

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

1 aprile 2021

**LUISS**

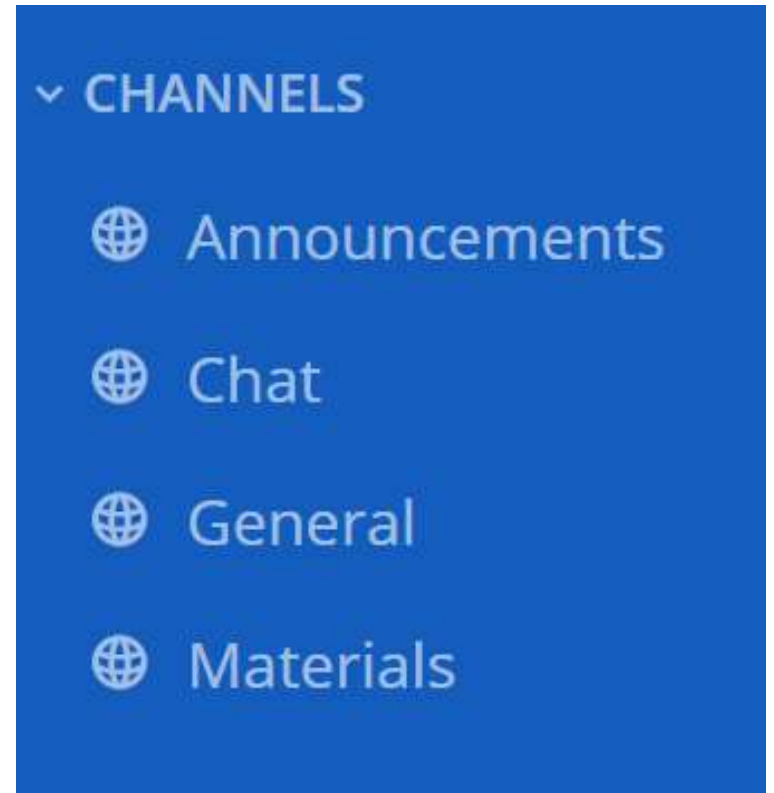


# Reminder: Mattermost

<https://coding.luiss.ml/mattermost>

1. *Announcements* (e.g., link video lezioni)
2. *Chat* (fai qui le tue domande durante le lezioni)
3. *General* (discussioni interesse generale)
4. *Materials* (materiale didattico, slide prima delle lezioni, etc...)


- Controlla i canali: spesso troverai qui le risposte alle tue domande!
- Non inquinare i canali! Contribuisci a mantenerli puliti e non postare contenuti impropri. Grazie!



# Lezioni

Puoi accedere alle lezioni tramite YouTube o Webex:

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

 (licenza limitata a 1000 utenti; puoi provare  
l'ebrezza di diventare tu docente per qualche minuto  
condividendo la tua soluzione ai problemi che consideriamo!)

# Programma di oggi (Algoritmi 1/2)

1. Soluzione esercizi della scorsa settimana
2. Introduzione (informale) agli algoritmi
3. Un problema dalle Territoriali (2014): Collatz
4. Complichiamo Collatz: Pollatz (Gator 2016)
5. Un problema dalle Territoriali (2015): Rispetta i versi


# Programma di oggi (Strutture Dati 2/2)

6. Introduzione (informale) alle strutture dati
7. Un problema dalle OIS (2019): Gasoline
8. Un problema dalle Territoriali (2017): Scommessa
9. Esercizi Olimpiadi (compiti da fare a casa per la prossima volta)

Soluzioni degli esercizi della scorsa settimana



# Compiti a casa della scorsa settimana

Se siete collegati tramite Webex alzate la mano (pulsante  vicino al vostro nome nella lista partecipanti) per esporre la vostra soluzione ai problemi della scorsa settimana:

Graduation Card (text):

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

Encrypted Contacts (ransomware):

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

Science against spam (spam):

[https://training.olinfo.it/#/task/ioit\\_spam/statement](https://training.olinfo.it/#/task/ioit_spam/statement)

# Introduzione (informale) agli algoritmi





# Cos'è un algoritmo?

## **algorithm**

*noun*

Word used by programmers when they do not want to explain what they did.

# 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

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

# 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).

# Altro esempio di algoritmo

1. Se hai domande relative a questo corso, non spedire email a [olimpiadi@luiss.it](mailto:olimpiadi@luiss.it)
2. Piuttosto, controlla i canali Mattermost: nel 90% dei casi la risposta alla tua domanda è già lì!
3. Se non trovi risposta su un canale Mattermost, fai la tua domanda sulla Chat: qualcuno ti risponderà! -- anche se la risposta è già su un canale Mattermost



# Numeri di Fibonacci

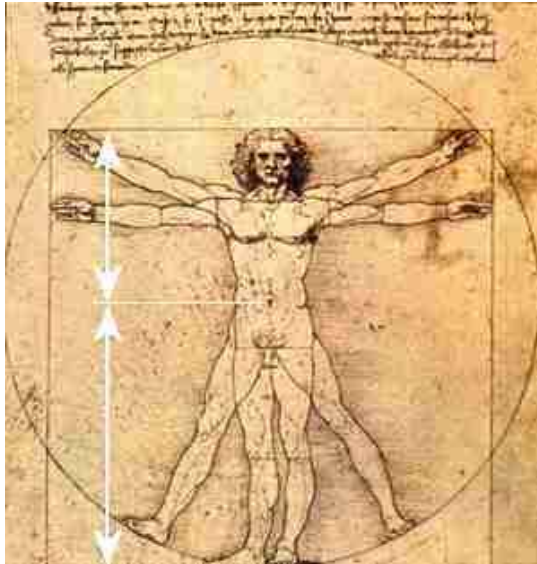
- Definiti dalla relazione di ricorrenza:

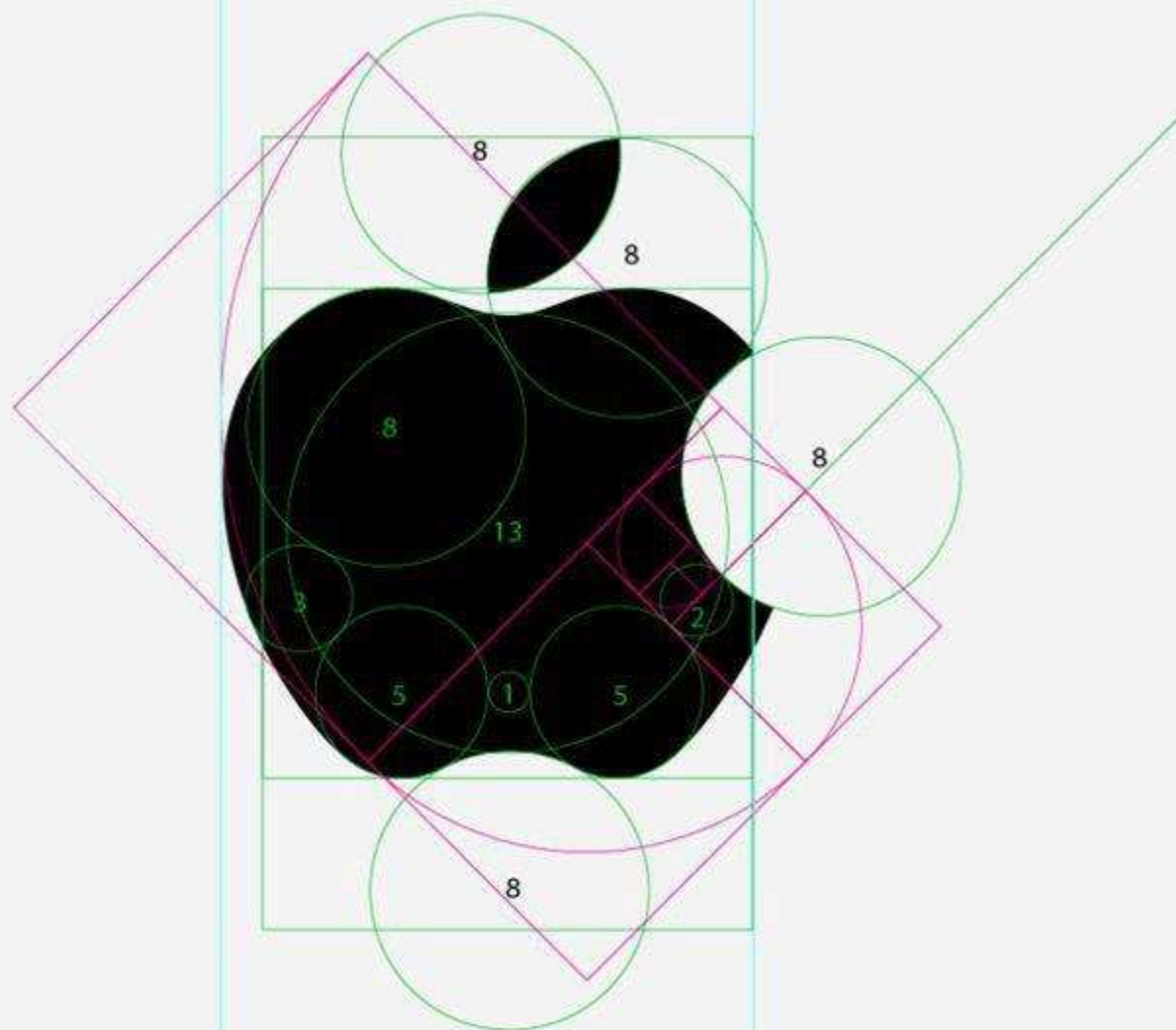
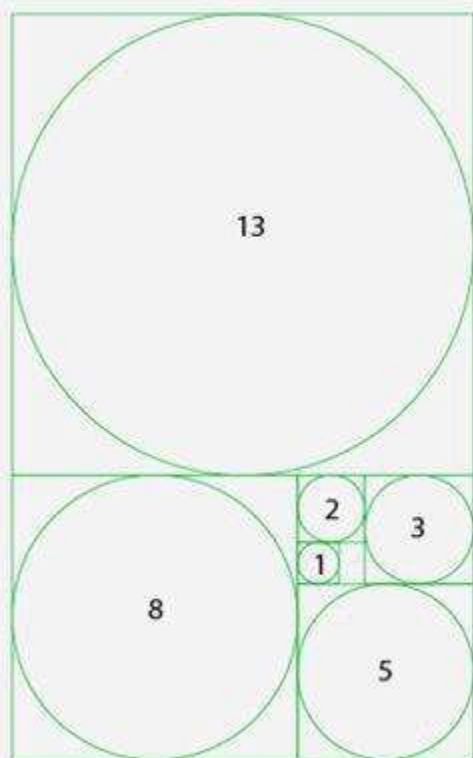
$$F_n = \begin{cases} F_{n-1} + F_{n-2} & \text{se } n \geq 3 \\ 1 & \text{se } n = 1, 2 \end{cases}$$

