Equipos

Hay una clase con N estudiantes, enumerados de 0 a $N-1$. Cada día, el profesor elige algunos proyectos para los estudiantes. Cada proyecto debe ser completado el mismo día por un equipo de estudiantes. Los proyectos pueden variar en dificultad. Para cada proyecto, el profesor determina el tamaño adecuado del equipo para el mismo.

Distintos estudiantes pueden preferir distintos tamaños de equipos. De una forma más precisa, el estudiante i sólo puede ser asignado a un equipo con tamaño entre $A[i]$ y $B[i]$, inclusive. En cada día, un estudiante puede ser asignado a no más de un equipo. Algunos estudiantes pueden quedarse sin equipos. Cada equipo trabajará en un único proyecto.

El profesor ya ha elegido los proyectos para los próximos Q días. Para cada uno de estos días, determina si es posible asignar estudiantes a equipos de manera que haya un único equipo trabajando en cada proyecto.

Ejemplo

Asume que hay $N = 4$ estudiantes y $Q = 2$ días. Las restricciones de estudiantes sobre los tamaños de equipos se dan en la tabla debajo.

estudiante	0	1	2	3
A	1	2	2	2
B	2	3	3	4

En el primer día hay $M = 2$ proyectos. Los tamaños requeridos de equipos son $K[0] = 1$ y $K[1] = 3$. Estos dos equipos pueden ser formados asignando el estudiante 0 a un equipo de tamaño 1, y los otros tres estudiantes a un equipo de tamaño 3.

En el segundo día hay $M = 2$ proyectos nuevamente, pero esta vez los tamaños requeridos de equipos son $K[0] = 1$ y $K[1] = 1$. En esta ocasión es imposible formar equipos ya que hay un solo estudiante que puede estar en un equipo de tamaño 1.

Tarea

Te dan la descripción de todos los estudiantes: N, A, y B, así como una secuencia de Q consultas — una por cada día. Cada consulta está compuesta del número M de proyectos en ése día y una secuencia K de longitud M que contiene los tamaños requeridos de equipos. Para cada consulta, tu programa debe determinar si es posible formar todos los equipos necesarios.

Necesitas implementar las funciones `init` y `can`:

- `init(N, A, B)` — El grader llamará esta función al principio y exactamente una vez.

- `N`: el número de estudiantes.
- `A`: un arreglo de longitud N: `A[i]` es el mínimo tamaño de equipo por el estudiante i.
- `B`: un arreglo de longitud N: `B[i]` es el máximo tamaño de equipo requerido por el estudiante i.
- La función no tiene valor de retorno.
- Puedes asumir que $1 \leq A[i] \leq B[i] \leq N$ para cada $i = 0, \ldots, N-1$.

- `can(M, K)` — Luego de llamar a `init` una vez, el grader llamará esta función Q veces en fila, una por cada día.
  - `M`: el número de proyectos para éste día.
  - `K`: un arreglo de longitud `M` que contiene el tamaño requerido de equipo para cada uno de estos proyectos.
  - La función deberá retornar 1 si es posible formar todos los equipos requeridos, y 0 de lo contrario.
  - Puedes asumir que $1 \leq$ `M` $\leq N$, y que para cada $i = 0, \ldots, M-1$ se tiene que $1 \leq$ `K[i]` $\leq N$. Ten en cuenta que la suma de todos los `K[i]` puede exceder N.

Sub-tareas

Sea S la suma de valores de `M` en todas las llamadas a `can(M, K)`.

sub-tareas	puntos	N	Q	Restricciones adicionales
1	21	$1 \leq N \leq 100$	$1 \leq Q \leq 100$	ninguna
2	13	$1 \leq N \leq 100,000$	$Q = 1$	ninguna
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 de Ejemplo

El grader de ejemplo lee la entrada en el siguiente formato:

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

Para cada consulta, el grader de ejemplo imprimirá el valor de retorno de `can`.