제6회 천하제일 코딩대회 본선

Official Problem Set

Sponsored By:

제6회 천하제일 코딩대회 본선

Problem list

#	Problem Name	Time limit				page
		C/C++	Java	Python	node.js	
A	Gravity Hackenbush	2 seconds	5 seconds	8 seconds	8 seconds	3 - 5
B	K-균형 잡힌 수	3 seconds	7 seconds	11 seconds	11 seconds	6 - 6
C	Merge the Tree and Sequence	2 seconds	5 seconds	8 seconds	8 seconds	7 - 8
D	바지 구매	2 seconds	5 seconds	8 seconds	8 seconds	9 - 10
E	반전 수와 쿼리	2 seconds	5 seconds	8 seconds	8 seconds	11 - 11
F	시간딱딱충	3 seconds	7 seconds	11 seconds	11 seconds	12 - 13
G	인공 신경망	3 seconds	7 seconds	11 seconds	11 seconds	14 - 15
H	최대 최소공배수	2 seconds	5 seconds	8 seconds	8 seconds	16 - 16
I	최장 최장 증가 부분 수열	2 seconds	5 seconds	8 seconds	8 seconds	17 - 17
J	행성 정렬	2 seconds	5 seconds	8 seconds	8 seconds	18 - 18

문제지에 있는 문제가 총 10문제가 맞는지 확인하시기 바랍니다.

모든 문제의 메모리 제한은 1GB로 동일합니다.

A. Gravity Hackenbush

성현이는 아래와 같은 방식으로 진행되는 2인용 게임인 Gravity Hackenbush를 만들어서 나정휘에게 선물로 줬다.

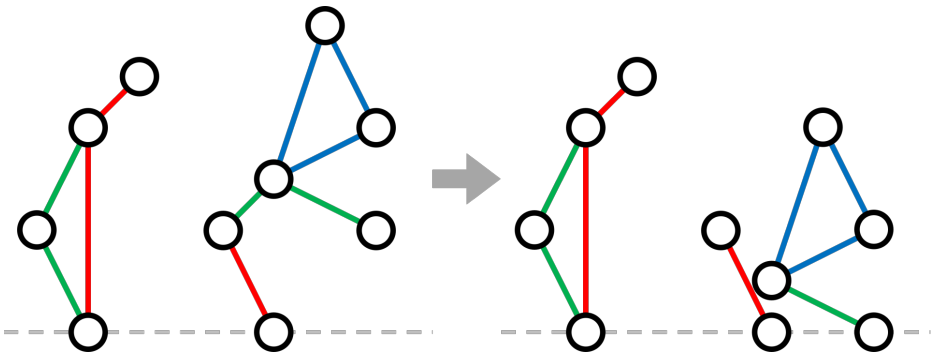
Gravity Hackenbush의 준비 과정은 다음과 같다.

1. 땅을 의미하는 직선 $y = 0$을 그린다.
2. N개의 점을 찍는다. 땅보다 더 낮은 곳에는 점을 찍을 수 없다.
3. 2번 과정에서 찍은 N개의 점 중 서로 다른 두 점을 연결하는 M개의 선을 그린다. 땅과 평행한 선은 그릴 수 없다.
4. 각 선을 빨간색, 파란색, 또는 초록색 중 하나로 칠한다.

Gravity Hackenbush의 게임은 다음과 같이 진행된다.

1. 두 플레이어는 승부가 정해질 때까지 번갈아가면서 턴을 갖는다. 1번 플레이어가 먼저 시작한다.
2. 각 플레이어는 특정 색깔의 선만 자를 수 있다.
    - 빨간색 선은 1번 플레이어만 자를 수 있다.
    - 파란색 선은 2번 플레이어만 자를 수 있다.
    - 초록색 선은 두 플레이어 모두가 자를 수 있다.
3. 각 플레이어는 자신의 턴에 자를 수 있는 선 중 원하는 것을 하나 골라서 자른다.
4. 자신의 턴이 되었을 때 자를 수 있는 선이 없는 플레이어는 패배한다.
5. 선을 자른 다음, 땅과 직간접적으로 연결되지 않은 점과 선들은 연결된 점 또는 선이 땅에 닿을 때까지 내려간다.
6. 떨어지는 과정에서 연결되어 있지 않은 점 또는 선들이 만나더라도 연결되지 않은 것으로 취급한다.

