Ricorsione e Ordinamento

Esercitazione 10

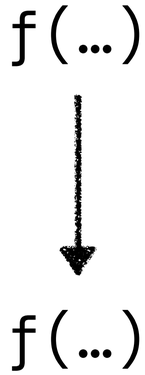
18/11/2024 - Alessandro Montenegro

Ricorsione

Ricorsione

Riassunto

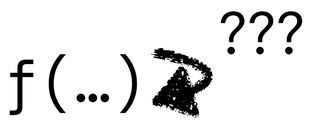
- Una funzione **ricorsiva** è una funzione che **chiama se stessa**
- Ogni volta che una funziona chiama se stessa, deve **attendere** che la sua **chiamata** ricorsiva **termini**
- Ogni attivazione della funzione è un'**istanza separata** (i.e., ha il suo spazio di variabili locali etc...)

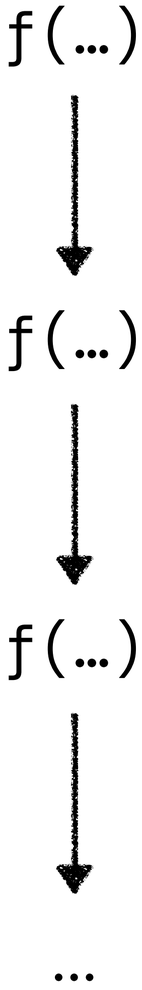


Ricorsione

Riassunto

- Ogni istanza della funzione, chiama a sua volta se stessa...
- IMPORTANTE: definire un caso base in cui la funzione non ricorre

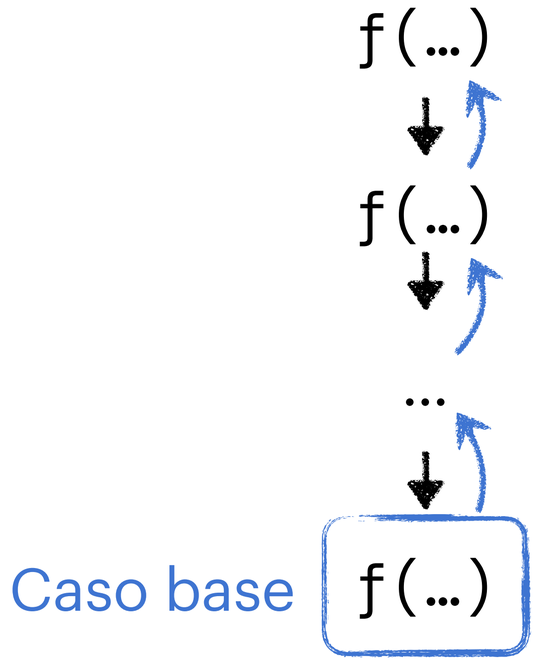




Ricorsione

Riassunto

- Definire un **caso base**
- Assicurarsi che ad ogni chiamata delle funzione essa **si avvicini al caso base**
- Come? Mettere una **condizione** su qualche **variabile** passata come **argomento**



Ricorsione

Riassunto

Pro

- Meccanismo **elegante**
- Molti problemi si risolvono molto più **intuitivamente**

Contro

- **Costosa** in termini di **memoria** (ogni volta bisogna allocare nuovo spazio per la funzione, come nel caso di Fibonacci che vedrete a lezione)
- Bisogna fare attenzione alla definizione di un **caso base**, a come **trattare** gli **output** di ogni funzione e al fatto che ogni chiamata deve **avvicinarsi** al **caso base**

Stampa Numeri

Ricorsione: Esercizio 1

- Si scriva una funzione ricorsiva che prende in input un intero positivo n e che stampi tutti i numeri interi da 1 a n
- Esempio: n = 10 -> 1 2 3 4 5 6 7 8 9 10

- Si scriva una funzione ricorsiva che prende in input un intero positivo n e che stampi tutti i numeri interi da n a 1
- Esempio: n = 10 -> 10 9 8 7 6 5 4 3 2 1

Potenze

Ricorsione: Esercizio 2

- Si scriva una funzione ricorsiva “potenza” che prende in input un numero base e un numero esponente e restituisce la base elevata all’esponente
- Esempio: base = 2, esponente = 6 -> 64

Triangolo di Tartaglia

Ricorsione: Esercizio 4

- Scrivere una funzione che stampi il triangolo di tartaglia dato un numero intero N
- Si utilizzi una funzione ricorsiva per il calcolo dei coefficienti del triangolo
- Nota: il coefficiente del triangolo in posizione n (riga) k (colonna), è il coefficiente binomiale (n,k) = n!/(k!(n-k)!) con (n,0) = (n,n) = 1

