ABC 2014

Bug (bug)

Descrizione del problema

Il professor Tarboto sta per terminare di scrivere il suo nuovo libro di reti e sistemi informatici, ma si è accorto di essere un po' indietro con il lavoro: la fine dell'anno scolastico si sta avvicinando e la casa editrice vuole che il lavoro sia assolutamente pronto entro giugno.
In particolare, restano da sistemare ancora **M** bug nei codici degli algoritmi presenti nel libro.
Grazie alla sua proverbiale esperienza, il professor Tarboto decide quindi di chiedere la collaborazione dei suoi **N** studenti affinché risolvano i problemi al suo posto. Ovviamente, gli studenti **possono lavorare in parallelo**.
Gli studenti del professore però, saputo dell'urgenza con cui il loro docente deve finire il lavoro, chiedono in cambio per l'essere impiegati (**indipendentemente dal volume di lavoro**, ovvero da quanti bug devono risolvere) un certo numero di punti (P) in più nella prossima verifica di Sistemi e Reti.
Il professor Tarboto è disposto a fare questa concessione purché il **totale** dei punti aggiuntivi che dovrà regalare ai suoi studenti non ecceda la somma **S** (non vuole che tutti prendano 10 alla prossima verifica... il preside potrebbe insospettirsi!).
Il professore sa che i bug, come gli studenti, non sono tutti uguali: ogni bug è caratterizzato dall'avere una complessità (C) e può essere risolto **in un giorno** (indipendentemente dalla complessità) da uno studente con un'abilità (A) tale che $A \geq C$.
Aiuta il professor Tarboto a stabilire a quale studente assegnare ciascun bug in modo da risolverli tutti **nel minor numero di giorni** possibile.

Dati di input

L'input è composto da 4 righe.
La prima riga contiene tre interi: N, M, S; rispettivamente il numero degli studenti, il numero dei bug e la massima quantità di punti che Tarboto può distribuire alla prossima verifica.
La seconda riga contiene M interi separati da uno spazio: C_i è la complessità dell'i-esimo bug.
La terza riga contiene N interi separati da uno spazio: A_i è l'abilità dell'i-esimo studente.
La quarta riga contiene N interi separati da uno spazio: P_i è la quantità di punti che l'i-esimo studente chiede in cambio del proprio lavoro.

Dati di output

La prima riga dell'output deve contenere la stringa "SI" (senza virgolette e senza accento) oppure "NO" (senza virgolette), a seconda se gli studenti del professor Tarboto riusciranno a risolvere tutti i problemi.
Se la risposta è "SI", l'output deve contenere una seconda riga costituita da M interi separati da spazio. L'i-esimo di questi numeri deve essere il numero dello studente che corregge l'i-esimo bug nella soluzione ottimale.
Se ci sono più soluzioni ottimali, è sufficiente stamparne una qualsiasi.

Assunzioni

- $1 \leq N, M \leq 10^5$
- $0 \leq S \leq 10^9$
- $1 \leq C_i, A_i \leq 10^9$
- $0 \leq P_i \leq 10^9$

Nel 50% dei casi vale che $N, M, S, A_i, C_i, P_i \leq 100$.

Nota importante per la valutazione

Soluzioni che stampano **solo** la prima riga di output ("SI" o "NO") verranno valutate con il 30% dei punti per ogni caso di prova.

Esempi di input/output

Input `(stdin)`	**Output** `(stdout)`

`3 4 9`
`1 3 1 2`
`2 1 3`
`4 3 6`
	`SI`
	`2 3 2 3`

Chiarificazione dell'esempio
Il terzo studente (con abilità 3) risolve il secondo e il quarto bug (con complessità 3 e 2, rispettivamente) e il secondo studente (con abilità 1) risolve il primo e il terzo bug (le loro complessità sono 1).
Risolvere ciascun bug richiede un giorno per ciascuno studente, quindi i giorni totali sono 2 (gli studenti possono lavorare in parallelo).
Il secondo studente vuole 3 punti, il terzo 6 punti e ciò incontra la disponibilità del professore che è pronto a distribuire al massimo 9 punti.

ABC 2014

Messaggi (`messaggi`)

Descrizione del problema

Luca sta preparando un sistema di messaggistica semplice per consentire la comunicazione tra gli studenti (soprattutto durante le verifiche, un sistema simile torna sempre utile).
Luca ha finito di realizzare il sistema e lo sta provando: purtroppo qualcosa non funziona e bisogna cercare di capire la situazione.
Da buon sistemista, la prima cosa che fa è analizzare il file di log conservato sul server che registra ogni singolo messaggi scambiato tra gli utenti.
Un utente è identificato con nome (con lunghezza massima 10 caratteri tra lettere minuscole e numeri).
Dopo un po' di analisi, Luca ha capito che il problema non risiede nel server: **tutti i messaggi sono memorizzati correttamente**. Probabilmente quindi è l'app che ha realizzato per gli smartphone dei suoi compagni ad avere qualche problema. Per verificare ciò, aiutalo a rispondere a due tipi di domande:

- `utente INVIATI`
  che deve restituire l'elenco degli utenti a cui *utente* ha inviato un messaggio
- `utente RICEVUTI`
  che deve restituire l'elenco degli utenti da cui *utente* ha ricevuto un messaggio

Così come il file di log è ordinato cronologicamente, allo stesso modo gli elenchi dell'utente dovranno essere nello stesso ordine con cui gli utenti compaiono nel log.

