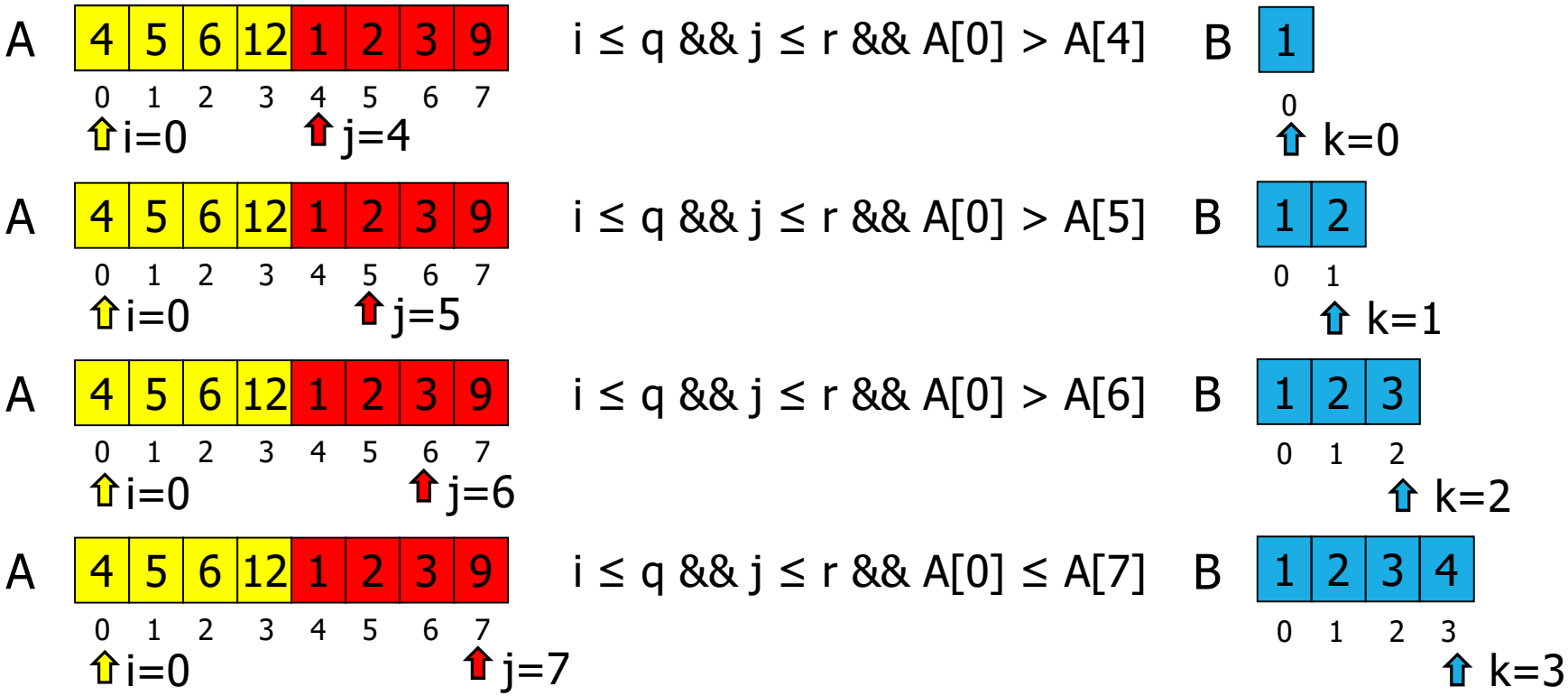
- ipotesi: la dimensione del vettore A è una potenza di 2 $N = 2^k$
- fusione di 2 (2-way) sottovettori di A ordinati di dimensione m per ottenere un vettore ordinato di dimensione 2m
- generalizzabile a k vettori (k-way Merge)
- indice q per dividere sottovettori di A a metà in 2 sottovettori sinistro e destro $q = i + m - 1$
- sottovettore sinistro con indice $l \leq i \leq q$
- sottovettore destro con indice $q+1 \leq j \leq r$
- vettore ausiliario B di dimensione N con indice $l \leq k \leq r$ per memorizzare i risultati delle fusioni passato come parametro

Approccio:

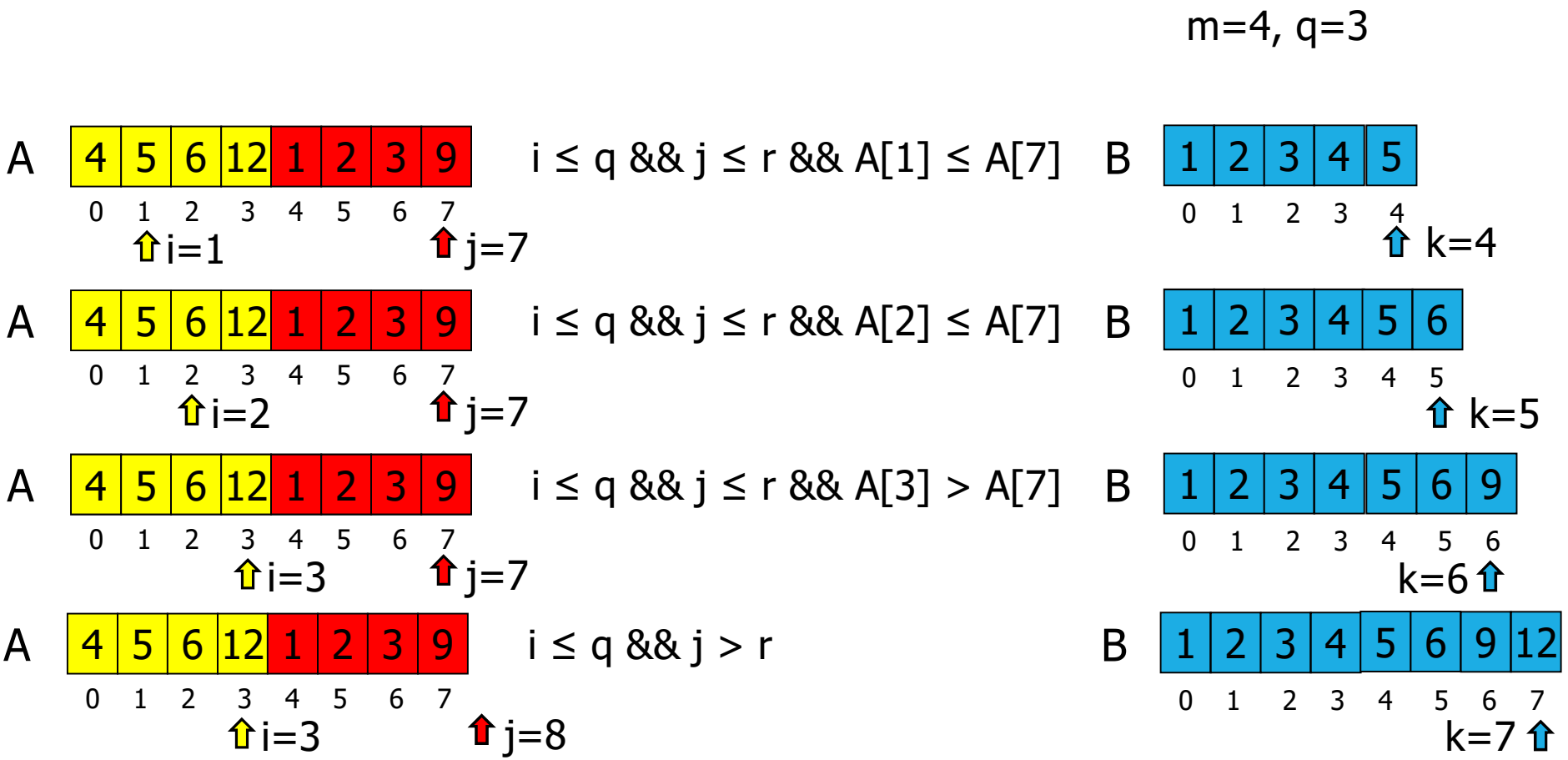
- scorrere i sottovettori sinistro e destro mediante gli indici i e j e il vettore B mediante l'indice k
- se è esaurito il sottovettore sinistro, ricopiare gli elementi rimanenti del sottovettore destro in B
- altrimenti se è finito il sottovettore destro, ricopiare gli elementi rimanenti del sottovettore sinistro in B
- altrimenti confrontare l'elemento corrente A[i] del sottovettore sinistro con quello del sottovettore destro A[j]
  - se A[i] ≤ A[j], ricopiare A[i] in B e avanzare i, j resta invariato
  - altrimenti ricopiare A[j] in B e avanzare j, i resta invariato.