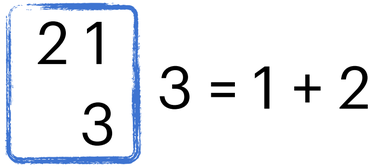
```
N = 3

N=0  |  1

N=1  |  1  1

N=2  |  1  2  1

N=3  |  1  3  3  1
```



3 = 1 + 2

Inversione Stringa

Ricorsione: Esercizio 5

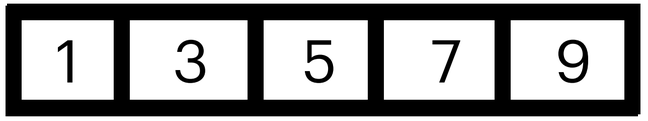
- Si scriva una funzione ricorsiva “potenza” in grado di invertire una stringa data come argomento. La stringa data in ingresso non deve essere modificata all fine dell’esecuzione.
- Esempio: “ciao mamma!” -> “!ammam oaic”

Ricerca

Ricerca

Scorrimento iterativo dell’array

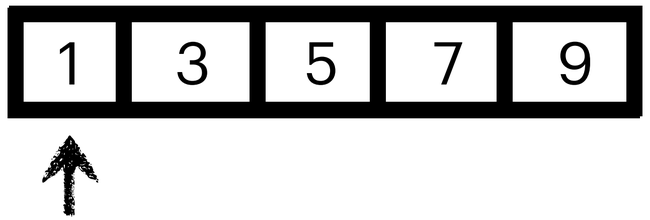
N = 7 ?



Ricerca

Scorrimento iterativo dell’array

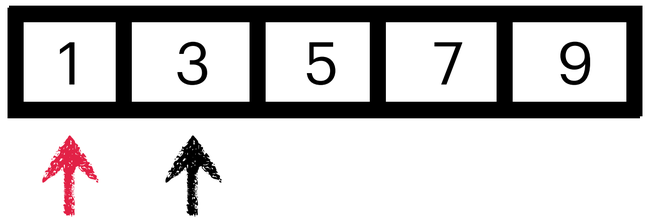
N = 7 ?



Ricerca

Scorrimento iterativo dell’array

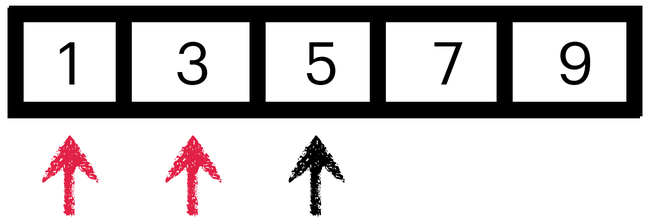
N = 7 ?



Ricerca

Scorrimento iterativo dell’array

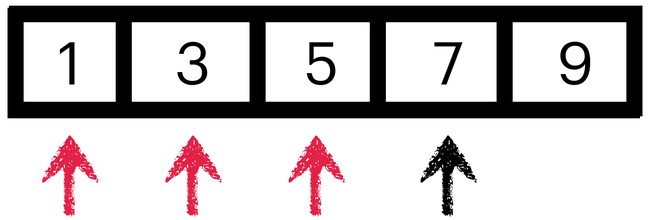
N = 7 ?



Ricerca

Scorrimento iterativo dell’array

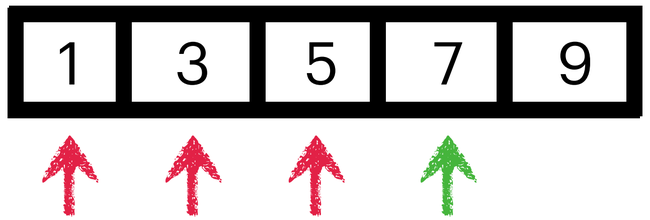
N = 7 ?



Ricerca

Scorrimento iterativo dell'array

N = 7 ?

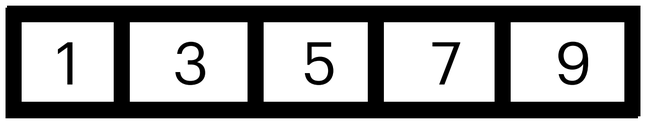


- **Caso peggiore**: scorriamo **tutto l'array** (come quando non è presente l'argomento)
- Funziona anche se l'array **non è ordinato**

Ricerca

Ricerca Binaria su Array Ordinato

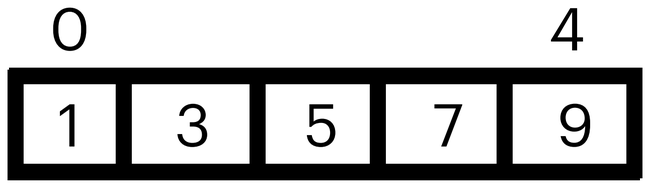
N = 7 ?



