

Torre di controllo (aeroporto)

Limite di tempo: 0.2 secondi
Limite di memoria: 256 MiB

Nella torre di controllo dell'aeroporto di Bologna è scattato il panico. La finale nazionale delle Olimpiadi di Informatica, infatti, ha attirato una moltitudine di tifosi che stanno raggiungendo la città con voli di linea e charter. L'insolita mole di traffico sta quindi costringendo gli operatori della torre di controllo a regolare meticolosamente tutte le richieste di atterraggio, per evitare che la situazione porti a un rischio di incidente aereo troppo elevato.

In particolare, la torre di controllo ha ricevuto R richieste di atterraggio. L' i -esima richiesta arrivata in ordine cronologico consiste in due valori $A[i]$ e $B[i]$: il primo di questi due valori indica l'istante di tempo in cui l'aereo corrispondente può atterrare se manovra direttamente verso la pista, mentre il secondo indica l'istante di tempo massimo entro cui può atterrare se cerca di allungare il tragitto (ma senza rischiare di finire il carburante).

La torre di controllo deve scegliere per ogni aereo un istante di tempo $T[i]$ compreso tra $A[i]$ e $B[i]$, di modo che la distanza *minima* K tra due istanti scelti *sia massima possibile*. Infatti, più K si riduce e più aumenta il rischio di incidenti aerei. La torre di controllo deve anche considerare che un pilota vuole atterrare prima di tutti quelli che hanno fatto richiesta dopo di lui, perciò deve far sì che $T[i] < T[i + 1]$, cioè che gli atterraggi avvengano nello stesso ordine delle richieste. Aiuta gli operatori a pianificare gli atterraggi!

Implementazione

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

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

Dovrai implementare la seguente funzione:

■ Funzione pianifica

C/C++	<code>void pianifica(int R, int A[], int B[], int T[]);</code>
Pascal	<code>procedure pianifica(R: longint; A, B, T: array of longint);</code>

- L'intero R rappresenta il numero di richieste di atterraggio.
- Gli array A e B , indicizzati da 0 a $R - 1$, contengono alla posizione i rispettivamente il minimo e il massimo istante di tempo possibile per l'atterraggio dell'aereo i .
- L'array T , indicizzato da 0 a $R - 1$, dovrà essere riempito dal tuo programma con gli istanti di tempo $T[i]$ compresi tra $A[i]$ e $B[i]$ tali da massimizzare la minima distanza tra due istanti scelti.

Il grader chiamerà la funzione `pianifica` e stamperà il valore calcolato in T sul file di output.

Grader di prova

Nella directory relativa 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 di

input dal file `input.txt`, chiama le funzioni che dovete implementare e scrive il file `output.txt`, secondo il seguente formato.

Il file `input.txt` è composto da $R + 1$ righe, contenenti:

- Riga 1: l'unico intero R .
- Righe 2, \dots , $R + 1$: i due interi $A[i]$, $B[i]$ per $i = 0, \dots, R - 1$.

Il file `output.txt` è composto da un'unica riga, contenente:

- Riga 1: gli R interi $T[i]$ per $i = 0, \dots, R - 1$ calcolati dalla funzione `pianifica`.

Assunzioni

- $1 \leq R \leq 100\,000$.
- $0 \leq T_{\max} \leq 1\,000\,000\,000$.
- $0 \leq A[i] \leq B[i] \leq T_{\max}$ per ogni $i = 0, \dots, R - 1$.
- È assicurato che in tutti i casi di prova sia possibile far atterrare gli aerei nello stesso ordine in cui sono giunte le richieste, in istanti di tempo distinti.

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.

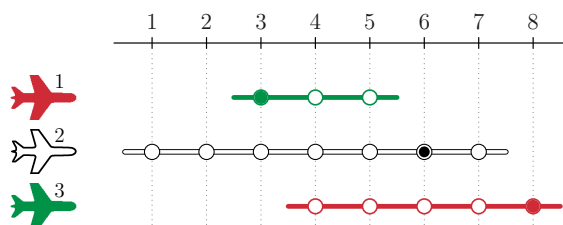
- **Subtask 1 [5 punti]**: Casi d'esempio.
- **Subtask 2 [7 punti]**: $A[i] = B[i]$ per ogni $i = 0, \dots, R - 1$.
- **Subtask 3 [14 punti]**: $T_{\max} \leq 10R$.
- **Subtask 4 [17 punti]**: $T_{\max} \leq 1000$.
- **Subtask 5 [15 punti]**: $T_{\max} \leq 1\,000\,000$.
- **Subtask 6 [20 punti]**: $B[i] - A[i] \leq 100$.
- **Subtask 7 [22 punti]**: Nessuna limitazione specifica.

Esempi di input/output

input.txt	output.txt
3 3 5 1 7 4 8	3 6 8
4 1 5 1 9 6 8 8 13	2 5 8 13

Spiegazione

Nel **primo caso di esempio**, si realizza $K = 2$ ed una possibile soluzione è:



Nel **secondo caso di esempio**, si realizza invece $K = 3$ ed una soluzione è:

