Microsoft
Learn

STUDENT AMBASSADOR

Microsoft Learn
Student Ambassadors

Algoritmi di ordinamento

Mario Cuomo

20.10.2021

Microsoft Learn
Student Ambassadors

MARIO CUOMO

mariocuomo.github.io

linkedin/in/mariocuomo

@mariocuomo.exe

@mariocuomoEXE

NET CODE

Semplici algoritmi di ordinamento: pseudocodice e implementazione in linguaggio C

Copertina flessibile – 5 marzo 2020

Amazon

Sommario

- ALGORITMI DI ORDINAMENTO
- ANALISI DEGLI ALGORITMI
- STUPID SORT
- SELECTION SORT
- MERGE SORT

Sommario

- ALGORITMI DI ORDINAMENTO
- ANALISI DEGLI ALGORITMI
- STUPID SORT
- SELECTION SORT
- MERGE SORT

ALGORITMO

ALGORITMO: procedura di calcolo **ben definita** che a partire da un input x produce un output y.

ALGORITMO

ALGORITMO: procedura di calcolo **ben definita** che a partire da un input x produce un output y.

Computabile dalla
$Random\ Access\ Machine$

ALGORITMO

ALGORITMO: procedura di calcolo **ben definita** che a partire da un **input** $\boldsymbol{x}$ produce un output y.

$Istanza$

Computabile dalla
$Random\ Access\ Machine$

ALGORITMO

ALGORITMO: procedura di calcolo **ben definita** che a partire da un **input** $\boldsymbol{x}$ produce un **output** $\boldsymbol{y}$.

$Istanza$

$Soluzione$

Computabile dalla
$Random\ Access\ Machine$

ALGORITMO

ALGORITMO: procedura di calcolo **ben definita** che a partire da un **input** $\boldsymbol{x}$ produce un **output** $\boldsymbol{y}$.

$Istanza$

$Soluzione$

Computabile dalla
$Random\ Access\ Machine$

ALGORITMO CORRETTO: ∀ istanza possibile $\boldsymbol{x}$ l'algoritmo produce la soluzione $\boldsymbol{y}$ corretta

ALGORITMO

ALGORITMO

ALGORITMO SICURO: ∀ istanza possibile x che soddisfa le $\boldsymbol{pre\text{-}condizioni}$ l’algoritmo produce la soluzione $\boldsymbol{y}$ corretta che rispetta le $\boldsymbol{post\text{-}condizioni}$.

Algoritmo di ordinamento

INPUT

sequenza di n elementi

$\langle a_1, a_2, a_3, \ldots, a_n \rangle$

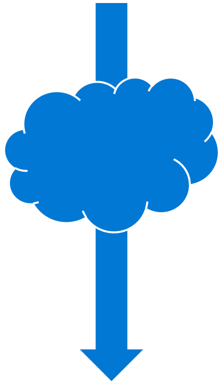
Algoritmo di ordinamento

INPUT
sequenza di n elementi
$\langle a_1, a_2, a_3, \dots, a_n \rangle$

OUTPUT
sequenza di n elementi
$\langle a_1, a_2, a_3, \dots, a_n \rangle$ tale che $a_1 \leq a_2 \leq \dots \leq a_n$

Algoritmo di ordinamento

$$\langle 3, 5, 1, 9, 5, 7 \rangle$$



$$\langle 1, 3, 5, 5, 7, 9 \rangle$$

Sommario

- ALGORITMI DI ORDINAMENTO
- ANALISI DEGLI ALGORITMI
- STUPID SORT
- SELECTION SORT
- MERGE SORT

Analisi degli algoritmi

Dato un problema P esistono diversi algoritmi A_i che producono la soluzione corretta.

Ci domandiamo quale tra questi sia quello *migliore*(?)

Analisi degli algoritmi

Dato un problema P esistono diversi algoritmi A_i che producono la soluzione corretta.

Ci domandiamo quale tra questi sia quello *migliore*(?)

tempo memoria

Analisi degli algoritmi

ASSUNZIONE ERRATA:
il tempo di calcolo di un algoritmo è una funzione della grandezza dell’input.

$$T(N) = f(n)$$

Analisi degli algoritmi

ASSUNZIONE ERRATA:
il tempo di calcolo di un algoritmo è una funzione della grandezza dell'input.

$$T(N) = f(n)$$

Il tempo di calcolo dipende anche (soprattutto) da come l'input mi viene fornito!

ESEMPIO

ALGORITMO DI ORDINAMENTO A

1. Verifica se la sequenza è ordinata (costo 3)
2. Se la sequenza non è ordinata, ordinala (costo 2)

ESEMPIO

ALGORITMO DI ORDINAMENTO A

1. Verifica se la sequenza è ordinata (costo 3)
2. Se la sequenza non è ordinata, ordinala (costo 2)

$A(\langle 1, 2, 3, 4 \rangle)$ ha costo 3.