아래 그림은 Gravity Hackenbush에서 선을 하나 자른 뒤의 상태를 보여준다.



게임을 선물받은 나정휘는 난정휘와 함께 게임을 여러 번 플레이하면서 필승법을 찾아냈다. 마침 천하제일 코딩대회에 낼 문제가 부족했던 나정휘는 필승법을 찾는 문제를 대회에 출제하기로 했다.

게임의 초기 상태가 주어지면 나정휘와 난정휘가 모두 최선을 다해 플레이했을 때 누가 이기는지 구해보자. 나정휘가 1번 플레이어, 난정휘가 2번 플레이어다.

입력 형식

첫째 줄에 점과 선의 개수를 나타내는 정수 N, M이 공백으로 구분되어 주어진다. ($1 \le N \le 200\,000$, $0 \le M \le 500\,000$)

둘째 줄부터 N개의 줄에 i번 점의 좌표 x_i, y_i가 한 줄에 하나씩 공백으로 구분되어 주어진다. ($-10^9 \le x_i \le 10^9$, $0 \le y_i \le 10^9$)

다음 M개의 줄에 i번째 선이 연결하는 두 점의 번호 v_i, w_i와 색깔 c_i가 공백으로 구분되어 주어진다. ($1 \le v_i, w_i \le N$, $v_i \neq w_i$, c_i는 R 또는 G 또는 B)

두 점의 좌표는 모두 서로 다르고, 연결하는 두 점의 쌍이 동일한 선이 여러 개 주어지지 않는다. 또한, x축과 평행한 선은 주어지지 않는다.

처음에 모든 점들은 땅과 직간접적으로 연결되어 있다.

입력으로 주어지는 수는 모두 정수이다.

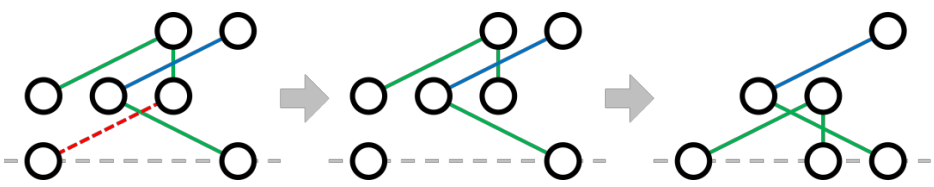
출력 형식

두 사람이 최선을 다해서 플레이할 때, 나정휘가 이긴다면 "jhnah917", 난정휘가 이긴다면 "jhnan917"을 출력한다.

예제

표준 입력(stdin)	표준 출력(stdout)
7 5 0 0 3 0 0 1 1 1 2 1 2 2 3 2 1 5 R 2 4 G 4 7 B 5 6 G 6 3 G	jhnah917

빨간 간선을 자르면 아래 그림과 같이 변한다.



표준 입력(stdin)	표준 출력(stdout)
10 10 -3 0 -4 2 -3 4 -2 5 3 0 2 2 3 3 5 4 4 6 5 2 1 2 G 1 3 R 2 3 G 3 4 R 5 6 R 6 7 G 7 8 B 7 9 B 8 9 B 7 10 G	jhnan917

지문에 나온 그림과 동일한 상태이다.

참고

- Python 사용자는 PyPy로 제출하는 것을 권장한다.

B. K-균형 잡힌 수

우리가 주로 사용하는 10진법은 모든 수를 0부터 9까지의 숫자들의 조합으로 표기한다.

양의 정수 N을 10진법으로 표기하기 위해 사용한 숫자들에 대해, 0으로 시작하지 않으면서 (가장 많이 사용한 숫자의 등장 횟수) - (가장 적게 사용한 숫자의 등장 횟수) $\leq K$이면 N을 K-균형 잡힌 수라고 하자.

양의 정수 X, K가 주어지면 X보다 작거나 같은 가장 큰 K-균형 잡힌 수를 구해보자.

입력 형식

첫째 줄에 테스트 케이스의 개수 T가 주어진다. ($1 \leq T \leq 10^5$)

둘째 줄부터 T개의 줄에 각각 양의 정수 X, K가 공백으로 구분되어 주어진다. ($1 \leq X < 10^{10^5}$, $1 \leq K \leq 10^5$)

모든 테스트 케이스에서 X의 길이의 합은 10^5 이하이다.