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



# Sezione aurea







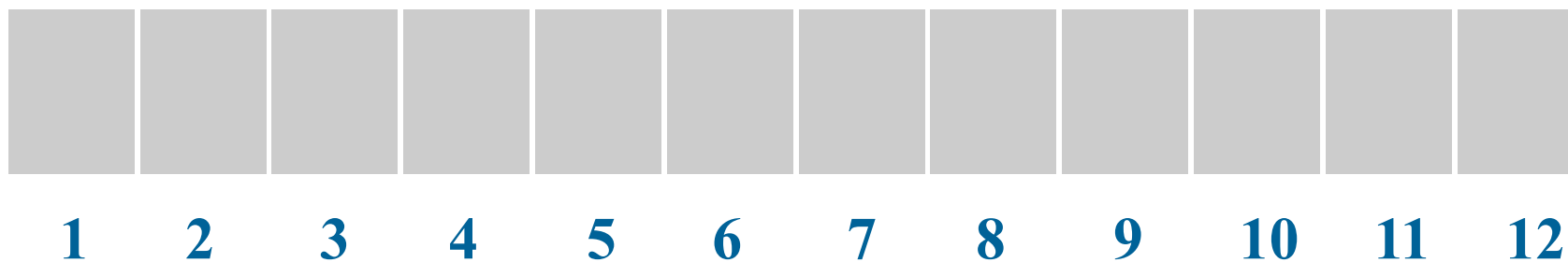
# Primo algoritmo (doppiamente ricorsivo)

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?

# Possiamo fare di meglio?



## Secondo algoritmo (iterativo)

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

# Un problema dalle Territoriali: Collatz

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



# La Congettura di Collatz (collatz)

DIFFICOLTÀ D=1

## Descrizione del problema

Consideriamo il seguente algoritmo, che prende in ingresso un intero positivo  $N$ :

1. Se  $N$  vale 1, l'algoritmo termina.
2. Se  $N$  è pari, dividi  $N$  per 2, altrimenti (se  $N$  è dispari) moltiplicalo per 3 e aggiungi 1.

Per esempio, applicato al valore  $N = 6$ , l'algoritmo produce la seguente sequenza (di lunghezza 9, contando anche il valore iniziale  $N = 6$  e il valore finale 1):

6, 3, 10, 5, 16, 8, 4, 2, 1.

La congettura di Collatz, chiamata anche congettura  $3N+1$ , afferma che l'algoritmo qui sopra termini sempre per qualsiasi valore  $N$ ; in altri termini, se prendo un qualsiasi numero intero maggiore di 1 applicare la regola numero 2 conduce sempre al numero 1.

È riferendosi a questa celebre congettura che il famoso matematico Erdős ha commentato sul come questioni semplici ma elusive mettono in evidenza quanto poco noi si possa accedere ai misteri del “grande Libro”.

Giovanni sta cercando di dimostrare la congettura, ed è interessato alla lunghezza della sequenza. Il vostro compito è quello di aiutare Giovanni scrivendo un programma che, ricevuto in ingresso un numero  $N$ , calcoli la lunghezza della sequenza che si ottiene a partire da  $N$ .

## Dati di input

Il file `input.txt` è composto da una riga contenente  $N$ , un intero positivo.

## Dati di output

Il file `output.txt` è composto da una sola riga contenente un intero positivo  $L$ : la lunghezza della sequenza a partire da  $N$ .

## Assunzioni e note

- $2 \leq N \leq 1\,000$ .
- E' noto che, per qualsiasi  $N$  minore di 1000, la lunghezza  $L$  della sequenza è minore di 200.

## Esempio di input/output

File input.txt	File output.txt
6	9

File input.txt	File output.txt
24	11

# Complichiamo Collatz: Pollatz

[https://training.olinfo.it/#/task/gator\\_pcollatz/statement](https://training.olinfo.it/#/task/gator_pcollatz/statement)







## Sequenza di Pollatz (pcollatz)

Consideriamo il seguente algoritmo, che prende in ingresso un intero positivo  $N$ :

1. Se  $N$  vale 1, l'algoritmo termina.
2. Se  $N$  è pari, dividi  $N$  per 2, altrimenti (se  $N$  è dispari) moltiplicalo per 3 e aggiungi 1.

Per esempio, applicato al valore  $N = 6$ , l'algoritmo produce la seguente sequenza (di lunghezza 9, contando anche il valore iniziale  $N = 6$  e il valore finale 1):

6, 3, 10, 5, 16, 8, 4, 2, 1.

La congettura di Collatz, chiamata anche congettura  $3N + 1$ , afferma che l'algoritmo qui sopra termini sempre per qualsiasi valore  $N$ ; in altri termini, se prendo un qualsiasi numero intero maggiore di 1 applicare la regola numero 2 conduce sempre al numero 1. È riferendosi a questa celebre congettura che il famoso matematico Erdős ha detto “*La matematica non è ancora pronta per problemi di questo tipo*”.