Ricerca

Ricerca Binaria su Array Ordinato

N = 7 ?



Start = 0 End = 4

m = 2

Controllo l'indice nel punto medio

$$m = \frac{start + end}{2}$$

Ricerca

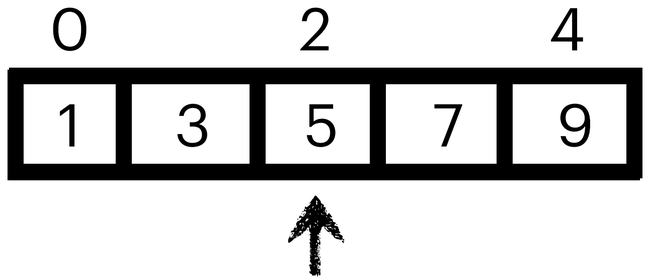
Ricerca Binaria su Array Ordinato

N = 7 ?

Start = 0

End = 4

m = 2



Controllo l'indice nel punto medio

$$m = \frac{start + end}{2}$$

Ricerca

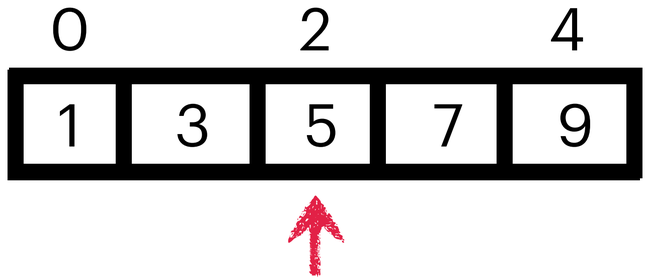
Ricerca Binaria su Array Ordinato

N = 7 ?

Start = 0

End = 4

m = 2



Controllo l'indice nel punto medio

$$m = \frac{start + end}{2}$$

L'elemento in posizione m non è quello che cerchiamo

L'elemento in posizione m è più piccolo di N e l'array è ordinato

Ricerca

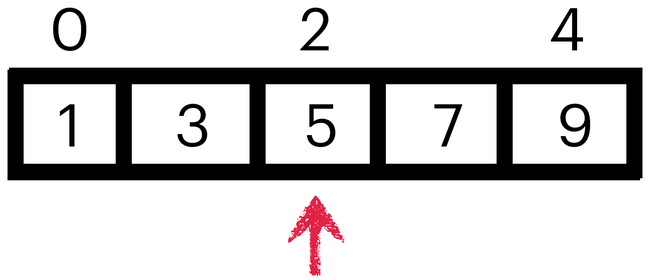
Ricerca Binaria su Array Ordinato

N = 7 ?

Start = 0

End = 4

m = 2



Controllo l'indice nel punto medio

$$m = \frac{start + end}{2}$$

Cerchiamo nella metà di destra dove ci sono gli elementi più grandi di quello in posizione m (quindi forse c'è N)

Start = m+1, End = End (nel caso opposto Start = Start e End = m-1)

Ricerca

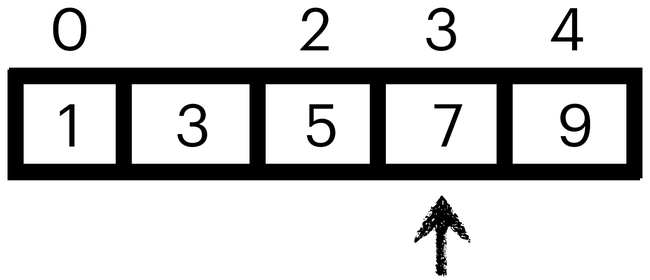
Ricerca Binaria su Array Ordinato

N = 7 ?

Start = 3

End = 4

m = 3



Controllo l'indice nel punto medio

$$m = \frac{start + end}{2}$$

Ricerca

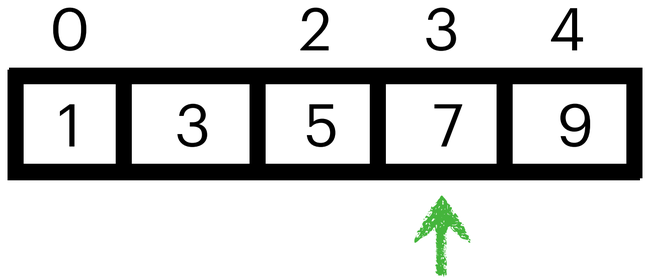
Ricerca Binaria su Array Ordinato

N = 7 ?

