

**Luiss**

Libera Università Internazionale  
degli Studi Sociali Guido Carli

# **Corso di preparazione per la selezione territoriale delle Olimpiadi di Informatica**

## **Lezione 2. Algoritmi (e Strutture Dati)**

**Giuseppe F. Italiano**

**6 marzo 2023**


**LUISS**



# Lezioni

Puoi accedere alle lezioni tramite YouTube o Webex:

 **YouTube** (preferibile se hai problemi di connessione)

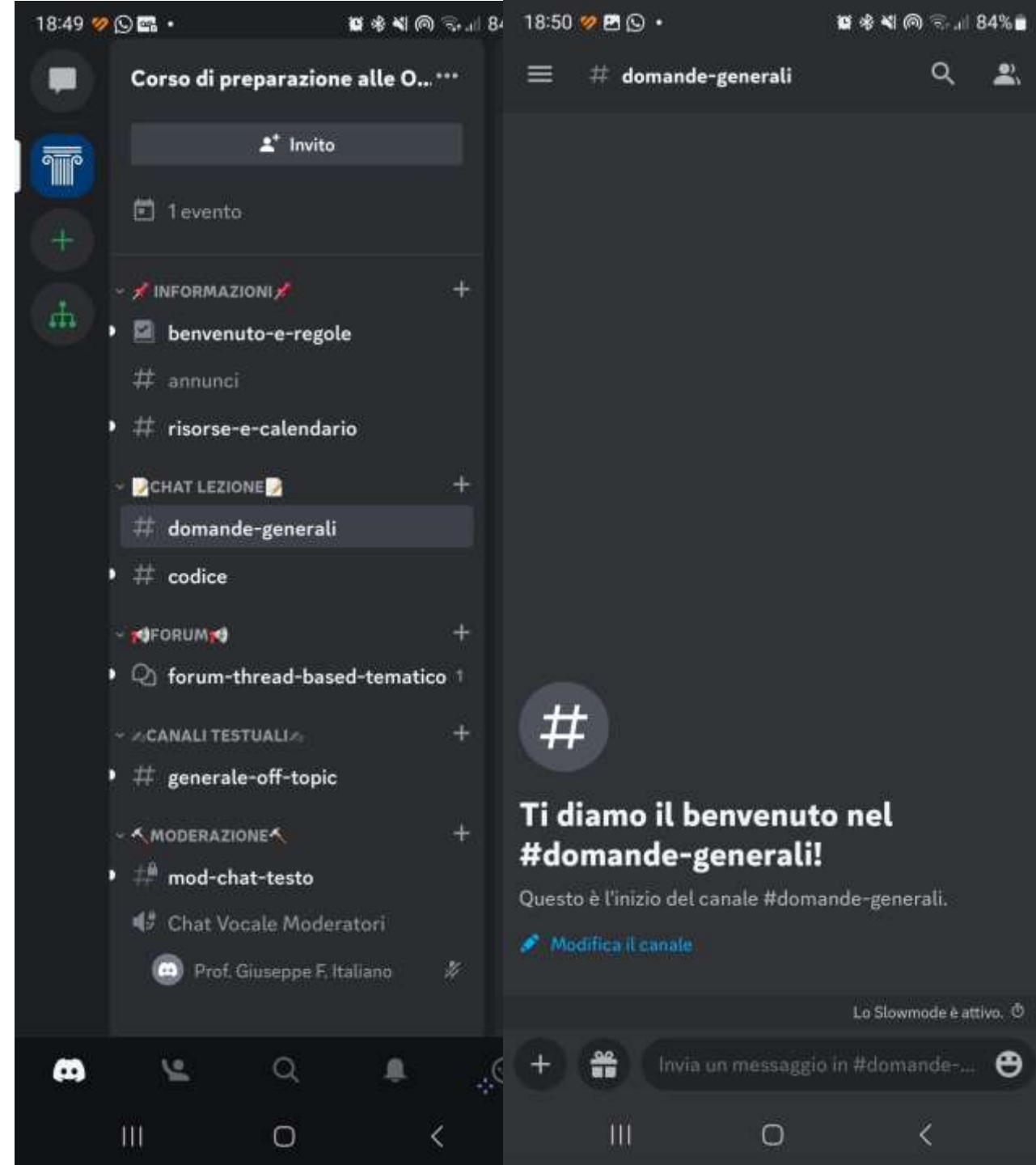
 **webex** (puoi diventare *Chad (Stacy)* per *un giorno* condividendo la tua soluzione ai problemi che consideriamo di volta in volta!)



# Reminder: Chat su Discord

<https://discord.gg/WFKeXDbS>

Non inquinare i canali! Contribuisci a mantenerli puliti e non postare contenuti impropri. Grazie!



# Programma di oggi

1. Soluzione esercizi della scorsa settimana
2. Introduzione (informale) agli algoritmi
3. Un problema dalle Territoriali (2017): Scommessa
4. (Scorsa settimana) Introduzione (informale) alle strutture dati
5. Un problema dalle OIS (2020): Ransomware
6. Compiti da fare a casa (per la prossima volta)

# **Soluzioni degli esercizi della scorsa settimana**




# Compiti a casa della scorsa settimana

- Festa estiva (festaestiva): [festa estiva](#)
- Soste in autostrada (autogrill): [autogrill](#)
- Tieni aggiornato il catalogo (catalogo): [catalogo](#)
- Musical fight (trap): [musical fight](#)
- Kill those bugs (blindpunch): [bugs](#)

# Tutti vogliamo diventare chad per un giorno



Se sei collegato/a tramite Webex alza la mano (pulsante  vicino al tuo nome nella lista partecipanti) per presentare la tua soluzione ai problemi della scorsa settimana

# Compiti a casa della scorsa settimana

- Festa estiva (festaestiva): [festa estiva](#)



# Compiti a casa della scorsa settimana

- Soste in autostrada (autogrill): [autogrill](#)

## Compiti a casa della scorsa settimana

- Tieni aggiornato il catalogo (catalogo): [catalogo](#)

# Compiti a casa della scorsa settimana

- Musical fight (trap): [musical fight](#)

# Compiti a casa della scorsa settimana

- Kill those bugs (blindpunch): [bugs](#)

# Introduzione (informale) agli algoritmi





Per preparare il tiramisù cominciate dalle uova (freschissime): quindi separate accuratamente gli albumi dai tuorli (1), ricordando che per montare bene gli albumi non dovranno presentare alcuna traccia di tuorlo. Poi montate i tuorli con le fruste elettriche, versando solo metà dose di zucchero (2). Non appena il composto sarà diventato chiaro e spumoso (3),



e con le fruste ancora in funzione, potrete aggiungere il mascarpone, poco alla volta (4). Incorporato tutto il formaggio avrete ottenuto una crema densa e compatta; tenetela da parte (5). Pulite molto bene le fruste e passate a montare gli albumi versando il restante zucchero un po' alla volta (6).

# Cos'è un algoritmo? (informalmente)

Insieme di istruzioni

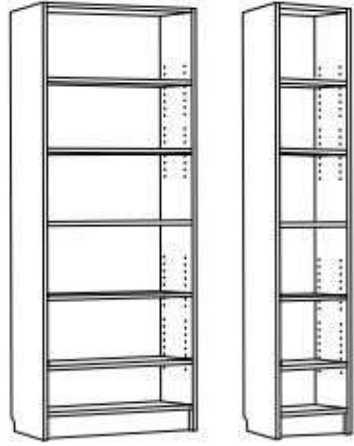
- Definite passo per passo,
- In modo da poter essere eseguite meccanicamente e
- Tali da produrre un determinato risultato