Patrizio ha creato una versione alternativa della sequenza, secondo la seguente regola:

1. Se  $N$  vale 1, l'algoritmo termina.
2. Se  $N$  è pari, dividi  $N$  per 2, altrimenti (se  $N$  è dispari) moltiplicalo per 5 e aggiungi 1.

Patrizio ha chiamato la sua versione *sequenza di Pollatz*, e si è accorto che, per alcuni numeri la sua sequenza è più corta di quella originale; per esempio, applicata al numero 6, otteniamo:

6, 3, 16, 8, 4, 2, 1.

ovvero una sequenza di lunghezza 7. Allo stesso tempo, però, Patrizio si è anche accorto che per altri numeri **la sequenza non termina mai**. Il vostro compito è quello di aiutare Patrizio a calcolare quante volte la sequenza di Pollatz termina generando una sequenza di numeri strettamente più corta di quella di Collatz.

## Dati di input

Il file `input.txt` è composto da una riga di testo, contenente i due interi  $A$  e  $B$  ( $A \leq B$ ).

## Dati di output

Il file `output.txt` è composto da una sola riga contenente un intero positivo: quanto sono gli  $N$  ( $A \leq N \leq B$ ) per cui la lunghezza della sequenza di Pollatz calcolata a partire da  $N$  termina e ha lunghezza strettamente minore della corrispondente sequenza di Collatz.

## Assunzioni

- $1 \leq A \leq B \leq 10\,000$ .

## Esempi di input/output

input.txt	output.txt
1 5	1
50 60	2

## Spiegazione

Nel primo caso di esempio, le lunghezze delle sequenze di Pollatz e di Collatz coincidono per  $N = 1, 2, 4$ . Per  $N = 3$  la sequenza di Pollatz  $(3, 16, 8, 4, 2, 1)$  è strettamente più corta di quella di Collatz  $(3, 10, 5, 16, 8, 4, 2, 1)$ . Per  $N = 5$  la sequenza di Pollatz non termina mai.

# Un altro problema dalle Territoriali (2015): Rispetta i versi

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





## Rispetta i versi (disuguaglianze)

Limite di tempo: 1.0 secondi

Limite di memoria: 256 MiB

Difficoltà: 2

Gabriele ha un nuovo rompicapo preferito, chiamato “Rispetta i versi”. Si tratta di un solitario giocato su una griglia formata da  $N$  caselle separate da un simbolo di disuguaglianza; in figura è mostrato un esempio con  $N = 6$ .



L'obiettivo del gioco è quello di riempire le celle vuote con tutti i numeri da 1 a  $N$  (ogni numero deve comparire esattamente una volta), in modo da rispettare le disuguaglianze tra caselle adiacenti. Per la griglia della figura, una delle possibili soluzioni al rompicapo è la seguente:





## Dati di input

Il file `input.txt` contiene due righe di testo. Sulla prima è presente l'intero  $N$ , il numero di caselle del gioco. Sulla seconda è presente una stringa di  $N - 1$  caratteri, ognuno dei quali può essere solo `<` o `>`, che descrive i vincoli tra le caselle, da sinistra a destra.

## Dati di output

Il file `output.txt` contiene su una sola riga una qualunque permutazione dei numeri da 1 a  $N$  - separati tra loro da uno spazio - che risolve il rompicapo. I numeri corrispondono ai valori scritti nelle caselle, leggendo da sinistra verso destra.

## Assunzioni

- $2 \leq N \leq 100\,000$ .
- Nel 30% dei casi, il valore di  $N$  non supera 10.
- Nel 60% dei casi, il valore di  $N$  non supera 20.
- Si garantisce l'esistenza di almeno una soluzione per ciascuno dei casi di test utilizzati nella verifica del funzionamento del programma.

## Esempi di input/output

input.txt	output.txt
6 <><<>	2 5 1 3 6 4
5 >><<	5 3 1 2 4
8 >><>><>	6 5 4 7 3 2 8 1



# 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)
- ...

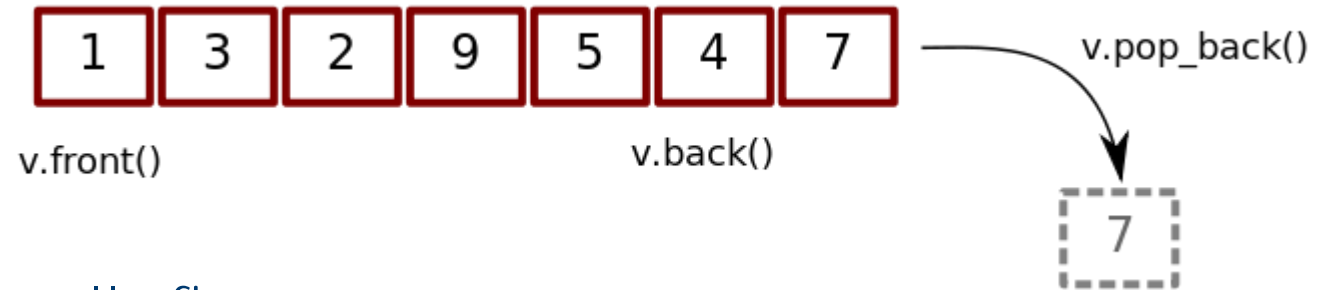
# std::vector (<http://www.cplusplus.com/reference/vector/vector/>)

