ABC 2019

Testi dei problemi

Innumerevoli progetti (`progetti`)

Leonardo è universalmente riconosciuto come uno degli inventori più prolifici di tutti i tempi. Com'era possibile che non si dimenticasse le numerose idee che gli balenavano continuamente in testa? Probabilmente Leonardo aveva cura per ciascuna idea di appuntarsi qualche bozza su un foglietto, in modo che non appena fosse stato possibile sarebbe stato in grado di recuperare le idee e portare a termine il progetto.

Figura 1: Si crede che Leonardo sia stato tra i primi a costruirsi e usare una penna stilografica.

Luca è strabiliato dall'impressionante numero di progetti e fatica a capacitarsi di come una singola persona abbia potuto lavorare a tutto ciò. È ragionevole supporre infatti che il massimo numero di progetti che si possono realizzare in un giorno sia un numero K piccolo. Inoltre, non tutti i giorni sono uguali: a volte si riesce a completarne solo uno... a volte molti!

Luca sta provando a stimare il numero di giorni necessari per completare N progetti ma è in confusione perché si è accorto che il risultato varia estremamente a seconda della quantità di progetti completati in ciascun giorno. Facendo un passo indietro, vuole allora capire un'altra cosa: quante sono le possibili combinazioni diverse con cui può completare tutti i progetti, scegliendo arbitrariamente quanti progetti completare giorno per giorno e senza avere alcun limite massimo di giorni?

Implementazione

Dovrai sottoporre un unico file con estensione `.cpp` o `.c`.

> ☞ Tra gli allegati a questo task troverai un template (`progetti.cpp` e `progetti.c`) con un esempio di implementazione.

Dovrai implementare la seguente funzione:

■ Funzione `progetti`

C/C++	`int progetti(int N, int K);`

- L'intero N rappresenta il numero dei progetti.
- L'intero K rappresenta il massimo numero di progetti completabili in un giorno.
- La funzione dovrà restituire il numero di modi diversi con cui è possibile completare i progetti. Poiché tale numero può essere grande, è necessario restituire soltanto il resto della divisione (operatore di *modulo*[1]) per $1\,000\,000\,007$.

Il grader chiamerà la funzione `progetti` e ne stamperà il valore restituito sul file di output.

Allegata a questo problema è presente una versione semplificata del grader usato durante la correzione, che potete usare per testare le vostre soluzioni in locale. Il grader di esempio legge i dati da `stdin`, chiama la funzione che dovete implementare e scrive su `stdout`, secondo il seguente formato.

Il file di input è composto da un'unica riga, contentente:

- Riga 1: gli interi N e K, separati da uno spazio.

Il file di output è composto da un'unica riga, contenente:

- Riga 1: il valore restituito dalla funzione `progetti`.

Assunzioni

- $2 \leq N \leq 10^9$.
- $2 \leq K \leq 5$.
- In ogni giorno Leonardo può completare da 1 a K progetti.
- Rigorosmente, due modi di completamento si considerano *diversi* se differiscono per il numero di giorni necessari al completamento di tutti i progetti oppure se, a parità di giorni, esiste almeno un giorno in cui nei due modi sono stati completati un numero diverso di progetti.

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 [0 punti]**: Casi d'esempio.
- **Subtask 2 [15 punti]**: $K = 2$, $N \leq 15$.
- **Subtask 3 [20 punti]**: $K = 2$, $N \leq 10^6$.
- **Subtask 4 [15 punti]**: $K = 3$, $N \leq 10^6$.
- **Subtask 5 [20 punti]**: $K = 2$, $N \leq 10^9$.
- **Subtask 6 [30 punti]**: Nessuna limitazione specifica.

[1] In C/C++ l'operatore modulo ha simbolo '`%`'.

Esempi di input/output

input	output
2 2	2
4 2	5

Spiegazione

Nel **primo caso di esempio** ci sono due modi diversi per completare i due progetti rispettando il vincolo dei due progetti al giorno: è possibile completarne uno nel primo giorno e uno nel secondo oppure si può finire in un solo giorno completandoli subito entrambi.

