UNIVERSITÀ
DI TRENTO

Dipartimento di Ingegneria e
Scienza dell'Informazione
DISI - Trento

Programmazione 1

13 - Esercitazione

Giovanni De Toni
giovanni.detoni@unitn.it

Attenzione

La presente esercitazione verrà trasmessa via Zoom. Essa verrà anche registrata e successivamente messa a disposizione degli studenti dell'Università degli Studi di Trento. Per gli utenti connessi attraverso Zoom, in caso non desideriate per qualunque motivo essere registrati, siete pregati di effettuare la disconnessione ora. La lezione sarà comunque visionabile in modo asincrono.

Anno Accademico 2021/2022

00 - Torre di Hanoi

Scrivere un programma che risolva la Torre di Hanoi tramite procedura ricorsiva.

A **B** **C**

00 - Torre di Hanoi

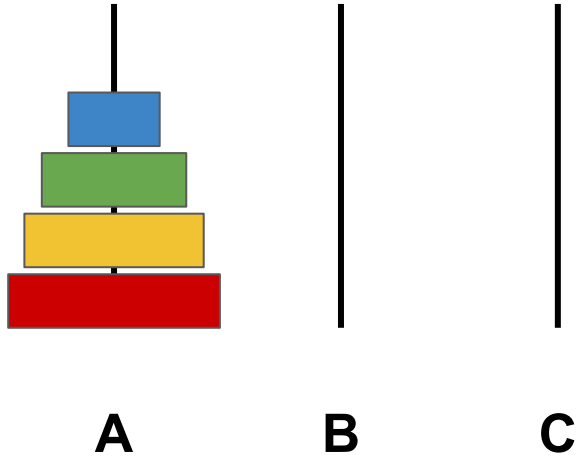
Scrivere un programma che risolva la Torre di Hanoi tramite procedura ricorsiva.

```
FUNZIONE MoveTower(disk, source, dest, spare):
IF disk == 0, THEN:
    move disk from source to dest
ELSE:
    MoveTower(disk - 1, source, spare, dest)   // Step 1
    move disk from source to dest              // Step 2
    MoveTower(disk - 1, spare, dest, source)   // Step 3
END IF
```

00 - Torre di Hanoi

Scrivere un programma che risolva la Torre di Hanoi tramite procedura ricorsiva.

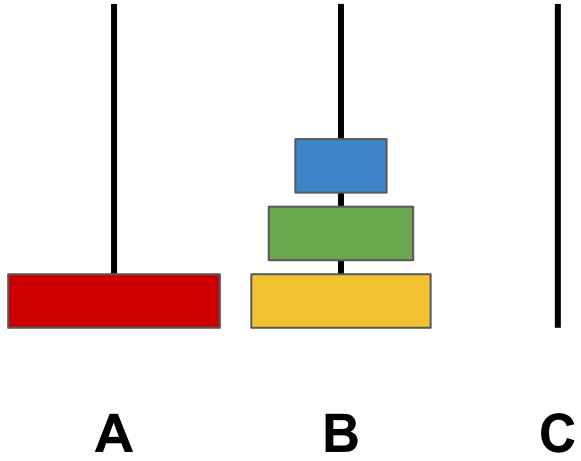
```
FUNZIONE MoveTower(disk, source, dest, spare):
IF disk == 0, THEN:
    move disk from source to dest
ELSE:
    MoveTower(disk - 1, source, spare, dest)   // Step 1
    move disk from source to dest              // Step 2
    MoveTower(disk - 1, spare, dest, source)   // Step 3
END IF
```



00 - Torre di Hanoi

Scrivere un programma che risolva la Torre di Hanoi tramite procedura ricorsiva.

```
FUNZIONE MoveTower(disk, source, dest, spare):
IF disk == 0, THEN:
    move disk from source to dest
ELSE:
    MoveTower(disk - 1, source, spare, dest)   // Step 1
    move disk from source to dest              // Step 2
    MoveTower(disk - 1, spare, dest, source)   // Step 3
END IF
```

