



ELEMENTI DI INFORMATICA

DOCENTE: FRANCESCO MARRA

INGEGNERIA CHIMICA

INGEGNERIA ELETTRICA

SCIENZE ED INGEGNERIA DEI MATERIALI

INGEGNERIA GESTIONALE DELLA LOGISTICA E DELLA PRODUZIONE

INGEGNERIA NAVALE


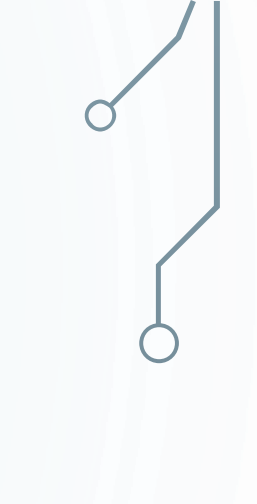
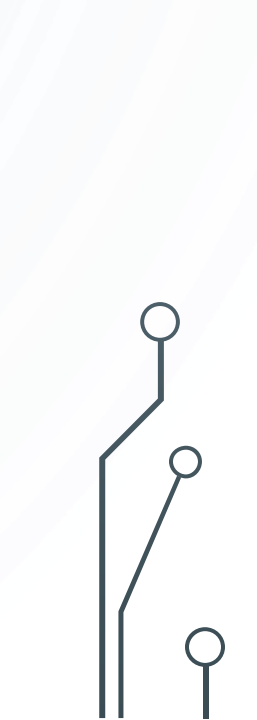
UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE



ALGORITMI DI RICERCA E ORDINAMENTO SU UN VETTORE



AGENDA

- Ricerca su vettore
 - Ricerca sequenziale
 - Ricerca binaria
 - Ordinamento di un vettore
 - Selection sort
 - Bubble sort
- 
- 
- 

OPERAZIONI SUI VETTORI: PROBLEMI DI RICERCA



RICERCA IN UN VETTORE

- Tra le operazioni più diffuse in programmazione è la ricerca di un elemento in un dato vettore (se esiste)
- L'algoritmo più semplice e intuitivo di ricerca è la **ricerca sequenziale**
 - Si scorre il vettore dall'inizio alla fine, fintanto che non viene individuato l'elemento cercato
 - Questo è anche l'unico algoritmo possibile su un vettore non ordinato
- **Se il vettore è ordinato** si può usare un algoritmo più efficiente: la **ricerca binaria** (o dicotomica)

RICERCA BINARIA

Valore Cercato: 10

 Primo

 Ultimo

Passo1

1	3	5	10	15	23	31	47	47	56	64
0	1	2	3	4	5	6	7	8	9	10

↑
Elemento Centrale

Passo2

1	3	5	10	15	23	31	47	47	56	64
0	1	2	3	4	5	6	7	8	9	10

↑
Elemento Centrale

Passo3

1	3	5	10	15	23	31	47	47	56	64
0	1	2	3	4	5	6	7	8	9	10

↑
Elemento Centrale

RICERCA BINARIA

Se il valore cercato non è all'interno del vettore...

Valore Cercato: 8

Primo

Ultimo

Passo4

1	3	5	10	15	23	31	47	47	56	64
0	1	2	3	4	5	6	7	8	9	10

Elemento Centrale

COMPLESSITA' COMPUTAZIONALE: RICERCA SEQUENZIALE

- **Best case $O(1)$:** ricerca con successo al primo caso (primo elemento del vettore), con un solo confronto
- **Worst case $O(N^2)$:** ricerca senza successo, l'algoritmo dovrà scorrere tutto l'array di dimensione N , quindi farà N confronti
- **Medium case $O(N^2)$:** mediamente l'algoritmo effettua $(N+1)/2$ confronti

COMPLESSITA' COMPUTAZIONALE: RICERCA BINARIA

- **Best case $O(1)$:** ricerca con successo al primo caso (elemento centrale del vettore), con un solo confronto
- **Worst case $O(\log_2(N))$:** ricerca senza successo; ad ogni iterazione l'insieme è dimezzato, il numero di confronti è pari a quante volte N può essere diviso per 2 fino a ridurlo a 0. In un vettore di dimensione $N = 2^h$, l'algoritmo deve compiere circa $h = \log_2(N)$ passi (e quindi confronti) per la ricerca senza successo.
- **Medium case $O(\log_2(N))$:** nel caso medio il numero è leggermente inferiore ma proporzionale a $\log_2(N)$