# Algoritmi

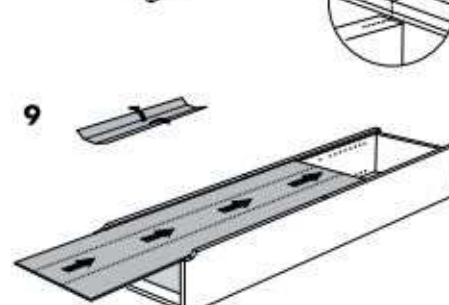
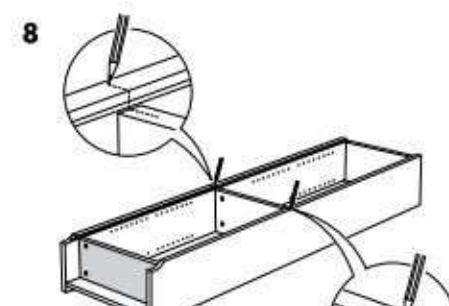
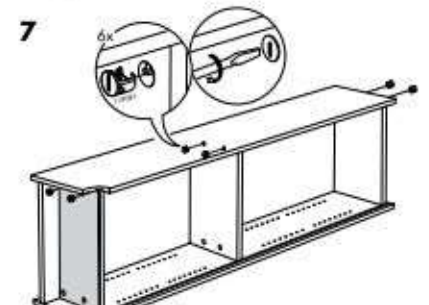
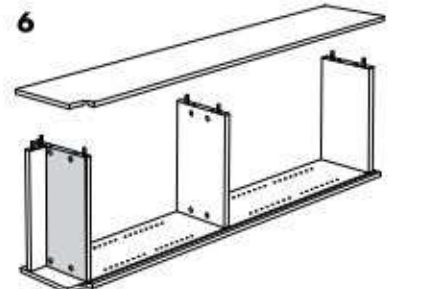
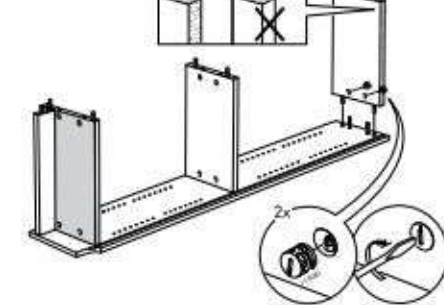
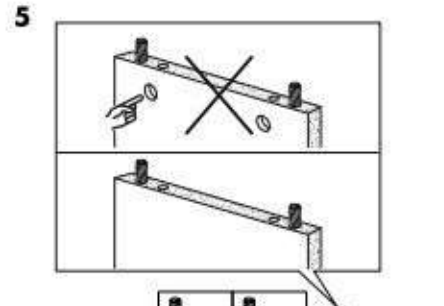
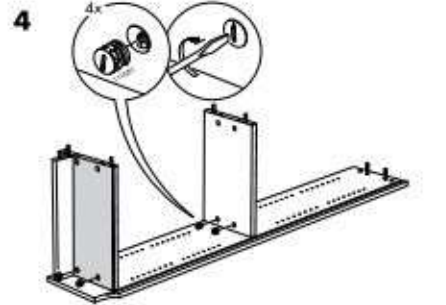
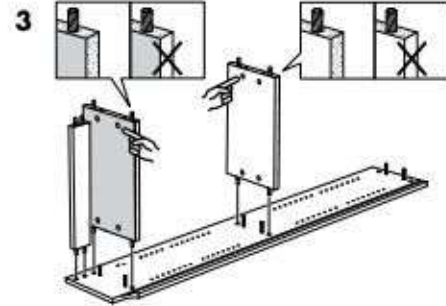
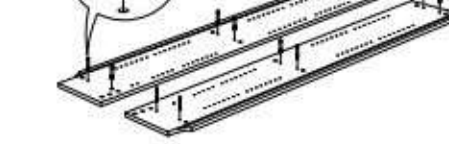
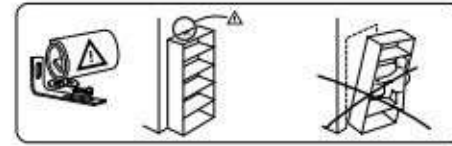
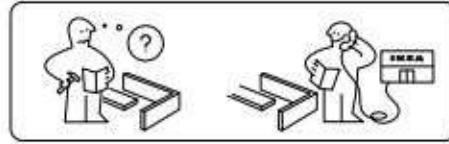
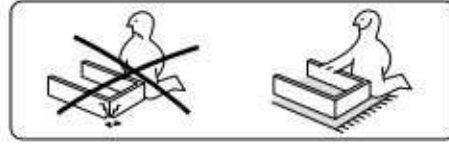
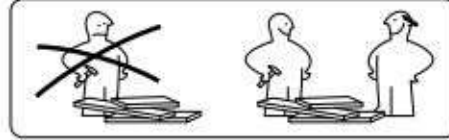
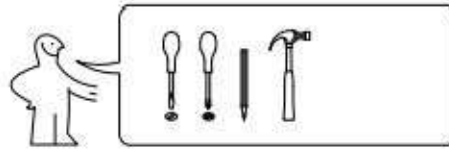
$$\begin{array}{r} 17 \times \\ 12 \\ \hline 34 \\ 17 \\ \hline 204 \end{array}$$



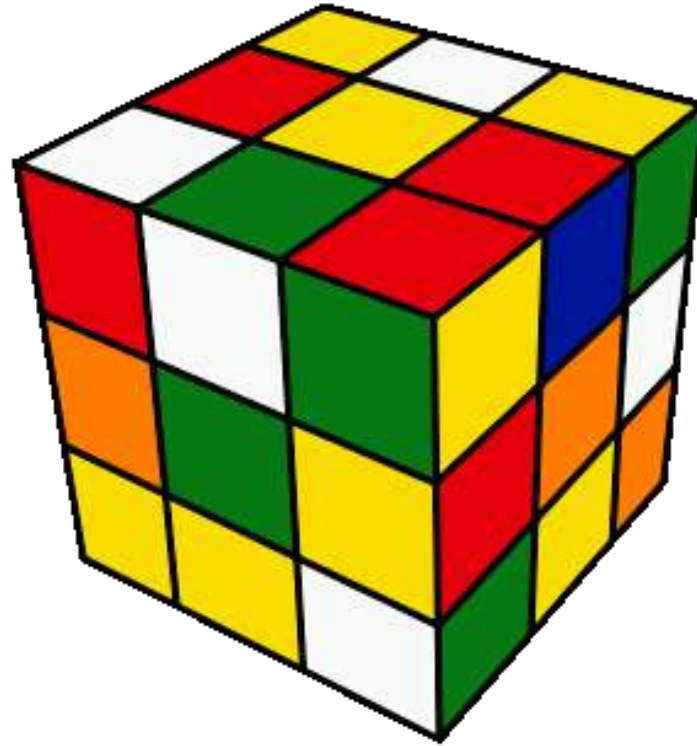
# BILLY



**IKEA**  
Design and Quality  
Since 1946



# Algoritmi



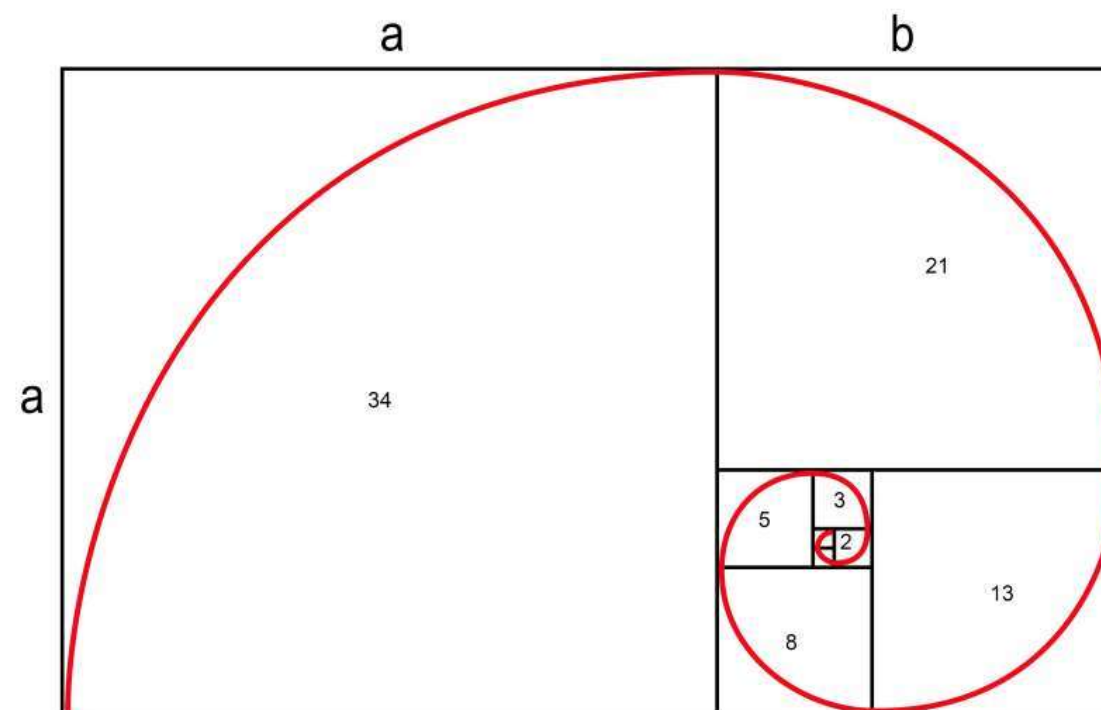
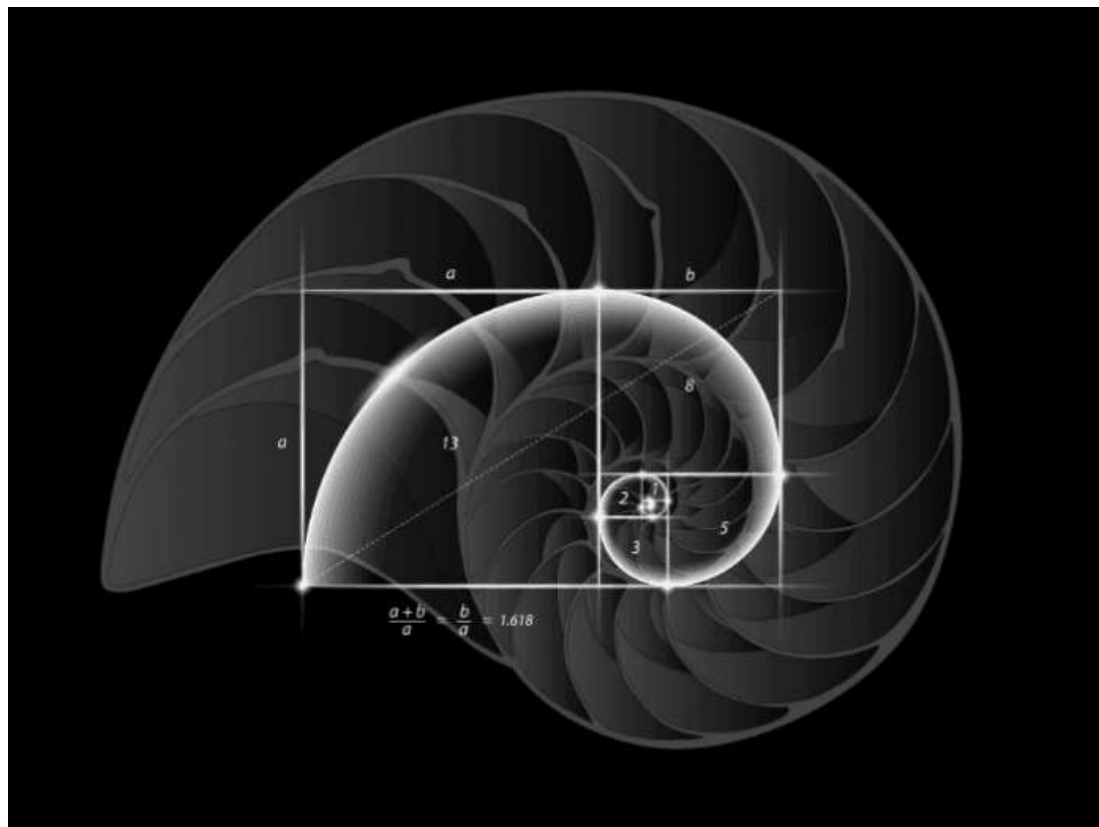
# Numeri di Fibonacci

- Definiti dalla relazione di ricorrenza:

$$F_n = \begin{cases} F_{n-1} + F_{n-2} & \text{se } n \geq 3 \\ 1 & \text{se } n = 1, 2 \end{cases}$$

- Problema (algoritmico): come calcoliamo  $F_n$  ?