Start = 3

End = 4

m = 3



Controllo l'indice nel punto medio

$$m = \frac{start + end}{2}$$

Possibilità: troviamo N oppure Start > End (non troviamo N)

Ricerca Binaria

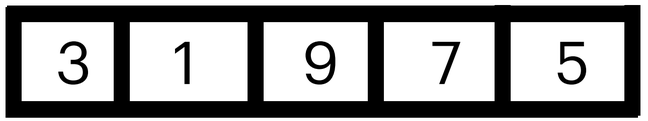
Ricorsione: Esercizio 6

- Si scriva una funzione ricorsiva che cerca un numero N in un array ordinato tramite ricerca binaria
- **Bonus**: si faccia lo stesso con le stringhe

Ordinamento

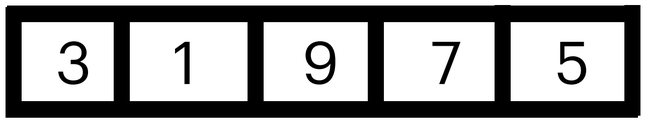
Selection Sort

Ordinamento Iterativo 1



Selection Sort

Ordinamento Iterativo 1

V = 

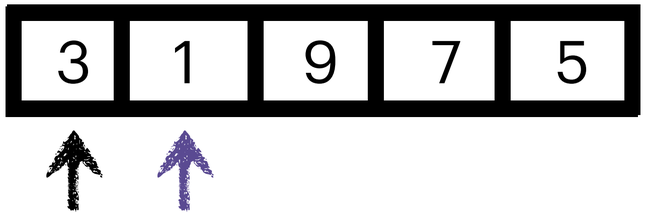
Logica

- Fisso un elemento in posizione i fino a LEN
- Scorro l’array in avanti a partire da quello in posizione j = i + 1 fino a LEN - 1
- Se V[i] > V[j] li scambio
- Alla fine del secondo ciclo fino a i siamo sicuri che l’i-esimo elemento più piccolo è in posizione i

Selection Sort

Ordinamento Iterativo 1

i = 0 , j = 1

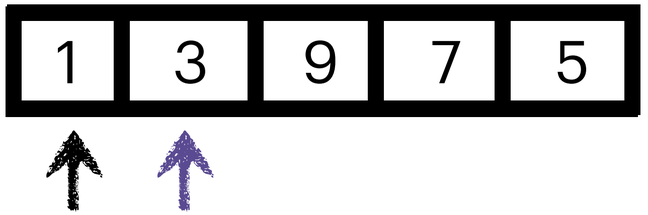


Selection Sort

Ordinamento Iterativo 1

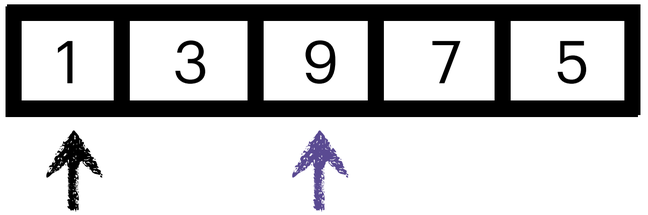
i = 0 , j = 1

Scambio



Selection Sort

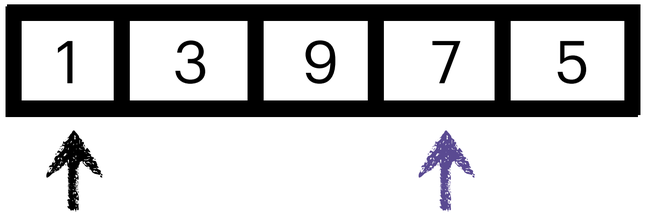
Ordinamento Iterativo 1



Selection Sort

Ordinamento Iterativo 1

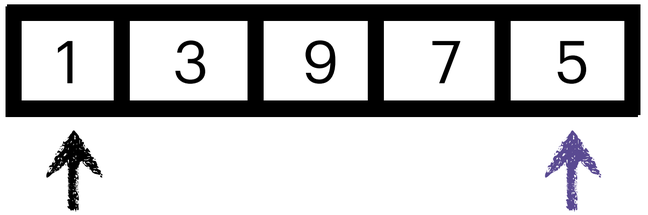
i = 0 , j = 1



Selection Sort

Ordinamento Iterativo 1

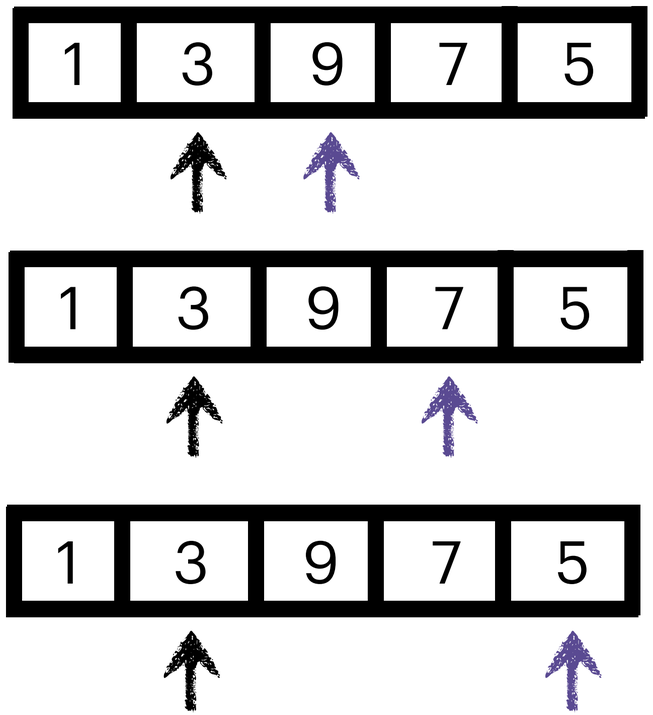
i = 0 , j = 1



Selection Sort

Ordinamento Iterativo 1

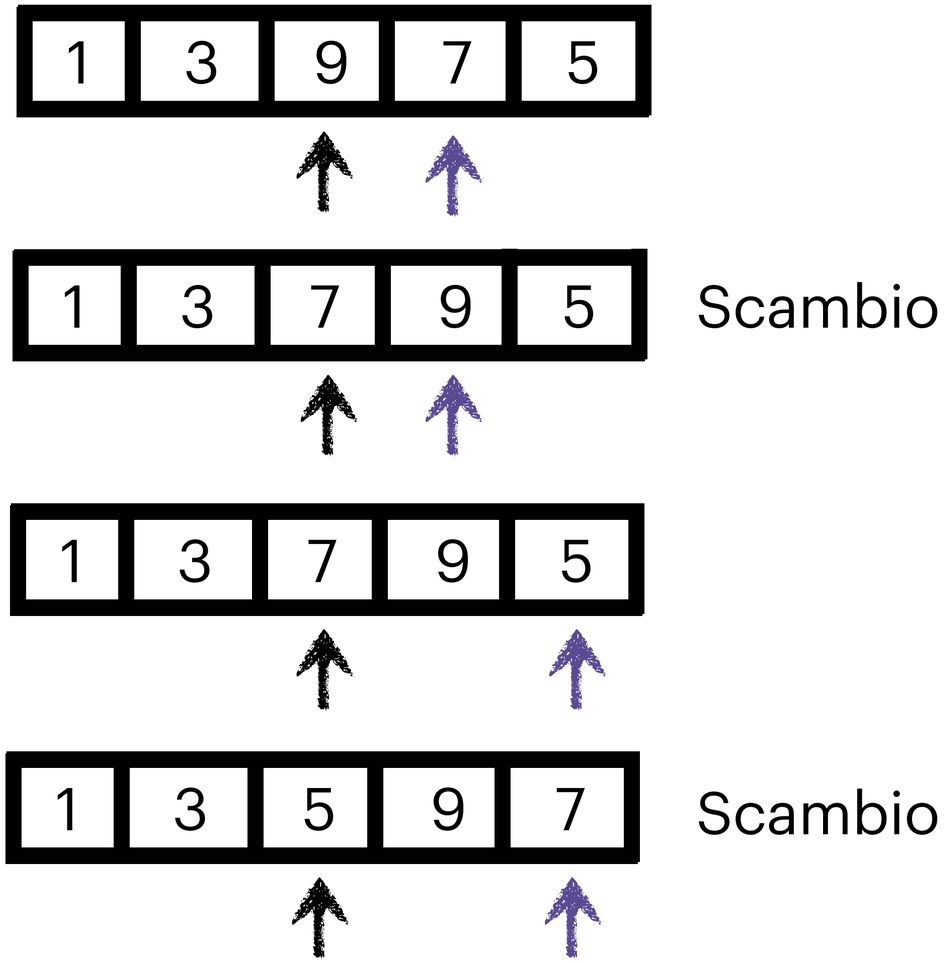
i = 1 , j = 2



Selection Sort

Ordinamento Iterativo 1

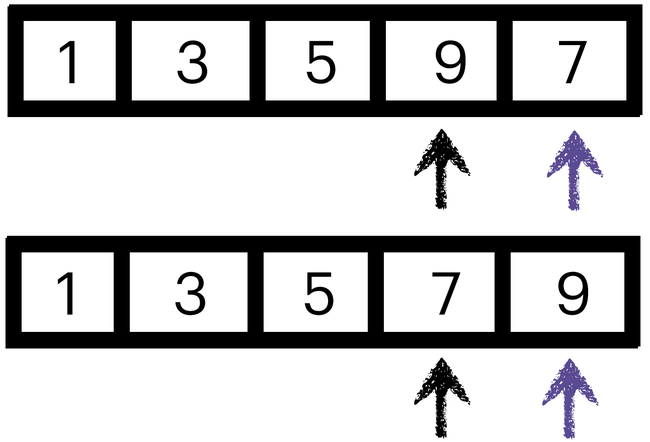
i = 2, j = 3