출력 형식

각 테스트 케이스의 정답을 한 줄에 하나씩 차례대로 출력한다.

예제

표준 입력(stdin)	표준 출력(stdout)
4 10000 1 10000 3 112222 1 32334837373633003033 3	9999 10000 112221 32334837373488887422

32334837373633003033은 2^{64}보다 크다.

참고

- 입력과 출력은 64비트 정수 범위를 넘을 수 있다.
- 사용하지 않은 숫자는 등장 횟수를 고려하지 않는다.
- Python 사용자는 PyPy로 제출하는 것을 권장한다.

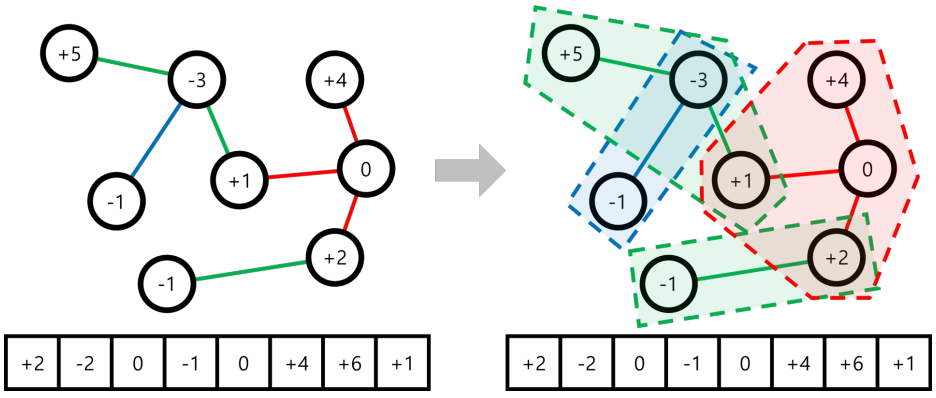
C. Merge the Tree and Sequence

정휘는 정점이 N개인 트리와 길이가 N인 수열을 갖고 있다. 트리의 각 정점에는 정수가 하나씩 적혀 있으며, 각 간선은 1 이상 20만 이하의 자연수로 표현되는 색깔 중 하나로 칠해져 있다.

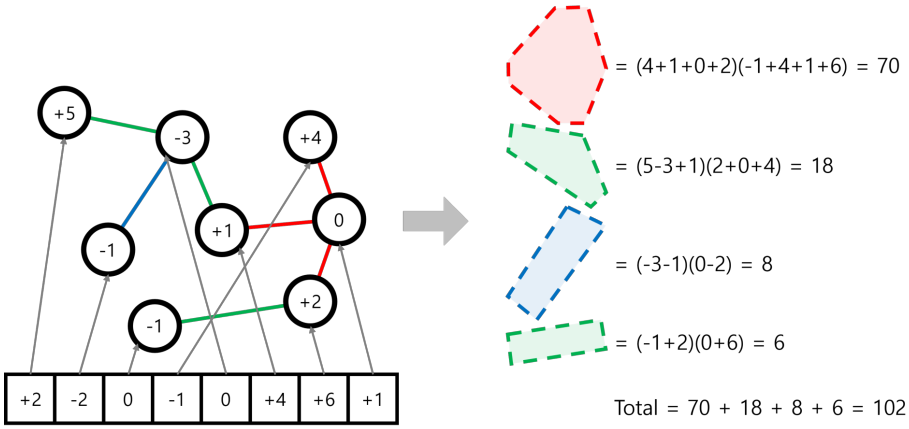
정휘는 트리와 수열을 따로 구분해야 하는 게 귀찮아서, 트리의 정점과 수열의 원소를 일대일대응시키는 방식으로 트리와 수열을 합치기로 했다. 물론 그냥 합치는 건 재미가 없으니, 트리와 수열을 합칠 때의 점수를 아래와 같이 정의하기로 했다.

