

Dadi truccati (dadi)

L'ultima partita a Monopoly tra Gabriele e Giorgio è finita in tragedia dopo che Gabriele è stato costretto a ipotecare metà dei suoi terreni per coprire il costo dell'albergo di Giorgio in Parco della Vittoria.

Per evitare che la spiacevole situazione si ripeta un'altra volta, Gabriele sta mettendo in pratica gli insegnamenti del corso di ingegneria per costruire dei dadi robotici truccati. Il piano è molto astuto: i dadi sono progettati per fare in modo che Giorgio finisca sempre sulla casella più sfortunata, e che Gabriele si salvi sempre e paghi il minimo possibile ad ogni turno. Per raggiungere l'obiettivo, i dadi hanno all'interno un minuscolo giroscopio che permette loro di ruotare in modo da mostrare la faccia che decide Gabriele.

Questo contratto vale L. 40.000	
CONTRATTO	
PARCO della VITTORIA	
Rendita - Solo terreno L.	5.000
» Con 1 Casa »	20.000
» Con 2 Case »	60.000
» Con 3 Case »	140.000
» Con 4 Case »	170.000
» Con Albergo »	200.000
Se un giocatore possiede tutti i terreni d'uno stesso Gruppo (colore), la rendita del solo terreno viene raddoppiata.	
Costo di ogni Casa L.	20.000
» di un Albergo »	20.000
più 4 Case	
Valore ipotecario . . L.	20.000

Gabriele non è uno sprovveduto, e sa che per non essere scoperto è necessario che nel lancio i dadi si muovano in modo realistico, compiendo molte rotazioni, all'apparenza casuali, ma che in realtà sono dettate dal software.

In questo momento Gabriele sta debuggando proprio questa parte di codice, verificando che i dadi compiano correttamente la sequenza di rotazioni imposta dal software. All'inizio della sequenza il dado si trova sempre nella posizione indicata in figura:

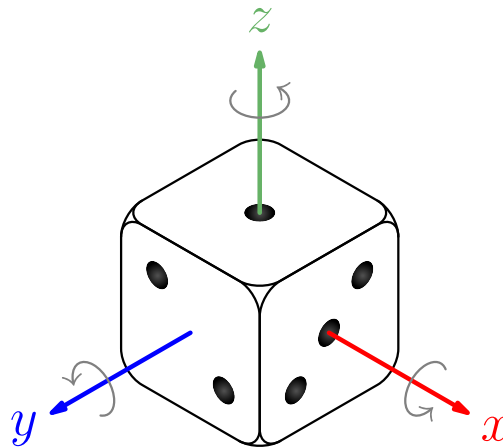


Figura 1. Situazione iniziale del dado.

Il driver che si occupa della rotazione accetta in input tre comandi: X, Y, e Z. Questi comandi hanno l'effetto di far compiere al dado una rotazione di 90 gradi rispetto all'asse scelto, in senso antiorario come mostrato dalle frecce grigie. Qui sotto sono rappresentati gli effetti dei comandi X, Y, e Z sul dado di Figura 1.

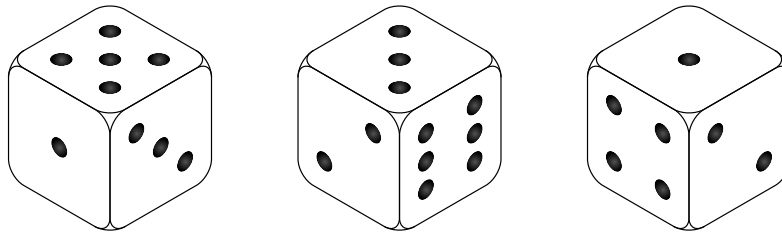


Figura 2. Effetti dei comandi X, Y, e Z sul dado di Figura 1.

Il driver supporta inoltre 3 comandi di interrogazione:

- **T**: ritorna il valore sulla faccia superiore del dado, ovvero quella corrispondente al verso positivo dell'asse z ;
- **F**: ritorna il valore sulla faccia frontale del dado, ovvero quella corrispondente al verso positivo dell'asse y ;
- **R**: ritorna il valore sulla faccia destra del dado, ovvero quella corrispondente al verso positivo dell'asse x .

Aiuta Gabriele a debuggare i dadi truccati, scrivendo un software di simulazione che, ricevuta la sequenza di comandi inviati al dado, calcoli quali sono le risposte corrette alle varie richieste T, F e R inoltrate al driver.