Selection Sort

Ordinamento Iterativo 1

i = 3, j = 4

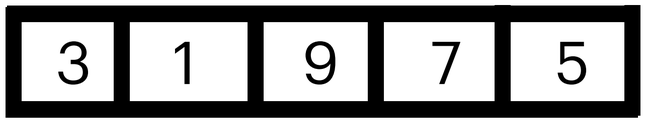


Scambio

Fine!

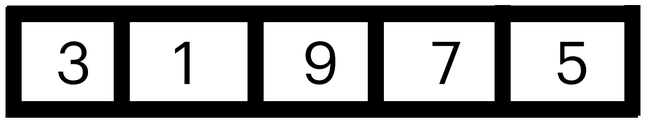
Bubble Sort

Ordinamento Iterativo 2



Bubble Sort

Ordinamento Iterativo 2

V = 

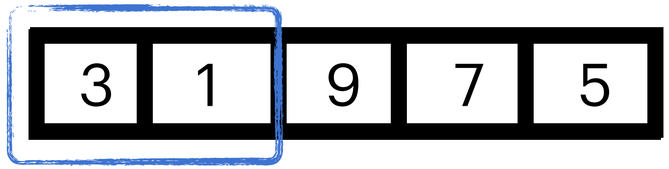
Logica

- Fisso un elemento in posizione i (scorro fino a LEN - 1)
- Scorro l’array in avanti a partire da quello in posizione j = 0 e fino alla posizione LEN - i - 1
- Se V[j] > V[j+1] li scambio (si usa una finestra di 2 elementi)
- Alla fine del secondo ciclo siamo sicuri che gli elementi da i in poi sono ordinati
- Ottimizzazione: se in una passata non ho cambiato nulla posso fermarmi

