

## Calcolatrice d'epoca (calcolatrice)

La calcolatrice che Giorgio conserva gelosamente da quando faceva le elementari si è rotta, e ora il suo schermo *LCD* vecchio stile penzola appeso ai tasti dal solo filo di alimentazione. Nonostante questo, funziona ancora perfettamente come un tempo e quindi Giorgio non ha intenzione di smettere di utilizzarla.

Con la sua affezionata calcolatrice, Giorgio ha appena fatto dei lunghi e complessi calcoli i cui risultati potrebbero portare alla soluzione di numerose congetture in innumerevoli campi della matematica, e ne ha trascritto i risultati su un foglio. Tuttavia solo dopo si è accorto che, per via del danno descritto sopra, non è in grado di capire l'orientamento giusto dello schermo e quindi in alcuni casi potrebbe aver trascritto il risultato sbagliato (e cioè come si leggerebbe ruotando lo schermo della calcolatrice di  $180^\circ$ ).

Da un'approssimazione a stima che si è fatto, gli sembra che i risultati dovrebbero essere numeri abbastanza piccoli. Pertanto, dato un numero  $N$ , vuole scrivere un programma che calcoli il corrispondente numero  $M$  ruotato di  $180^\circ$ , e se questo è effettivamente un numero sensato scelga il minore tra i due.



### Implementazione

Dovrai sottoporre esattamente un file con estensione `.c`, `.cpp` o `.pas`.

📎 Tra gli allegati a questo task troverai un template (`calcolatrice.c`, `calcolatrice.cpp`, `calcolatrice.pas`) con un esempio di implementazione da completare.

Se sceglierai di utilizzare il template, dovrai implementare la seguente funzione:

C/C++	<code>int rimedia(int N);</code>
Pascal	<code>function rimedia(N: longint): longint;</code>

In cui:

- L'intero  $N$  rappresenta il numero segnato sul foglio da Giorgio.
- La funzione dovrà restituire il minimo tra  $N$  e il corrispondente  $M$  ruotato di  $180^\circ$  (se sensato), che verrà stampato sul file di output.

### Dati di input

Il file `input.txt` è composto da un'unica riga contenente l'unico intero  $N$ .

### Dati di output

Il file `output.txt` è composto da un'unica riga contenente un unico intero, la risposta a questo problema.



## Assunzioni

- $0 \leq N \leq 1\,000\,000\,000$ .

## Assegnazione del punteggio

Il tuo programma verrà testato su diversi test case raggruppati in subtask. Per ottenere il punteggio relativo ad un subtask, è necessario risolvere correttamente tutti i test relativi ad esso.

- **Subtask 1 [10 punti]:** Casi d'esempio.
- **Subtask 2 [20 punti]:**  $N \leq 10$ .
- **Subtask 3 [30 punti]:**  $N \leq 100$ .
- **Subtask 4 [40 punti]:** Nessuna limitazione specifica.

## Esempi di input/output

input.txt	output.txt
14	14

input.txt	output.txt
12650	12650

input.txt	output.txt
806129	621908

## Spiegazione

Nel **primo caso di esempio**, il numero 14 ruotato non genera un numero sensato, per via della presenza della cifra 4.

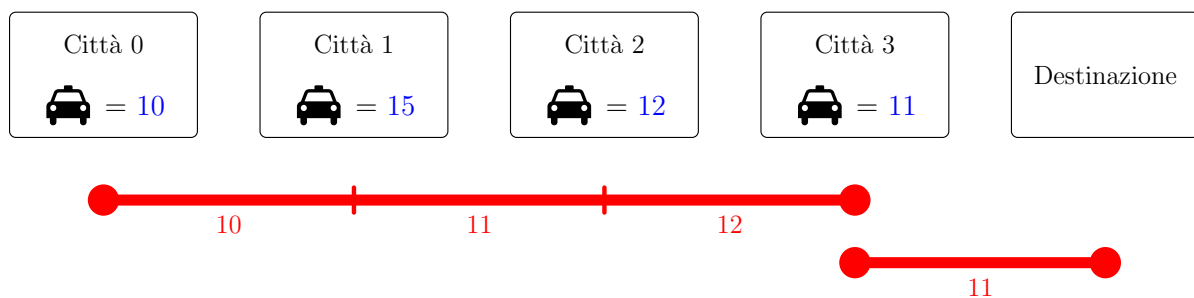
Nel **secondo caso di esempio**, il numero 12650 ruotato genera il numero 05921 che non è da considerarsi sensato, per via della presenza della cifra 0 come prima cifra.