ESEMPIO

ALGORITMO DI ORDINAMENTO A

1. Verifica se la sequenza è ordinata (costo 3)
2. Se la sequenza non è ordinata, ordinala (costo 2)

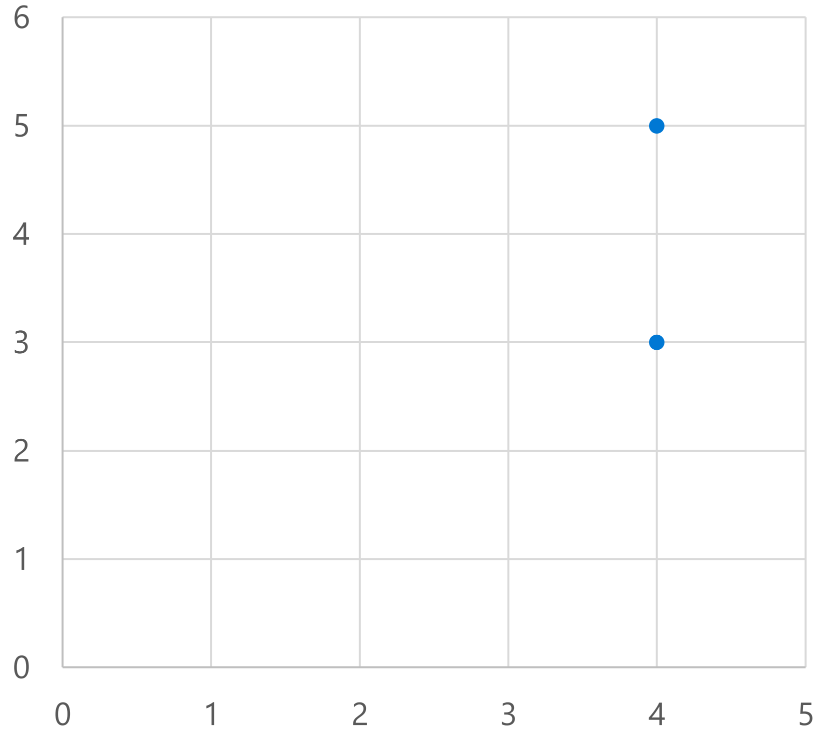
$A(\langle 1, 2, 3, 4 \rangle)$ ha costo 3.
$A(\langle 2, 1, 4, 3 \rangle)$ ha costo $3 + 2$.

ESEMPIO

ALGORITMO DI ORDINAMENTO A

1. Verifica se la sequenza è ordinata (costo 3)
2. Se la sequenza non è ordinata, ordinala (costo 2)

$A(\langle 1,2,3,4\rangle)$ ha costo 3.
$A(\langle 2,1,4,3\rangle)$ ha costo $3+2$.



Analisi degli algoritmi

$$T(N) = f(n)$$

È il tempo di calcolo che l'algoritmo A impiega per terminare la computazione su un input di dimensione n nel caso peggiore.

Analisi degli algoritmi

$$T(N) = f(n)$$

È il tempo di calcolo che l'algoritmo A impiega per terminare la computazione su un input di dimensione n nel caso peggiore.
Possiamo applicare l'$analisi\ asintotica$ alla funzione $T(N)$ per determinare un limite inferiore e superiore al tempo di calcolo (O, Ω, Θ)

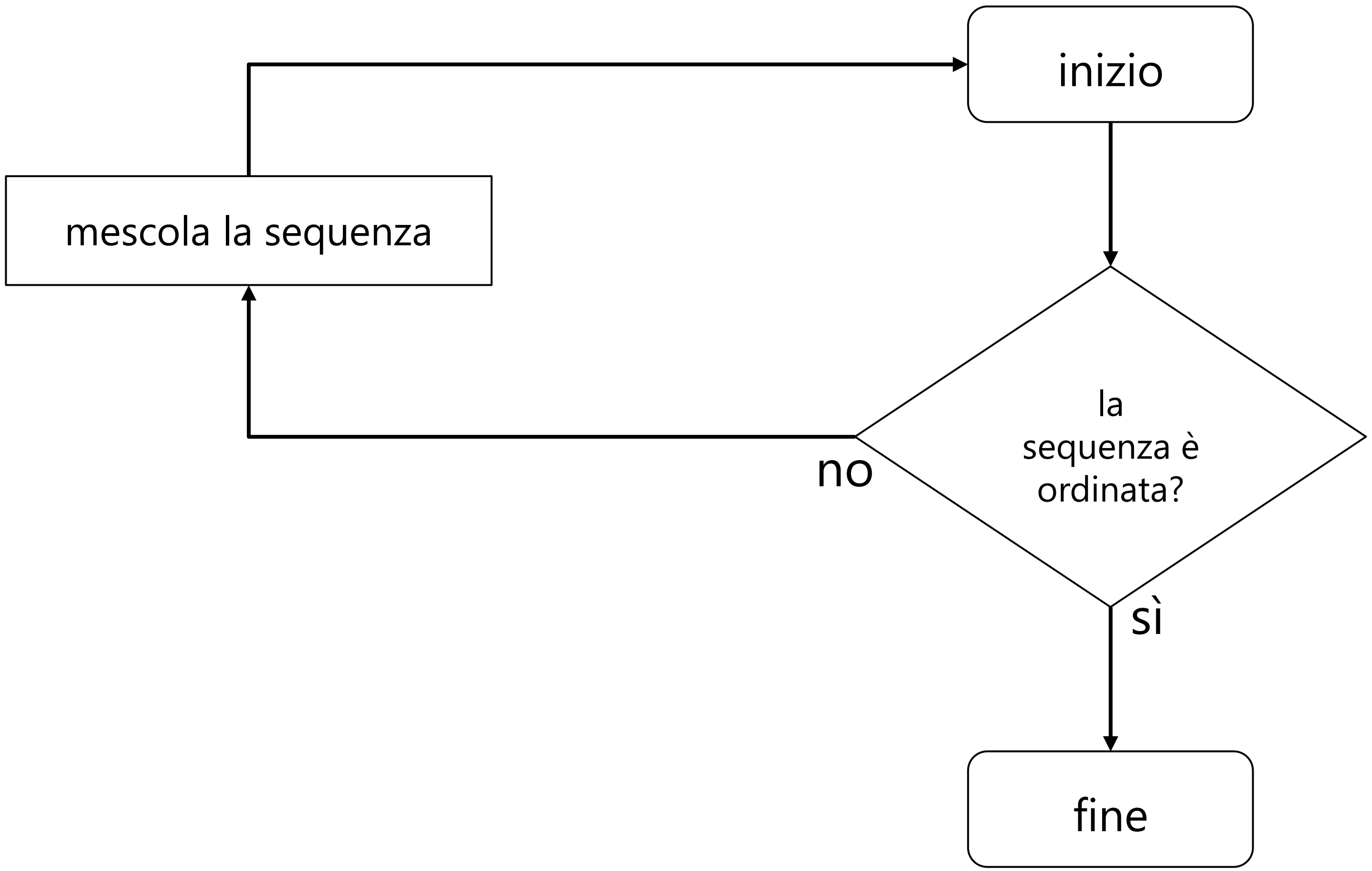
Sommario

- ALGORITMI DI ORDINAMENTO
- ANALISI DEGLI ALGORITMI
- STUPID SORT
- SELECTION SORT
- MERGE SORT

Stupid sort

Approccio probabilistico

STUPID SORT


inizio
mescola la sequenza
la sequenza è ordinata?
no
sì
fine

STUPID SORT

```
stupid_sort(A,n)
   while not isSorted(A,n)
        shuffle(A,n);
```

STUPID SORT

$caso\ peggiore$	???
$caso\ medio$	???
$caso\ migliore$	???