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::set` (<http://www.cplusplus.com/reference/set/set/>)  
<chiave>

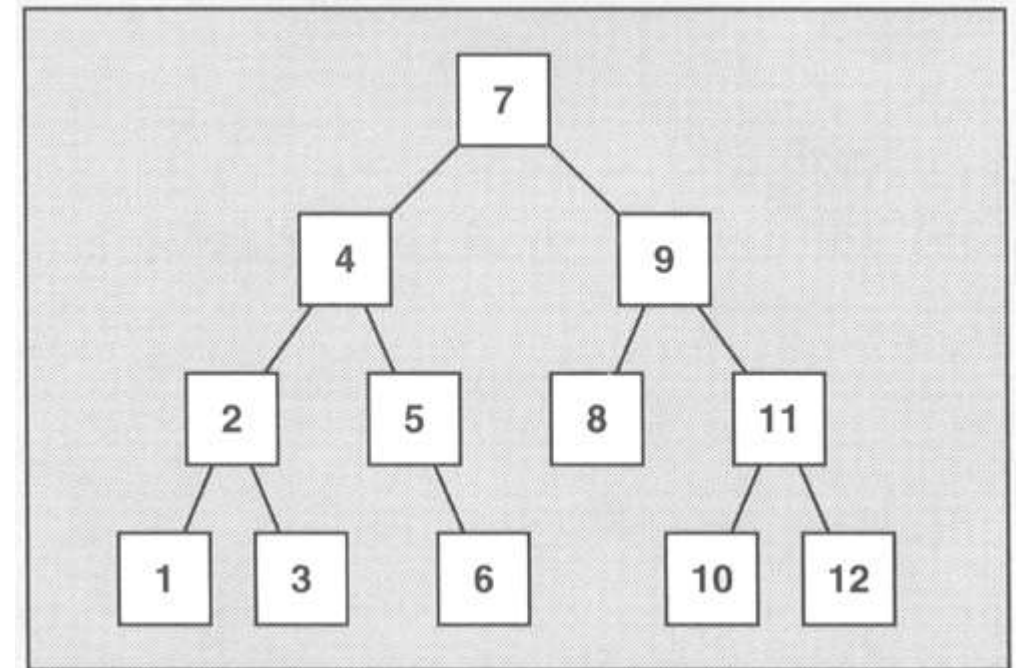
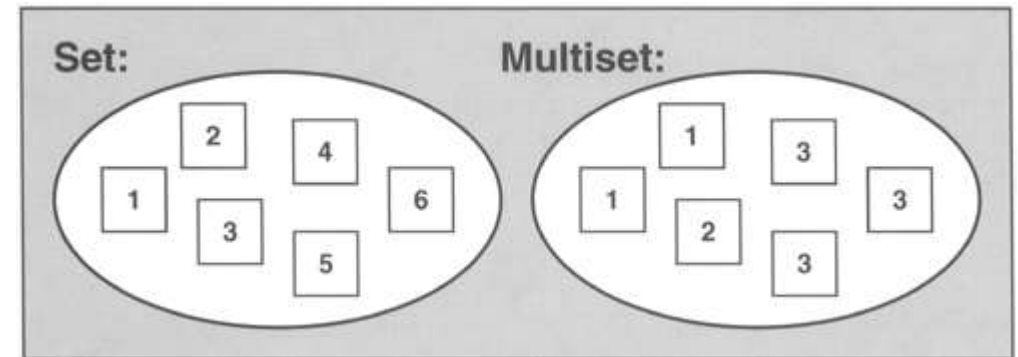
Iterator: *begin*, ***end***, ...

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

Modifier:

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

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



# std::unordered\_set

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

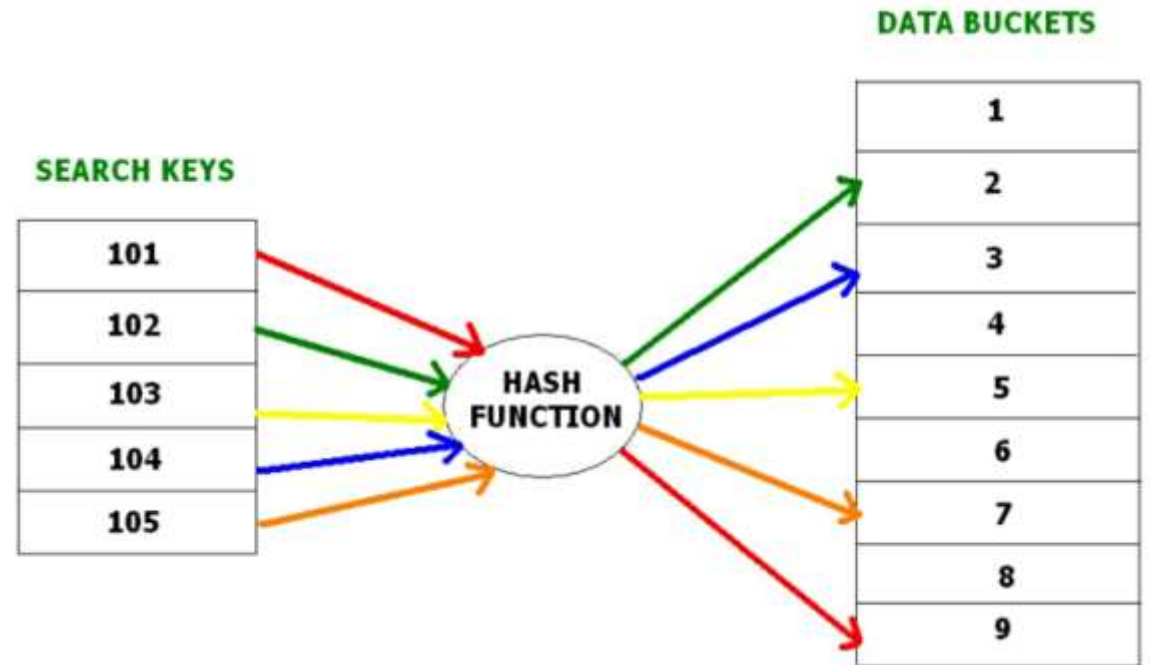
Iterator: *begin*, ***end***, ...