Nel **terzo caso di esempio**, il numero 806129 ruotato genera il numero 621908 che è sensato e minore del precedente.

## Viaggio in taxi (taxi)

Gabriele deve andare a trovare Giorgio per mettere a punto la finale delle *OIS*. La strada da Brescia a Pinerolo è lunga, e attraversa  $N + 1$  città (numerate da 0 a  $N$ ) tutte alla stessa distanza di 1 tantometro. Dato che Gabriele ha recentemente racimolato un po' di liquidità, sta valutando se viaggiare comodo spostandosi di città in città in taxi. In ogni città c'è una stazione di taxi, ma la situazione è resa complicata dal fatto che i prezzi dei taxi variano notevolmente di città in città. Prendere un taxi nella città  $i$  ha un prezzo base di  $C_i$  euro per il primo tantometro percorso, e poi questo prezzo aumenta di uno per ogni ulteriore tantometro percorso (i tassisti sono reticenti ad allontanarsi dalla loro città natia!).

Per esempio, nella seguente situazione:



Gabriele inizia prendendo il taxi a Brescia, la città numero 0, con un prezzo base di 10 euro al tantometro. Quindi lo utilizza per andare fino alla città numero 3, pagando  $10 + 11 + 12 = 33$  euro. A questo punto, preferisce cambiare taxi e per arrivare a Pinerolo spende ancora 11 euro, per un totale di 44.

Aiuta Gabriele a valutare se può permettersi di viaggiare in taxi, calcolando quanto costerebbe al minimo un tragitto da Brescia (città numero 0) a Pinerolo (città numero  $N$ ) in taxi!

## Implementazione

Dovrai sottoporre esattamente un file con estensione `.c`, `.cpp` o `.pas`.

🔗 Tra gli allegati a questo task troverai un template (`taxi.c`, `taxi.cpp`, `taxi.pas`) con un esempio di implementazione da completare.

Se sceglierai di utilizzare il template, dovrai implementare la seguente funzione:

C/C++	<code>int viaggia(int N, int C[]);</code>
Pascal	<code>function viaggia(N: longint; var C: array of longint): longint;</code>

In cui:

- L'intero  $N$  rappresenta il numero di città tra Brescia e Pinerolo (incluse).
- L'array  $C$ , indicizzato da 0 a  $N - 1$ , contiene i prezzi base dei taxi nelle varie città. Non è riportato il prezzo dei taxi a Pinerolo, che non è rilevante per la soluzione di questo problema.
- La funzione dovrà restituire il costo minimo in euro per un tragitto in taxi da 0 a  $N$ , che verrà stampato sul file di output.



## Dati di input

Il file `input.txt` è composto da due righe. La prima riga contiene l'unico intero  $N$ . La seconda riga contiene gli  $N$  interi  $C_i$  separati da uno spazio.

## Dati di output

Il file `output.txt` è composto da un'unica riga contenente un unico intero, la risposta a questo problema.

## Assunzioni

- $1 \leq N \leq 10\,000$ .
- $1 \leq C_i \leq 100\,000$  per ogni  $i = 0 \dots N - 1$ .

## Assegnazione del punteggio

Il tuo programma verrà testato su diversi test case raggruppati in subtask. Per ottenere il punteggio relativo ad un subtask, è necessario risolvere correttamente tutti i test relativi ad esso.