ESEMPIO

1	4	3	8	3

ESEMPIO

1	4	3	8	3

ESEMPIO

1	4	3	8	3

1	3	4	8	3

ESEMPIO

1	4	3	8	3

1	3	4	8	3

ESEMPIO

1	4	3	8	3

1	3	4	8	3

3	3	8	4	1

ESEMPIO

1	4	3	8	3

1	3	4	8	3

3	3	8	4	1

1	8	3	3	4

ESEMPIO

1	4	3	8	3

1	3	4	8	3

3	3	8	4	1

1	8	3	3	4

… … … … … … …

1	3	3	4	8

DEMO

TIME

Sommario

- ALGORITMI DI ORDINAMENTO
- ANALISI DEGLI ALGORITMI
- STUPID SORT
- SELECTION SORT
- MERGE SORT

SELECTION SORT

Approccio greedy.

SELECTION SORT

Approccio **greedy**.

a ogni iterazione
scegli la soluzione
più appetibile

SELECTION SORT

Approccio greedy.

a ogni iterazione
scegli la soluzione
più **appetibile**

ATTENZIONE
la soluzione ottima locale non coincide per forza con la soluzione ottima globale!

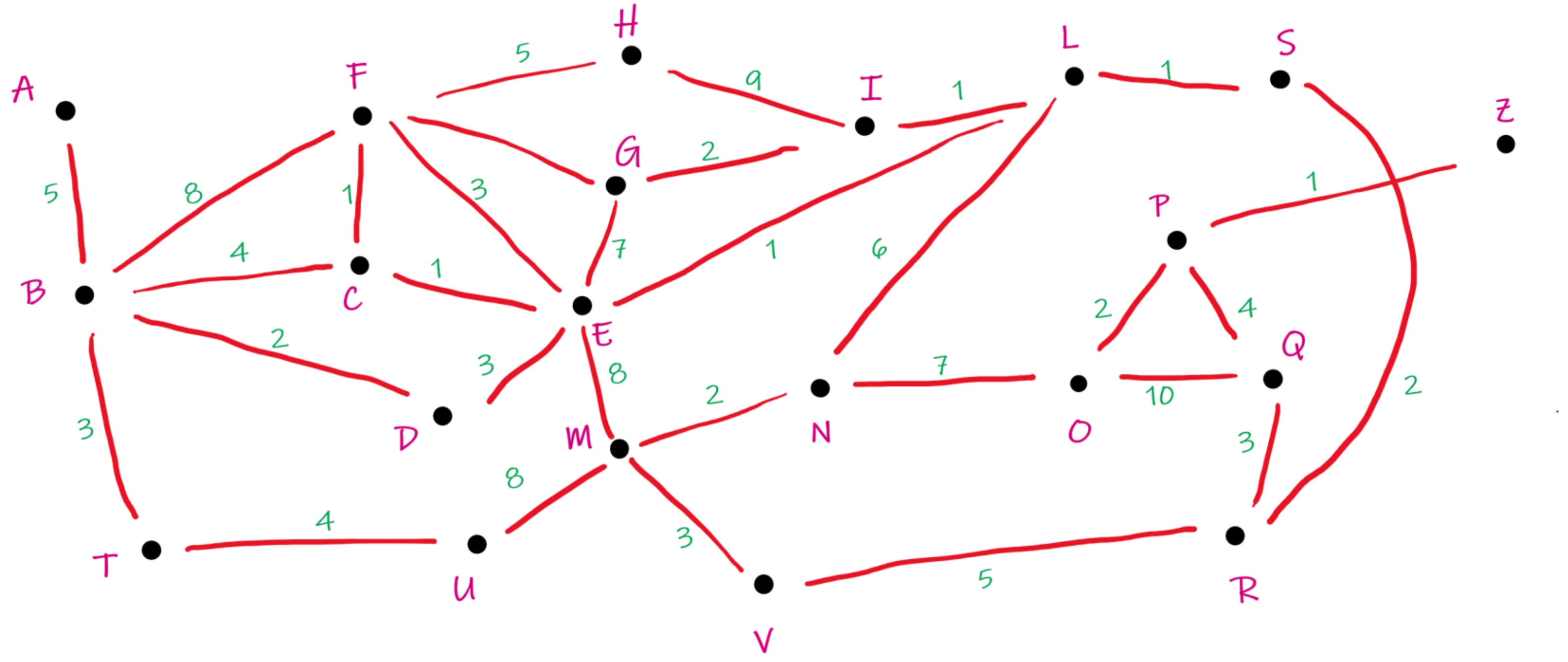
SELECTION SORT

Approccio greedy.

a ogni iterazione
scegli la soluzione
più appetibile

ATTENZIONE
la soluzione ottima locale non coincide per forza con la soluzione ottima globale!

ESEMPIO



$$M \to Q$$

strada migliore: $M \to V \to R \to Q$ di costo 11

strada scelta con algoritmo greedy: $M \to N \to L \to S \to R \to Q$ di costo 14

SELECTION SORT

L’idea è quella di dividere la sequenza in due porzioni.

- la prima porzione è ordinata
- la seconda non è ordinata

SELECTION SORT

L’idea è quella di dividere la sequenza in due porzioni.

- la prima porzione è ordinata
- la seconda non è ordinata

A ogni passo prendo un elemento (il più piccolo) della porzione non ordinata e lo inserisco nella porzione ordinata.