COMPLESSITA' COMPUTAZIONALE: RICERCA BINARIA

```
int a[N]={1,5,8,10,15};
int primo=0, ultimo=N-1, medio;
bool trovato=false;

while (primo<=ultimo && !trovato)
{
    medio = (int)(primo+ultimo)/2;
    if(a[medio]==x)
        trovato=true;
    else if (x > a[medio])
        primo = medio + 1;
    else /* x< a[medio] */
        ultimo = medio - 1;
}
if(trovato)
    cout<<"numero trovato in posizione: "<<medio<<endl;
else
    cout<<"numero non trovato";
```

ESERCIZI

- Realizzare un programma che permetta di inserire un vettore in ingresso dal lato utente di N numeri interi positivi (N massimo 20).

Il programma:

- Controlla se il vettore è ordinato e lo comunica all'utente
- chiede all'utente un numero da cercare:
- dà come risposta se il numero è presente o meno e quanti confronti sono stati effettuati, utilizzando la ricerca sequenziale
- ripete la ricerca finché l'utente non inserisce un numero negativo

ESERCIZI

- Realizzare un programma che permetta di inserire un vettore in ingresso dal lato utente di N numeri interi positivi (N massimo 20).
 - Il programma chiederà all'utente un numero da cercare:
 - Il programma darà come risposta se il numero è presente o meno e quanti confronti sono stati effettuati, utilizzando la ricerca binaria
 - Si ripete la ricerca finchè l'utente non inserisce un numero negativo
- N.B.: si presuppone che il vettore sia ordinato

ESERCIZI

- Realizzare un programma che permetta di generare un vettore in ingresso dal lato utente di N numeri interi positivi tra 0 e 10 pseudo-randomici. N dovrà essere minimo 20 e massimo 100 scelto dall'utente. Il programma:
 - genera un vettore somma cumulata partendo dal vettore generato. (N.B. l'elemento i-esimo del vettore è pari alla somma di tutti gli elementi da 0 a i), e lo mostra all'utente.
 - chiede all'utente un numero da cercare
 - darà come risposta se il numero è presente o meno nel vettore somma cumulata e quanti confronti sono stati effettuati, utilizzando la ricerca sequenziale e quella binaria.
 - ripete finché l'utente non inserisce un numero negativo

OPERAZIONI SUI VETTORI: PROBLEMI DI ORDINAMENTO



ORDINAMENTO DI UN VETTORE: PROBLEMA

- Tra le operazioni più diffuse in programmazione è l'ordinamento degli elementi di un vettore
- Definizione del problema:
 - Input:
 - una sequenza di n elementi a_1, a_2, \dots, a_n
 - (N.B. gli elementi della sequenza sono tipi ordinati, cioè si possono applicare gli operatori relazionali)
 - Output:
 - una permutazione della sequenza a'_1, a'_2, \dots, a'_n tale che $a'_1 \leq a'_2 \leq \dots \leq a'_n$ (ordinamento crescente)

ORDINAMENTO DI UN VETTORE

SELECTION SORT

- Un algoritmo molto semplice e intuitivo di ordinamento è il **selection sort**
 1. Si cerca l'elemento più piccolo(grande) del vettore
 2. Si scambia l'elemento più piccolo con il primo elemento del vettore
 3. Si ripetono i passi 1. e 2. sul sotto-vettore rimanente, finchè il sotto-vettore rimanente è nullo