- **Subtask 1 [10 punti]:** Casi d'esempio.
- **Subtask 2 [20 punti]:**  $N \leq 10$ .
- **Subtask 3 [40 punti]:**  $N \leq 100$ .
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

## Esempi di input/output

input.txt	output.txt
4 10 15 12 11	44
input.txt	output.txt
12 27 21 99 35 71 23 64 5 10 44 1 1	184

## Spiegazione

Nel **primo caso di esempio**, Gabriele prende i taxi nelle città 0 e 3.

Nel **secondo caso di esempio**, Gabriele prende i taxi nelle città 0, 1, 5, 7, 10, 11.

## Finanza creativa (bilancio)

La *SteamPower S.P.A.*, azienda leader mondiale nel campo delle macchine a vapore portatili, non accenna ad uscire dal periodo di crisi nonostante i forti e ben ponderati tagli al personale di recente effettuati. Per fortuna il *CEO* ha avuto una nuova geniale idea: affidarsi al massimo esperto mondiale in campo di finanza creativa. L'esperto ha già ricevuto il bilancio dal reparto contabilità, e la situazione è disastrosa: arrivati a questo punto l'unico modo che conosce per assicurare gli azionisti e prendere tempo è quello di effettuare qualche piccolo ritocco ai libri contabili.

Più precisamente, vuole ricorrere al suo fedele bianchetto per correggere il totale  $U$  delle uscite. Inoltre, grazie agli insegnamenti di lunghi anni di esperienza, sa che per contenere i rischi dell'operazione non deve assolutamente eccedere le  $K$  cifre cancellate (sulle  $N$  complessive del totale  $U$ ).

Aiuta l'esperto di finanza creativa a trovare il minimo intero ottenibile cancellando  $K$  cifre dall'intero  $U$ !

## Implementazione

Dovrai sottoporre esattamente un file con estensione `.c`, `.cpp` o `.pas`.

📎 Tra gli allegati a questo task troverai un template (`bilancio.c`, `bilancio.cpp`, `bilancio.pas`) con un esempio di implementazione da completare.

Se sceglierai di utilizzare il template, dovrai implementare la seguente funzione:

C/C++	<code>void bianchetta(int N, int K, int U[], int C[]);</code>
Pascal	<code>procedure bianchetta(N, K: longint; var U, C: array of longint);</code>

In cui:

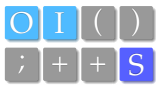
- L'intero  $N$  rappresenta il numero di cifre del totale delle uscite.
- L'intero  $K$  rappresenta il numero di cifre da cancellare.
- L'array  $U$ , indicizzato da 0 a  $N - 1$ , contiene l'elenco delle cifre del totale delle uscite, con la cifra più significativa in posizione 0 e così via fino alla cifra delle unità in posizione  $N - 1$ .
- La funzione dovrà riempire l'array  $C$ , indicizzato da 0 a  $N - K - 1$ , con l'elenco delle cifre rimaste dopo la cancellazione (dalla più alla meno significativa), che verrà stampato sul file di output.

## Dati di input

Il file `input.txt` è composto da due righe. La prima riga contiene i due interi  $N$  e  $K$ . La seconda riga contiene le  $N$  cifre  $U_i$  del totale delle uscite  $U$ , separate da uno spazio.

## Dati di output

Il file `output.txt` è composto da un'unica riga contenente  $N - K$  cifre separate da uno spazio, la risposta a questo problema.



## Assunzioni

- $1 \leq K < N \leq 1\,000\,000$ .
- $0 \leq U_i \leq 9$  per ogni  $i = 0 \dots N - 1$ , e  $U_0 \geq 1$ .

## Assegnazione del punteggio

Il tuo programma verrà testato su diversi test case raggruppati in subtask. Per ottenere il punteggio relativo ad un subtask, è necessario risolvere correttamente tutti i test relativi ad esso.