Esempio

m=4, q=3



m=4, q=3


A
4 5 6 12 1 2 3 9
0 1 2 3 4 5 6 7
i=1
j=7
i ≤ q && j ≤ r && A[1] ≤ A[7]
B
1 2 3 4 5
0 1 2 3 4
k=4
A
4 5 6 12 1 2 3 9
0 1 2 3 4 5 6 7
i=2
j=7
i ≤ q && j ≤ r && A[2] ≤ A[7]
B
1 2 3 4 5 6
0 1 2 3 4 5
k=5
A
4 5 6 12 1 2 3 9
0 1 2 3 4 5 6 7
i=3
j=7
i ≤ q && j ≤ r && A[3] > A[7]
B
1 2 3 4 5 6 9
0 1 2 3 4 5 6
k=6
A
4 5 6 12 1 2 3 9
0 1 2 3 4 5 6 7
i=3
j=8
i ≤ q && j > r
B
1 2 3 4 5 6 9 12
0 1 2 3 4 5 6 7
k=7

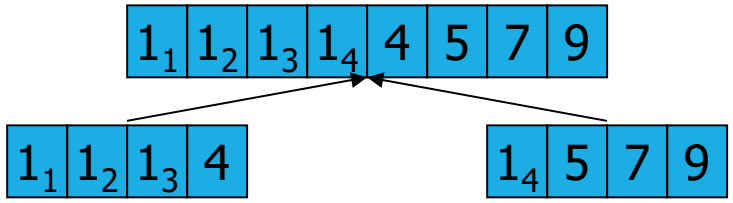
```
void Merge(int A[], int B[], int l, int q, int r) {
  int i, j, k;
  i = l;
  j = q+1;
  for (k = l; k <= r; k++)
    if (i > q)
      B[k] = A[j++];
    else if (j > r)
      B[k] = A[i++];
    else if ((A[i]< A[j]) || (A[i]== A[j]))
      B[k] = A[i++];
    else
      B[k] = A[j++];
  for ( k = l; k <= r; k++ )
    A[k] = B[k];
  return;
}
```

esaurito sottovettore sinistro

esaurito sottovettore destro

Caratteristiche del Merge sort

- non in loco, in quanto usa il vettore ausiliario B di dimensione N
- stabile: in quanto la funzione Merge prende dal sottovettore sinistro in caso di chiavi uguali:



Si noti che l'assenza della condizione A[i] == A[j] nell'istruzione

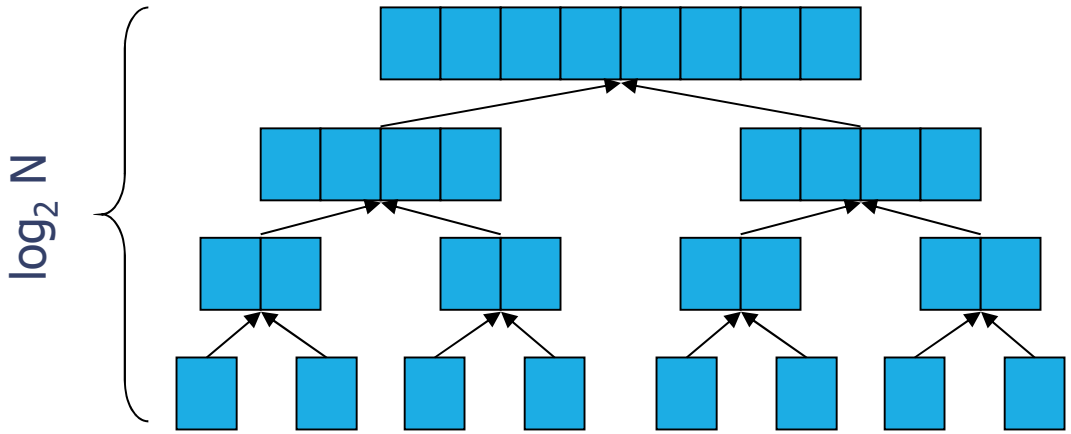
```
if ((A[i]< A[j]) || (A[i]== A[j]))
```

avrebbe reso non stabile l'algoritmo!