SELECTION SORT

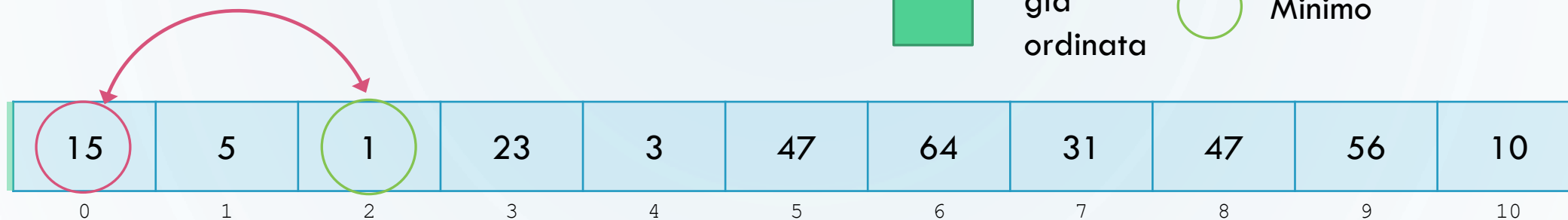


Parte
già
ordinata



Minimo

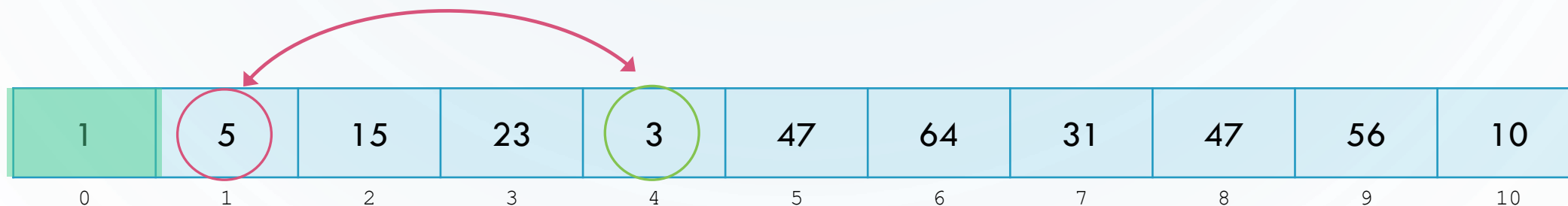
Passo 1



↑
P.e.
sottovettore

↑
Minimo

Passo 2



↑
P.e.
sottovettore

↑
Minimo

SELECTION SORT

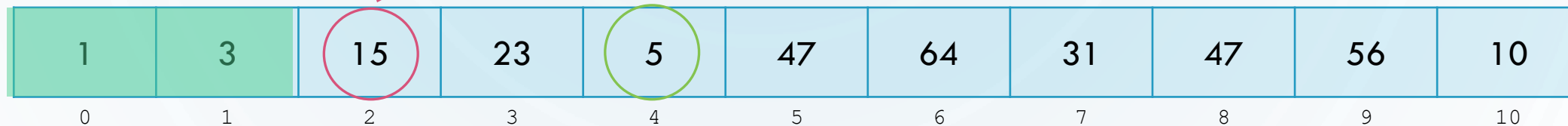


Parte
già
ordinata



Minimo

Passo3



↑
P.e.
sottovettore

↑
Minimo



SELECTION SORT



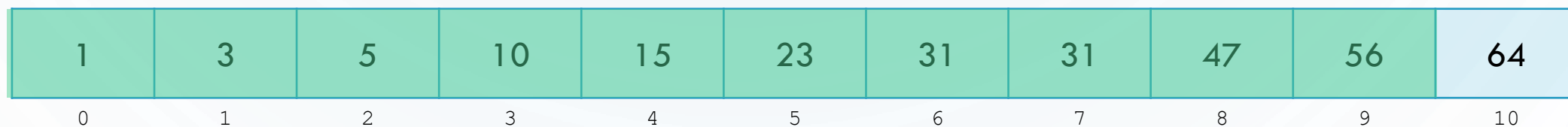
Parte
già
ordinata



Minimo



Passo n



↑
P.e.
sottovettore

COMPLESSITA' COMPUTAZIONALE: SELECTION SORT

- L'algoritmo dovrà al primo passo lavora su $n-1$ elementi (*selezione del minimo del sottovettore*) facendo $n-1$ confronti. La seconda volta agirà su un sottovettore più piccolo di $n-2$ elementi, e così via. Fino ad eseguire 1 solo confronto per l'ultimo passo.
- In totale si eseguono $n(n-1)/2$ confronti
- La complessità computazionale sarà una $O(n^2)$, indipendentemente dai valori dei dati di ingresso