Implementazione

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

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

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

C/C++	<code>void simula(int N, char C[], int R[]);</code>
Pascal	<code>procedure simula(N: longint; var C: array of char; var R: array of longint);</code>

In cui:

- L'intero N rappresenta il numero di comandi inoltrati al driver.
- L'array C , indicizzato da 0 a $N - 1$, contiene i comandi (un carattere tra X, Y, Z, T, F, R).
- La funzione dovrà riempire l'array di interi R , indicizzato da 0 a $M - 1$, dove M rappresenta il numero di comandi di interrogazione (ovvero T, F, R). Nella posizione i scrivere la risposta all' i -esimo comando di interrogazione.

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 caratteri C_i , senza spazi.

Dati di output

Il file `output.txt` è composto da un'unica riga contenente gli M interi di risposta.

Assunzioni

- $1 \leq N \leq 100\,000$.
- I comandi C_i sono sempre un carattere tra X, Y, Z, T, F e R.

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.

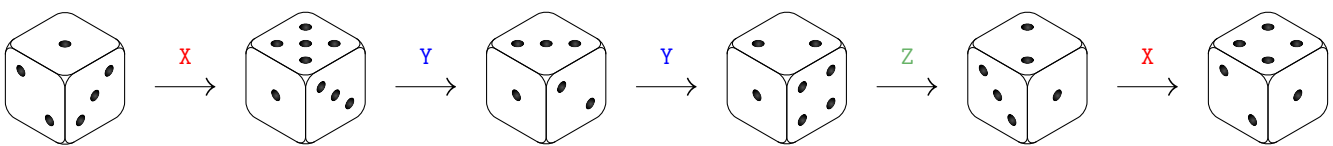
- **Subtask 1 [10 punti]:** Casi d'esempio.
- **Subtask 2 [20 punti]:** $N \leq 100$.
- **Subtask 3 [40 punti]:** $N \leq 1000$.
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

Esempi di input/output

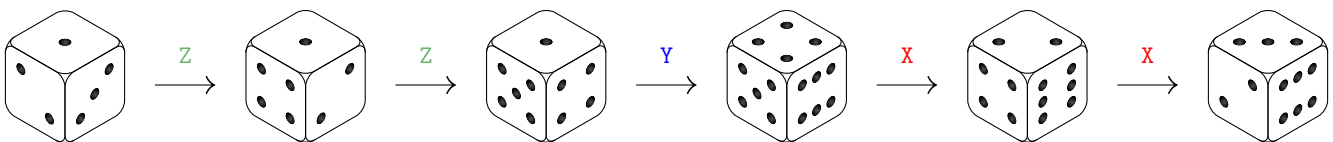
input.txt	output.txt
9 XYTZFXFR	2 3 2 1
input.txt	output.txt
10 TZZYXFRTXR	1 4 6 2 6

Spiegazione

Nel **primo caso di esempio** il dado subisce queste rotazioni:



Nel **secondo caso di esempio** il dado subisce queste rotazioni:



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

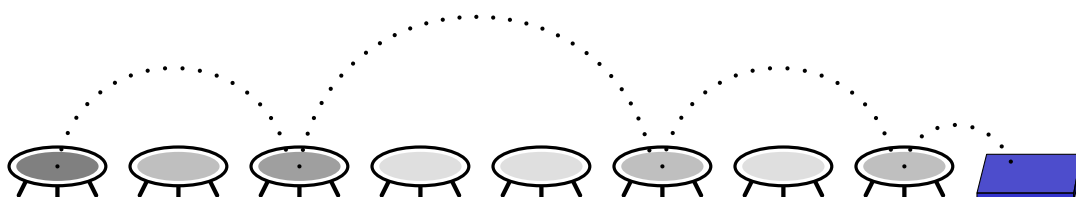
input.txt	output.txt
4 2 3 1 1	2
input.txt	output.txt
5 5 2 3 4 5	1
input.txt	output.txt
8 4 2 3 1 1 2 1 2	4

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.



Edilizia fantasiosa (costruzioni)

La *SteamPower S.P.A.* è di nuovo alla ricerca di una soluzione rapida ed efficace per far quadrare i conti. Per risolvere la questione, il consiglio di amministrazione ha deciso di nascondere gli illeciti della società tramite la costruzione di un lunghissimo centro sportivo destinato allo spettacolo dei trampolini di Giorgio. Con una sola mossa, in altre parole, la società riuscirà a celare le irregolarità nei bilanci e farsi una grossa pubblicità, e tutti i dirigenti sono entusiasti (già si parla di un torneo di rugby...).