- **Subtask 1 [10 punti]:** Casi d'esempio.
- **Subtask 2 [20 punti]:**  $N \leq 10$ .
- **Subtask 3 [30 punti]:**  $N \leq 200$ .
- **Subtask 3 [20 punti]:**  $N \leq 10\,000$ .
- **Subtask 4 [20 punti]:** Nessuna limitazione specifica.

## Esempi di input/output

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

## Spiegazione

Nel **primo caso di esempio** conviene cancellare le due cifre più alte (5 e 9).

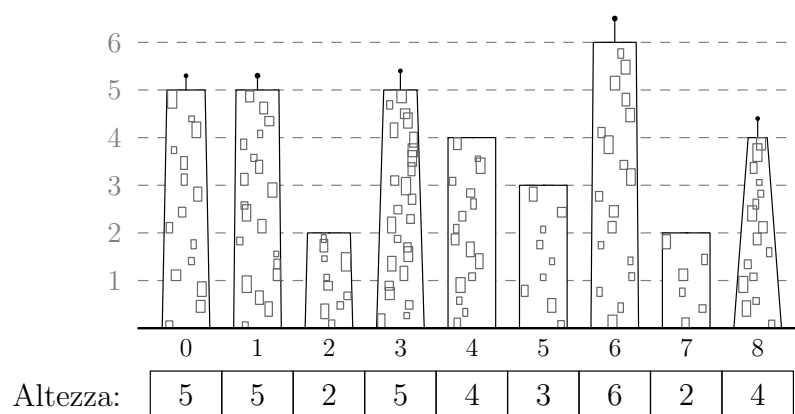
Nel **secondo caso di esempio** conviene cancellare le due cifre più significative (9 e 4), e la restante cifra più alta (il 7).

Nel **terzo caso di esempio** conviene cancellare le tre cifre più significative (1, 3 e 2), il 2 ancora successivo e la restante cifra più alta (l'8).

## Fila di grattacieli (grattacieli)

Giorgio è in visita a Toronto, città famosa per il suo skyline di  $N$  grattacieli di varia altezza tutti disposti lungo un'unica fila. Giorgio è appassionato di grattacieli, e quindi intende salire su uno di essi e approfittare dell'altezza per ammirare il panorama (e cioè, gli altri grattacieli). Giorgio sa che dalla cima di un grattacielo ammirerà appieno tutti i grattacieli (compreso quello su cui si trova) che non sono coperti da un altro grattacielo. Un grattacielo viene coperto da un altro (più vicino di questo al grattacielo su cui Giorgio si trova) se questo altro grattacielo è sia alto almeno quanto il grattacielo che copre e sia alto almeno quanto il grattacielo su cui Giorgio si trova.

Ad esempio, consideriamo questo skyline:



In questo scenario, se Giorgio salisse sul grattacielo 3, di altezza 5, sarebbe in grado di osservare solo i grattacieli da 1 a 6 compresi. Infatti il grattacielo 0 è coperto dal grattacielo 1, alto 5; i grattacieli 7 e 8 sono coperti alla vista dal grattacielo 6.

Aiuta Giorgio a scegliere su quale grattacielo salire, calcolando qual è il massimo numero di grattacieli che Giorgio può ammirare se sceglie nel modo ottimo il grattacielo su cui salire.

### Dati di input

Il file `input.txt` è composto da due righe. La prima riga contiene l'unico intero  $N$ . La seconda riga contiene  $N$  interi separati da uno spazio, le altezze  $H_i$  dei grattacieli.

### Dati di output

Il file `output.txt` è composto da un'unica riga contenente un unico intero, la risposta a questo problema.

### Implementazione

Dovrai sottoporre esattamente un file con estensione `.c`, `.cpp` o `.pas`.

📁 Tra gli allegati a questo task troverai un template (`grattacieli.c`, `grattacieli.cpp`, `grattacieli.pas`) con un esempio di implementazione da completare.