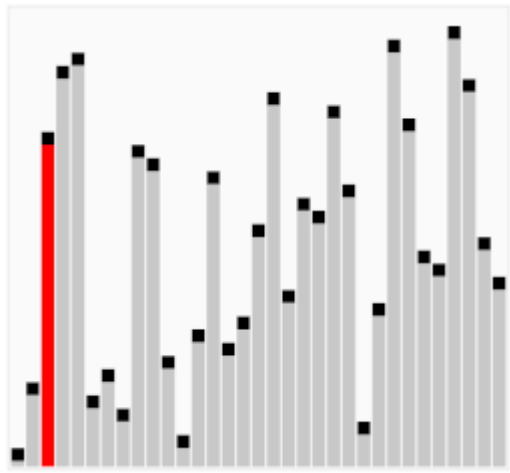
ESERCIZIO 1

- Realizzare un programma che permetta di generare un vettore in ingresso dal lato utente di N numeri interi positivi tra 0 e 100 pseudo-randomici. N dovrà essere minimo 20 e massimo 100 scelto dall'utente. Il programma:
 - Visualizza il vettore generato
 - Ordina il vettore in ordine crescente utilizzando il selection sort
 - chiede all'utente un numero da cercare
 - darà come risposta se il numero è presente o meno nel vettore ordinato utilizzando la ricerca binaria
 - ripete finché l'utente non inserisce un numero negativo
 - Si utilizzi la programmazione strutturata (funzioni)

ORDINAMENTO DI UN VETTORE

BUBBLE SORT

- L'algoritmo deve il suo nome al modo in cui gli elementi vengono ordinati in una lista: quelli più piccoli "risalgono" verso un'estremità della lista, mentre quelli più grandi "affondano" verso l'estremità opposta della lista, come le bolle in un bicchiere di champagne



Bubble sort in inseramento – fonte: Wikipedia



ORDINAMENTO DI UN VETTORE

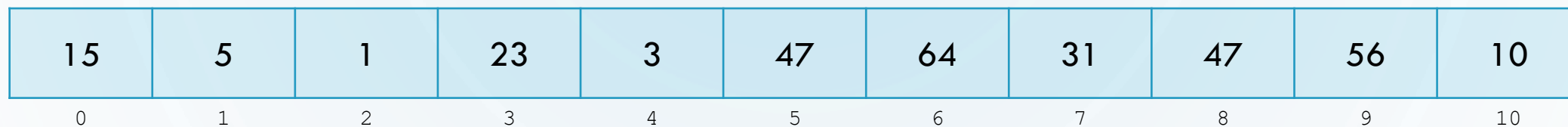
BUBBLE SORT

- Il bubble sort è un algoritmo iterativo. La singola iterazione dell'algoritmo prevede che gli elementi dell'array siano confrontati a due a due, procedendo in un verso stabilito e scambiati di posto se non in ordine:
 1. Partendo dal primo elemento, si confrontano due elementi successivi
 2. Si scambiano di posto se il primo è più grande del secondo (ordinamento crescente)
 3. Si ripetono i passi 1. e 2. su tutto il vettore finchè in un ciclo viene effettuato almeno uno scambio