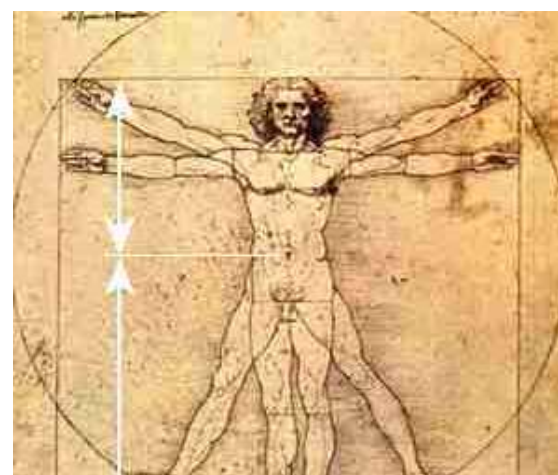
# Spirale di Fibonacci

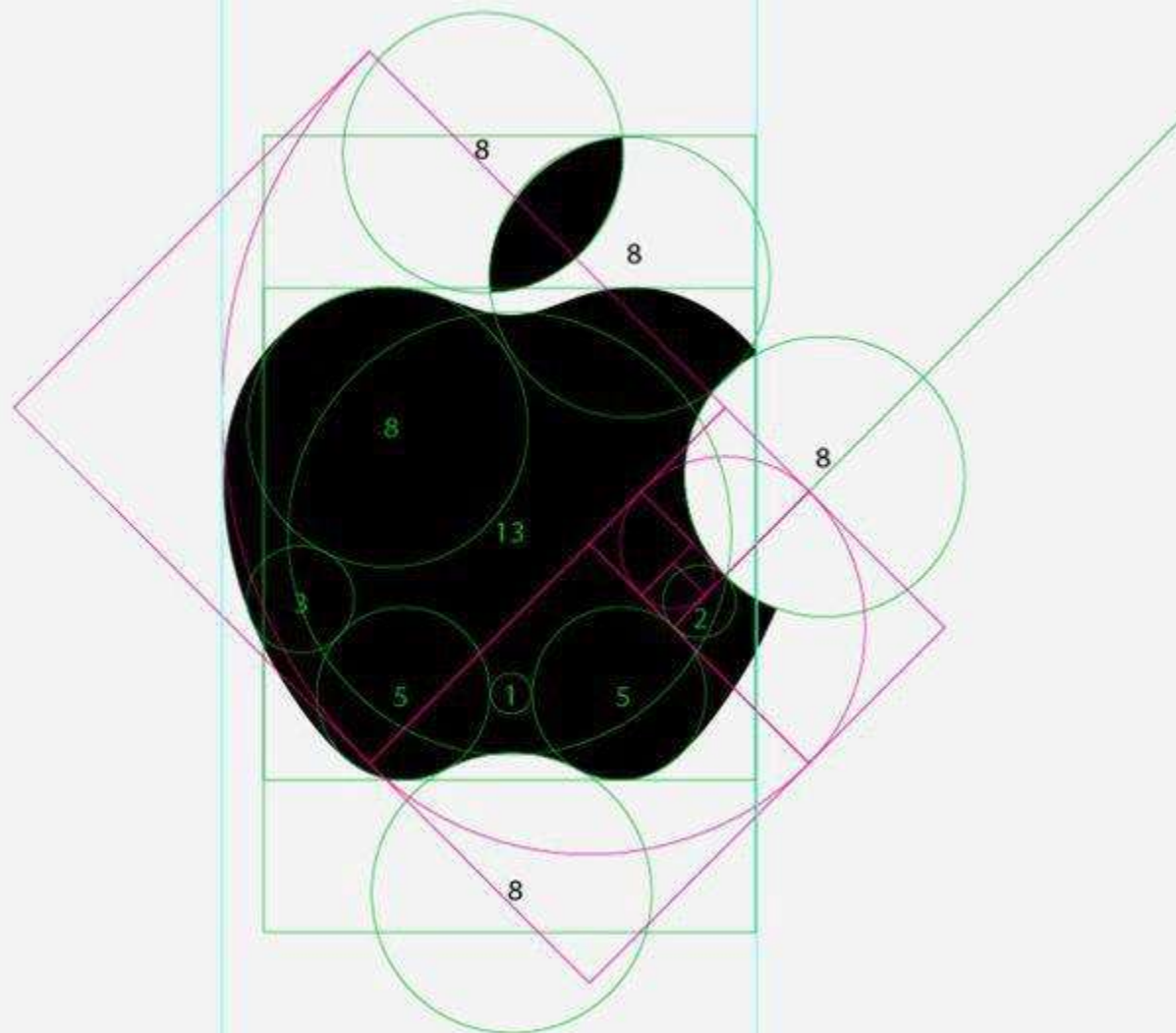
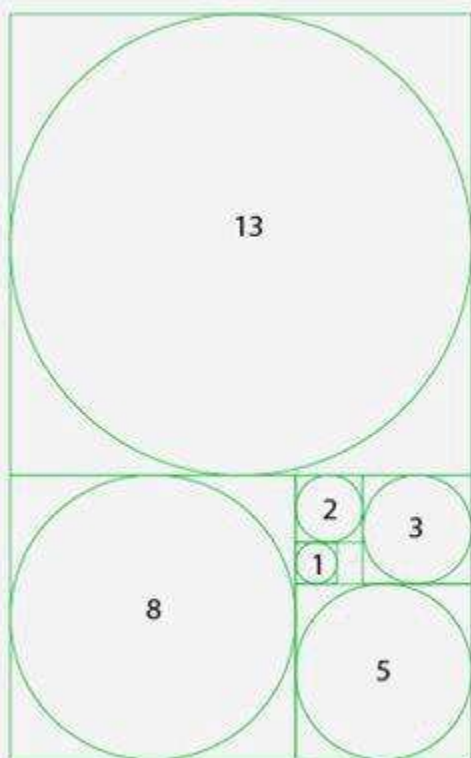


$$\Phi = \frac{a+b}{a} = \frac{a}{b} = 1,618$$

$$F_n = F_{n-1} + F_{n-2}$$







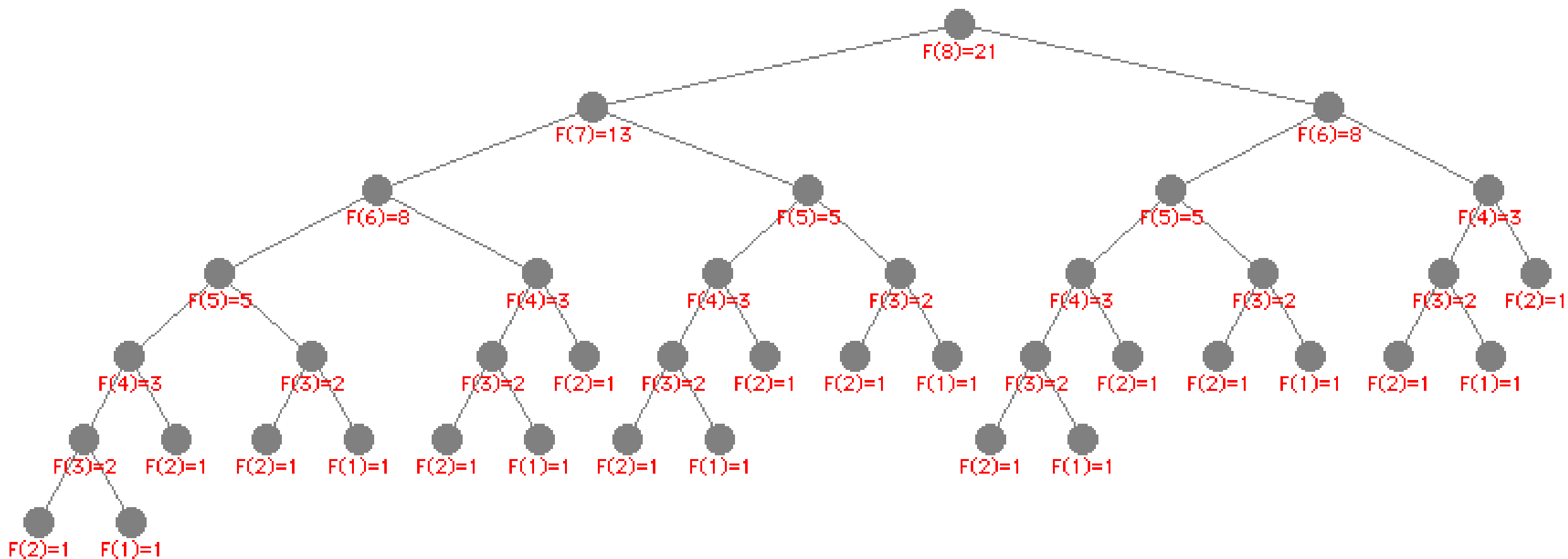
# Primo algoritmo (doppiamente ricorsivo)

Potremmo utilizzare direttamente la definizione (ricorsiva) :

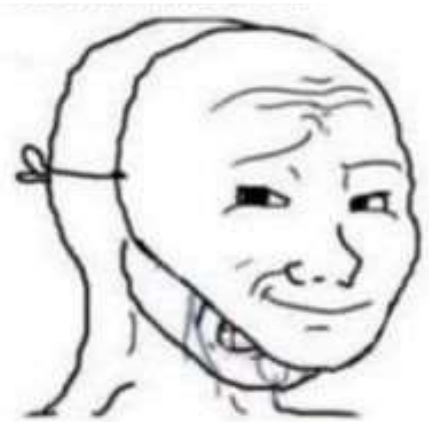
```
algoritmo fibonacci(intero  $n$ )  $\rightarrow$  intero  
  if ( $n \leq 2$ ) then return 1  
  else return fibonacci( $n-1$ ) + fibonacci( $n-2$ )
```

E' una soluzione accettabile?