- 아래 조건을 만족하도록 트리의 간선들을 1개 이상의 구역으로 나눈다. 아래 방식으로 트리를 나누는 방법은 유일하다.
  - 트리의 모든 간선은 정확히 하나의 구역에 속한다.
  - 모든 구역은 적어도 1개 이상의 간선을 포함한다.
  - 만약 한 끝점을 공유하는 두 간선의 색깔이 같다면 두 간선은 같은 구역에 속한다.
- 어떤 구역의 점수는 (구역에 속한 간선들의 끝점에 적힌 수의 합) × (구역에 속한 간선들의 끝점에 대응되는 수열의 원소의 합)으로 정의된다.
- 트리와 수열을 합칠 때의 점수는 모든 구역의 점수를 더한 값이다.

예를 들어, 아래 트리는 4개의 구역으로 나누어진다.



그리고, 만약 정휘가 아래와 같은 방식으로 수열과 트리를 합친다면, 아래와 같은 102점을 얻게 된다.



정휘는 트리와 수열을 합칠 때 얻을 수 있는 점수의 최솟값과 최댓값이 궁금해졌지만, 트리와 수열의 크기가 너무 커서 계산하지 못하고 있다. 컴퓨터를 잘 다루는 여러분들이 정휘를 도와 얻을 수 있는 점수의 최솟값과 최댓값을 구해보자.

입력 형식

첫째 줄에 트리의 정점의 개수와 수열의 길이를 의미하는 정수 N이 주어진다. ($2 \le N \le 200\,000$)

둘째 줄부터 $N-1$개의 줄에 걸쳐, 트리의 간선을 의미하는 세 정수 v_i, w_i, c_i가 한 줄에 하나씩 공백으로 구분되어 주어진다. 이는 v_i번 정점과 w_i번 정점을 연결하는 간선의 색깔이 c_i라는 것을 의미한다. ($1 \le v_i, w_i \le N$, $1 \le c_i \le 200\,000$)

다음 줄에는 트리의 $1, 2, \cdots, N$번 정점에 적힌 정수 $A_1, A_2, \cdots, A_N$이 공백으로 구분되어 주어진다. ($-1\,000 \le A_i \le 1\,000$)

다음 줄에는 길이가 N인 수열의 원소를 나타내는 정수 $B_1, B_2, \cdots, B_N$이 공백으로 구분되어 주어진다. ($-1\,000 \le B_i \le 1\,000$)

출력 형식

첫째 줄에 정휘가 얻을 수 있는 점수의 최솟값을 출력한다.

둘째 줄에 정휘가 얻을 수 있는 점수의 최댓값을 출력한다.

예제

표준 입력(stdin)	표준 출력(stdout)
8 1 2 1 2 5 1 2 4 3 5 6 2 6 3 2 6 8 2 8 7 1 5 -3 4 -1 1 0 -1 2 2 -2 0 -1 0 4 6 1	-51 122

참고

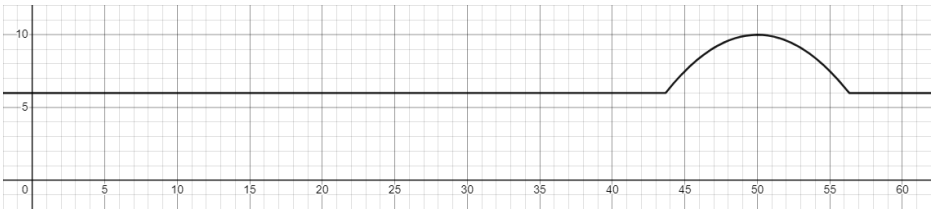
- 정답이 32비트 정수 범위를 넘을 수 있다.

D. 바지 구매

시루는 가을에 입을 바지를 미리 사기 위해 백화점에 왔다. 다리가 길고 저체중인 시루는 길이가 맞는 바지를 사면 허리가 너무 크고, 허리가 맞는 바지를 사면 길이가 짧아서 잘 맞는 바지를 찾지 못하고 있다.

길이가 맞는 바지를 산 다음 허리둘레를 수선을 하거나 허리띠를 하면 되지만, 수선하는 것은 귀찮고 허리띠를 불편해하는 시루는 멋진 아이디어를 생각해냈다. 허리가 조금 크고 길이가 조금 짧은 바지를 산 다음, 허리가 아닌 엉덩이에 바지를 걸치는 방식으로 입는 것이다.