BUBBLE SORT

 Elementi
confrontati

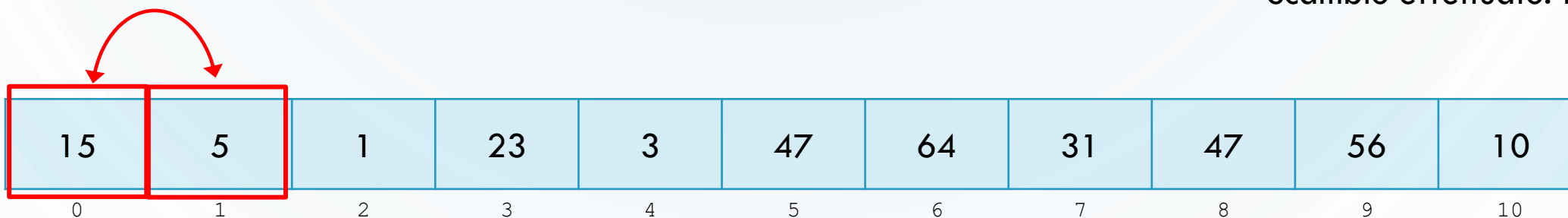
Passo0



↑
indice

Scambio effettuato: false

Passo1.1



↑
indice

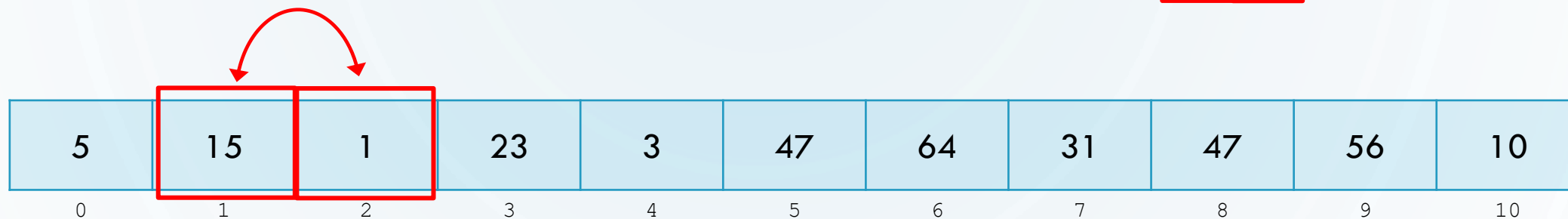
Scambio effettuato: true

BUBBLE SORT



Elementi
confrontati

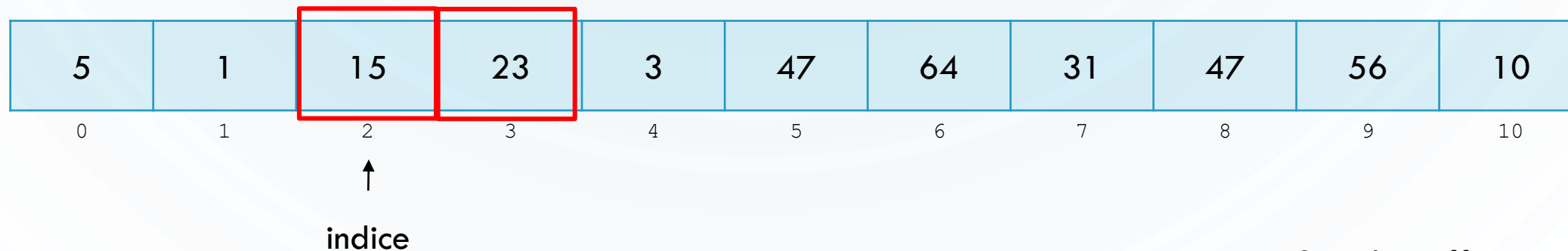
Passo 1.2



Scambio effettuato: true



Passo 1.3



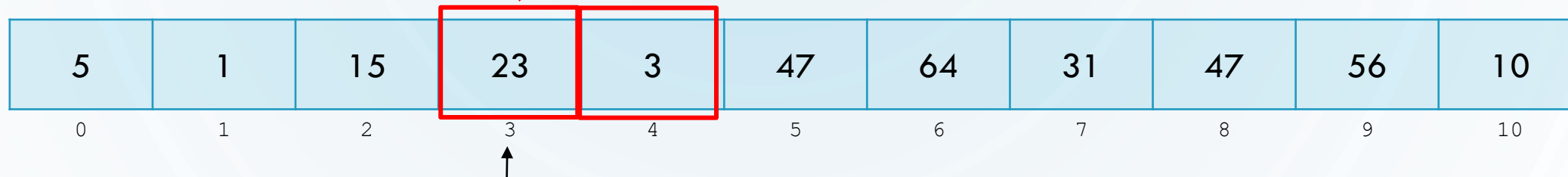
Scambio effettuato: true

BUBBLE SORT



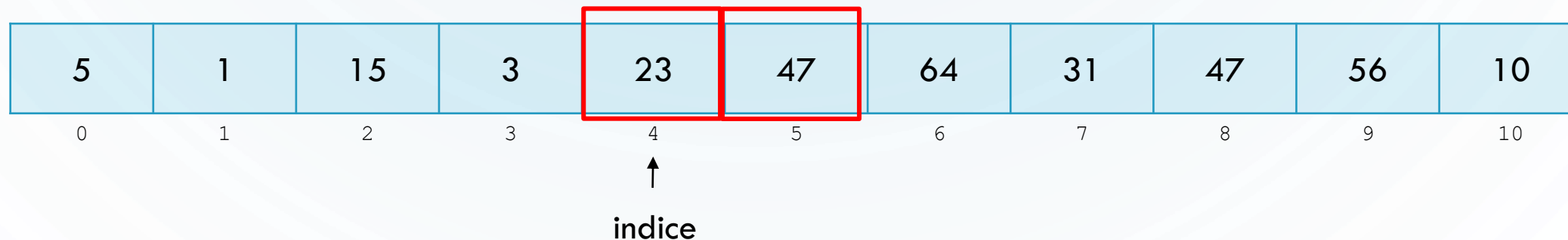
Elementi
confrontati

Passo 1.4



Scambio effettuato: true