Nel **secondo caso di esempio** ci sono cinque modi diversi per completare i quattro progetti:

- uno al giorno per quattro giorni;
- due al giorno per due giorni;
- uno il primo giorno, due il secondo giorno, uno il terzo giorno;
- uno il primo giorno, uno il secondo giorno, due il terzo giorno;
- due il primo giorno, uno il secondo giorno, uno il terzo giorno.

Lotteria di quadri (`quadri`)

Facendo pulizia in soffitta Fabio ha ritrovato diversi quadri, alcuni del tutto privi di valori e altri invece potenzialmente interessanti (tra cui diverse copie di alcuni dipinti di Leonardo da Vinci). Per racimolare qualche soldo, ha intenzione di venderli: ha pertanto portato a valutare gli N quadri da un esperto che ha attribuito a ciascuno un valore V_i approssimativo.

Per far sì che tutti i quadri di minor valore non restino invenduti, continunando a ingombrare la soffitta, Fabio è intenzionato a organizzare una lotteria con una regola particolare: le opere saranno disposte in fila e l'acquirente, dopo aver pagato un "biglietto di ingresso", potrà scegliere a propria discrezione un "blocco" consecutivo di B opere (non di più né di meno).

Figura 1: Alcuni dei quadri esposti (foto di Muhammad Raufan Yusup).

Fabio non vuole che la somma dei valori delle opere che un acquirente può portarsi a casa sia maggiore di un certo massimale M, altrimenti ci starebbe perdendo troppo. D'altro canto, pur rispettando questo principio, vorrebbe rendere B (il numero di quadri che ci si porta a casa comprando il biglietto d'ingresso) più alto possibile. Aiutalo a capire qual è il valore massimo di B per rendere il più appetibile possibile la lotteria!

Implementazione

Dovrai sottoporre un unico file con estensione `.cpp` o `.c`.

> ☞ Tra gli allegati a questo task troverai un template (`quadri.cpp` e `quadri.c`) con un esempio di implementazione.

Dovrai implementare la seguente funzione:

■ Funzione `quadri`

C/C++	`int quadri(int N, long long M, int V[]);`

- L'intero N rappresenta il numero dei quadri.
- L'intero M rappresenta il massimale da non superare.
- L'array `V`, indicizzato da 0 a $N-1$, contiene alla posizione i il valore dell'i-esimo quadro nella fila.
- La funzione dovrà restituire il valore massimo per B come descritto nel testo.

Il grader chiamerà la funzione `quadri` e ne stamperà il valore restituito sul file di output.

Allegata a questo problema è presente una versione semplificata del grader usato durante la correzione, che potete usare per testare le vostre soluzioni in locale. Il grader di esempio legge i dati da `stdin`, chiama la funzione che dovete implementare e scrive su `stdout`, secondo il seguente formato.

Il file di input è composto da due righe, contententi:

- Riga 1: gli interi N e M, separati da uno spazio.
- Riga 2: N interi `V[`i`]` per $i = 0, \ldots, N-1$.

Il file di output è composto da un'unica riga, contenente:

- Riga 1: il valore restituito dalla funzione `quadri`.

Assunzioni

- $1 \leq N \leq 200\,000$.
- $1 \leq M \leq 10^{12}$.
- $1 \leq V_i \leq 10^6$ per ogni $i = 0 \ldots N-1$.
- Nella scelta del blocco di B quadri, l'acquirente **non** può scegliere un blocco agli estremi della fila in modo da prenderne meno di B.

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** [**0 punti**]: Casi d'esempio.
- **Subtask 2** [**15 punti**]: $M < V_i$ per ogni $i = 0, \ldots, N-1$ oppure $M > V_0 + V_1 + \ldots + V_{N-1}$.
- **Subtask 3** [**20 punti**]: $N \leq 500$.
- **Subtask 4** [**25 punti**]: $N \leq 5\,000$.
- **Subtask 5** [**40 punti**]: Nessuna limitazione specifica.