Se sceglierai di utilizzare il template, dovrai implementare la seguente funzione:



C/C++	<code>int osserva(int N, int H[]);</code>
Pascal	<code>function osserva(N: longint; var H: array of longint): longint;</code>

In cui:

- L'intero  $N$  rappresenta il numero di grattacieli presenti nella skyline.
- L'array  $H$ , indicizzato da  $0$  a  $N - 1$ , contiene le altezze  $H_i$  dei grattacieli nell'ordine in cui sono disposti lungo la skyline.
- La funzione dovrà restituire il massimo numero di grattacieli che è possibile ammirare appieno, che verrà stampato sul file di output.

## Assunzioni

- $1 \leq N \leq 100\,000$ .
- $1 \leq H_i \leq 1\,000\,000$  per ogni  $i = 0 \dots N - 1$ .

## Assegnazione del punteggio

Il tuo programma verrà testato su diversi test case raggruppati in subtask. Per ottenere il punteggio relativo ad un subtask, è necessario risolvere correttamente tutti i test relativi ad esso.

- **Subtask 1 [10 punti]:** Casi d'esempio.
- **Subtask 2 [20 punti]:**  $N \leq 10$ .
- **Subtask 3 [40 punti]:**  $N, H_i \leq 1000$ .
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

## Esempi di input/output

input.txt	output.txt
9 5 5 2 5 4 3 6 2 4	9
input.txt	output.txt
6 9 4 2 9 9 7	5

## Spiegazione

Nel **primo caso di esempio**, corrispondente all'esempio del testo, è possibile ammirare appieno tutti i grattacieli salendo sul grattacielo 6, di altezza 6.

Nel **secondo caso di esempio**, è possibile ammirare al massimo 5 grattacieli, salendo sul grattacielo 3, di altezza 9.

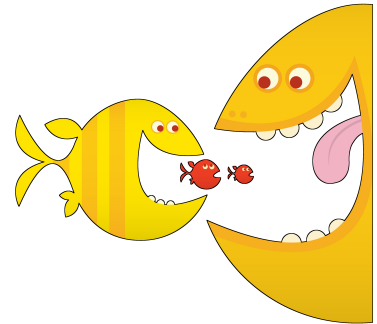


## Pesci Mangioni (pesci)

Al corso di *Ittiologia Computazionale* Gabriele sta studiando le dinamiche algoritmiche del mondo dei pesci. Un argomento particolarmente interessante è quello dell'alimentazione dei *Pesci Mangioni*.

Questi voracissimi pesci, di dimensioni molto variegata, sono molto aggressivi e ogni volta che si sentono osservati da un'altro pesce reagiscono mangiandoselo (se possono), senza mai sentire il senso di sazietà. Per compito Gabriele ha stilato un modello semplificato del loro comportamento, che dovrebbe essere valido almeno nel caso in cui i pesci si trovino in un tubo da sperimentazione. Secondo il modello:

- ogni pesce può nuotare da sinistra verso destra o da destra verso sinistra, e stando nel tubo questa direzione non cambierà mai,
- ogni pesce ha una grandezza ben definita, e tutte le grandezze sono distinte,
- un incrocio di pesci è una coppia di pesci adiacenti, per cui il più a sinistra nuota verso destra, e il più a destra nuota verso sinistra,
- durante un incrocio di pesci il pesce più grande mangia il pesce più piccolo e continua nel suo moto, eventualmente incrociando altri pesci.



Per verificare quanto è buono il modello che ha derivato di questa specie curiosa, Gabriele intende ora liberare in un tubo  $N$  Pesci Mangioni e monitorare cosa succede. Quanti pesci dovrebbero rimanere vivi dopo che tutti gli incroci si saranno risolti?

## Implementazione

Dovrai sottoporre esattamente un file con estensione `.c`, `.cpp` o `.pas`.

🔗 Tra gli allegati a questo task troverai un template (`pesci.c`, `pesci.cpp`, `pesci.pas`) con un esempio di implementazione da completare.