# Albero delle chiamate ricorsive







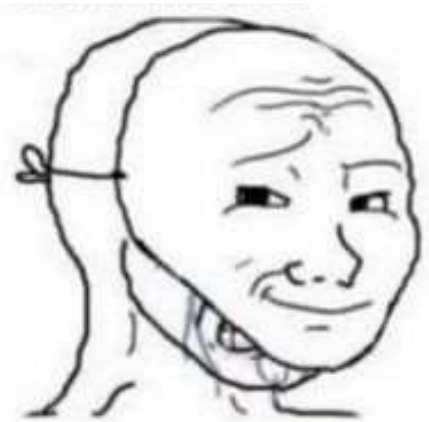
Possiamo fare di meglio?

# Sì, possiamo fare di meglio!

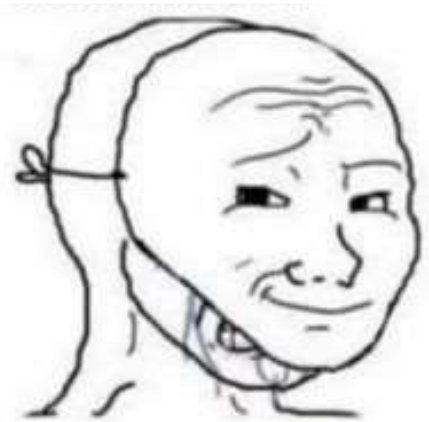
1	1	2	3	5	8	13	21	34	55	89	144
1	2	3	4	5	6	7	8	9	10	11	12

# Secondo algoritmo (iterativo)

```
algoritmo fibonacci(intero n)  $\rightarrow$  intero  
  sia Fib un array di n interi  
  Fib[1]  $\leftarrow$  Fib[2]  $\leftarrow$  1  
  for i = 3 to n do  
    Fib[i]  $\leftarrow$  Fib[i-1] + Fib[i-2]  
  return Fib[n]
```



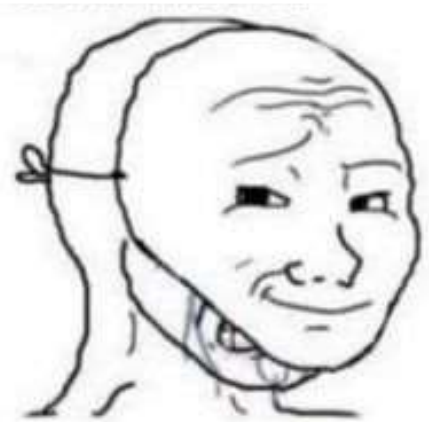
Possiamo fare di meglio?



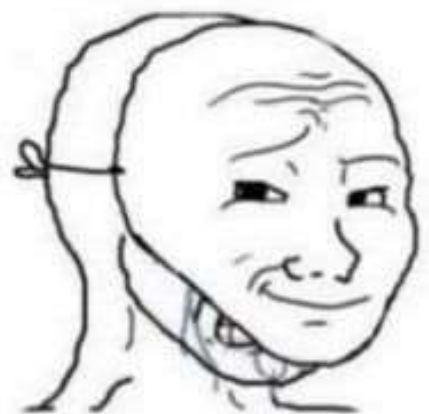
Possiamo fare di meglio?



Sì, ma non c'è tempo! Dobbiamo fare tante cose più importanti!



Possiamo fare di meglio?



Faster Fibonacci?



Sì, ma non c'è tempo! Dobbiamo fare tante cose più importanti!

**Indian man on YouTube**



I will teach you my son

# **Un problema dalle Territoriali: Sport intellettuali (scommessa)**

<https://training.olinfo.it/#/task/scommessa/statement>





## Sport intellettuali (scommessa)

Difficoltà: 2

Romeo è un grande appassionato di sport intellettuali, e adora ritrovarsi con gli amici per seguire le competizioni internazionali più avvincenti di questo tipo. Di recente, il gruppo di amici si è appassionato a uno sport molto particolare. In questo gioco, un mazzo di carte numerate da 0 a  $N-1$  (dove  $N$  è dispari) viene prima mescolato, e poi le carte vengono affiancate in linea retta sul tavolo. Ai telespettatori, per aumentare la suspense, vengono mostrati i numeri delle carte  $C_0, C_1, \dots, C_i, \dots, C_{N-1}$  nell'ordine così ottenuto. A questo punto i giocatori<sup>1</sup> possono scoprire due carte disposte consecutivamente sul tavolo, e prenderle nel solo caso in cui queste due carte *abbiano somma dispari*. Se queste carte vengono prese, le altre vengono aggiustate quanto basta per riempire il buco lasciato libero. Il gioco prosegue quindi a questo modo finché nessun giocatore può più prendere carte.

Romeo e i suoi amici, per sentirsi più partecipi, hanno oggi deciso di fare un “gioco nel gioco”: all'inizio della partita, scommettono su quali carte pensano rimarranno sul tavolo una volta finita la partita. Aiuta Romeo, determinando quali carte *potrebbero* rimanere sul tavolo alla fine del gioco!



📖 Una carta *potrebbe* rimanere sul tavolo a fine gioco, se esiste una sequenza di mosse (rimozioni di coppie di carte consecutive con somma dispari) tale per cui dopo di esse nessuna altra mossa è possibile (il gioco è finito) e la carta suddetta è ancora sul tavolo.

## Dati di input

Il file `input.txt` è composto da 2 righe, contenenti:

- Riga 1: l'unico intero  $N$ .
- Riga 2: gli  $N$  interi  $C_i$  separati da spazio, nell'ordine in cui sono disposti sul tavolo.

## Dati di output

Il file `output.txt` deve essere composto da due righe, contenenti:

- Riga 1: il numero di diverse carte  $K$  che *potrebbero* rimanere sul tavolo a fine partita.
- Riga 2: i  $K$  interi che identificano le carte che *potrebbero* rimanere sul tavolo a fine partita.

## Assunzioni

- $1 \leq N \leq 100$ .
- $N$  è sempre un numero dispari.
- $0 \leq C_i \leq N - 1$  per ogni  $i = 0 \dots N - 1$ .
- Ogni numero tra 0 e  $N - 1$  compare esattamente una volta nella sequenza dei  $C_i$ .

## Esempi di input/output

input.txt	output.txt
3 1 2 0	1 0
11 1 0 2 6 4 5 3 9 8 10 7	2 2 8

## Spiegazione

Nel primo caso di esempio, l'unica mossa possibile è eliminare le carte 1 e 2 per cui rimane sul tavolo necessariamente la carta 0.

Nel secondo caso di esempio sono invece possibili diverse sequenze di mosse. Una delle sequenze che lasciano la carta 2 è la seguente:

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

Una delle sequenze di mosse che lasciano la carta 8 è la seguente:

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

# Introduzione (informale) alle strutture dati



# Strutture Dati

“Struttura” per “organizzare” dati  
Caratterizzata da operazioni

# Strutture Dati

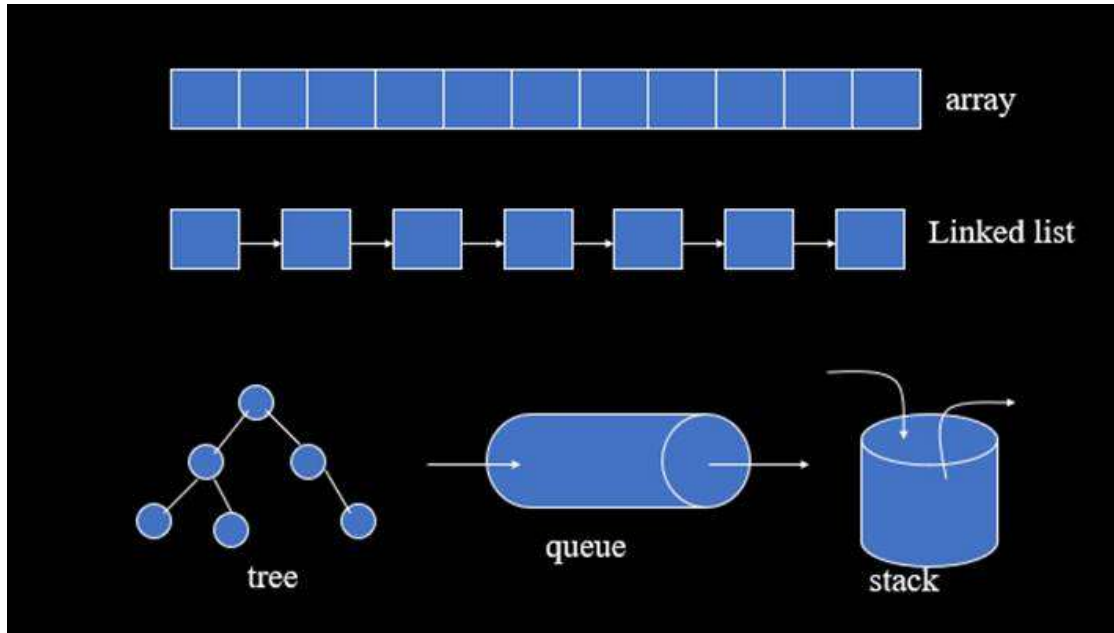


“Struttura” per “organizzare” uova

Caratterizzata da operazioni:

- Quante uova? (size)
- E' vuoto? (empty)
- Prendi un uovo (erase)
- Aggiungi un uovo (insert)
- ...