SELECTION SORT

L’idea è quella di dividere la sequenza in due porzioni.

- la prima porzione è ordinata
- la seconda non è ordinata

A ogni passo prendo un elemento (il più piccolo) della porzione non ordinata e lo inserisco nella porzione ordinata.

A ogni passo la porzionata ordinata cresce di un elemento, mentre quella non ordinata decresce di un elemento.

SELECTION SORT

9	4	1	8	3

sequenza ordinata

sequenza non ordinata

SELECTION SORT

9	4	1	8	3

sequenza ordinata

sequenza non ordinata

1	4	9	8	3

SELECTION SORT

sequenza ordinata

sequenza non ordinata

9	4	1	8	3

1	4	9	8	3

1	3	9	8	4

SELECTION SORT

sequenza ordinata

sequenza non ordinata

9	4	1	8	3

1	4	9	8	3

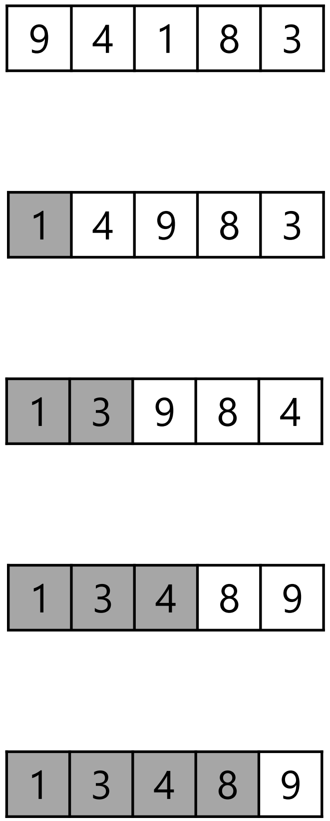
1	3	9	8	4

1	3	4	8	9

SELECTION SORT

sequenza ordinata

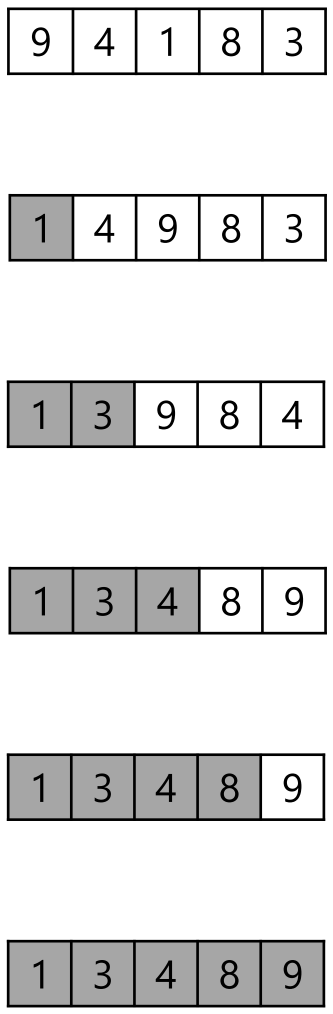
sequenza non ordinata



SELECTION SORT

sequenza ordinata

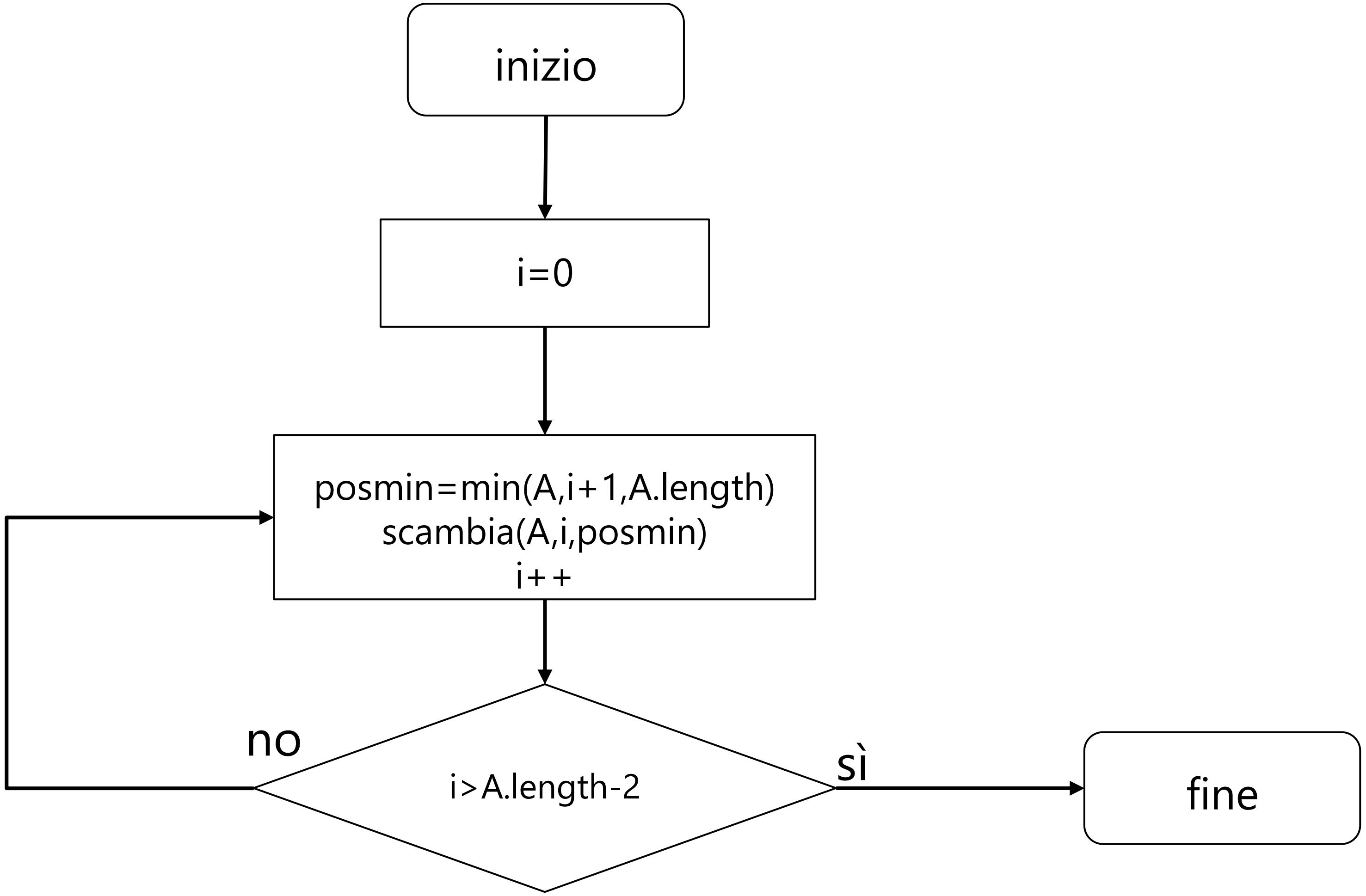
sequenza non ordinata



SELECTION SORT

```
selection_sort(A)
   for i=0 to A.length-2
       posmin ← i
       for j=i+1 to A.length-1
           if A[j]<A[posmin]
              posmin ← j
       if posmin!=i
          tmp ← A[i]
          A[i] ← A[posmin]
          A[posmin] ← tmp
```

SELECTION SORT



DEMO TIME

SELECTION SORT

$\boldsymbol{caso\ peggiore}$	$\Theta(N^2)$
$\boldsymbol{caso\ medio}$	$\Theta(N^2)$
$\boldsymbol{caso\ migliore}$	$\Theta(N^2)$

SELECTION SORT

$\boldsymbol{caso\ peggiore}$	$\Theta(N^2)$
$\boldsymbol{caso\ medio}$	$\Theta(N^2)$
$\boldsymbol{caso\ migliore}$	$\Theta(N^2)$

La computazione non dipende dall'entropia della sequenza in input.

SELECTION SORT

$\boldsymbol{caso\ peggiore}$	$\Theta(N^2)$
$\boldsymbol{caso\ medio}$	$\Theta(N^2)$
$\boldsymbol{caso\ migliore}$	$\Theta(N^2)$

La computazione non dipende dall'entropia della sequenza in input.

Se ho n elementi di input:

- Alla prima iterazione effettuo $n - 1$ confronti

SELECTION SORT

$\boldsymbol{caso\ peggiore}$	$\Theta(N^2)$
$\boldsymbol{caso\ medio}$	$\Theta(N^2)$
$\boldsymbol{caso\ migliore}$	$\Theta(N^2)$

La computazione non dipende dall'entropia della sequenza in input.

Se ho n elementi di input:

- Alla prima iterazione effettuo $n - 1$ confronti