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***, ...

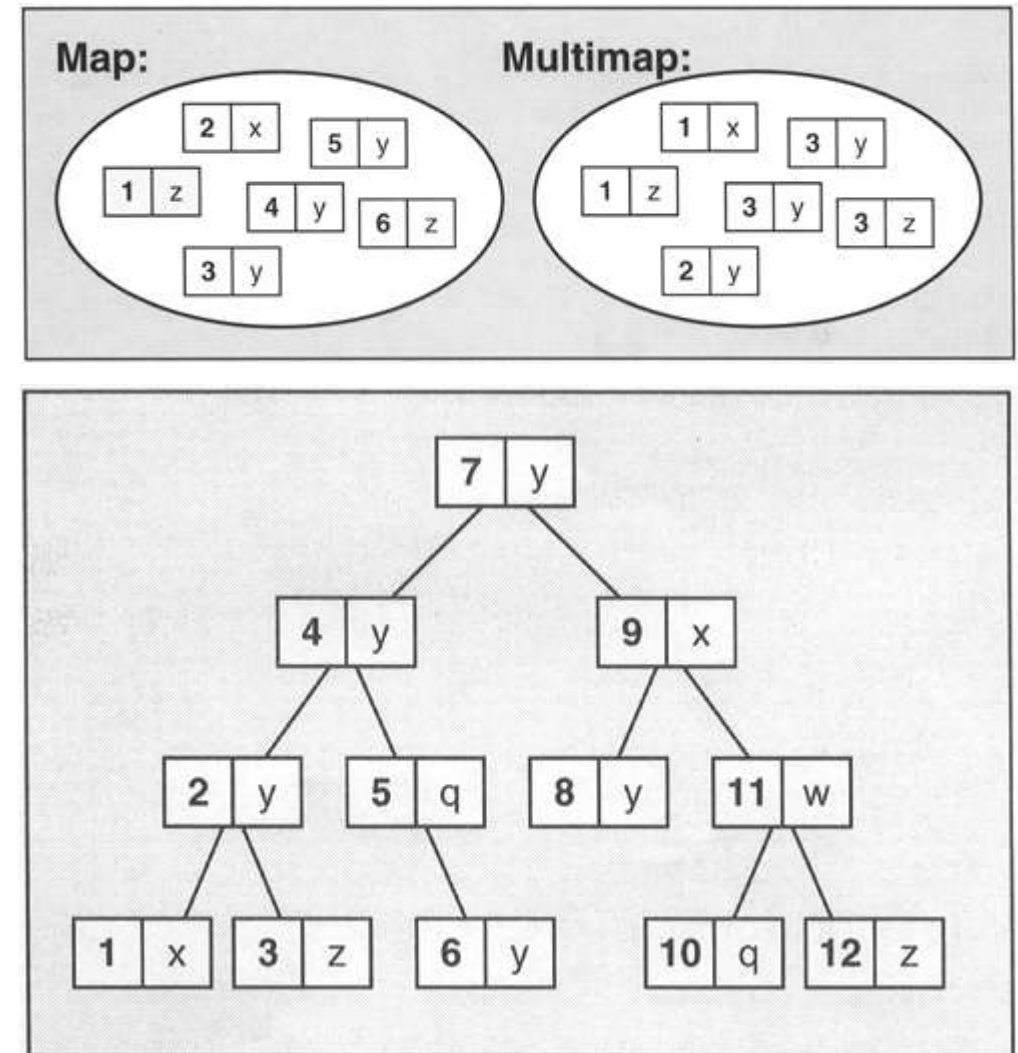
Operations: *find*, *count*, ...