Giorgio è stato tassativo sui requisiti della costruzione: questa dovrà essere lunga esattamente K metri, non uno di meno e non uno di più. Per accomodare la richiesta la società si è affidata al proprio consulente di fiducia, Gabriele.

Gabriele ha individuato la via più lunga della città, e sta valutando in che punto esatto posizionare la struttura per risparmiare il più possibile. Infatti, la via è completamente edificata e per poter iniziare a costruire il palazzo è necessario prima demolire le costruzioni già esistenti sul terreno, pagando un costo proporzionale al loro valore.

La strada è lunga N metri, e per ogni metro di edifici Gabriele ne ha stimato il valore V_i . Ora deve scegliere K metri consecutivi di strada, in modo da minimizzare la somma dei valori degli edifici da demolire per fare spazio al centro sportivo.

Implementazione

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

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

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

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

In cui:

- L'intero N rappresenta la lunghezza, in metri, della strada.
- L'intero K rappresenta la lunghezza, in metri, del centro sportivo da costruire.
- L'array V , indicizzato da 0 a $N - 1$, contiene il valore degli edifici presenti nel metro i -esimo di strada.
- La funzione dovrà restituire la minima somma di valori di un segmento di esattamente K metri consecutivi di strada, che verrà stampato sul file di output.

Dati di input

Il file `input.txt` è composto da due righe. La prima riga contiene gli interi N e K . La seconda riga contiene gli N interi V_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

- $1 \leq K \leq N \leq 100\,000$.
- $1 \leq V_i \leq 10\,000$ 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]:** $N \leq 100$.
- **Subtask 3 [40 punti]:** $N \leq 10\,000$, $K \leq 50$.
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

Esempi di input/output

input.txt	output.txt
5 3 5 7 6 4 8	17
input.txt	output.txt
9 7 1 2 3 4 5 4 3 1 1	21

Spiegazione

Nel **primo caso di esempio** conviene demolire gli edifici di valore 7, 6, 4.

Nel **secondo caso di esempio** conviene demolire gli ultimi 7 edifici.

Illuminazione di sala (luci)

Il nuovo centro sportivo è pronto, e lo spettacolo di trampolini di Giorgio sta per debuttare! Come sempre, durante lo spettacolo Giorgio starà alla regia occupandosi personalmente dell'illuminazione di sala, di modo da ricreare l'atmosfera giusta in ogni momento dello spettacolo.


Il lungo e stretto palco è illuminato da una fila di N lampadine (numerata da 1 a N) che, come richiesto da Giorgio stesso, sono controllate da altrettanti interruttori (numerati da 1 a N) in modo un po' peculiare. Precisamente, l'interruttore i -esimo si occupa di cambiare lo stato di tutte le lampadine in posizione multipla di i . Solo in questo modo Giorgio potrà cambiare velocemente e con precisione la luminosità globale del palco nella maniera che preferisce.

Per lo spettacolo, Giorgio ha stabilito che le lampadine partiranno tutte accese, e che successivamente verranno premuti tutti gli N interruttori una singola volta. Più precisamente, Giorgio inizierà dall'interruttore M , e poi inizierà a fare la "conta": tra gli interruttori non ancora premuti selezionerà il K -esimo dopo M (circolarmente, quindi continuando a contare dal primo interruttore ogni volta che arriva all'ultimo), poi ancora il K -esimo ancora non premuto dopo il precedente, e così via.

Aiuta Giorgio a scrivere un programma che calcoli quante lampadine rimarranno accese alla fine!

Implementazione

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

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

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

C/C++	<code>int spegni(int N, int M, int K);</code>
Pascal	<code>function spegni(N, M, K: longint): longint;</code>

In cui:

- L'intero N rappresenta il numero di lampadine e interruttori.
- L'intero M rappresenta il primo interruttore a venire premuto.
- L'intero K rappresenta ogni quanti interruttori si svolge la conta.
- La funzione dovrà restituire il numero di lampadine ancora accese alla fine, che verrà stampato sul file di output.

Dati di input

Il file `input.txt` è composto da una riga contenente i tre interi N , M , K .

Dati di output

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



Assunzioni

- $1 \leq M \leq N \leq 1\,000\,000\,000$.
- $1 \leq K \leq 1\,000\,000\,000$.