Se sceglierai di utilizzare il template, dovrai implementare la seguente funzione:

C/C++	<code>int mangia(int N, int direzione[], int dimensione[]);</code>
Pascal	<code>function mangia(N: longint; var direzione, dimensione: array of longint): longint;</code>

In cui:

- L'intero  $N$  rappresenta il numero di pesci liberati da Gabriele.
- L'array `direzione`, indicizzato da  $0$  a  $N - 1$ , indica se l' $i$ -esimo pesce nuota da sinistra verso destra (`direzione[i] = 0`) o da destra verso sinistra (`direzione[i] = 1`).
- L'array `dimensione`, indicizzato da  $0$  a  $N - 1$ , indica la dimensione dell' $i$ -esimo pesce.
- La funzione dovrà restituire il numero di Pesci Mangioni sopravvissuti dopo che tutti gli incroci sono stati risolti, che verrà stampato sul file di output.

## Dati di input

Il file `input.txt` è composto da  $N + 1$  righe. La prima riga contiene l'unico intero  $N$ . Le successive  $N$  righe contengono ciascuna i due interi `direzione[i]`, `dimensione[i]`.



## Dati di output

Il file `output.txt` è composto da un'unica riga contenente un unico intero, la risposta a questo problema.

## Assunzioni

- $1 \leq N \leq 100\,000$ .
- `direzione[i]`  $\in \{0, 1\}$  per ogni  $i = 0 \dots N - 1$ .
- $1 \leq \text{dimensione}[i] \leq 10^9$  per ogni  $i = 0 \dots N - 1$ ; inoltre, le dimensioni dei pesci sono tutte distinte.

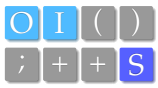
## Assegnazione del punteggio

Il tuo programma verrà testato su diversi test case raggruppati in subtask. Per ottenere il punteggio relativo ad un subtask, è necessario risolvere correttamente tutti i test relativi ad esso.

- **Subtask 1 [10 punti]**: Casi d'esempio.
- **Subtask 2 [20 punti]**:  $N \leq 10$ .
- **Subtask 3 [40 punti]**:  $N \leq 100$ .
- **Subtask 4 [30 punti]**: Nessuna limitazione specifica.

## Esempi di input/output

input.txt	output.txt
2 0 6 1 9	1
input.txt	output.txt
2 1 6 1 9	2
input.txt	output.txt
5 0 12 0 9 1 3 0 4 1 6	2



## Spiegazione

Nel **primo caso di esempio** il secondo pesce, di dimensione 9, mangia il primo, di dimensione 6.

Nel **secondo caso di esempio** i pesci non si incrociano.

Nel **terzo caso di esempio** il pesce di dimensione 4 viene mangiato dal pesce di dimensione 6, ma il pesce di dimensione 9 mangia sia il pesce di dimensione 3 che quello di dimensione 6.

## Tagli al personale (licenziamenti)

La *SteamPower S.P.A.*, azienda leader mondiale nel campo delle macchine a vapore portatili, sta attraversando un brutto periodo di crisi e per tenerla in piedi nell'attesa che l'economia riparta la dirigenza ha stabilito un regime di forti tagli al personale.

Per stabilire chi deve essere licenziato, il reparto personale ha distribuito un test con cui ha stabilito accuratamente di ognuno degli  $N$  dipendenti il suo personale valore di bravura  $B_i$ . Inoltre si baserà anche sullo studio dell'organigramma (cioè il grafico ad albero in cui tutti i dipendenti dell'azienda sono organizzati secondo le relazioni di dipendenza), e conosce quindi il capo diretto  $C_i$  di ogni dipendente  $i$ . Per convenzione, il presidente dell'azienda è segnato al numero 0 e ha  $C_i = -1$  per indicare l'assenza di un capo diretto.

Le direttive presidenziali hanno stabilito che per cominciare verranno licenziati tutti i dipendenti *inadequati*, e cioè quelli che hanno valore di bravura  $B_i$  strettamente maggiore di quello di uno dei loro capi (diretti o indiretti). Aiuta il reparto personale a determinare quante persone dovranno essere licenziate!