Esempi di input/output

input	output
4 8 1 2 3 4	2
5 1 3 1 3 2 10	0

Spiegazione

Nel **primo caso di esempio** è possibile impostare $B = 2$: l'acquirente potrebbe scegliere i quadri di valore $1 + 2 = 3$, $2 + 3 = 5$ oppure $3 + 4 = 7$ senza superare mai il massimale $M = 8$. Non sarebbe stato possibile scegliere $B = 3$ perché in quel caso l'acquirente avrebbe potuto il scegliere il blocco di valore $2 + 3 + 4 = 9$, superando il massimale.

Nel **secondo caso di esempio** anche scegliere $B = 1$ consentirebbe all'acquirente di superare il massimale qualunque scelta egli faccia (ad eccezione del secondo quadro). La risposta corretta è quindi $B = 0$.

Cenacolo (`ultimacena`)

Tra le varie opere di Leonardo, l'affresco raffigurante l'Ultima Cena è considerato uno dei maggiori capolavori. Un amico di Luca sta svolgendo una ricerca approfondita con il compito di fornire una descrizione esaustiva e particolareggiata di queste meraviglie.

Analizzando l'affresco, ha già individuato N "zone di interesse" (come quelle racchiuse tra ellissi nella figura sotto), descritte con interi da 1 a N, e M relazioni tra di esse (indicate in figura con delle linee). I motivi per definire una relazione tra due zone sono molti: potrebbero contenere due personaggi che si stanno guardando, oppure due aree dipinte con tecniche similiari oppure completamente opposte, e così via.

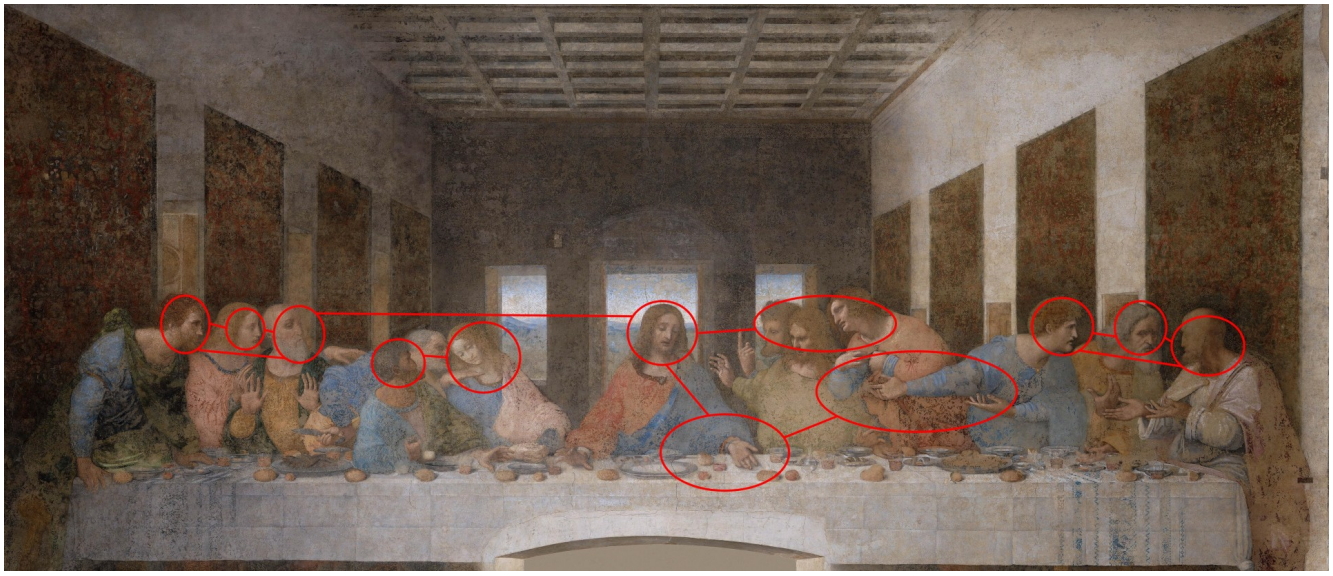

Figura 1: Cenacolo vinciano (annotato per evidenziare alcune delle zone di interesse e le loro relazioni).

Il report finale del progetto di ricerca sarà costituito da un elenco delle zone di interesse. Per ognuna verranno descritte, oltre alla zona in sè, anche tutte le relazioni in cui è coinvolta.

