Montagne russe

Anna lavora in un parco di divertimenti ed ha il compito di costruire una nuova montagna russa. Ha già progettato n sezioni speciali (numerate da 0 a $n-1$) che influenzano la velocità dei treni che le percorrono. Ora deve solo metterle insieme per finire il progetto della montagna russa! Per i propositi di questo problema, puoi assumere che la lunghezza del treno sia zero.

Per ogni i tra 0 e $n-1$ inclusi, la sezione speciale i ha due proprietà:

- un limite di velocità in ingresso: la velocità di un treno in ingresso deve essere **minore o uguale a** s_i km/h (chilometeri all'ora),
- una velocità in uscita: la velocità t_i km/h che un treno possiede all'uscita della sezione speciale, **indipendentemente** dalla velocità con cui è entrato.

Il progetto finito deve essere una singola linea contenente ciascuna delle n sezioni in qualche ordine. Ognuna delle n sezioni deve essere presente esattamente una volta. Inoltre, tra ogni due sezioni consecutive deve esserci un *collegamento*. Anna deve quindi scegliere l'ordine delle n sezioni e la lunghezza di ogni collegamento. La lunghezza di un collegamento è misurata in metri e può essere uguale a un qualunque intero non-negativo (possibilmente zero).

Ogni metro del collegamento tra due sezioni speciali rallenta il treno di 1 km/h. All'inizio della corsa, il treno entra nella prima sezione speciale scelta da Anna con la velocità di 1 km/h.

Il progetto finale deve soddisfare i requisiti seguenti:

- il treno non deve mai violare il limite di velocità all'ingresso di una sezione speciale;
- la velocità del treno deve rimanere positiva in ogni momento.

In tutti i subtask tranne il terzo, il tuo compito è di trovare l'ordine delle n sezioni speciali e le lunghezze dei corrispettivi collegamenti, di modo che la lunghezza totale dei collegamenti sia minima possibile. Nel subtask 3 devi soltanto controllare se esiste una montagna russa valida in cui ogni collegamento ha lunghezza zero.

Dettagli di implementazione

Devi implementare la seguente funzione (metodo):

- `int64 plan_roller_coaster(int[] s, int[] t)`
  - `s`: array di lunghezza n, massima velocità in entrata consentita.
  - `t`: array di lunghezza n, velocità in uscita.
  - La funzione deve restituire la minima lunghezza totale possibile per i collegamenti tra sezioni speciali. Nel subtask 3, deve restituire 0 se esiste un

progetto valido in cui tutti i collegamenti hanno lunghezza zero, e un qualsiasi intero positivo altrimenti (vedi dettagli nella sezione Subtask).

Per il linguaggio C la signature della funzione è leggermente diversa:

- `int64 plan_roller_coaster(int n, int[] s, int[] t)`
  - `n`: il numero di elementi in `s` e `t` (cioè il numero di sezioni speciali),
  - gli altri parametri sono come sopra.

Esempi

`plan_roller_coaster([1, 4, 5, 6], [7, 3, 8, 6])`
In questo esempio ci sono 4 sezioni speciali. La soluzione migliore è di metterle nell'ordine 0, 3, 1, 2, e poi connetterle con collegamenti di lunghezza 1, 2, 0 rispettivamente. Con questo progetto, un treno che percorre la montagna russa procede in questo modo:

- Inizialmente la velocità del treno è di 1 km/h.
- Il treno inizia la corsa entrando nella sezione speciale 0.
- Il treno lascia la sezione 0 viaggiando a 7 km/h.
- A questo punto c'è un collegamento di 1 m di lunghezza. Dopo che il treno lo ha percorso, la sua velocità scende a 6 km/h.
- Il treno entra nella sezione speciale 3 viaggiando a 6 km/h e la lascia alla medesima velocità.
- Dopo aver lasciato la sezione 3, il treno percorre un collegamento di 2m per cui la sua velocità scende a 4 km/h.
- Il treno entra nella sezione speciale 1 viaggiando a 4 km/h e la lascia a 3 km/h.
- Subito dopo la sezione 1 il treno entra nella sezione speciale 2.
- Il treno lascia la sezione speciale 2 a una velocità finale di 8 km/h.

La funzione deve quindi restituire la somma delle lunghezze dei collegamenti tra sezioni speciali: $1 + 2 + 0 = 3$.

Subtask

In tutti i subtask $1 \le s_i \le 10^9$ e $1 \le t_i \le 10^9$.

1. (11 punti): $2 \le n \le 8$,
2. (23 punti): $2 \le n \le 16$,
3. (30 punti): $2 \le n \le 200\,000$.
   In questo subtask il tuo programma deve solo controllare se la risposta è zero oppure no. Se la risposta non è zero, ogni intero positivo è considerato una risposta corretta.
4. (36 punti): $2 \le n \le 200\,000$.

Grader di esempio

Il grader di esempio legge l'input nel formato seguente:

- riga 1: l'intero n.
- righe 2 + i, per i tra 0 e $n - 1$: gli interi s_i e t_i.