Teams

In einer Klasse von N Schülern sind die Schüler von 0 bis $N-1$ nummeriert. Jeden Tag hat der Lehrer einige Projekte für die Schüler. Jedes Projekt muss von einem Schülerteam noch an diesem Tag abgeschlossen werden. Die Projekte sind möglicherweise unterschiedlich schwierig. Für jedes Projekt kennt der Lehrer die genaue Grösse des Teams, welches an diesem Projekt arbeiten soll.

Verschiedene Schüler bevorzugen unter Umständen unterschiedliche Teamgrössen. Genauer gesagt, Schüler i kann nur einem Team zugeordnet werden, dessen Grösse zwischen $A[i]$ und $B[i]$ liegt ($A[i]$ und $B[i]$ inklusive). An jedem Tag kann ein Schüler höchstens einem Team zugeordnet sein. Möglicherweise werden einige Schüler gar keinem Team zugeordnet. Jedes Team arbeitet an einem einzigen Projekt.

Der Lehrer hat die Projekte für die nächsten Q Tage bereits festgelegt. Für jeden dieser Tage sollst du bestimmen, ob es möglich ist, aus den Schülern so Teams zu bilden, dass an jedem Projekt ein Team arbeitet.

Beispiel

Angenommen es gibt $N = 4$ Schüler und $Q = 2$ Tage. Die Teamgrössenbeschränkungen der Schüler sind in der Tabelle unten aufgeführt.

Schüler	0	1	2	3
A	1	2	2	2
B	2	3	3	4

Am ersten Tag gibt es $M = 2$ Projekte. Die Teamgrössen sind $K[0] = 1$ und $K[1] = 3$. Für diese beiden Projekte können Teams passend gebildet werden, indem wir Schüler 0 einem Team der Grösse 1 zuweisen, und die übrigen drei Schüler einem Team der Grösse 3.

Auch am zweiten Tag sind es $M = 2$ Projekte, aber dieses Mal sind die Teamgrössen $K[0] = 1$ und $K[1] = 1$. In diesem Fall ist es nicht möglich die Teams zu bilden, da es nur einen Schüler gibt, der in einem Team der Grösse 1 sein möchte.

Aufgabe

Du bekommst die Beschreibung aller Schüler: N, A, und B, sowie eine Folge von Q Fragen — eine pro Tag. Jede Frage besteht aus der Anzahl M der Projekte an diesem Tag und einer Folge K der Länge M, welche die Teamgrössen beschreibt. Dein Programm muss für jede dieser Fragen ausgeben, ob es möglich ist, die Teams passend zu bilden.

Du musst die Funktionen `init` und `can` implementieren:

- `init(N, A, B)` — Der Grader wird diese Funktion genau einmal aufrufen.

- `N`: die Anzahl Schüler.
- `A`: ein Array der Länge `N`: `A[i]` ist die minimale Teamgrösse für Schüler i.
- `B`: ein Array der Länge `N`: `B[i]` ist die maximale Teamgrösse für Schüler i.
- Die Funktion hat keinen Rückgabewert.
- Du kannst annehmen, dass $1 \leq$`A[i]`$\leq$`B[i]`$\leq$`N` gilt, für jedes $i = 0, \ldots,$`N`-1.

- `can(M, K)` — Nach dem einzigen Aufruf von `init` wird der Grader diese Funktion Q Mal hintereinander aufrufen, einmal für jeden Tag.
  - `M`: die Anzahl Projekte an diesem Tag.
  - `K`: ein Array der Länge `M` mit den erforderten Teamgrössen der Projekte.
  - Die Funktion soll `1` zurückgeben, falls es möglich ist, alle erforderten Teams zu bilden und `0` andernfalls.
  - Du darfst annehmen, dass $1 \leq$`M`$\leq N$ gilt, und dass für jedes $i = 0, \ldots, M-1$ gilt, dass $1 \leq$`K[i]`$\leq N$. Beachte, dass die Summe aller `K[i]` möglicherweise N übersteigt.

Teilaufgaben

Wir definieren S als die Summe der Werte von `M` in allen Aufrufen `can(M, K)`.

Teilaufgabe	Punkte	N	Q	Weitere Einschränkungen
1	21	$1 \leq N \leq 100$	$1 \leq Q \leq 100$	none
2	13	$1 \leq N \leq 100\,000$	$Q = 1$	none
3	43	$1 \leq N \leq 100\,000$	$1 \leq Q \leq 100\,000$	$S \leq 100\,000$
4	23	$1 \leq N \leq 500\,000$	$1 \leq Q \leq 200\,000$	$S \leq 200\,000$

Beispielgrader

Der Beispielgrader liest die Eingabe im folgenden Format ein:

- Zeile 1: `N`
- Zeilen $i + 2$ (i=0, .., `N` - 1): `A[i] B[i]`
- Zeile `N` + 2: `Q`
- Zeilen `N` + 3, …, `N` + `Q` + 2: `M K[0] K[1] … K[M - 1]`

Für jede Frage gibt der Grader den Rückgabewert der Funktion `can` aus.