Analisi di complessità del Merge sort

Analisi informale, nell'ipotesi $N = 2^k$

- ad ogni livello si eseguono complessivamente N operazioni nelle diverse Merge
- inizialmente i sottovettori ordinati hanno dimensione 1
- ad ogni livello la dimensione dei sottovettori ordinati raddoppia, quindi all'i-esimo passo la dimensione è 2^i
- si termina quando $2^i = N$, quindi servono $i = \log_2 N$ livelli
- poiché ogni livello costa N, il costo complessivo è $N\log_2 N$
- la complessità è linearitmica T(n) = O(NlogN).



Livelli: $\log_2 N$

Operazioni per livello: N

Operazioni totali: $N \log_2 N$

Generalizzazione a N qualsiasi ($\neq 2^k$)

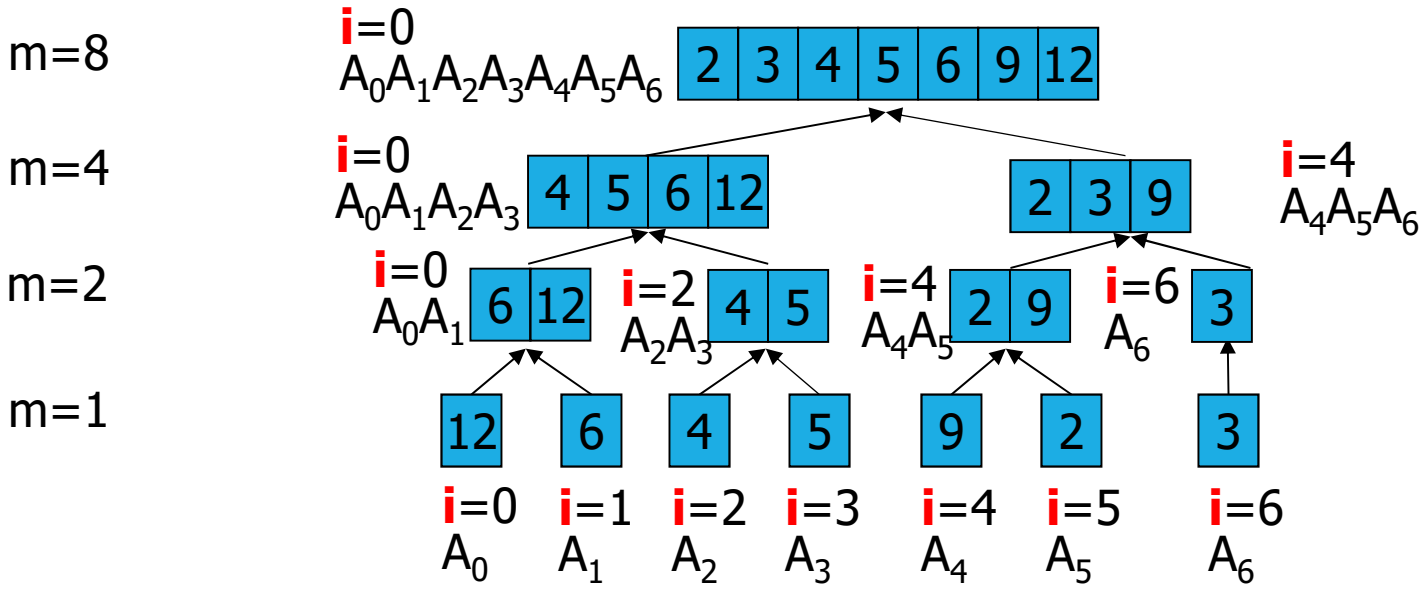
L'estremo destro del vettore nel Merge è il più piccolo valore tra r e il valore che si avrebbe se la lunghezza fosse una potenza di 2
i + m + m -1

```
void BottomUpMergeSort(int A[], int B[], int N)
  {
  int i, q, m, l=0, r=N-1;
for (m = 1; m <= r - l; m = m + m)
    for (i = l; i <= r - m; i += m + m) {
      q = i+m-1;
      Merge(A, B, i, q, min(i+m+m-1,r));
    }
}
```

Esempio

N=7


12 6 4 5 9 2 3


m=8
m=4
m=2
m=1
i=0
A0A1A2A3A4A5A6
2 3 4 5 6 9 12
i=0
A0A1A2A3
4 5 6 12
2 3 9
i=4
A4A5A6
i=0
A0A1
6 12
i=2
A2A3
4 5
i=4
A4A5
2 9
i=6
A6
3
12
6
4
5
9
2
3
i=0
A0
i=1
A1
i=2
A2
i=3
A3
i=4
A4
i=5
A5
i=6
A6