- Alla seconda iterazione effettua $n - 2$ confronti

SELECTION SORT

$\boldsymbol{caso\ peggiore}$	$\Theta(N^2)$
$\boldsymbol{caso\ medio}$	$\Theta(N^2)$
$\boldsymbol{caso\ migliore}$	$\Theta(N^2)$

La computazione non dipende dall'entropia della sequenza in input.

Se ho n elementi di input:

- Alla prima iterazione effettuo $n-1$ confronti
- Alla seconda iterazione effettua $n-2$ confronti
- Alla iterazione i effettuo $n-i$ confronti

In media effettuo quindi $\frac{n}{2}$ confronti.

SELECTION SORT

$\boldsymbol{caso\ peggiore}$	$\Theta(N^2)$
$\boldsymbol{caso\ medio}$	$\Theta(N^2)$
$\boldsymbol{caso\ migliore}$	$\Theta(N^2)$

La computazione **non** dipende dall'entropia della sequenza in input.

Se ho n elementi di input:

- Alla prima iterazione effettuo $n - 1$ confronti
- Alla seconda iterazione effettua $n - 2$ confronti
- Alla iterazione i effettuo $n - i$ confronti

In media effettuo quindi $\frac{n}{2}$ confronti.

Ho n elementi, per ognuno effettuo in media $\frac{n}{2}$ confronti.

Effettuo in totale $\frac{n}{2} \cdot n = \frac{n^2}{2}$ confronti.

Sommario

- ALGORITMI DI ORDINAMENTO
- ANALISI DEGLI ALGORITMI
- STUPID SORT
- SELECTION SORT
- MERGE SORT

MERGE SORT

Approccio *divide et impera*.

MERGE SORT

Approccio *divide et impera*.

- Dividi il problema in problemi più piccoli dello stesso tipo
- Risolvi i sottoproblemi (eventualmente iterando)
- Combina le soluzioni parziali per ottenere il risultato del problema iniziale

MERGE SORT

Approccio *divide et impera* nel *merge sort*

MERGE SORT

Approccio *divide et impera* nel *merge sort*

Hai una sequenza di n valori.
Dividi la sequenza in due sottosequenze di $\frac{n}{2}$ valori ciascuna.

Ordina le due sottosequenze.

Utilizza le due sottosequenze ordinate per ottenere una unica sequenza ordinata.

MERGE SORT

Approccio *divide et impera* nel *merge sort*

Hai una sequenza di n valori.

Dividi la sequenza in due sottosequenze di $\frac{n}{2}$ valori ciascuna.

Ordina le due sottosequenze.

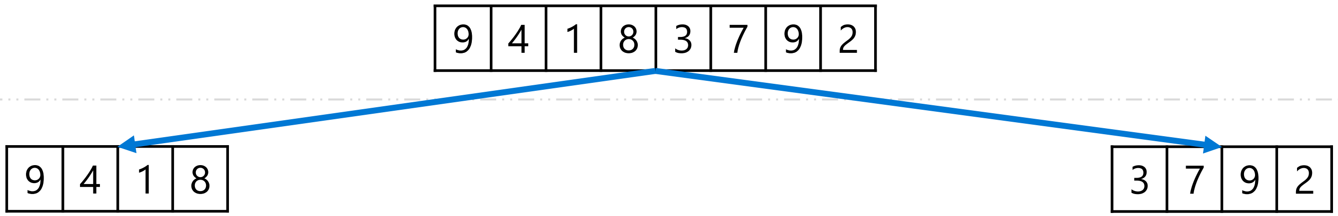
riapplicando in merge sort e la tecnica divide et impera

Utilizza le due sottosequenze ordinate per ottenere una unica sequenza ordinata.

