



## Quadrati gradevoli (quadrati)

Il professore di disegno tecnico di Gabriele è un uomo stravagante. I ragazzi hanno scoperto che il professore ama moltissimo i quadrati, e che spesso per prendere un buon voto in un compito è sufficiente che la tavola ne contenga uno. Tuttavia non tutti i quadrati sono uguali agli occhi del professore: sono per lui gradevoli alla vista solo quelli con area compresa tra i valori  $A$  e  $B$  (inclusi).

Quanti sono i quadrati diversi che il professore trova gradevoli?

### Dati di input

Il file `input.txt` ha questo formato:

- Riga 1: contiene gli interi  $A$  e  $B$ .


### Dati di output

Il file `output.txt` ha questo formato:

- Riga 1: contiene la risposta al problema.

### Implementazione

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

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

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

C/C++	<code>int conta(int A, int B);</code>
Pascal	<code>function conta(A, B: longint): longint;</code>

La funzione riceverà come parametri i valori  $A$  e  $B$  e dovrà ritornare la risposta al problema, che verrà stampata sul file di output.

### Assunzioni

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



## 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]:**  $A \leq B \leq 100$ .
- **Subtask 3 [40 punti]:**  $A \leq B \leq 1\,000$ .
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

## Esempi di input/output

input.txt	output.txt
6 16	2

input.txt	output.txt
4 5	1



## La congettura di Collatz (collatz)

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 appena descritto termina per qualsiasi valore  $N$ ; in altri termini, preso un qualsiasi numero intero maggiore di 1 applicare la regola numero 2 conduce sempre al numero 1.

Riferendosi a questa celebre congettura il famoso matematico Erdős ha commentato che questioni semplici ma elusive come questa mettono in evidenza quanto poco sia facile accedere ai misteri del “grande Libro”.

Giorgio sta cercando di dimostrare la congettura, ed è interessato alla lunghezza della sequenza. Il vostro compito è quello di aiutare Giorgio scrivendo una funzione che, ricevuto in ingresso il numero  $N$ , calcoli la lunghezza della sequenza che si ottiene a partire da  $N$ , seguendo le regole spiegate.

### Dati di input


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

### Dati di output

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

### Implementazione

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

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

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

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

La funzione riceverà come parametro il valori  $N$  e dovrà ritornare la risposta al problema, che verrà stampata sul file di output.



## Assunzioni

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

## 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]:**  $N \leq 500$ .
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

## Esempi di input/output

input.txt	output.txt
6	9

input.txt	output.txt
24	11



## Cubetti colorati (cubetti)

Grazie alle sue grandi abilità di cecchinaggio<sup>1</sup>, Giorgio si è aggiudicato una fantastica collezione di preziosissimi cubetti colorati da collezione presso un'asta online. Purtroppo l'entusiasmo per il ricco bottino è scemato di colpo quando aprendo il cofanetto in pelle di porcospino ha scoperto che i cubetti non sono tutti di colori diversi come si aspettava. Come tutti i collezionisti di cubetti colorati sanno, una collezione di cubetti è di valore solo se i cubetti sono tutti di colori diversi. Per fortuna non è impreparato e per emergenze di questo tipo può fare affidamento alla sua fedele Vernici-o-matic<sup>TM</sup>, una verniciatrice ad alta precisione per cubetti colorati. La macchina funziona in modo molto semplice: si inserisce un cubetto colorato, si imposta il nuovo colore e il cubetto viene verniciato di quel colore. Dal momento che il processo di verniciatura è molto lento e che Giorgio non vede l'ora di mostrare ai suoi amici la nuova collezione, scrivi un programma che determini il numero minimo di cubetti da verniciare affinché alla fine tutti i cubetti abbiano colori diversi.

### Dati di input

La prima riga del file di input contiene l'intero  $N$ , il numero di cubetti colorati all'interno del cofanetto. La seconda riga contiene  $N$  interi  $a_1, \dots, a_N$  che rappresentano i colori dei diversi cubetti. In particolare vale sempre  $1 \leq a_i \leq N$  per ogni  $i = 1, \dots, N$  e numeri uguali corrispondono a colori uguali e viceversa.

### Dati di output

In output, stampare il numero minimo di cubetti da verniciare per ottenere cubetti di colori diversi.

### Implementazione

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

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

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

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

La funzione riceverà come parametro il numero di cubetti  $N$  e l'array dei colori dei cubetti, e dovrà ritornare la risposta al problema, che verrà stampata sul file di output.