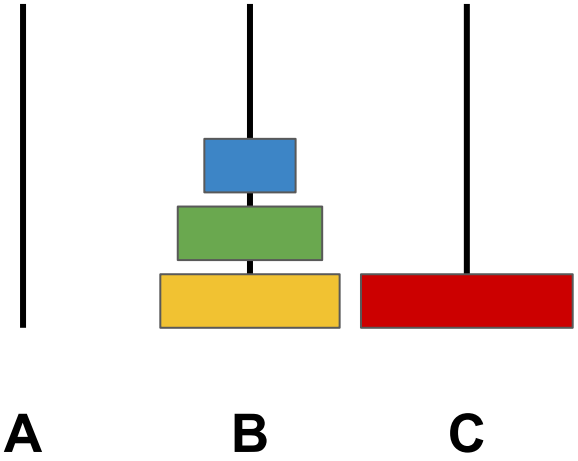


Step 1: Muovi i primi n-1 dischi sul perno B (spare)

00 - Torre di Hanoi

Scrivere un programma che risolva la Torre di Hanoi tramite procedura ricorsiva.

```
FUNZIONE MoveTower(disk, source, dest, spare):
IF disk == 0, THEN:
    move disk from source to dest
ELSE:
    MoveTower(disk - 1, source, spare, dest)    // Step 1
    move disk from source to dest               // Step 2
    MoveTower(disk - 1, spare, dest, source)    // Step 3
END IF
```

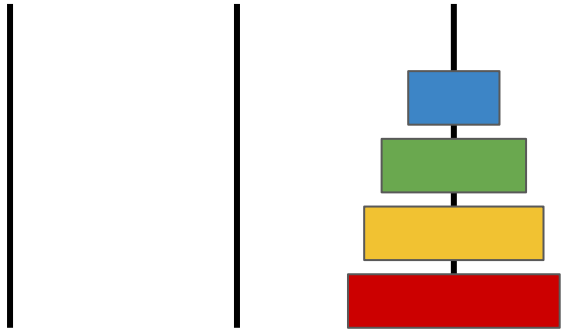


Step 2: Muovi il disco n dischi sul perno C (dest)

00 - Torre di Hanoi

Scrivere un programma che risolva la Torre di Hanoi tramite procedura ricorsiva.

```
FUNZIONE MoveTower(disk, source, dest, spare):
IF disk == 0, THEN:
    move disk from source to dest
ELSE:
    MoveTower(disk - 1, source, spare, dest)    // Step 1
    move disk from source to dest               // Step 2
    MoveTower(disk - 1, spare, dest, source)    // Step 3
END IF
```



Step 3: Muovi i n-1 dischi sul perno C (dest)

00 - Torre di Hanoi

Scrivere un programma che risolva la Torre di Hanoi tramite procedura ricorsiva.

```
FUNZIONE MoveTower(disk, source, dest, spare):
IF disk == 0, THEN:
    move disk from source to dest
ELSE:
    MoveTower(disk - 1, source, spare, dest)   // Step 1
    move disk from source to dest              // Step 2
    MoveTower(disk - 1, spare, dest, source)   // Step 3
END IF
```