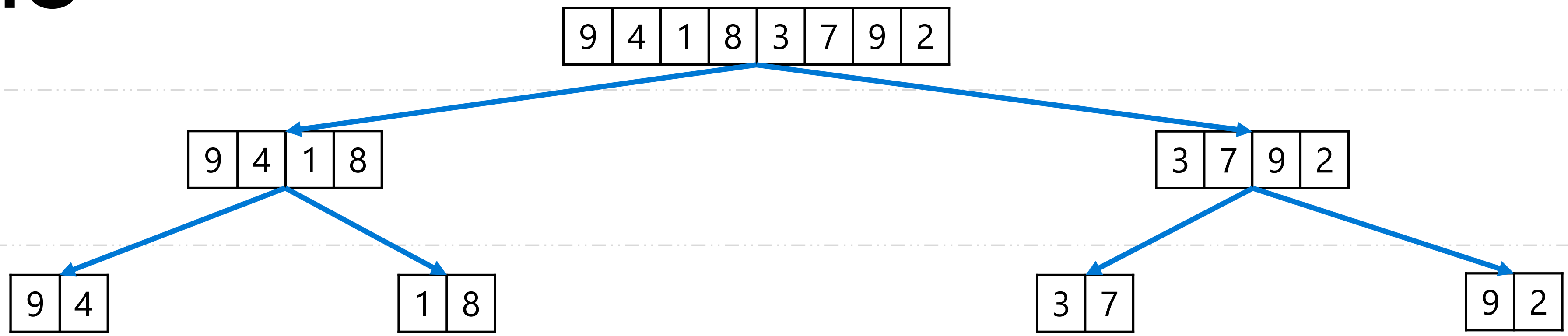
ESEMPIO

9	4	1	8	3	7	9	2

ESEMPIO

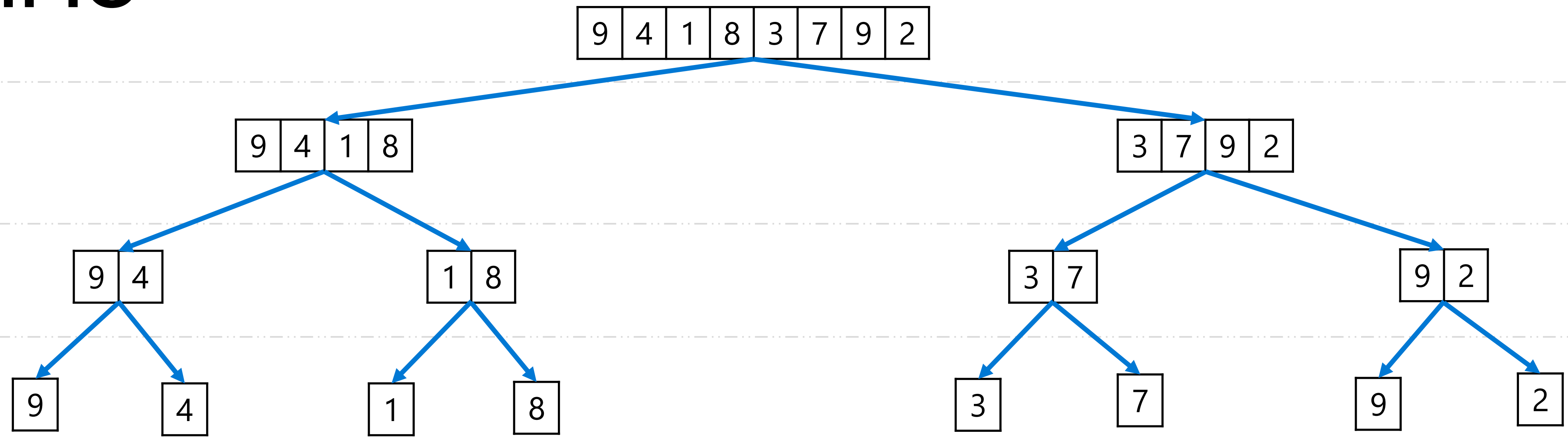


ESEMPIO



ESEMPIO

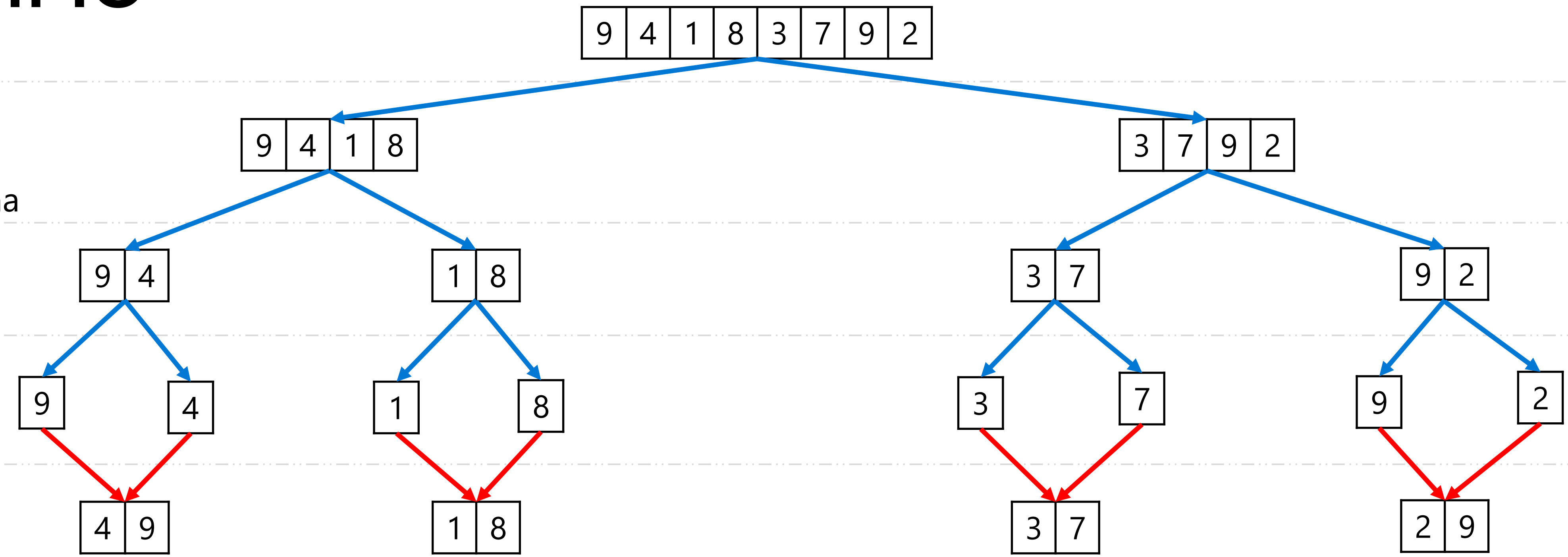
divide



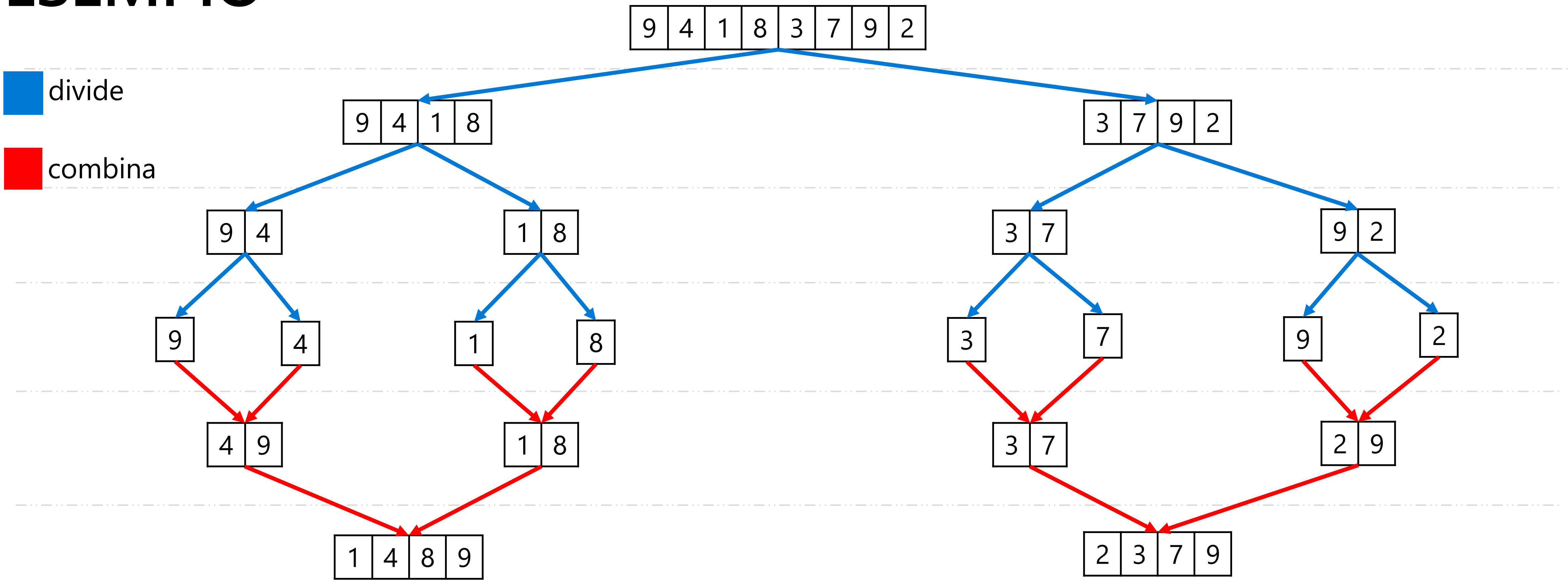
ESEMPIO

divide

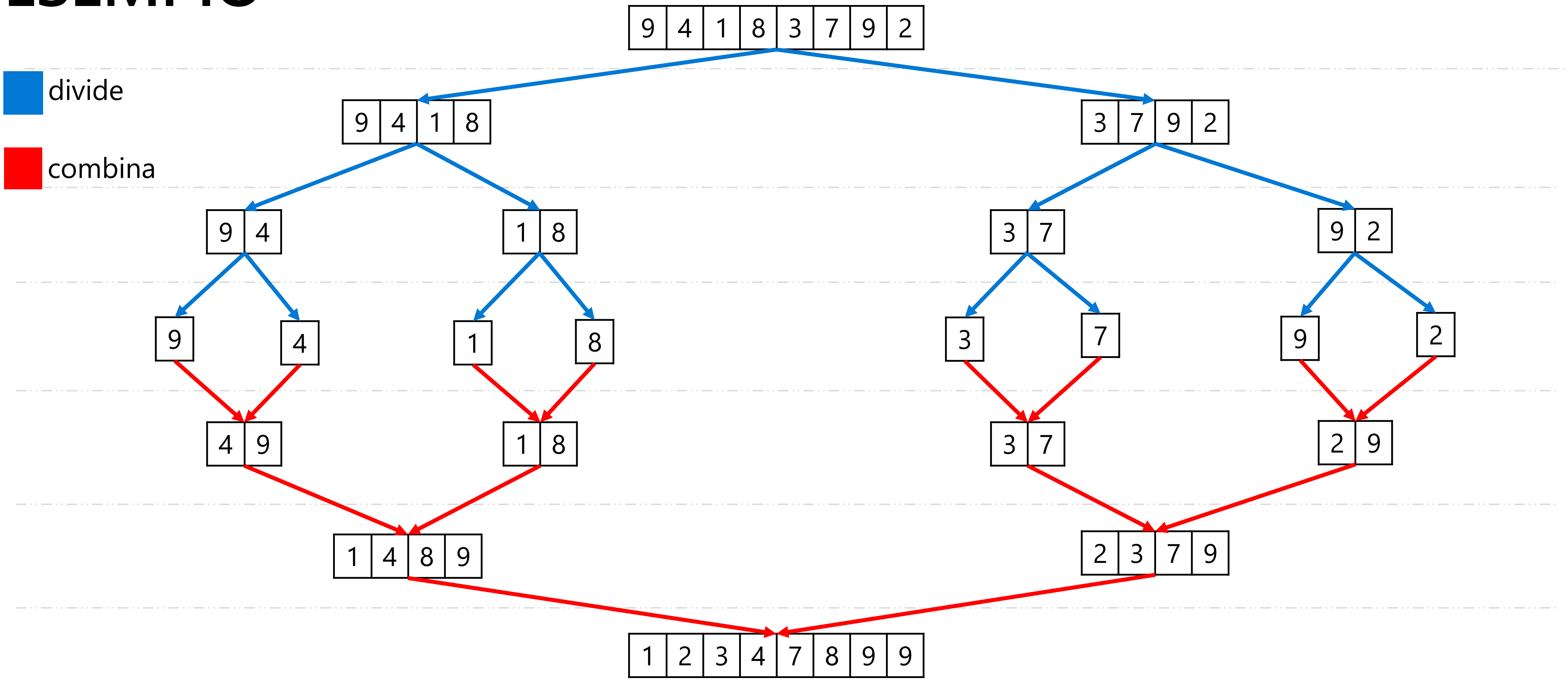
combina



ESEMPIO


divide
combina
9 4 1 8 3 7 9 2
9 4 1 8
3 7 9 2
9 4
1 8
3 7
9 2
9
4
1
8
3
7
9
2
4 9
1 8
3 7
2 9
1 4 8 9
2 3 7 9

ESEMPIO


divide
combina
9 4 1 8 3 7 9 2
9 4 1 8
3 7 9 2
9 4
1 8
3 7
9 2
9
4
1
8
3
7
9
2
4 9
1 8
3 7
2 9
1 4 8 9
2 3 7 9
1 2 3 4 7 8 9 9

MERGE SORT

```
mergesort (A, left, right)
    if left < right
        center ← (left + right) / 2
        mergesort(A, left, center)
        mergesort(A, center+1, right)
        merge(A, left, center, right)
```

MERGE SORT

```
merge (A, left, center, right)
     i ← left
     j ← center + 1
     k ← 0
     B ← array temp size=right-left+1


     while i ≤ center and j ≤ right
          if A[i] ≤ A[j]
             B[k] ← A[i]
             i ← i+1
          else
            B[k] ← A[j]
            j ← j + 1

       k ← k + 1

    …
```

```
merge (A, left, center, right)