# Strutture Dati



“Struttura” per “organizzare” dati così che possono essere utilizzati più efficientemente

Caratterizzata da operazioni:

- Quanti elementi? (size)
- E' vuota? (empty)
- Prendi un elemento (erase)
- Aggiungi un elemento (insert)
- ...

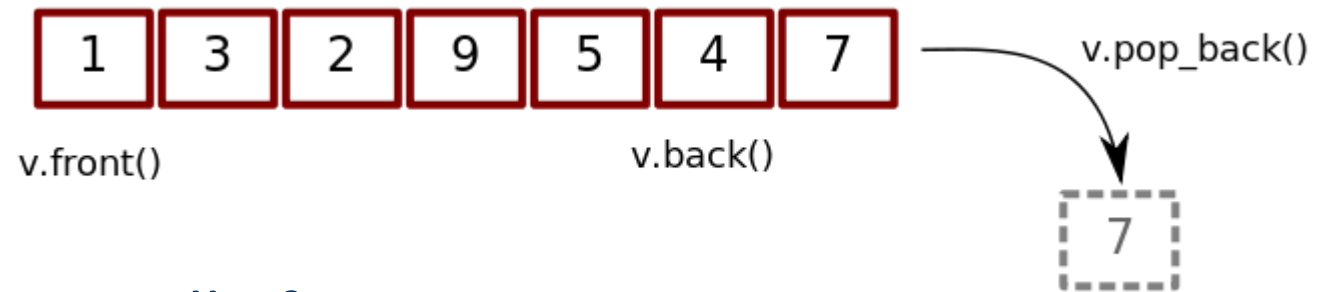
# **std::vector** (<http://www.cplusplus.com/reference/vector/vector/>)

Iterator: *begin, end, ...*

Access: *operator[], front, back, ...*

Modifier:

- ***insert*** : inserisce un elemento
- ***push\_back*** : inserisce un elemento alla fine
- ***erase*** : cancella un elemento
- ***pop\_back*** : cancella ultimo elemento



Capacity: *empty, size, ...*

# std::vector (<http://www.cplusplus.com/reference/vector/vector/>)

```
# include <vector > # definisce std :: vector
# include <algorithm > # definisce diversi algoritmi
using namespace std;

int main (){
vector <int > x; // struttura dati : vettore di interi
    x. push_back (8); // operazione : aggiungi in fondo
}
```





# std::vector (<http://www.cplusplus.com/reference/vector/vector/>)

```
# include <vector > # definisce std :: vector
# include <algorithm > # definisce diversi algoritmi
using namespace std;

int main (){
    vector <int > x; // struttura dati : vettore di interi
        x. push_back (8); // operazione : aggiungi in fondo
        x. push_back (3);
    }
```



# std::vector (<http://www.cplusplus.com/reference/vector/vector/>)

```
# include <vector > # definisce std :: vector
# include <algorithm > # definisce diversi algoritmi
using namespace std;

int main (){
    vector <int > x; // struttura dati : vettore di interi
        x. push_back (8); // operazione : aggiungi in fondo
        x. push_back (3);
        x. push_back (5);
    }
```



# std::vector (<http://www.cplusplus.com/reference/vector/vector/>)

# include <vector > # definisce std :: vector

# include <algorithm > # definisce diversi algoritmi

using namespace std;

int main (){

vector <int > x; // struttura dati : vettore di interi

x. push\_back (8); // operazione : aggiungi in fondo

x. push\_back (3);

x. push\_back (5);

x. push\_back (9);

}



# std::vector (<http://www.cplusplus.com/reference/vector/vector/>)

```
1 // erasing from vector
2 #include <iostream>
3 #include <vector>
4
5 int main ()
6 {
7     std::vector<int> myvector;
8
9     // set some values (from 1 to 10)
10    for (int i=1; i<=10; i++) myvector.push_back(i);
11
12    // erase the 6th element
13    myvector.erase (myvector.begin()+5);
14
15    // erase the first 3 elements:
16    myvector.erase (myvector.begin(),myvector.begin()+3);
17
18    std::cout << "myvector contains:";
19    for (unsigned i=0; i<myvector.size(); ++i)
20        std::cout << ' ' << myvector[i];
21    std::cout << '\n';
22
23    return 0;
24 }
```

[first,last)

myvector contains: 4 5 7 8 9 10

# **std::set** (<http://www.cplusplus.com/reference/set/set/>)

<chiave>

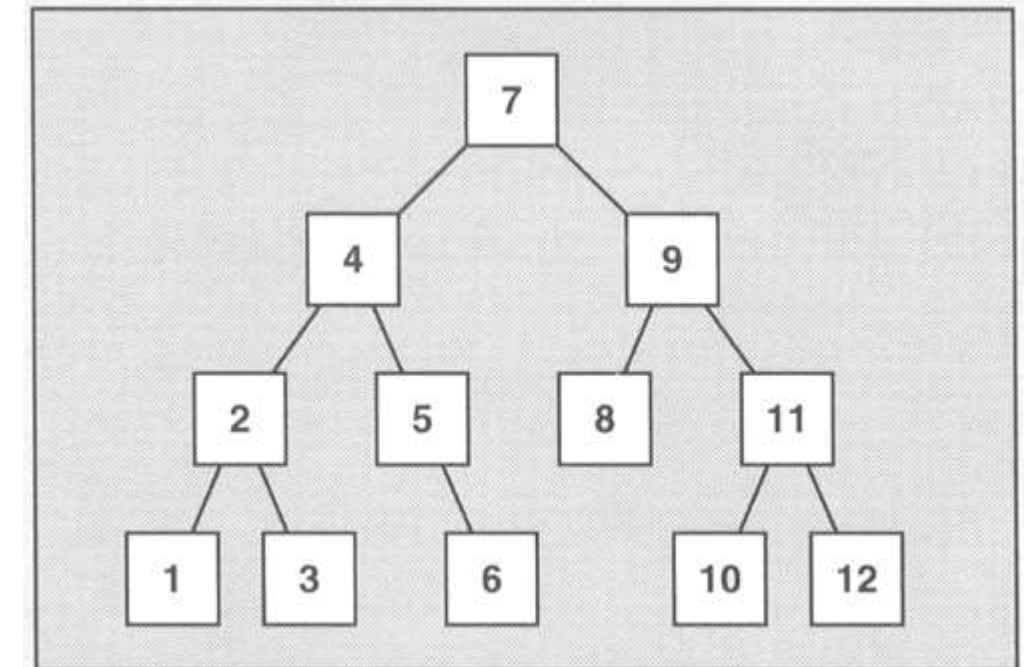
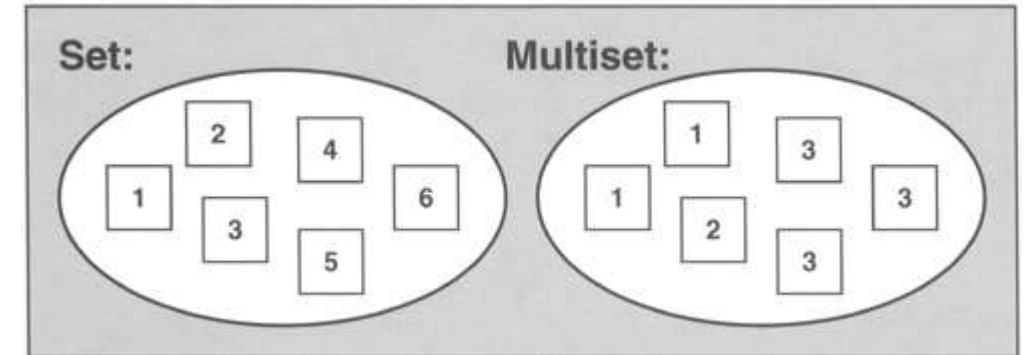
Iterator: ***begin, end, ...***

Operations: ***find, count, ...***

Modifier:

- ***insert*** : inserisce un elemento
- ***erase*** : cancella un elemento

Capacity: ***empty, size, ...***



# std::set (<http://www.cplusplus.com/reference/set/set/>)

```
1 // erasing from set
2 #include <iostream>
3 #include <set>
4
5 int main ()
6 {
7     std::set<int> myset;
8     std::set<int>::iterator it;
9
10    // insert some values:
11    for (int i=1; i<10; i++) myset.insert(i*10); // 10 20 30 40 50 60 70 80 90
12
13    it = myset.begin();
14    ++it; // "it" points now to 20
15
16    myset.erase (it);
17
18    myset.erase (40);
19
20    it = myset.find (60);
21    myset.erase (it, myset.end());
22
23    std::cout << "myset contains:";
24    for (it=myset.begin(); it!=myset.end(); ++it)
25        std::cout << ' ' << *it;
26    std::cout << '\n';
27
28    return 0;
29 }
```

myset contains: 10 30 50

# std::unordered\_set

([http://www.cplusplus.com/reference/unordered\\_set/unordered\\_set/](http://www.cplusplus.com/reference/unordered_set/unordered_set/))