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

Modifier:

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

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



# std::unordered\_map

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

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

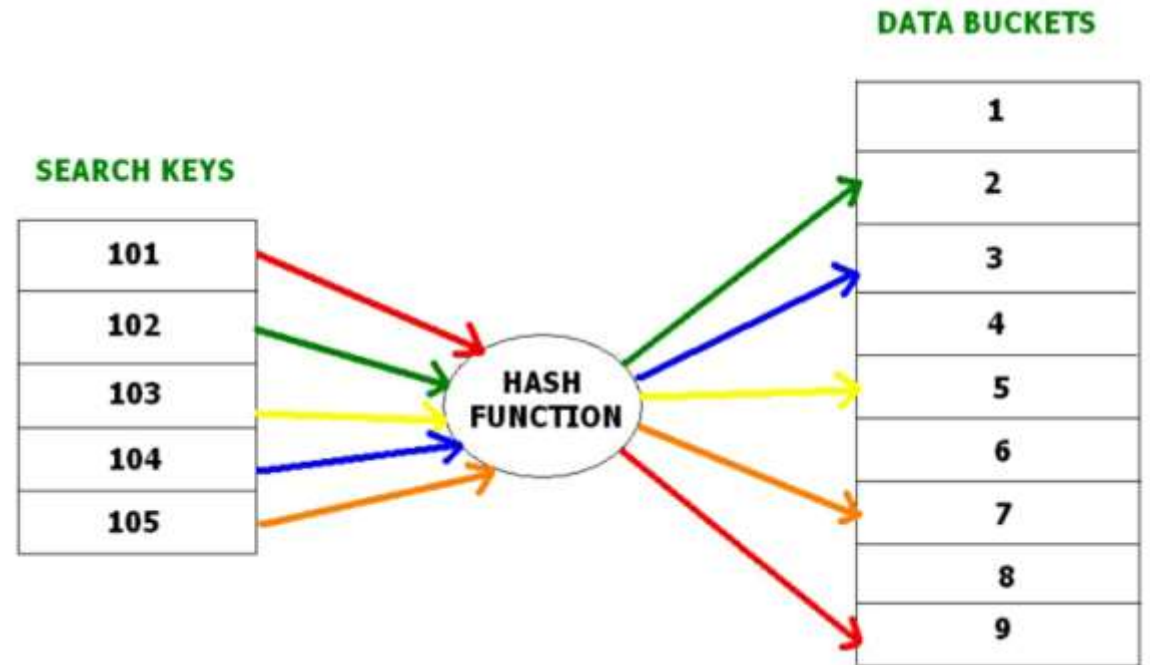
Operations: *find*, *count*, ...

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

Modifier:

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

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

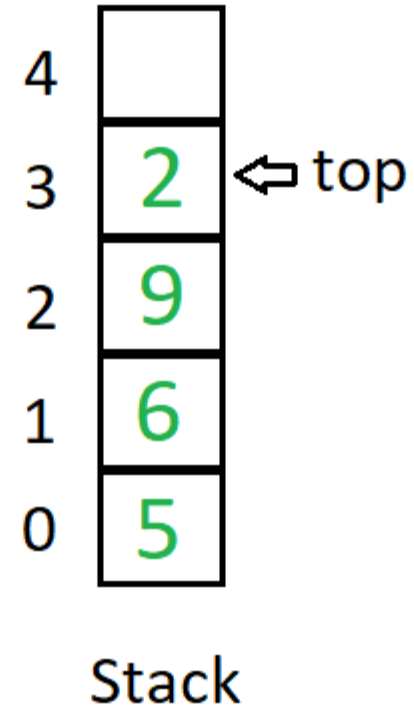




# std::stack (<http://www.cplusplus.com/reference/stack/stack/>)

Member functions:

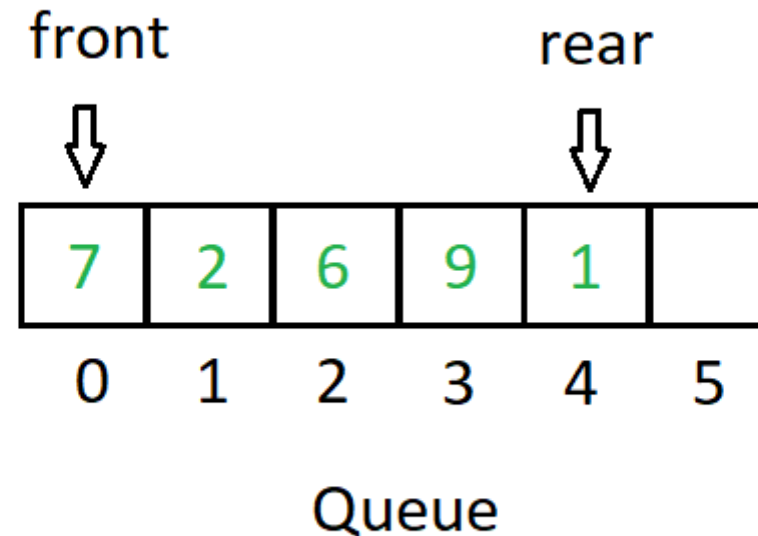
- *empty*
- *size*
- *top*
- *push*
- *pop*
- ...