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]:** $K, N \leq 100$.
- **Subtask 3 [20 punti]:** $N \leq 10\,000$.
- **Subtask 4 [20 punti]:** $N \leq 1\,000\,000$.
- **Subtask 5 [30 punti]:** Nessuna limitazione specifica.

Esempi di input/output

input.txt	output.txt
2 1 5	1

input.txt	output.txt
6 3 3	4

Spiegazione

Nel **primo caso di esempio**, vengono premute le lampadine 1 e poi 2 nell'ordine e alla fine rimane accesa la lampadina numero 2.

Nel **secondo caso di esempio**, vengono premute le lampadine 3, 6, 4, 2, 5, 1 e alla fine rimangono accese tutte le lampadine tranne la numero 1 e la numero 4.

Torneo di rugby (rugby)

La *SteamPower S.P.A.*, azienda leader mondiale nel campo delle macchine a vapore portatili, grazie ai tagli al personale, agli espedienti di finanza creativa e alle operazioni di immagine recentemente effettuate sembra essere momentaneamente fuori pericolo. Tuttavia, il fallimento è sempre dietro l'angolo e il *CEO* non vuole farsi cogliere impreparato. Per recuperare popolarità e ottenere un po' di pubblicità gratuita, ha quindi deciso di impegnare alcuni dei suoi dipendenti nel *Torneo Intercostale* di Rugby, una competizione che coinvolge squadre provenienti da zone costiere di tutto il mondo.

L'incarico di formare la squadra è stato affidato al consulente di fiducia dell'azienda, Gabriele, che ha iniziato effettuando dei test attitudinali individuando la bravura B_i degli N dipendenti dell'azienda. Gabriele sa bene che la bravura di una squadra si ottiene molto semplicemente sommando i valori di bravura dei suoi componenti. Ma sa anche che ancor più della bravura dei singoli conta lo spirito di squadra: e questo si perde completamente se due dei componenti della squadra sono in *conflitto di interessi*, e cioè uno il capo (diretto o indiretto) dell'altro nella gerarchia dell'azienda.

Pertanto Gabriele si è anche procurato l'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. Aiuta Gabriele a trovare la squadra più forte (con massima somma della bravura dei componenti) senza conflitti di interessi!

Implementazione

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

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

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

C/C++	<code>int recluta(int N, int B[], int C[]);</code>
Pascal	<code>function recluta(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 la massima bravura per una squadra senza conflitti di interessi, che verrà stampata 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 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.

Assunzioni

- $1 \leq N \leq 10\,000$.
- $1 \leq B_i \leq 100\,000$ 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.
- Non ci sono limiti minimi o massimi per la dimensione di una squadra nel Torneo Intercostale.

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]:** Con l'eccezione del presidente, tutti gli altri dipendenti hanno al massimo un dipendente diretto.
- **Subtask 4 [40 punti]:** Nessuna limitazione specifica.