$$\text{minimo numero di mosse} = 2^n - 1$$

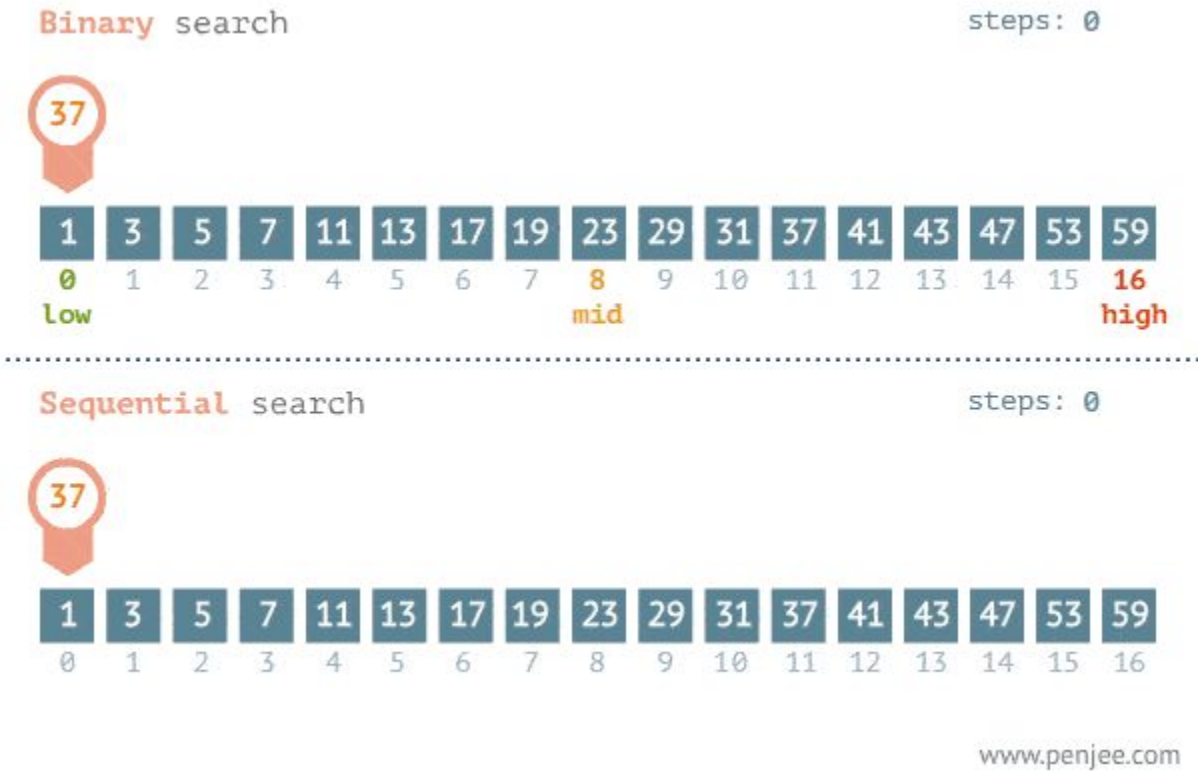
00 - Ordering

Problema: ricerca di un elemento in un array di N elementi

Se l'array **non** è ordinato => ricerca sequenziale (`O(n)`)

Se l'array è ordinato => ricerca binaria (`O(log(n))`)

00 - Ordering


Binary search
steps: 0
37
1 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
low
mid
high
Sequential search
steps: 0
37
1 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
www.penjee.com

00 - Ordering

Ordinamento

Algoritmi di Ordinamento:

- Bubble Sort
- Merge Sort
- Quicksort
- Heapsort
- Selection Sort
- Shellsort
- Tree Sort
- Bucket Sort
- Pigeonhole Sort
- Spaghetti Sort
- Random Sort
- ...

00 - Ordering

Avere array ordinati garantisce efficienza in:

- Ricerca di un elemento in un array
- Merge di due array
- Inserimento ordinato
- ...

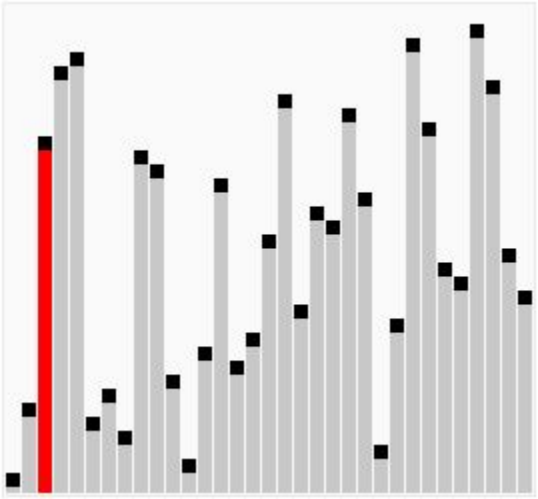
01 - Sequential Search

Scrivere un programma che dichiari e poi inizializzi (con valori random) un array di 10 interi (da 0 a 9) e lo stampi a video.

Chiedere poi in input all'utente un intero da 0 a 9 ed implementare la ricerca sequenziale per trovare la posizione (indice) della prima occorrenza del numero in input.

02 - Bubble Sort

Scrivere un programma che dichiari e poi inizializzi (con valori random) un array di 10 interi (da 0 a 9) e lo stampi a video. Ordinare poi l'array implementando l'algoritmo Bubble Sort.



03 - Duplicati

Scrivere un programma che, dato un array di 10 interi (da 0 a 9) ordinato, rimuova eventuali interi duplicati e li sostituisca con -1.

```
[3, 0, 4, 0, 3, 7, 5] => [0, -1, 3, -1, 4, 5, 7]
```

04 - Binary Search

Scrivere un programma che, dato un array di 10 interi (da 0 a 9) ordinato, chieda in input all'utente un intero da 0 a 9 ed implementi la ricerca binaria tramite funzione ricorsiva per trovare la posizione (indice) della prima occorrenza del numero in input.

05 - Merge

Scrivere un programma che, dati due array ordinati, ne faccia il merge in un terzo array.