    …
    while i ≤ center
      B[k] ← B[i]
       i ← i + 1
       k ← k + 1

    while j ≤ right
       B[k] ← B[j]
       j ← j + 1
       k ← k + 1

    for k ← left to right do
      A[k] ← B[k-left]
```

MERGE SORT

Analizziamo il costo $\mathrm{T}(N)$ dell'algoritmo merge sort

MERGE SORT

Analizziamo il costo $\mathrm{T}(N)$ dell'algoritmo merge sort

Se la sequenza è di un solo elemento il costo è $\Theta(1)$

MERGE SORT

Analizziamo il costo $\mathrm{T}(N)$ dell'algoritmo merge sort

Se la sequenza è di un solo elemento il costo è $\Theta(1)$

Il costo per effettuare la divisione della sequenza di due è $\Theta(1)$ perché devo calcolare solamente $\frac{N}{2}$.

MERGE SORT

Analizziamo il costo $\mathrm{T}(N)$ dell'algoritmo merge sort

Se la sequenza è di un solo elemento il costo è $\Theta(1)$

Il costo per effettuare la divisione della sequenza di due è $\Theta(1)$ perché devo calcolare solamente $\frac{N}{2}$.

Ho due sottoproblemi di dimensione $\frac{N}{2}$ ciascuno.
Il costo è $2 \cdot \mathrm{T}\left(\frac{N}{2}\right)$.

MERGE SORT

Analizziamo il costo $\mathrm{T}(N)$ dell'algoritmo merge sort