### Dati di input

Il file `input.txt` è composto da  $N + 1$  righe. La prima riga contiene l'unico intero  $N$ . Le successive  $N$  righe contengono ciascuna due interi separati da uno spazio, i valori  $B_i$  e  $C_i$  del dipendente  $i$ -esimo.

### Dati di output

Il file `output.txt` è composto da un'unica riga contenente un unico intero, la risposta a questo problema.

### Implementazione

Dovrai sottoporre esattamente un file con estensione `.c`, `.cpp` o `.pas`.

🔗 Tra gli allegati a questo task troverai un template (`licenziamenti.c`, `licenziamenti.cpp`, `licenziamenti.pas`) con un esempio di implementazione da completare.

Se sceglierai di utilizzare il template, dovrai implementare la seguente funzione:

C/C++	<code>int licenzia(int N, int B[], int C[]);</code>
Pascal	<code>function licenzia(N: longint; var B, C: array of longint): longint;</code>

In cui:

- L'intero  $N$  rappresenta il numero totale di dipendenti dell'azienda.
- L'array  $B$ , indicizzato da 0 a  $N - 1$ , contiene i valori di bravura  $B_i$  dei dipendenti.
- L'array  $C$ , indicizzato da 0 a  $N - 1$ , contiene gli indici dei capi diretti  $C_i$  (sempre da 0 a  $N - 1$ ) dei dipendenti corrispondenti. Il valore  $C_0$  è garantito essere pari a  $-1$  e indica che il presidente è sempre il dipendente numero 0.
- La funzione dovrà restituire il numero di dipendenti inadeguati, che verrà stampato sul file di output.

### Assunzioni

- $1 \leq N \leq 100\,000$ .
- $0 \leq B_i \leq 1\,000\,000$  per ogni  $i = 0 \dots N - 1$ .
- $0 \leq C_i \leq N - 1$  per ogni  $i = 1 \dots N - 1$ , mentre  $C_0 = -1$ .
- È garantito che l'organigramma rappresenti effettivamente un albero, e cioè che risalendo di capo in capo a partire da ogni dipendente  $i$  si arrivi sempre al presidente 0.

## Assegnazione del punteggio

Il tuo programma verrà testato su diversi test case raggruppati in subtask. Per ottenere il punteggio relativo ad un subtask, è necessario risolvere correttamente tutti i test relativi ad esso.

- **Subtask 1 [10 punti]:** Casi d'esempio.
- **Subtask 2 [20 punti]:**  $N \leq 10$ .
- **Subtask 3 [30 punti]:**  $N \leq 100$ .
- **Subtask 4 [10 punti]:** L'organigramma forma una linea retta, e quindi  $C_i = i - 1$  per ogni  $i$ .
- **Subtask 5 [30 punti]:** Nessuna limitazione specifica.

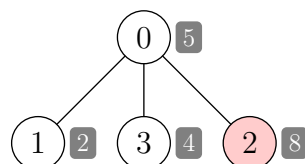
## Esempi di input/output

input.txt	output.txt
4 5 -1 2 0 8 0 4 0	1

input.txt	output.txt
7 4 -1 2 0 1 4 2 4 6 0 3 1 5 4	3

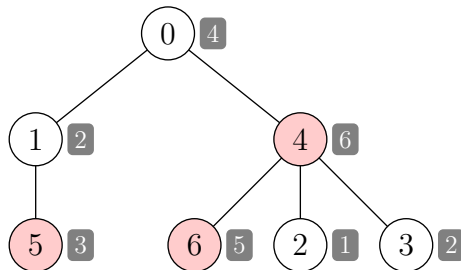
## Spiegazione

Il **primo caso di esempio** corrisponde a questo organigramma, in cui i numeri bianchi nei riquadri scuri rappresentano il valore di bravura dei dipendenti:



Il dipendente 2 ha un valore di bravura superiore a quello del suo capo (il presidente dell'azienda), e per questo deve essere licenziato.

