



## Teams

Há uma classe de  $N$  estudantes, numerados de  $0$  a  $N - 1$ . Cada dia o professor da classe tem alguns projetos para os estudantes. Cada projeto tem que ser executado por uma equipa de estudantes nesse mesmo dia. Os projetos podem ter dificuldades variadas. Para cada projeto, o professor sabe o tamanho exato de uma equipa que deve trabalhar nesse projeto.

Estudantes diferentes podem preferir trabalhar em equipas de diferentes tamanhos. Mais precisamente, o estudante  $i$  somente pode ser alocado a uma equipa de tamanho entre  $A[i]$  e  $B[i]$  inclusive. A cada dia, um estudante pode ser alocado a no máximo uma equipa. Alguns estudantes podem não ser alocados a nenhuma equipa. Cada equipa trabalhará em um único projeto.

O professor já escolheu os projetos para cada um dos próximos  $Q$  dias. Para cada um desses  $Q$  dias, determine se é possível alocar estudantes a equipas de tal forma que exista uma equipa trabalhando em cada projeto.

## Exemplo

Suponha que existam  $N = 4$  estudantes e  $Q = 2$  dias. As restrições dos estudantes para os tamanhos das equipas são dadas na tabela abaixo.

estudante	0	1	2	3
$A$	1	2	2	2
$B$	2	3	3	4

No primeiro dia há  $M = 2$  projetos. Os tamanhos das equipas devem ser  $K[0] = 1$  e  $K[1] = 3$ . Estas duas equipas podem ser formadas alocando o estudante 0 a uma equipa de tamanho 1 e os demais estudantes a uma equipa de tamanho 3.

No segundo dia há novamente  $M = 2$  projetos, mas desta vez os tamanhos das equipas devem ser  $K[0] = 1$  e  $K[1] = 1$ . Neste caso não é possível formar as equipas, pois existe apenas um estudante que pode estar em uma equipa de tamanho 1.

## Tarefa

Você receberá a descrição de todos os estudantes:  $N$ ,  $A$  e  $B$ , assim como uma sequência de  $Q$  questões — uma sobre cada dia. Cada questão consiste de um número  $M$  de projetos naquele dia e uma sequência  $K$  de tamanho  $M$  contendo os tamanhos requeridos das equipas. Para cada questão, seu programa deve retornar se é possível formar todas as equipas.

Você deve implementar as funções `init` e `can`:

- `init(N, A, B)` — O avaliador chamará esta função no início, uma única vez.

- $N$ : o número de estudantes.
  - $A$ : um vetor de comprimento  $N$ :  $A[i]$  é o tamanho mínimo da equipa para o estudante  $i$ .
  - $B$ : um vetor de comprimento  $N$ :  $B[i]$  é o tamanho máximo da equipa para o estudante  $i$ .
  - A função não retorna qualquer valor.
  - Você pode considerar que  $1 \leq A[i] \leq B[i] \leq N$  para cada  $i = 0, \dots, N-1$ .
- $\text{can}(M, K)$  — Após chamar `init` uma vez, o avaliador chamará esta função  $Q$  vezes consecutivas, uma para cada dia.
- $M$ : o número de projetos para esse dia.
  - $K$ : um vetor de comprimento  $M$  contendo o tamanho da equipa requerido para cada um desses projetos.
  - A função deve retornar 1 se é possível formar todas as equipas, e 0 caso contrário.
  - Você pode considerar que  $1 \leq M \leq N$ , e que para cada  $i = 0, \dots, M-1$  temos  $1 \leq K[i] \leq N$ . Note que a soma de todos os  $K[i]$  pode exceder  $N$ .

## Subtarefas

Vamos denotar por  $S$  a soma dos valores de  $M$  em todas as chamadas a  $\text{can}(M, K)$ .

subtarefa	pontos	$N$	$Q$	Restrições adicionais
1	21	$1 \leq N \leq 100$	$1 \leq Q \leq 100$	nenhuma
2	13	$1 \leq N \leq 100,000$	$Q = 1$	nenhuma
3	43	$1 \leq N \leq 100,000$	$1 \leq Q \leq 100,000$	$S \leq 100,000$
4	23	$1 \leq N \leq 500,000$	$1 \leq Q \leq 200,000$	$S \leq 200,000$

## Avaliador exemplo

O avaliador exemplo lê a entrada no seguinte formato:

- linha 1:  $N$
- linhas 2, ...,  $N+1$ :  $A[i] B[i]$
- linha  $N+2$ :  $Q$
- linhas  $N+3, \dots, N+Q+2$ :  $M K[0] K[1] \dots K[M-1]$

Para cada questão, o avaliador exemplo escreve na saída o valor retornado por `can`.