### Assunzioni

- $2 \leq N \leq 100\,000$ .
- La verniciatrice può applicare una vernice di colore arbitrario (da 1 a  $N$ ) ad ogni cubetto.
- L'input contiene almeno due cubetti di colore uguale.

### 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.

<sup>1</sup>Nel gergo delle aste online, con *cecchinaggio* ci si riferisce all'atto di fare offerte all'ultimo secondo.



- **Subtask 1** [10 punti]: Caso d'esempio.
- **Subtask 2** [20 punti]:  $N \leq 100$ .
- **Subtask 3** [40 punti]:  $N \leq 10\,000$ .
- **Subtask 4** [30 punti]: Nessuna limitazione specifica.

## Esempi di input/output

input.txt	output.txt
5 5 1 2 2 2	2

## Spiegazione

Nel caso di esempio è sufficiente verniciare il terzo elemento del colore 3 e il quinto elemento del colore 4.

## Insieme k-free (kfree)

Il professore di informatica di Gabriele, noto scienziato delle merendine, è convinto che tutti i problemi NP-completi possano essere ridotti al problema del massimo sottoinsieme  $K$ -free. Dato un numero intero positivo  $K$ , un insieme  $A$  di numeri interi positivi è detto  $K$ -free se rispetta la seguente condizione:

$$K \cdot a \notin A \quad \text{per ogni } a \in A.$$

In altre parole, se un insieme  $K$ -free contiene  $a$ , non può contenere anche  $K \cdot a$ .

Diventa a questo punto cruciale determinare la dimensione del più grande sottoinsieme di un insieme dato, che ha la proprietà di essere  $K$ -free<sup>2</sup>.

## Dati di input


La prima riga del file di input contiene gli interi  $N$  e  $K$ , il numero di elementi nell'insieme iniziale e il valore di  $K$ . La seconda riga contiene gli  $N$  interi distinti  $a_1, \dots, a_N$  dell'insieme.

## Dati di output

In output, stampare la dimensione del più grande sottoinsieme  $K$ -free dell'insieme dato in input.

## Implementazione

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

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

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

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

La funzione riceverà come parametri i valori  $N$  e  $K$  e un array di interi che rappresenta l'insieme iniziale e dovrà ritornare la risposta al problema, che verrà stampata sul file di output.

## Assunzioni

- $1 \leq N \leq 100\,000$ .
- $1 \leq K \leq 1000$ .
- $1 \leq a_i \leq 100\,000$ , per ogni  $i = 1, \dots, N$ .
- Gli interi  $a_1, \dots, a_N$  sono distinti.

<sup>2</sup>Notare che la proprietà di un sottoinsieme di essere  $K$ -free è una proprietà solo del sottoinsieme, che non dipende in alcun modo dall'insieme di partenza.



## 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]:** Caso d'esempio.
- **Subtask 2 [20 punti]:**  $N \leq 100, K = 1$ .
- **Subtask 3 [40 punti]:**  $N \leq 500, K \leq 100$ .
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

## Esempi di input/output

input.txt	output.txt
6 2 2 3 6 5 4 10	3

## Spiegazione

Nel caso di esempio un sottoinsieme 2-free di dimensione massima è  $\{4, 5, 6\}$ , infatti non contiene né 8, né 10 né 12. Esistono altri sottoinsiemi 2-free della stessa dimensione, come ad esempio  $\{2, 3, 5\}$  o  $\{2, 3, 10\}$ , mentre altri sottoinsiemi della stessa dimensione non sono 2-free (come ad esempio  $\{2, 3, 4\}$ ).



## Giochiamo con Mojito (mojito)



Mojito, il jackrussell di Monica, è ormai diventato la mascotte dei Probabili Olimpici, i ragazzi che sono candidati a rappresentare l'Italia alle Olimpiadi Internazionali di Informatica 2015 a Astana, Kazakistan. Negli allenamenti a Volterra, Mojito gioca a palla con i ragazzi nel prato: lui porta la pallina al ragazzo più vicino che la calcia via; a quel punto Mojito rincorre la palla, l'acchiappa e la porta di nuovo al ragazzo che ha più vicino... e così via! Possiamo rappresentare questo gioco con una griglia: supponendo di avere tre ragazzi che giocano con Mojito, rappresentiamo la loro posizione nella griglia, rispettivamente, con R1, R2 e R3. Tutti i ragazzi sono piuttosto metodici, e ogni volta che tirano la palla questa finisce sempre nella stessa posizione (a seconda di chi tira!): sulla griglia indichiamo con P1 il punto in cui finisce la palla tirata da R1, P2 il punto in cui finisce la palla tirata da R2, ecc... La posizione iniziale di Mojito, con la palla, è rappresentata nella griglia da una M. Mojito misura la distanza come il minimo numero di spostamenti orizzontali e/o verticali per andare da una casella a un'altra.