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

Member functions:

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

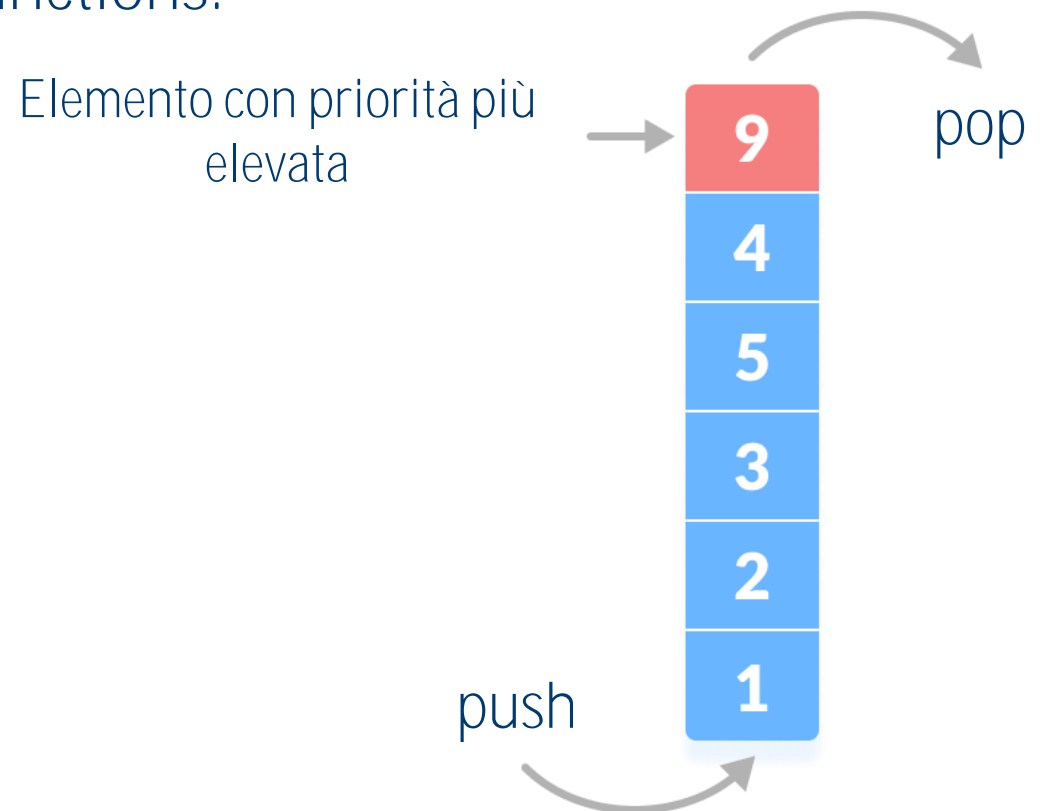


# std::priority\_queue

([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*
- ...



# Un problema dalle OIS (2019): Gasoline

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



## Gasoline Stations (gasoline)

After many days of uninterrupted studying, William needs to take a break: a romantic trip with his girlfriend from Milan to Pordenone, to visit the famous Saint Valentine Park!

During the travel, he will encounter  $N$  gas stations where he can buy some fuel for his car. William knows the price-per-liter  $P_i$  of each station and he has already estimated how much gasoline  $G_i$  at minimum he has to have at each station in order to reach the next one (and, at the last station, to arrive in Pordenone) safely. Being very wise, he brought with him many tanks so that he can buy as much gasoline as he wants.



Figure 1: To fill or not to fill, that is the question.

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

## Input

The first line contains the only integer  $N$ . The second line contains  $N$  integers  $P_i$ , the price-per-liter at the  $i$ -th station. The third line contains  $N$  integers  $G_i$ , the amount of liters William has to have in the tanks to reach the next station (or Pordenone) from the  $i$ -th station.

## Output







You need to write a single line with an integer: the minimum amount of money needed for the whole trip.

## Constraints

- $1 \leq N \leq 1\,000\,000$ .
- $1 \leq P_i, G_i \leq 1\,000$  for each  $i = 0 \dots N - 1$ .

## 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** (10 points)       $N \leq 10$ .  

- **Subtask 3** (10 points)      All  $P_i$  are equal.  

- **Subtask 4** (10 points)      All  $G_i$  are equal.  

- **Subtask 5** (30 points)       $N \leq 1000$ .  

- **Subtask 6** (40 points)      No additional limitations.  


## Examples

input	output
5 30 25 30 10 10 5 6 4 2 4	460
5 10 30 20 10 15 7 10 100 20 3	1400

## Explanation

In the **first sample case**, an optimal solution is:

- Buy 5 liters at the first station spending  $5 \cdot 30 = 150$ .
- Buy 10 liters at the second station spending  $10 \cdot 25 = 250$ .
- Don't buy anything at the next station.
- Buy 6 liters at the next station  $6 \cdot 10 = 60$ .
- Don't buy anything at the last station.

The total amount is  $150 + 250 + 60 = 460$ .

In the **second sample case**, an optimal solution is to buy all the gasoline needed at the first station spending  $140 \cdot 10 = 1400$ .



# Un problema dalle Territoriali (2017): 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!

<sup>1</sup> 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
```

# Compiti a casa (Esercizi Olimpiadi)



# Compiti a casa

Prova a risolvere i seguenti problemi:

- Rope escapes [https://training.olinfo.it/#/task/ois\\_ropes/statement](https://training.olinfo.it/#/task/ois_ropes/statement)
- Smartphone [https://training.olinfo.it/#/task/ois\\_smartphone/statement](https://training.olinfo.it/#/task/ois_smartphone/statement)
- Somme costose [https://training.olinfo.it/#/task/gator\\_somme/statement](https://training.olinfo.it/#/task/gator_somme/statement)
- Walkway [https://training.olinfo.it/#/task/ois\\_walkway/statement](https://training.olinfo.it/#/task/ois_walkway/statement)

# Compiti a casa