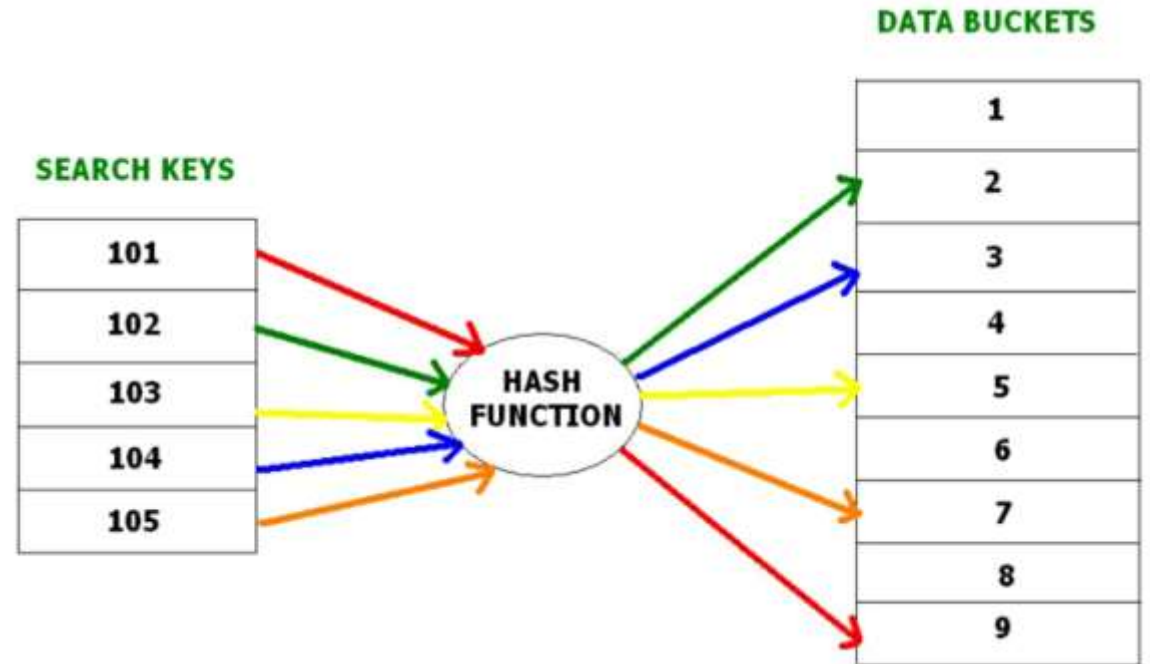
Iterator: *begin, end, ...*

Operations: *find, count, ...*

Modifier:

- ***insert*** : inserisce un elemento
- ***erase*** : cancella un elemento

Capacity: *empty, size, ...*





# **std::map** (<http://www.cplusplus.com/reference/map/map/>)

<chiave, valore>

Iterator: ***begin, end, ...***

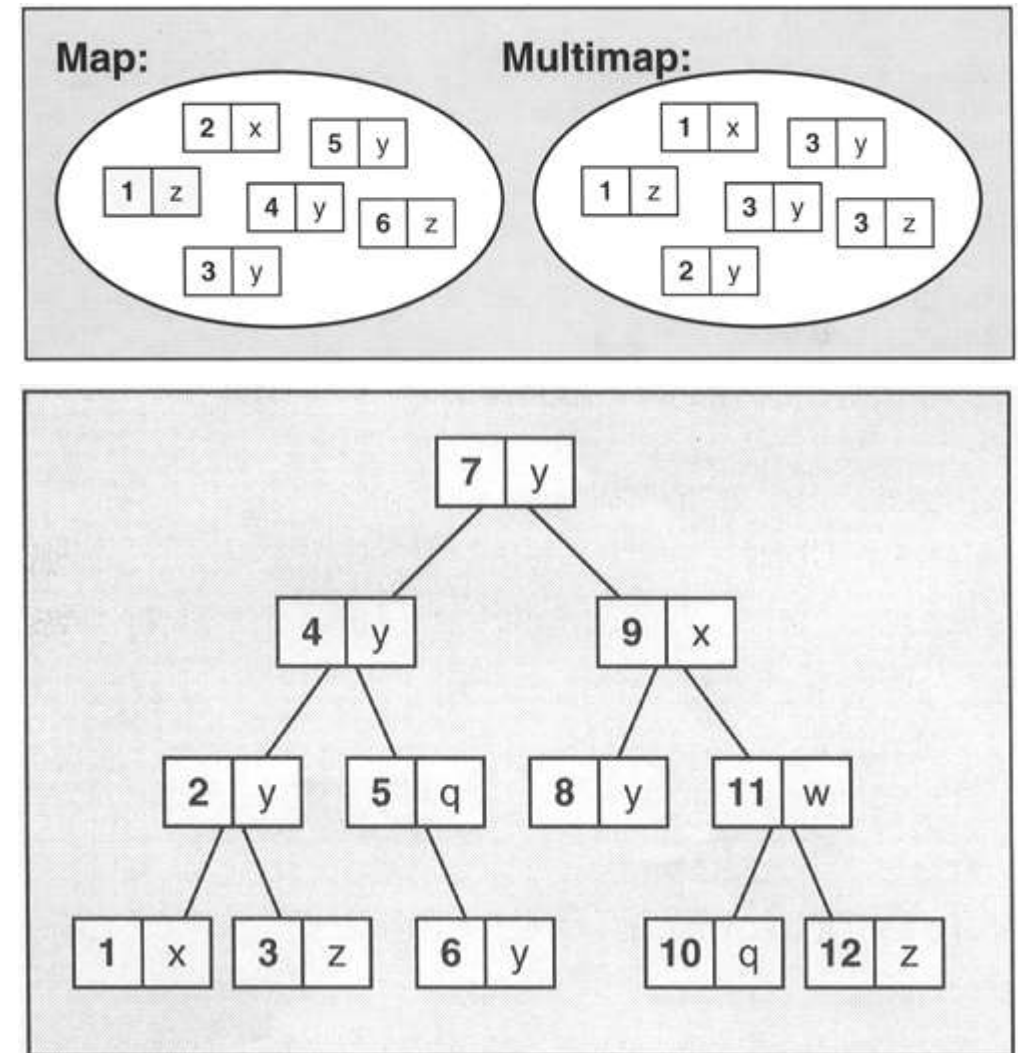
Operations: ***find, count, ...***

Access: ***operator[], ...***

Modifier:

- ***insert*** : inserisce elemento
- ***erase*** : cancella elemento

Capacity: ***empty, size, ...***





# std::map (<http://www.cplusplus.com/reference/map/map/> )

```
1 // erasing from map
2 #include <iostream>
3 #include <map>
4
5 int main ()
6 {
7     std::map<char,int> mymap;
8     std::map<char,int>::iterator it;
9
10    // insert some values:
11    mymap['a']=10;
12    mymap['b']=20;
13    mymap['c']=30;
14    mymap['d']=40;
15    mymap['e']=50;
16    mymap['f']=60;
17
18    it=mymap.find('b');
19    mymap.erase (it);           // erasing by iterator
20
21    mymap.erase ('c');         // erasing by key
22
23    it=mymap.find ('e');
24    mymap.erase ( it, mymap.end() ); // erasing by range
25
26    // show content:
27    for (it=mymap.begin(); it!=mymap.end(); ++it)
28        std::cout << it->first << " => " << it->second << '\n';
29
30    return 0;
31 }
```

a => 10
d => 40

# std::unordered\_map

([http://www.cplusplus.com/reference/unordered\\_map/unordered\\_map/](http://www.cplusplus.com/reference/unordered_map/unordered_map/))

Iterator: *begin, end, ...*

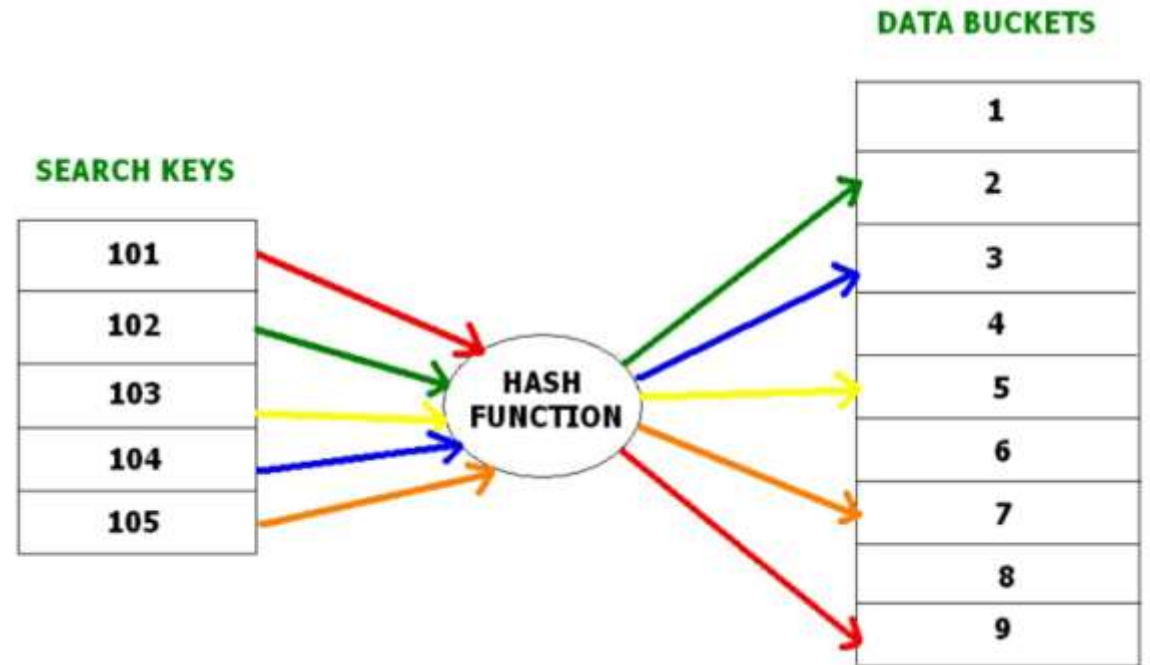
Operations: *find, count, ...*

Access: *operator[], ...*

Modifier:

- *insert* : inserisce elemento
- *erase* : cancella elemento

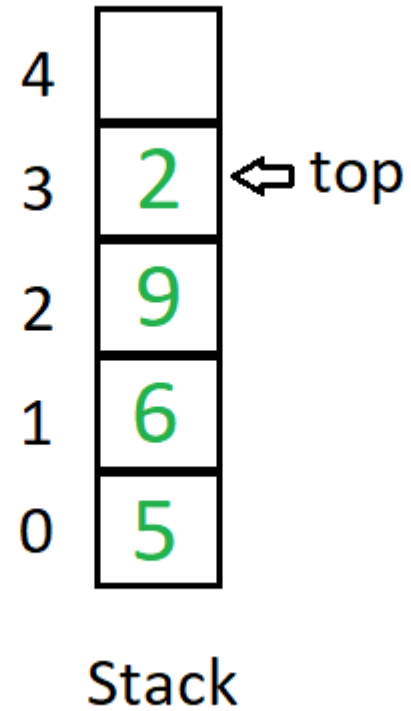
Capacity: *empty, size, ...*



# `std::stack` (<http://www.cplusplus.com/reference/stack/stack/>)

Funzioni che possono essere utili:

- *empty*
- *size*
- *top*
- *push*
- *pop*
- ...



# std::stack (<http://www.cplusplus.com/reference/stack/stack/>)

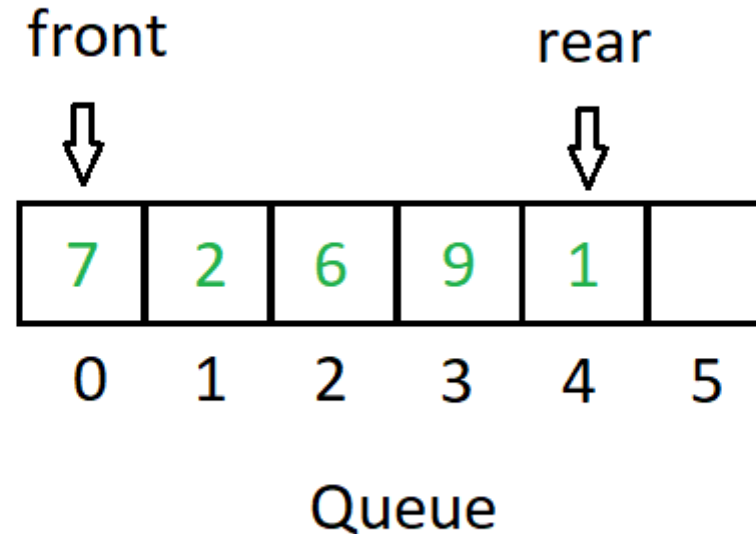
```
1 // stack::push/pop
2 #include <iostream>           // std::cout
3 #include <stack>              // std::stack
4
5 int main ()
6 {
7     std::stack<int> mystack;
8
9     for (int i=0; i<5; ++i) mystack.push(i);
10
11     std::cout << "Popping out elements...";
12     while (!mystack.empty())
13     {
14         std::cout << ' ' << mystack.top();
15         mystack.pop();
16     }
17     std::cout << '\n';
18
19     return 0;
20 }
```

Popping out elements... 4 3 2 1 0

# **std::queue** (<http://www.cplusplus.com/reference/queue/queue>)

Funzioni che possono essere utili:

- *empty*
- *size*
- *front*
- *back*
- *push*
- *pop*
- ...

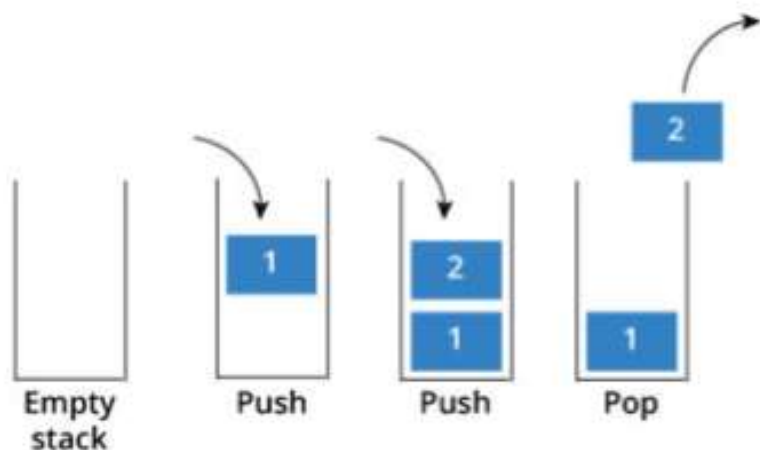


# std::queue (<http://www.cplusplus.com/reference/queue/queue> )

```
1 // queue::push/pop
2 #include <iostream>           // std::cin, std::cout
3 #include <queue>              // std::queue
4
5 int main ()
6 {
7     std::queue<int> myqueue;
8     int myint;
9
10    std::cout << "Please enter some integers (enter 0 to end):\n";
11
12    do {
13        std::cin >> myint;
14        myqueue.push (myint);
15    } while (myint);
16
17    std::cout << "myqueue contains: ";
18    while (!myqueue.empty())
19    {
20        std::cout << ' ' << myqueue.front();
21        myqueue.pop();
22    }
23    std::cout << '\n';
24
25    return 0;
26 }
```

# Stack vs. Queue

## LIFO (Last In First Out)



**Stack**

## FIFO (First In First Out)



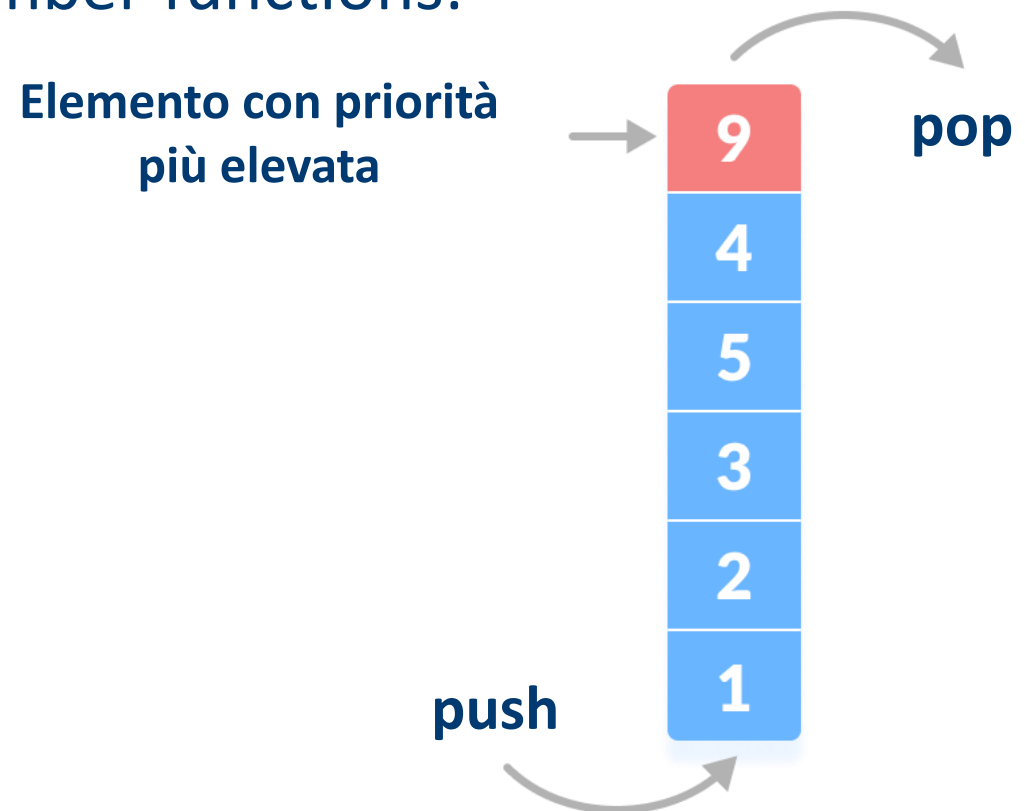
**Queue**

# std::priority\_queue

([http://www.cplusplus.com/reference/queue/priority\\_queue/](http://www.cplusplus.com/reference/queue/priority_queue/))

Heap: il primo elemento è sempre quello dal valore più grande (priorità più elevata). Member functions:

- *empty*
- *size*
- *top*
- *push*
- *pop*
- ...