3	R1		P2			P1		
2					M			
1				R3		P3	R2	
	1	2	3	4	5	6	7	8

Per esempio, consideriamo la griglia qui sopra, di dimensione  $8 \times 3$ . All'inizio Mojito si trova, insieme con la palla, nella casella (5,2); il ragazzo più vicino è R3, nella posizione (4,1), che dista due caselle da lui; il gioco inizia:

- Mojito porta la palla a R3, che la tira nella casella (6,1);
- a questo punto Mojito, presa la palla, la porta a R2, nella casella (7,1), che è il più vicino a lui; da qui la palla viene tirata nella casella (3,3);
- Mojito recupera la palla e la porta a R1, nella casella (1,3); R1 tira la palla nella casella (6,3);
- da qui in poi saranno solo R1 e R2 a giocare, visto che quando tira R1 poi Mojito porta la palla a R2 e viceversa.

Notiamo che, nel caso appena descritto, tutti e tre i ragazzi hanno giocato (anche se R3 ha toccato palla solo una volta). Se Mojito ha due o più ragazzi alla stessa distanza, sceglie quello che ha la coordinata  $X$  (orizzontale) minore e, se ve ne sono due o più con lo stesso valore, tra questi sceglie quello che ha la coordinata  $Y$  (verticale) minore. Mojito è molto concentrato sulla palla, e non riesce a ricordarsi se tutti i ragazzi l'hanno tirata. Il vostro compito è quello di scrivere un programma che calcoli il numero di ragazzi che lanciano la palla almeno una volta!



## Dati di input

Il file `input.txt` è composto da  $N+3$  righe. La prima riga contiene due interi positivi  $X$  e  $Y$ : le dimensioni della griglia. La seconda riga contiene una coppia di interi positivi: le coordinate della posizione iniziale di Mojito con la palla. La terza riga contiene  $N$ , il numero di ragazzi che giocano con Mojito. Ognuna delle successive  $N$  righe contiene due coppie di interi: le coordinate dell' $i$ -esimo ragazzo (prima coppia di interi) e le coordinate di dove l' $i$ -esimo ragazzo tirerà la palla.

## Dati di output

Il file `output.txt` è composto da una sola riga contenente un solo intero non negativo: il numero di ragazzi che giocano con Mojito, ovvero il numero di ragazzi che tirano la palla almeno una volta, a partire dalla posizione iniziale di Mojito.

## Implementazione

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

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

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

C/C++	<code>int gioca(int X, int Y, int MX, int MY, int N, int *RX, int *RY, int *PX, int *PY);</code>
Pascal	<code>function gioca(X, Y, MX, MY, N: longint; var RX, RY, PX, PY: array of longint): longint;</code>

In cui:

- Gli interi  $X$ ,  $Y$  rappresentano le dimensioni della griglia.
- Gli interi  $MX$ ,  $MY$  rappresentano le coordinate iniziali di Mojito.
- L'intero  $N$  rappresenta il numero di ragazzi.
- Gli array  $RX$ ,  $RY$ ,  $PX$ ,  $PY$ , indicizzati da 0 a  $N-1$ , contengono rispettivamente le posizioni dei ragazzi e dove lanciano la palla.
- La funzione dovrà restituire la risposta al problema, che verrà stampata sul file di output.

## Assunzioni

- $1 \leq N \leq 10\,000$
- $1 \leq X, Y \leq 1\,000\,000$
- Le coordinate della griglia vanno da 1 a  $X$  e da 1 a  $Y$  (inclusi).
- Tutte le posizioni nel file di input sono distinte: non ci possono essere due ragazzi nella stessa casella, non ci sono due ragazzi che tirano nella stessa casella, nessun ragazzo tira nella casella dove c'è un altro ragazzo.
- Mojito, inizialmente, è in una casella non occupata da nessun ragazzo e dove nessun ragazzo tira la palla.
- Mojito, piccolo com'è, riesce agevolmente a passare tra le gambe dei ragazzi; non viene quindi ostacolato nel suo movimento da ragazzi presenti in una cella tra lui e la palla.



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

## Esempi di input/output

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

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