Bubble Sort

Ordinamento Iterativo 2

i = 0 -> LEN - 1, j = 0 -> LEN - 1

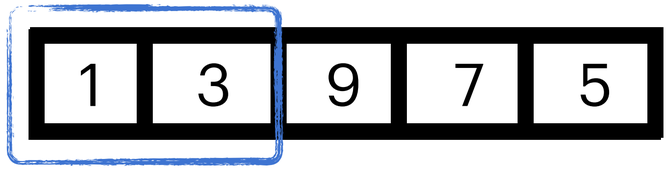


Bubble Sort

Ordinamento Iterativo 2

i = 0 -> LEN - 1, j = 0 -> LEN - 1

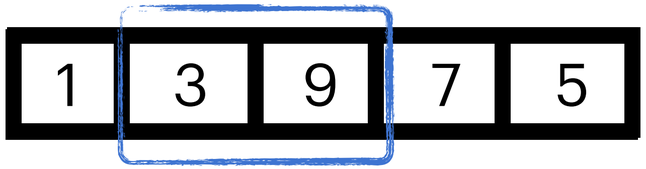
Scambio



Bubble Sort

Ordinamento Iterativo 2

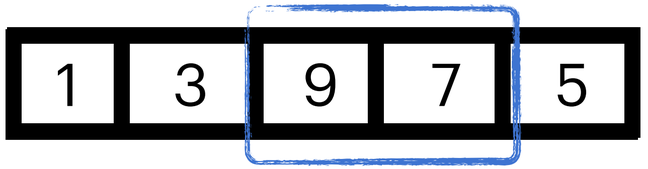
i = 0 -> LEN - 1, j = 0 -> LEN - 1



Bubble Sort

Ordinamento Iterativo 2

i = 0 -> LEN - 1, j = 0 -> LEN - 1



Bubble Sort

Ordinamento Iterativo 2

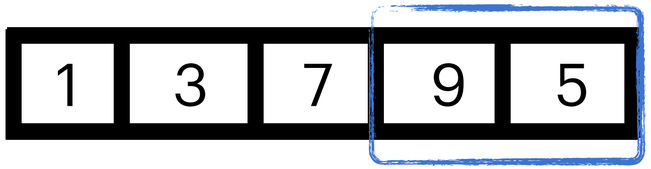
i = 0 -> LEN - 1, j = 0 -> LEN - 1

Scambio

Bubble Sort

Ordinamento Iterativo 2

i = 0 -> LEN - 1, j = 0 -> LEN - 1

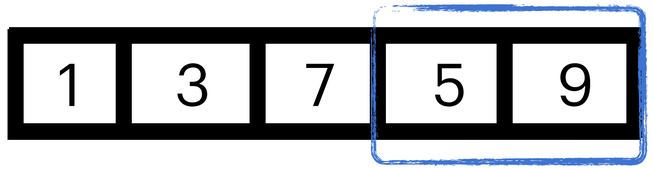


Bubble Sort

Ordinamento Iterativo 2

i = 0 -> LEN - 1, j = 0 -> LEN - 1

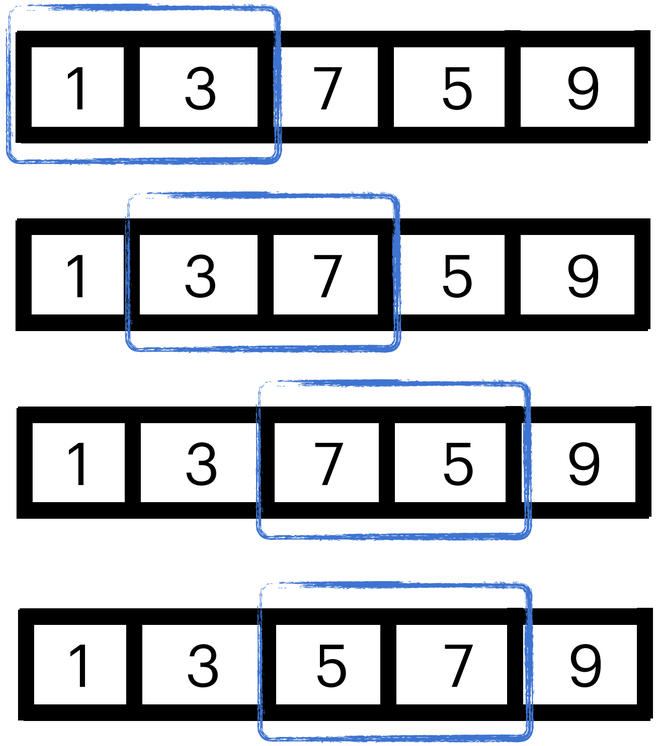
Scambio



Bubble Sort

Ordinamento Iterativo 2

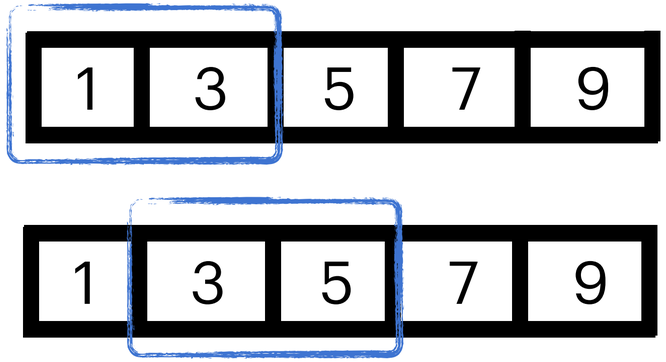
i = 1 -> LEN - 1, j = 0 -> LEN - 2



Bubble Sort

Ordinamento Iterativo 2

i = 1 -> LEN - 1, j = 0 -> LEN - 3



Fine!
Nell’ultima passata dell’array non ho effettuato scambi

Selection Sort vs Bubble Sort

Ordinamento Iterativo

- Entrambi non sono algoritmi ottimi (hanno la stessa complessità nel caso peggiore)
- Bubble Sort fa più scambi di Selection Sort
- Se i dati da scambiare sono pesanti, è preferibile Selection Sort
- Bubble Sort può essere ottimizzato fermandosi se non sono avvenuti scambi