Passo 1.5



Scambio effettuato: true

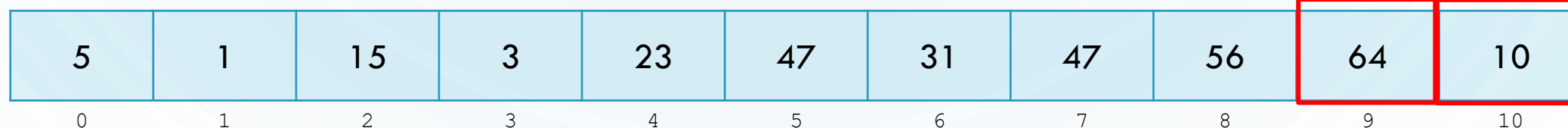
BUBBLE SORT



Elementi
confrontati



Passo 1.10

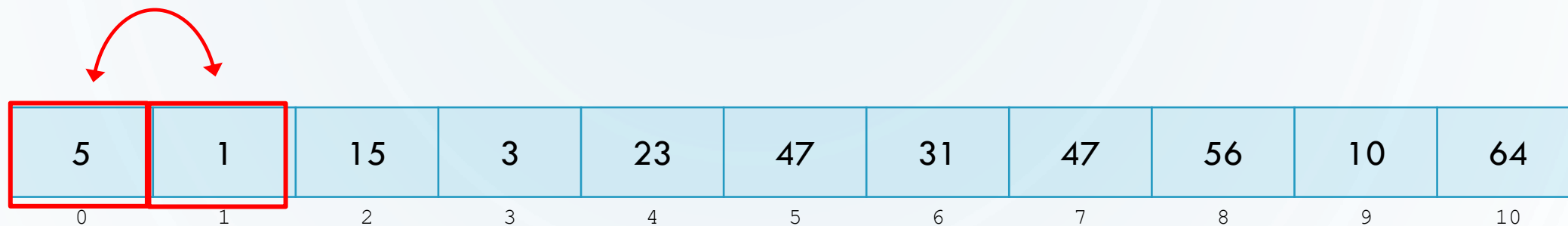


↑
indice

Scambio effettuato: true

BUBBLE SORT

Passo 2.0



Elementi
confrontati

Scambio effettuato: true



BUBBLE SORT

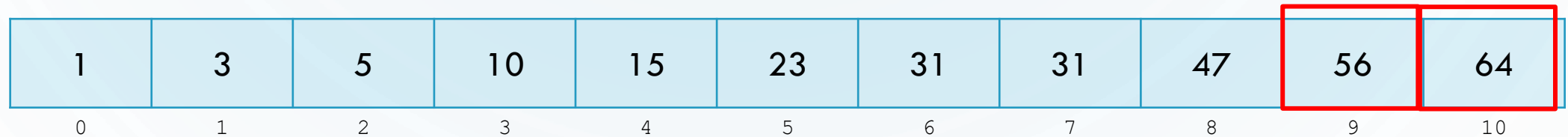


Elementi
confrontati

Si continua finchè in un ciclo non siano effettuati scambi



Passo M.10



↑
indice

Scambio effettuato: false

ORDINAMENTO DI UN VETTORE

BUBBLE SORT (MIGLIORATO)

