Aqueduct Materials (aqueducts)

I romani costruiscono gli acquedotti utilizzando uno di N materiali disponibili nell'Impero. L'i-esimo materiale ha una resistenza S_i e un prezzo P_i.

Diciamo che l'i-esimo materiale *domina* il j-esimo materiale se e solo se:

- è (strettamente) più resistente ($S_i > S_j$) e
- è (strettamente) più economico ($P_i < P_j$).

Il Senato vuole riservare l'uso di **un** materiale esclusivamente per Roma per assicurarsi che la Caput Mundi mantenga il suo aspetto unico. Gli altri $N-1$ materiali saranno *disponibili* per il resto dell'impero.

Un materiale si dice *ottimale* per il resto dell'impero se:

- è disponibile e
- non è dominato da nessun materiale disponibile.

Per ogni k da 0 a $N-1$, supponendo il materiale riservato a Roma sia k, conta il numero C_k di materiali ottimali per il resto dell'impero.

Implementazione

Dovrai inviare un singolo file sorgente `.cpp`.

> ⚠ Tra gli allegati di questo problema troverai un template `aqueducts.cpp` con un'implementazione di esempio.

Dovrai implementare la seguente funzione:

C++	`vector<int> count(int N, vector<int> S, vector<int> P)`

- L'intero N rappresenta il numero di materiali.
- L'array S, indicizzato da 0 a $N-1$, contiene i valori $S_0, S_1, ..., S_{N-1}$.
- L'array P, indicizzato da 0 a $N-1$, contiene i valori $P_0, P_1, ..., P_{N-1}$.
- La funzione deve restituire un array C contenente i valori $C_0, C_1, ..., C_{N-1}$ dove C_k è il numero di materiali ottimali rimasti per il resto dell'impero se k è riservato a Roma.

Grader di prova

Tra gli allegati di questo problema troverai una versione semplificata del grader usato durante la valutazione, che puoi usare per testare le tue soluzioni in locale. Il grader di esempio legge i dati da `stdin`, chiama la funzione `count` e scrive su `stdout` usando il seguente formato.

L'input è composto da $N+1$ righe, contenenti:

- Riga 1: l'intero N.
- Riga $2+i$ ($0 \le i < N$): gli interi S_i e P_i.

L'output è composto da una singola riga, contenente i valori restituiti dalla funzione `count`.

Assunzioni

- $2 \le N \le 300\,000$.
- $1 \le S_i, P_i \le 1\,000\,000$.

- Per tutti gli $0 \leq i < j \leq N-1$, si ha $S_i \neq S_j$ e $P_i \neq P_j$.

Assegnazione del punteggio

Il tuo programma verrà testato su diversi casi di test raggruppati in subtask. Per ottenere il punteggio di un subtask, il tuo programma deve risolvere correttamente tutti i suoi casi di test.

- **Subtask 0 [0 punti]**: Casi di esempio.
- **Subtask 1 [13 punti]**: $N \leq 100$.
- **Subtask 2 [22 punti]**: $N \leq 5000$.
- **Subtask 3 [28 punti]**: $|S_i - P_i| \leq 2$.
- **Subtask 4 [37 punti]**: Nessun vincolo aggiuntivo.

Esempi di input/output

stdin	stdout
3 10 20 30 10 20 30	1 2 1
9 10 11 4 6 11 13 14 7 1 4 6 10 9 2 12 12 3 5	2 2 2 3 2 2 4 2 2

Spiegazione

Spiegazione del primo esempio.

- Se $k = 0$, il materiale 1 è ottimale.
- Se $k = 1$, i materiali 0 e 2 sono ottimali.
- Se $k = 2$, il materiale 1 è ottimale.

Spiegazione del secondo esempio.

- Se $k = 0$, i materiali 3 e 6 sono ottimali.
- Se $k = 1$, i materiali 3 e 6 sono ottimali.
- Se $k = 2$, i materiali 3 e 6 sono ottimali.
- Se $k = 3$, i materiali 0, 6 e 7 sono ottimali.
- Se $k = 4$, i materiali 3 e 6 sono ottimali.
- Se $k = 5$, i materiali 3 e 6 sono ottimali.
- Se $k = 6$, i materiali 1, 3, 4 e 8 sono ottimali.
- Se $k = 7$, i materiali 3 e 6 sono ottimali.
- Se $k = 8$, i materiali 3 e 6 sono ottimali.