Come suo solito, l'amico di Luca è stato particolarmente prolisso e ora il professore a capo del progetto di ricerca non vuole accettare il suo lavoro finché non farà una drastica selezione, includendo al più **dieci** zone di interesse.

Dopo tutto questo lavoro di accurata descrizione delle relazioni, l'amico è costretto a obbedire al professore ma non vuole sacrificare nemmeno una delle descrizioni già prodotte: aiutalo a selezionare un sottoinsieme S delle N zone in modo che il report finale contenga comunque la descrizione di *tutte* le relazioni!

Implementazione

Dovrai sottoporre un unico file con estensione `.cpp` o `.c`.

> ☞ Tra gli allegati a questo task troverai un template (`ultimacena.cpp` e `ultimacena.c`) con un esempio di implementazione.

Dovrai implementare la seguente funzione:

■ Funzione `riassumi`

C/C++	`int riassumi(int M, int M, int A[], int B[], int S[]);`

- L'intero N rappresenta il numero delle zone di interesse.
- L'intero M rappresenta il numero delle relazioni tra le zone.
- Gli array `A` e `B`, indicizzati da 0 a $M-1$, contengono alla posizione i la seguente informazione: esiste una relazione tra le zone `A[`i`]` e `B[`i`]`.
- La funzione dovrà restituire il numero S di zone da includere nel report finale e riempire l'array `S` (nelle posizioni da 0 a $S-1$) con gli identificativi di tali zone.

Il grader chiamerà la funzione `riassumi` in accordo a quanto specificato di seguito.

Allegata a questo problema è presente una versione semplificata del grader usato durante la correzione, che potete usare per testare le vostre soluzioni in locale. Il grader di esempio legge i dati da `stdin`, chiama la funzione che dovete implementare e scrive su `stdout`, secondo il seguente formato.

Il file di input è composto da $M+1$ righe, contenenti:

- Riga 1: gli interi N e M, separati da uno spazio.
- Righe $2, \ldots, M+1$: i due interi `A[`i`]` e `B[`i`]` per $i = 0, \ldots, M-1$.

Il file di output è composto da due righe, contenenti:

- Riga 1: il valore S restituito dalla funzione `riassumi`.
- Riga 2: i valori contenuti nell'array `S` (posizioni da 0 a $S-1$) così come modificato dalla funzione `riassumi`.

Assunzioni

- $2 \leq N \leq 50\,000$.
- $1 \leq M \leq 100\,000$.
- È sempre garantita l'esistenza di una soluzione, ovvero esiste sempre un sottoinsieme costituito da al più dieci zone che rispetta quanto richiesto.
- Se esistono più soluzioni è possibile stamparne una qualsiasi.
- L'ordine con cui vengono riportate le zone in una soluzione è irrilevante.
- Una relazione non viene mai ripetuta in input; inoltre $A_i \neq B_i$ per ogni $i = 0 \ldots M-1$.
- Ogni zona d'interesse è coinvolta in almeno una relazione.

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** [**0 punti**]: Casi d'esempio.
- **Subtask 2** [**40 punti**]: $N \leq 15$.
- **Subtask 3** [**40 punti**]: $N \leq 1000$.
- **Subtask 4** [**20 punti**]: Nessuna limitazione specifica.

Esempi di input/output

input	output
5 3 1 4 3 5 2 4	5 1 2 3 4 5
11 15 4 1 1 3 2 4 8 7 5 8 9 10 11 5 6 9 10 6 3 8 7 11 3 5 8 6 9 2 5 6	10 3 4 2 5 6 7 8 9 10 11

Spiegazione

Nel **primo caso di esempio** le zone di interesse sono così poche da poter essere incluse tutte nel report senza problemi. Naturalmente sarebbe stato ammissibile produrre soluzioni diverse (ad esempio includendo le sole zone 3 e 4).

Nel **secondo caso di esempio** ci sono $11 > 10$ zone di interesse, quindi dobbiamo escluderne (almeno) una. Possiamo escludere la zona 1, le cui uniche relazioni sono con le zone 3 e 4 già incluse nel report finale.