- Una volta effettuato il primo ciclo, ci si ferma all'ultimo scambio avvenuto (dopo di esso il vettore risulta già ordinato)
 1. Partendo dal primo elemento, si confrontano due elementi successivi
 2. Si scambiano di posto se il primo è più grande del secondo (ordinamento crescente)
 3. Si ripetono i passi 1. e 2. su tutto il vettore fino all'ultimo scambio effettuato nell'iterazione precedente

N.B. la prima iterazione verrà fatta su tutto il vettore

COMPLESSITA' COMPUTAZIONALE: BUBBLE SORT

- Il Bubble sort è più efficiente rispetto al più semplice algoritmo di Ordinamento Ingenuo perché, invece di continuare ad eseguire sempre fino alla fine i due cicli annidati, si interrompe appena si accorge di non effettuare più scambi quando l'ordinamento è già completo. Esso ha comunque una complessità computazionale dell'ordine di $O(n^2)$

ESERCIZIO 2: BUBBLE SORT

- Realizzare un programma che permetta di generare un vettore in ingresso dal lato utente di N numeri interi positivi tra 0 e 100 pseudo-randomici. N dovrà essere minimo 20 e massimo 100 scelto dall'utente. Il programma:
 - Visualizza il vettore generato
 - Ordina il vettore in ordine crescente utilizzando il bubble sort
 - chiede all'utente un numero da cercare
 - darà come risposta se il numero è presente o meno nel vettore ordinato utilizzando la ricerca binaria
 - ripete finché l'utente non inserisce un numero negativo
 - Si utilizzi la programmazione strutturata (funzioni)

ESERCIZIO 3: BUBBLE SORT (MIGLIORATO)

- Realizzare un programma che permetta di generare un vettore in ingresso dal lato utente di N numeri interi positivi tra 0 e 100 pseudo-randomici. N dovrà essere minimo 20 e massimo 100 scelto dall'utente. Il programma:
 - Visualizza il vettore generato
 - Ordina il vettore in ordine crescente utilizzando il bubble sort migliorato
 - chiede all'utente un numero da cercare
 - darà come risposta se il numero è presente o meno nel vettore ordinato utilizzando la ricerca binaria
 - ripete finché l'utente non inserisce un numero negativo
 - Si utilizzi la programmazione strutturata (funzioni)

ESERCIZIO 4: CONFRONTO COMPLESSITÀ


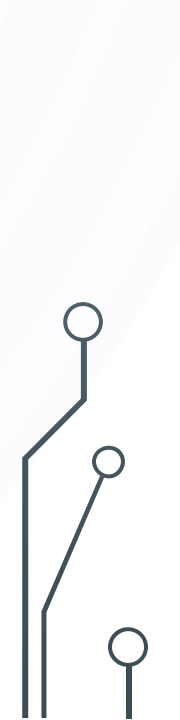
- Si vogliono realizzare i confronti della complessità computazionale dei tre algoritmi di ordinamento considerati (in ordine di numero di confronti effettuati e scambi effettuati)
- A tal proposito si modifichino le funzioni finora scritte per tener conto del passaggio di queste variabili aggiuntive
- (continua)

ESERCIZIO 4: CONFRONTO COMPLESSITÀ

-
- Realizzare un programma che permetta di generare randomicamente M vettori diversi (M scelto dall'utente), con grandezza N randomica tra 20 e 1000 di numeri reali tra -2 e 3 pseudo-randomici, e ordinarli con i tre diversi algoritmi
- Il programma visualizzerà la media di confronti e scambi effettuati dai tre algoritmi sia in termini assoluti che normalizzati a N_i^2 (N_i grandezza dell' i -esimo vettore generato)



ESERCIZIO 5: ORDINAMENTO CON RICORSIONE

- Scrivere gli algoritmi di ordinamento studiati utilizzando funzioni ricorsive
- 
- 