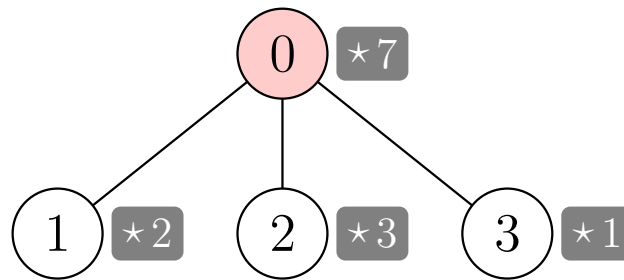
Esempi di input/output

input.txt	output.txt
<pre>4 7 -1 2 0 3 0 1 0</pre>	<pre>7</pre>

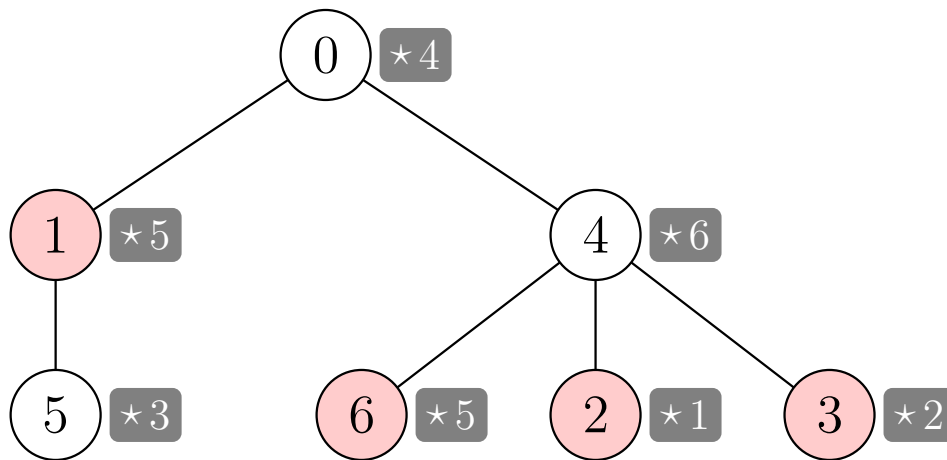
input.txt	output.txt
<pre>7 4 -1 5 0 1 4 2 4 6 0 3 1 5 4</pre>	<pre>13</pre>

Spiegazione

Nel **primo caso di esempio**, la squadra più forte consiste del solo presidente, come evidenziato nel seguente organigramma in cui i numeri bianchi nei riquadri scuri rappresentano il valore di bravura dei dipendenti.



Nel **secondo caso di esempio**, la squadra più forte è evidenziata nel seguente organigramma.



Spesa lampo (spesa)

Gabriele ha appena finito la sua ultima consulenza, e si prepara ad andare a casa della nonna per un meritato pranzo. Date le precedenti esperienze, decide però che è saggio prima passare da un supermercato a procurarsi un po' di bicarbonato, nel caso in cui i suoi propositi di mangiare il meno possibile falliscano miseramente dopo la dodicesima fetta di tiramisù.

Per non fare aspettare la nonna, questa deviazione deve costare il minor tempo possibile. Per questo Gabriele si è procurato la mappa dei supermercati della città, composta da N punti di interesse (incroci, piazze, edifici importanti) e M vie (a doppio senso di marcia) che collegano i punti di interesse. In particolare, K di questi punti rappresentano dei supermercati, mentre il punto di interesse indicato col numero 1 rappresenta il palazzo della *SteamPower S.P.A.*, dove si trova Gabriele in questo momento, e il punto numerato N è il condominio dove abita la nonna.

Data la mappa sopra descritta, Gabriele si chiede quanto tempo ci metterà (al minimo) per raggiungere la nonna, dovendo prima passare da un supermercato, e sapendo che ci vuole 1 minuto a percorrere ognuna delle M strade.

Implementazione

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

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

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

C/C++	<code>int compra(int N, int M, int K, int supermercati[], int da[], int a[]);</code>
Pascal	<code>function compra(N, M, K: longint; var supermercati, da, a: array of longint): longint;</code>

In cui:

- L'intero N rappresenta il numero di punti di interesse della città (numerati da 1 a N).
- L'intero M rappresenta il numero di strade (bidirezionali) che collegano i punti di interesse.
- L'intero K rappresenta il numero di supermercati presenti.
- L'array `supermercati`, indicizzato da 0 a $K - 1$, contiene i numeri dei punti di interesse che corrispondono ai supermercati della città.
- Gli array `da` e `a`, indicizzati da 0 a $M - 1$, contengono gli estremi delle strade, cioè i numeri dei punti di interesse che queste strade collegano.
- La funzione dovrà restituire il minimo numero di minuti necessari per raggiungere un supermercato e poi andare a casa della nonna, che verrà stampato sul file di output.

Dati di input

Il file `input.txt` è composto da $M + 2$ righe. La prima riga contiene gli interi N, M, K . La seconda riga contiene i K interi `supermercati[i]` separati da uno spazio. Le successive M righe contengono le descrizioni delle strade: la i -esima di queste righe contiene i due interi `da[i]` e `a[i]`.

Dati di output

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

Assunzioni

- $3 \leq N \leq 10\,000$.
- $2 \leq M \leq 100\,000$.
- $1 \leq K \leq N - 2$.
- $1 < \text{supermercati}_i < N$ per ogni $i = 0 \dots K - 1$.
- Nei punti 1 ed N non sono presenti supermercati, e i valori $\text{supermercati}[i]$ non sono ripetuti.
- Tutte le strade sono bidirezionali e non sono duplicate nell'input.
- È possibile da ogni punto raggiungere ogni altro punto della città.
- Gabriele può percorrere la stessa strada più di una volta, o passare più volte da uno stesso punto.
- Gabriele impiega 1 minuto a percorrere ogni strada.
- Il tempo speso all'interno del supermercato è trascurabile.

