Teams

Avem o clasă cu N studenți numerotați de la 0 la $N-1$. În fiecare zi profesorul clasei are unele proiecte pentru studenți. Fiecare proiect trebuie realizat de o echipă de studenți în decursul aceleiași zi. Proiectele pot avea diverse solicitări. Pentru fiecare proiect profesorul cunoaște numărul exact de studenți care formează echipa ce va lucra la el.
Fiecare student preferă să lucreze în echipe cu număr de membri dintr-un anumit interval. Mai exact, studentul i poate fi inclus numai într-o echipă care conține între $A[i]$ și $B[i]$ membri. În fiecare zi un student poate fi inclus în cel mult o echipă. Unii studenți pot să nu fie incluși în nicio echipă. Fiecare echipă va lucra la un singur proiect.
Profesorul a ales deja proiectele pentru fiecare din următoarele Q zile. Pentru fiecare dintre aceste zile, determinați dacă este posibil să asociați studenții cu echipele astfel încât să existe câte o echipă care să lucreze la fiecare proiect.

Exemplu

Presupunem $N = 4$ studenți și $Q = 2$ zile. Constrângerile de dimensiune a echipelor pentru studenți sunt date în următorul tabel:

student	0	1	2	3
A	1	2	2	2
B	2	3	3	4

În prima zi sunt $M = 2$ proiecte. Dimensiunile cerute pentru echipe sunt
$K[0] = 1$ si $K[1] = 3$. Aceste două echipe pot fi alcătuite incluzând studentul 0 în echipa de dimensiune 1 și ceilalti 3 studenți în echipa de dimensiune 3.
În a doua zi sunt din nou $M = 2$ proiecte dar timpii ceruți pentru echipe sunt $K[0] = 1$ și $K[1] = 1$. În acest caz nu este posibil să formam echipele fiind un singur student ce poate inclus într-o echipă de dimensiune 1.

Cerință

Se dă descrierea pentru toți studenții: N, A, și B, precum și o secvență de Q întrebări — câte una pentru fiecare zi. Fiecare întrebare constă din numarul M al proiectelor din acea zi și o secvență K de lungime M conținând dimensiunile solicitate ale echipelor. Pentru fiecare întrebare programul tău trebuie să returneze dacă este posibil sa formezi toate echipele.

Trebuie să implementezi funcțiile `init` și `can`:

- `init(N, A, B)` — Grader-ul va apela această funcție la început exact o dată.
  - `N`: numărul de studenți.
  - `A`: un șir de lungime `N`: `A[i]` dimensiunea minimă a unei echipe în care poate să lucreze

studentul i.

- `B`: un șir de lungime `N`: `B[i]` dimensiunea maximă a unei echipe în care poate să lucreze studentul i.
- Aceasta funcție nu returnează nicio valoare.
- Se știe că $1 \leq A[i] \leq B[i] \leq N$ pentru fiecare $i = 0, \ldots, N-1$.

- `can(M, K)` — După apelul lui `init`, grader-ul va apela aceasta funcție de Q ori la rând, câte o dată pentru fiecare zi.
  - `M`: numărul de proiecte pentru această zi.
  - `K`: un șir de lungime `M` conținând dimensiunile cerute pentru echipă fiecarui proiect.
  - Funcția trebuie să returneze 1 dacă este posibil să se formeze toate echipele cerute și 0 în caz contrar.
  - Se știe ca $1 \leq$ `M` $\leq N$, și pentru fiecare $i = 0, \ldots, M-1$ avem $1 \leq$ `K[i]` $\leq N$. De remarcat că suma tuturor `K[i]` poate depăși N.

Subprobleme

Să presupunem că S este suma tuturor valorilor lui `M` în toate apelurile `can(M, K)`.

subproblema	puncte	N	Q	Constrangeri aditionale
1	21	$1 \leq N \leq 100$	$1 \leq Q \leq 100$	nu
2	13	$1 \leq N \leq 100,000$	$Q = 1$	nu
3	43	$1 \leq N \leq 100,000$	$1 \leq Q \leq 100,000$	$S \leq 100,000$
4	23	$1 \leq N \leq 500,000$	$1 \leq Q \leq 200,000$	$S \leq 200,000$

Grader-ul de pe calculatorul tău

Grader-ul de pe calculatorul tău citește intrarea în următorul format:

- linia 1: `N`
- liniile 2, …, `N` + 1: `A[i] B[i]`
- linia `N` + 2: `Q`
- liniile `N` + 3, …, `N` + `Q` + 2: `M K[0] K[1] … K[M - 1]`

Pentru fiecare întrebare, grader-ul de pe calculatorul afișează valoarea returnată de `can`.