Se la sequenza è di un solo elemento il costo è $\Theta(1)$

Il costo per effettuare la divisione della sequenza di due è $\Theta(1)$ perché devo calcolare solamente $\frac{N}{2}$.

Ho due sottoproblemi di dimensione $\frac{N}{2}$ ciascuno.
Il costo è $2 \cdot \mathrm{T}\left(\frac{N}{2}\right)$.

Il costo per combinare due sottosequenze di $\frac{N}{2}$ elementi ciascuno è di $\Theta(N)$.

MERGE SORT

$$T(N) = \begin{cases} \Theta(1) & se\ N = 0\ oppure\ 1 \\ 2 \cdot \mathrm{T}\left(\frac{N}{2}\right) + \Theta(N) & se\ N > 1 \end{cases}$$

MERGE SORT

$$T(N) = \begin{cases} \Theta(1) & se\ N = 0\ oppure\ 1 \\ 2 \cdot \mathrm{T}\left(\frac{N}{2}\right) + \Theta(N) & se\ N > 1 \end{cases}$$

Questa equazione di ricorrenza ha come soluzione $T(N) = \Theta(N \cdot \log N)$

DEMO TIME

Semplici algoritmi di ordinamento: pseudocodice e implementazione in linguaggio C

Copertina flessibile – 5 marzo 2020

Amazon

Altri argomenti trattati

- INSERTION SORT
- TREE SORT
- BUBBLE SORT
- QUICK SORT
- COUNTING SORT
- …

Microsoft Learn
Student Ambassadors

RISORSE

github.com/mariocuomo/talks

GitHub

Grazie