Merge Sort

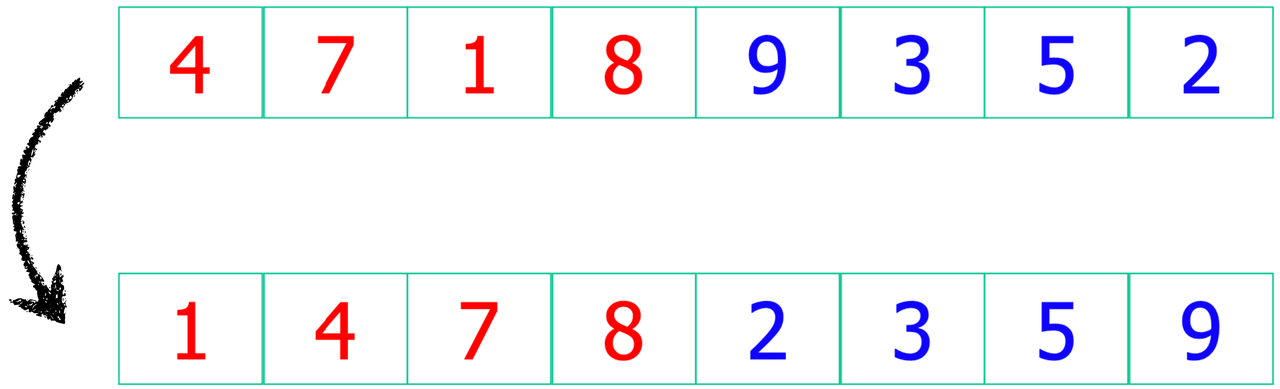
Ordinamento Ricorsivo

Logica

- Si divide l’array in due
- Si copia in un nuovo array in modo ordinato, rendendo gli elementi da entrambe le metà dell’array

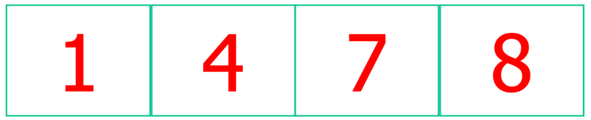
Merge Sort

Ordinamento Ricorsivo



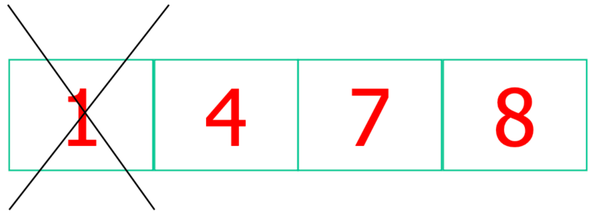
Merge Sort

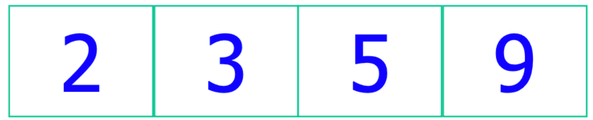
Ordinamento Ricorsivo



Merge Sort

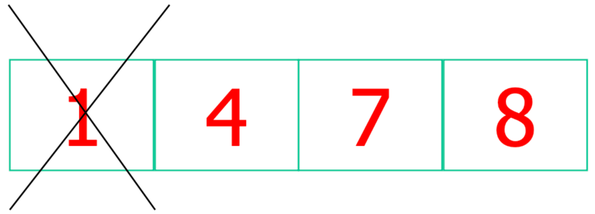
Ordinamento Ricorsivo

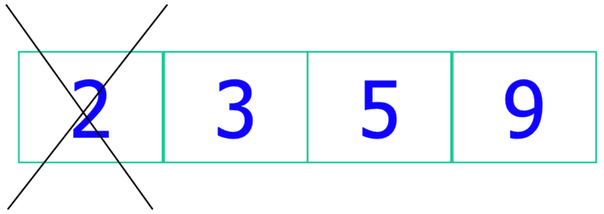




Merge Sort

Ordinamento Ricorsivo

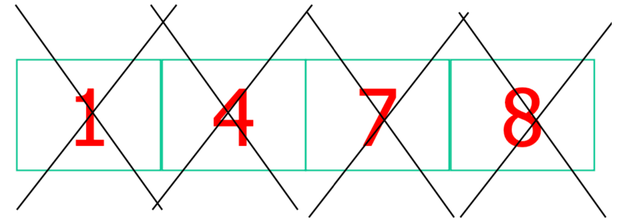


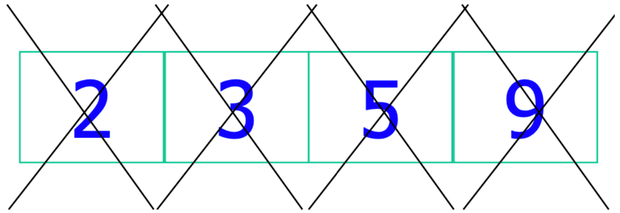


Merge Sort

Ordinamento Ricorsivo

[...]





1	2	3	4	5	7	8	9

Merge Sort

Ordinamento Ricorsivo

Divisione a metà

Caso base: porzioni di dimensione 1

Merge di tutte le porzioni

Array ordinato!

Ordinamento

Esercizio 7

- Si scrivano le funzioni di Selection Sort (iterativo), Bubble Sort (iterativo) e Merge Sort (ricorsivo) per l'ordinamento di un array di interi.
- **Bonus**: si faccia lo stesso con le stringhe

Contatti

Alessandro Montenegro

Mail: alessandro.montenegro@polimi.it

Sito: https://montenegroalessandro.github.io/InfoA2425/index.html