Dati di input

L'input è composto da 1 + N + R righe.
La prima riga contiene due interi, N e R, rispettivamente il numero di righe del file di log (ovvero, quanti messaggi sono stati scambiati) e il numero di richieste a cui si deve rispondere.

Le successive N righe sono nella forma

`mittente destinatario`

e indicano che un messaggio è stato inviato da *mittente* a *destinatario*.

Le ultime R righe sono nella forma

`utente TIPORICHIESTA`

indicando che Luca vuole sapere l'elenco degli utenti da cui *utente* ha ricevuto un messaggio (nel caso di `RICEVUTI`) oppure l'elenco degli utenti a cui *utente* ha inviato un messaggio (nel caso di `INVIATI`)

Dati di output

L'output deve essere formato da R righe.
Ciascuna di queste righe deve contenere un intero non negativo U_i, il numero di utenti che corrispondono alla R-esima richiesta, seguito dagli i nomi degli utenti.

Assunzioni

$N, R \leq 100000$.
Nel 50% dei casi vale che $N, R \leq 1000$.

Esempi di input/output

Input (stdin)	**Output** (stdout)
3 2 luca mario luca pietro francesca pietro luca INVIATI pietro RICEVUTI	2 mario pietro 2 luca francesca

Note

- Più messaggi possono essere inviati dallo stesso mittente anche verso lo stesso destinatario. Tutti i messaggi dovranno comparire nell'output del problema.
- In ogni messaggio, il mittente **non coincide mai** con il destinatario.

ABC 2014

Previsioni meteorologiche (`previsioni`)

Descrizione del problema

Il colonnello Bernacca ha studiato a lungo il tempo meteorologico del paese di Tailia e ha scoperto le regole per prevedere se in un certo giorno piove (P) o c'è il sole (S). La previsione funziona a partire dal giorno della creazione e funziona all'infinito per un numero di giorni qualsiasi. La sequenza del tempo su Tailia è iniziata così:

`P S S P S S S P S S P S S S S P S S P S S S P S S P S S S S S`

e significa che il giorno della creazione su Tailia ha piovuto, il secondo e il terzo c'è stato il sole e così via. Per ricavare il tempo in un certo giorno si possono usare una serie di regole che permettono di costruire previsioni lunghe a piacere, secondo la seguente regola: la **K-esima** previsione, si ottiene prendendo la previsione **K-1**, seguita da un giorno di pioggia (P) e da **K+2** giorni di sole (S), seguiti di nuovo dalla previsione **K-1**.
Le prime previsioni risultano quindi:

- Previsione 0 = `P S S`
- Previsione 1 = `P S S P S S S P S S`
- Previsione 2 = `P S S P S S S P S S P S S S S P S S P S S S P S S`

e così via...
Aiutate il colonnello Bernacca, che dato un giorno a partire dal giorno della creazione di Tailia, vuole sapere se, seguendo il suo modello di previsioni, in quel giorno pioverà (P) o ci sarà il sole (S).

Dati di input

L'input è costituito da un unico numero intero positivo **N**.

Dati di output

L'output è costituito da un unico carattere, P o S: se il giorno N pioverà (P) o ci sarà il sole (S).

Assunzioni

$1 \leq N \leq 10^9$.

Esempi di input/output

Input (`stdin`)	**Output** (`stdout`)
2	S

Input (stdin)	**Output** (stdout)
11	P

ABC 2014

Semina del campo (`semina`)

Descrizione del problema

Il giardiniere Mario è nel periodo dell'anno dove lavora di più: la primavera.
Il suo compito è seminare l'erba di un nuovo campo da calcio.
Essendo molto pigro, Mario non riesce a completare la semina del campo in un'unica volta e deve quindi tornarci. Sfortunatamente per lui è anche molto distratto e non riesce a ricordarsi esattamente quali parti del campo ha già seminato e può quindi capitare che in una delle volte successive rimetta i semi in parti dove aveva già seminato.
Aiuta Mario a capire quanti semi ha messo, al termine di tutte le semine, nella zona dove ve ne sono di più.

Il campo da seminare è suddiviso in quadrati di lato **1** metro utilizzando un sistema di coordinate cartesiane con origine nel centrocampo. **Un'unità del piano cartesiano corrisponde a 1 metro di campo**.
Ogni volta che Mario semina copre una zona rettangolare mettendo esattamente **1** seme in ognuno dei quadrati che compongono la zona.

Dati di input

L'input è costituito da 1 + N linee.
La prima linea è costituita da un unico numero intero positivo **N**, il numero di semine che Mario ha effettuato sul campo.
Le successive N linee sono costituite da 4 interi (Xi, Yi, Xf, Yf): le coordinate del vertice in alto a sinistra e del vertice in basso a destra della i-esima zona rettangolare di campo seminata.

Dati di output

L'output è costituito da un unico numero intero non negativo: quanti semi Mario ha messo, al termine di tutte le semine, nella zona dove ve ne sono di più.

Assunzioni

- $1 \leq N \leq 100$.
- $-100 < Xi, Yi, Xf, Yf < 100$.
- Per ogni semina, vale che $Xi < Xf$ e $Yf < Yi$.

Esempi di input/output

Input `(stdin)`	**Output** `(stdout)`
2 2 5 6 2 0 5 3 1	2

Nel caso di esempio mostrato, l'area tra (2, 5) e (3, 2) è stata seminata due volte.
L'immagine sotto riportata esemplifica il caso di input.