ESERCIZIO 6: ORDINAMENTO VETTORI DI RECORD

- Si vuole realizzare un programma che permetta di ordinare in ordine alfabetico una classe di studenti {Nome,Cognome,età} rispetto al loro cognome. A tal proposito:
 - si chiede all'utente di inserire in ordine sparso gli studenti (per ogni studente si richiede se esiste un ulteriore studente o meno)
 - Si ordinano gli studenti con uno degli algoritmi di ordinamento conosciuti
 - Si visualizzano gli studenti ordinati (uno per ogni riga):
 - 1) Alighieri Dante - 11 anni
 - 2) Boccaccio Giovanni - 13 anni
 - 3) Manzoni Alessandro - 14 anni

OPERAZIONI SUI VETTORI: PROBLEMI DI ORDINAMENTO SENZA SCAMBI



ORDINAMENTO DI UN VETTORE: PROBLEMA

- Gli algoritmi di ordinamento studiati ordinano gli elementi di un vettore scambiandoli di posto nel vettore stesso: per questo motivo sono detti «ordinamenti in loco» (in-place)
- Seppur efficienti, a volte l'operazione di scambio (fisico) degli elementi potrebbe essere onerosa perché si ha a che fare con strutture dati molto grandi o vettori di grandi dimensioni, in cui l'accesso in lettura è sensibilmente molto più veloce di quello in scrittura.
- Per questo tipo di problemi si può ricorrere ad ordinare i soli indici agli elementi

ORDINAMENTO DI UN VETTORE

SELECTION SORT (SUGLI INDICI)

- **Selection sort sul vettore degli indici:**
 1. Si crea un vettore degli indici da 1 a N (0 a N-1 in C)
 2. Si cerca l'elemento più piccolo(grande) del vettore
 3. Si scambia l'indice dell'elemento più piccolo con il primo elemento del vettore indici
 4. Si ripetono i passi 1. e 2. sul sotto-vettore degli indici rimanente, finchè il sotto-vettore rimanente è nullo

SELECTION SORT (SU INDICI)



Parte
già
ordinata



Minimo

Passo 1

15	5	1	23	3	47	64	31	47	56	10
0	1	2	3	4	5	6	7	8	9	10

0	1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	---	----

↑
P.e.
sottovettore

↑
Minimo

SELECTION SORT (SU INDICI)



Parte
già
ordinata



Minimo

Passo2

15	5	1	23	3	47	64	31	47	56	10
0	1	2	3	4	5	6	7	8	9	10

2	1	0	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	---	----

P.e.

sottovettore

Minimo

SELECTION SORT (SU INDICI)



Parte
già
ordinata



Minimo

Passo n

15	5	1	23	3	47	64	31	47	56	10
0	1	2	3	4	5	6	7	8	9	10


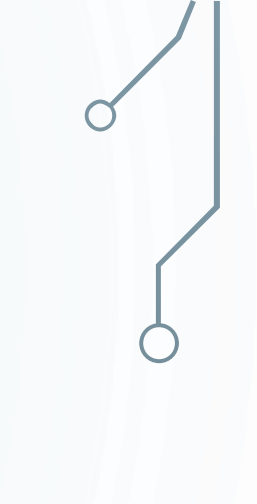
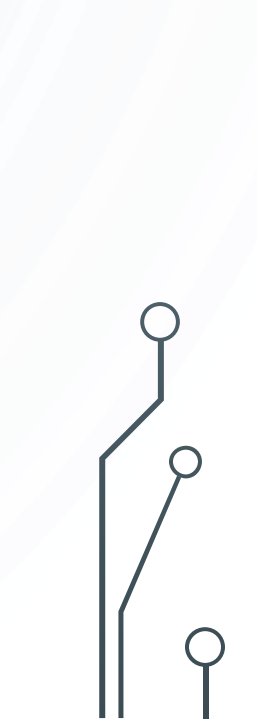
2	4	1	10	0	3	7	5	8	9	6
---	---	---	----	---	---	---	---	---	---	---



P.e.
sottovettore



ESERCIZIO 7: ORDINAMENTO VETTORI DI RECORD

- Riscrivere l'esercizio 6 utilizzando l'ordinamento sugli indici per la visualizzazione in ordine alfabetico
- 
- 
- 

DOMANDE, DUBBI, PERPLESSITÀ