Il **secondo caso di esempio** corrisponde invece a questo organigramma:



I dipendenti 4 e 5 hanno un valore di bravura maggiore dei loro rispettivi capi, e quindi vanno licenziati. Il dipendente 6 ha un valore di bravura minore di quello del suo capo, ma maggiore di quello del capo del suo capo, e quindi va licenziato.

## Trampolino elastico (trampolino)

Dopo il successo dello spettacolo con le piroette, Giorgio si è assicurato una brillante carriera nel mondo della coreografia. Per il prossimo spettacolo Giorgio sta pensando a qualcosa di decisamente più audace e dinamico: una lunghissima fila di trampolini elastici, ognuno a un metro di distanza dal precedente. Al termine della fila di trampolini è posto un tappetone elastico.

Ogni trampolino elastico è dotato di una elasticità  $E$ , che rappresenta il numero massimo di metri di lunghezza che è possibile compiere con un salto su quel trampolino. Ad esempio, se  $E = 1$ , l'acrobata dopo un balzo può trovarsi solo al trampolino successivo, mentre se il trampolino corrente ha  $E = 3$  l'atleta può dosare la forza del salto e trovarsi in uno dei 3 trampolini successivi al corrente.

Data la sequenza dei trampolini e delle loro elasticità, aiuta Giorgio a determinare quale è il minimo numero di salti che è necessario che compiano gli acrobati per terminare sul tappetone, sapendo che il primo balzo avviene obbligatoriamente sul primo trampolino.

### Implementazione

🔗 Tra gli allegati a questo task troverai un template (`trampolino.c`, `trampolino.cpp`, `trampolino.pas`) con un esempio di implementazione da completare.

Se sceglierai di utilizzare il template, dovrai implementare la seguente funzione:

C/C++	<code>int salta(int N, int E[]);</code>
Pascal	<code>function salta(N: longint; var E: array of longint): longint;</code>

In cui:

- L'intero  $N$  rappresenta il numero di trampolini presenti in scena.
- L'array  $E$ , indicizzato da 0 a  $N - 1$ , contiene l'elasticità dei trampolini.
- La funzione dovrà restituire il minimo numero di salti necessari per finire sul tappetone, che verrà stampato sul file di output.

### Dati di input

Il file `input.txt` è composto da due righe. La prima riga contiene l'unico intero  $N$ . La seconda riga contiene gli  $N$  interi  $E_i$  separati da uno spazio.

### Dati di output

Il file `output.txt` è composto da un'unica riga contenente un unico intero, la risposta a questo problema.

## Assunzioni

- $1 \leq N \leq 100\,000$ .
- $1 \leq E_i \leq 100\,000$  per ogni  $i = 0 \dots N - 1$ .
- Non è possibile saltare all'indietro.
- È obbligatorio che il primo salto avvenga sul primo trampolino.

## Assegnazione del punteggio

Il tuo programma verrà testato su diversi test case raggruppati in subtask. Per ottenere il punteggio relativo ad un subtask, è necessario risolvere correttamente tutti i test relativi ad esso.

- **Subtask 1 [10 punti]:** Casi d'esempio.
- **Subtask 2 [20 punti]:**  $N \leq 100$ .
- **Subtask 3 [40 punti]:**  $N \leq 1000$ .
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

## Esempi di input/output

input.txt	output.txt
4 2 3 1 1	2
input.txt	output.txt
5 5 2 3 4 5	1
input.txt	output.txt
8 4 2 3 1 1 2 1 2	4

## Spiegazione

Nel **primo caso di esempio** conviene dosare il salto sul primo trampolino in modo da arrivare al secondo trampolino e da qui arrivare al tappetone. Saltare dal primo trampolino al terzo sarebbe costato 3 salti invece di 2.

Nel **secondo caso di esempio** gli atleti saltano dal primo trampolino direttamente sul tappetone.

Il **terzo caso di esempio** corrisponde alla figura. Il colore del centro dei trampolini ha intensità proporzionale all'elasticità.