# std::priority\_queue

([http://www.cplusplus.com/reference/queue/priority\\_queue/](http://www.cplusplus.com/reference/queue/priority_queue/))

```
1 // priority_queue::push/pop
2 #include <iostream>          // std::cout
3 #include <queue>             // std::priority_queue
4
5 int main ()
6 {
7     std::priority_queue<int> mypq;
8
9     mypq.push(30);
10    mypq.push(100);
11    mypq.push(25);
12    mypq.push(40);
13
14    std::cout << "Popping out elements...";
15    while (!mypq.empty())
16    {
17        std::cout << ' ' << mypq.top();
18        mypq.pop();
19    }
20    std::cout << '\n';
21
22    return 0;
23 }
```

Popping out elements... 100 40 30 25

# Un problema dalle OIS (2020): ransomware

[https://training.olinfo.it/#/task/ois\\_ransomware/statement](https://training.olinfo.it/#/task/ois_ransomware/statement)



# Encrypted Contacts (ransomware)

Marco's phone has been attacked by hackers with a ransomware: they have remotely encrypted the phone book with all his contacts and now he has been asked to pay a ransom in bitcoins to recover his data.

He has decided to have some fun trying to perform a full reverse engineering of the malware to recover data without having to pay money. Fortunately for him, the encryption scheme is not too sophisticated. The malware encrypts each digit in isolation and substitutes it with a *code*. For every digit from 0 to 9, Marco has been able to determine which was the corresponding code.

For instance, suppose that digit 0 has been replaced with the code 12345 and that the digit 1 has been replaced with the code 1235; in this scenario an hypothetical number 010 would have been encrypted with the sequence of digits 12345123512345.

After all this grueling work, Marco asks you a small help: recover the original unencrypted numbers for his  $N$  contacts in the phone book.

📎 Among the attachments of this task you may find a template file `ransomware.*` with a sample incomplete implementation.



Figure 1: An Android phone infected by ransomware



## Input

(source: [welivesecurity.com](http://welivesecurity.com))

The first line contains the number of contacts  $N$ . The following  $N$  lines contain each a string of digits representing the encrypted number. The  $i$ -th of the following (and last) 10 lines contains the encrypted code used to replace the digit  $i$ .

## Output





You need to write  $N$  lines where the  $i$ -th contains the unencrypted number of the  $i$ -th contact.

## Constraints

- $1 \leq N \leq 100$ .
- Each number has at most 1000 digits in its encrypted version.
- Each code used to encrypt a digit is at most 100 digits long.
- It is guaranteed that a solution exists and is unique.

## Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- Subtask 1 (0 points)      Examples.  

- Subtask 2 (30 points)      All codes have length 1.  

- Subtask 3 (35 points)      All codes have the same length.  

- Subtask 4 (35 points)      No additional limitations.  


## Examples

input	output
2	330170
333333000111777000	330171
333333000111777111	
000	
111	
222	
333	
444	
555	
666	
777	
888	
999	

1 717171500500211150050089 500 3 21 71 501 11 0 98 89 42	3330025008
---	------------

## Quale struttura dati utilizzare?

1 717171500500211150050089 500 3 21 71 501 11 0 98 89 42	3330025008
---	------------



# Quale struttura dati utilizzare?

<div>1</div> <div>717171500500211150050089</div> <div>500</div> <div>3</div> <div>21</div> <div>71</div> <div>501</div> <div>11</div> <div>0</div> <div>98</div> <div>89</div> <div>42</div>	3330025008	Chiave	Valore
		500	0
		3	1
		21	2
		71	3
		501	4
		11	5
		0	6
		98	7
		89	8
		42	9

# Quale struttura dati utilizzare?

<div>1</div> <div>717171500500211150050089</div> <div>500</div> <div>3</div> <div>21</div> <div>71</div> <div>501</div> <div>11</div> <div>0</div> <div>98</div> <div>89</div> <div>42</div>		Chiave	Valore
		500	0
		3	1
		21	2
		71	3
		501	4
		11	5
		0	6
		98	7
		89	8
		42	9

# Quale struttura dati utilizzare?

<div>1</div> <div>717171500500211150050089</div> <div>500</div> <div>3</div> <div>21</div> <div>71</div> <div>501</div> <div>11</div> <div>0</div> <div>98</div> <div>89</div> <div>42</div>		Chiave	Valore
		500	0
		3	1
		21	2
		71	3
		501	4
		11	5
		0	6
		98	7
		89	8
		42	9

# Quale struttura dati utilizzare?

1 717171500500211150050089 500 3 21 71 501 11 0 98 89 42	3	Chiave	Valore
		500	0
		3	1
		21	2
		71	3
		501	4
		11	5
		0	6
		98	7
		89	8
		42	9

# Quale struttura dati utilizzare?

<div>1</div> <div>717171500500211150050089</div> <div>500</div> <div>3</div> <div>21</div> <div>71</div> <div>501</div> <div>11</div> <div>0</div> <div>98</div> <div>89</div> <div>42</div>	3	Chiave	Valore
		500	0
		3	1
		21	2
		71	3
		501	4
		11	5
		0	6
		98	7
		89	8
		42	9

# Quale struttura dati utilizzare?

<div><div>1</div><div>717171500500211150050089</div><div>500</div><div>3</div><div>21</div><div>71</div><div>501</div><div>11</div><div>0</div><div>98</div><div>89</div><div>42</div></div>	3	Chiave	Valore
		500	0
		3	1
		21	2
		71	3
		501	4
		11	5
		0	6
		98	7
		89	8
		42	9

# Quale struttura dati utilizzare?

<div><div>1</div><div>717171500500211150050089</div><div>500</div><div>3</div><div>21</div><div>71</div><div>501</div><div>11</div><div>0</div><div>98</div><div>89</div><div>42</div></div>	33	Chiave	Valore
		500	0
		3	1
		21	2
		71	3
		501	4
		11	5
		0	6
		98	7
		89	8
		42	9

# Quale struttura dati utilizzare?

<div><div>1</div><div>717171500500211150050089</div><div>500</div><div>3</div><div>21</div><div>71</div><div>501</div><div>11</div><div>0</div><div>98</div><div>89</div><div>42</div></div>	33	Chiave	Valore
		500	0
		3	1
		21	2
		71	3
		501	4
		11	5
		0	6
		98	7
		89	8
		42	9



# Quale struttura dati utilizzare?

<div><div>1</div><div>717171500500211150050089</div><div>500</div><div>3</div><div>21</div><div>71</div><div>501</div><div>11</div><div>0</div><div>98</div><div>89</div><div>42</div></div>	33	Chiave	Valore
		500	0
		3	1
		21	2
		71	3
		501	4
		11	5
		0	6
		98	7
		89	8
		42	9

# Quale struttura dati utilizzare?

<div><div>1</div><div>717171500500211150050089</div><div>500</div><div>3</div><div>21</div><div>71</div><div>501</div><div>11</div><div>0</div><div>98</div><div>89</div><div>42</div></div>	333	Chiave	Valore
		500	0
		3	1
		21	2
		71	3
		501	4
		11	5
		0	6
		98	7
		89	8
		42	9

# Quale struttura dati utilizzare?

<div><div>1</div><div>717171500500211150050089</div><div>500</div><div>3</div><div>21</div><div>71</div><div>501</div><div>11</div><div>0</div><div>98</div><div>89</div><div>42</div></div>	333	Chiave	Valore
		500	0
		3	1
		21	2
		71	3
		501	4
		11	5
		0	6
		98	7
		89	8
		42	9

# Quale struttura dati utilizzare?

<div><div>1</div><div>717171500500211150050089</div><div>500</div><div>3</div><div>21</div><div>71</div><div>501</div><div>11</div><div>0</div><div>98</div><div>89</div><div>42</div></div>	333	Chiave	Valore
		500	0
		3	1
		21	2
		71	3
		501	4
		11	5
		0	6
		98	7
		89	8
		42	9

# Quale struttura dati utilizzare?

<div><div>1</div><div>717171500500211150050089</div><div>500</div><div>3</div><div>21</div><div>71</div><div>501</div><div>11</div><div>0</div><div>98</div><div>89</div><div>42</div></div>	333	Chiave	Valore
		500	0
		3	1
		21	2
		71	3
		501	4
		11	5
		0	6
		98	7
		89	8
		42	9

# Quale struttura dati utilizzare?

<div><div>1</div><div>717171500500211150050089</div><div>500</div><div>3</div><div>21</div><div>71</div><div>501</div><div>11</div><div>0</div><div>98</div><div>89</div><div>42</div></div>	3330	Chiave	Valore
		500	0
		3	1
		21	2
		71	3
		501	4
		11	5
		0	6
		98	7
		89	8
		42	9

# Quale struttura dati utilizzare?

<div>1</div> <div>717171500500211150050089</div> <div>500</div> <div>3</div> <div>21</div> <div>71</div> <div>501</div> <div>11</div> <div>0</div> <div>98</div> <div>89</div> <div>42</div>	333002500	Chiave	Valore
		500	0
		3	1
		21	2
		71	3
		501	4
		11	5
		0	6
		98	7
		89	8
		42	9

# Quale struttura dati utilizzare?

<div><div>1</div><div>717171500500211150050089</div><div>500</div><div>3</div><div>21</div><div>71</div><div>501</div><div>11</div><div>0</div><div>98</div><div>89</div><div>42</div></div>	3330025008	Chiave	Valore
		500	0
		3	1
		21	2
		71	3
		501	4
		11	5
		0	6
		98	7
		89	8
		42	9



# Quale struttura dati utilizzare?

<div>1</div> <div>717171500500211150050089</div> <div>500</div> <div>3</div> <div>21</div> <div>71</div> <div>501</div> <div>11</div> <div>0</div> <div>98</div> <div>89</div> <div>42</div>	3330025008	Chiave	Valore
		500	0
		3	1
		21	2
		71	3
		501	4
		11	5
		0	6
		98	7
		89	8
		42	9

# **Compiti a casa (Esercizi Olimpiadi)**



# Compiti a casa

Risolvi i seguenti problemi:

- Paletta: [oii paletta](#)
- Regali: [ois regali](#)
- Seats: [ois seats](#)
- Maxim: [ois maxim](#)
- Tecnico pazzo: [pazzo](#)