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]:** $N, K \leq 100$.
- **Subtask 3 [40 punti]:** $K \leq 10$.
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

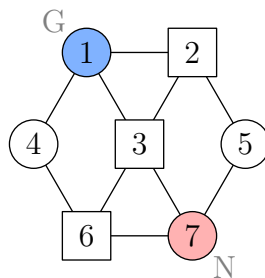
Esempi di input/output

input.txt	output.txt
<pre> 7 10 3 3 6 2 1 2 2 5 7 3 4 6 3 6 6 7 7 5 1 4 3 1 2 3 </pre>	<pre> 2 </pre>

input.txt	output.txt
<pre> 9 13 3 3 6 5 2 5 9 2 7 1 6 3 9 3 1 8 2 6 6 5 7 9 2 8 4 7 4 5 8 9 </pre>	<pre> 4 </pre>

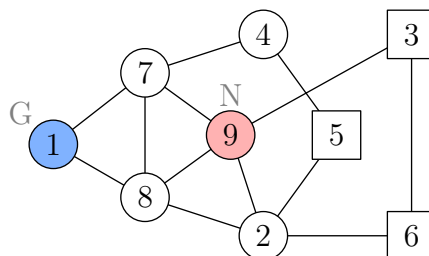
Spiegazione

Il primo caso di esempio corrisponde alla mappa



dove i nodi quadrati rappresentano i supermercati. A Gabriele conviene andare dal punto 1 al punto 3, comprare il bicarbonato, e poi ripartire, raggiungendo la casa della nonna al punto 7.

Il secondo caso di esempio corrisponde alla mappa



A Gabriele conviene andare dal punto 1 al punto 3 (in 3 minuti), e poi dal 3 tornare a casa della nonna, al nodo 9.

Pranzo dalla nonna (nonna)

Gabriele, stanco del duro lavoro di consulenza a cui è stato sottoposto recentemente, può finalmente rilassarsi a pranzo dalla nonna, che è molto premurosa e gli prepara sempre un succulento pranzetto composto di N portate. L'unica pecca è che la nonna si lascia sovente andare un po' la mano, e le ultime volte Gabriele è poi tornato a casa tutto dolorante dai postumi di una brutta indigestione. Questa volta ha quindi deciso di prendere precauzioni, e pianificare con cura cosa mangiare e cosa no.

Dalle sue precedenti esperienze, è riuscito a stimare quanti grammi di cibo K deve mangiare al minimo affinché la nonna si senta soddisfatta e non si offenda dell'inappetenza del nipotino. Inoltre, appena arrivato in casa, è riuscito a sbirciare il menù scoprendo quale peso P_i ha ciascuna portata. Aiuta Gabriele a trovare l'insieme di portate con peso totale minimo possibile ma almeno K !

Implementazione

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

📎 Tra gli allegati a questo task troverai un template (`nonna.c`, `nonna.cpp`, `nonna.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 K, int P[]);</code>
Pascal	<code>function mangia(N, K: longint; var P: array of longint): longint;</code>

In cui:

- L'intero N rappresenta il numero di portate.
- L'intero K rappresenta il peso minimo da mangiare per non offendere la nonna.
- L'array P , indicizzato da 0 a $N - 1$, contiene i pesi P_i delle portate preparate oggi dalla nonna.
- La funzione dovrà restituire il peso minimo per un insieme di portate di peso almeno K , 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 gli N interi P_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

- $1 \leq N \leq 5000$.
- $1 \leq K \leq 5000$.
- $1 \leq P_i \leq 1\,000\,000$ per ogni $i = 0 \dots N - 1$.
- È sempre possibile mangiare K grammi di cibo (il peso totale di tutte le portate è almeno K).

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]:** $N, K \leq 100$.
- **Subtask 4 [20 punti]:** $K, P_i \leq 1000$ per ogni i .
- **Subtask 5 [20 punti]:** Nessuna limitazione specifica.

Esempi di input/output

input.txt	output.txt
3 500 230 260 230	720
input.txt	output.txt
8 600 140 160 180 220 20 100 40 80	600

Spiegazione

Nel **primo caso di esempio**, Gabriele è costretto a mangiare tutte e tre le portate (due non bastano ad accontentare la nonna).

Nel **secondo caso di esempio**, Gabriele può mangiare le portate di pesi 140, 160, 220 e 80 totalizzando proprio i 600g necessari ad accontentare la nonna.