지면에서 x 만큼 떨어진 시루의 하체 둘레는 $f(x) = \max(a(x-b)^2 + c, d)$로 계산할 수 있다. 예를 들어 $f(x) = \max(-0.1(x-50)^2 + 10, 6)$이라고 하면, 시루가 엎드려 있을 때 하체는 다음과 같은 형태이다.



시루는 백화점에서 n개의 바지를 골랐다. i번째 바지의 허리둘레는 u_i, 길이는 v_i이다. 바지를 위에서부터 내려가는 방식으로 허리둘레가 시루의 하체 둘레와 딱 맞도록 바지를 입었을 때, 바지가 끌리지 않으면서 끝부분의 높이가 지면과 일치하는지 확인해 보자. 바지의 허리 부분은 시루의 하체에서 둘레가 가장 큰 위치보다 높거나 같은 곳에서만 걸린다.

입력 형식

첫째 줄에 시루의 하체 둘레를 의미하는 네 정수 a, b, c, d가 공백으로 구분되어 주어진다. ($-10 \le a \le -1$, $1 \le b \le 10\,000$, $1 \le d < c \le 10\,000$)

둘째 줄에 바지의 개수 N이 주어진다. ($1 \le N \le 100\,000$)

셋째 줄부터 N개의 줄에 걸쳐, i번째 줄에 i번째 바지의 둘레와 길이를 의미하는 두 정수 u_i, v_i가 공백으로 구분되어 주어진다. ($d < u_i \le c$, $b \le v_i \le 10\,000$)

출력 형식

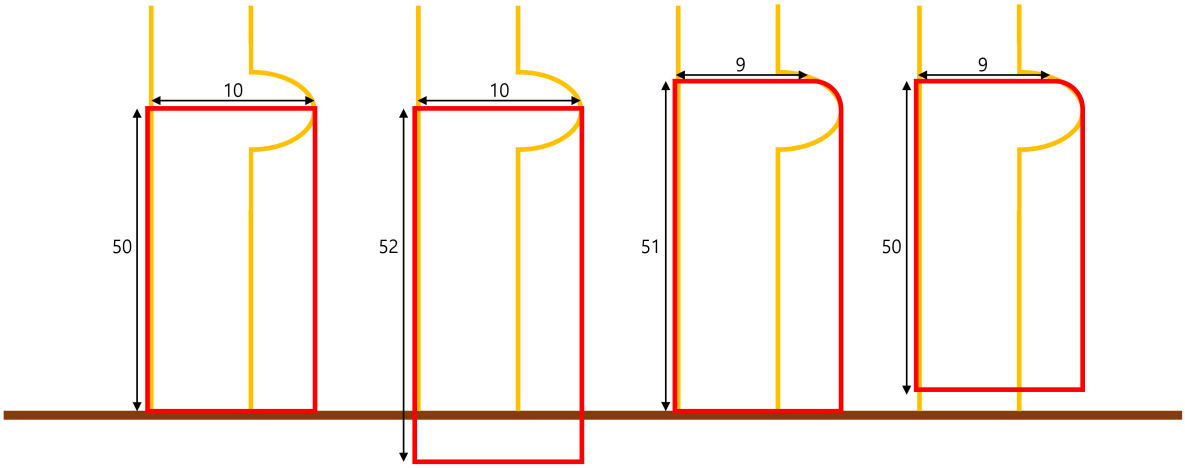
바지가 땅에 끌리지 않고, 바지의 끝부분의 높이가 지면과 일치하는 바지의 개수를 출력한다.

예제

표준 입력(stdin)	표준 출력(stdout)
-1 50 10 6 4 10 50 10 52 9 51 9 50	2

첫 번째 바지와 두 번째 바지는 지면으로부터 50 만큼 떨어진 위치에서 하체에 걸린다. 따라서 길이가 50인 첫 번째 바지는 끝부분의 높이가 지면과 일치하고, 길이가 52인 두 번째 바지는 땅에 끌린다.

세 번째 바지와 네 번째 바지는 지면으로부터 51 만큼 떨어진 위치에서 하체에 걸린다. 따라서 길이가 51인 세 번째 바지는 끝부분의 높이가 지면과 일치하고, 길이가 50인 네 번째 바지의 끝부분은 지면으로부터 1 만큼 떨어진 곳에 위치한다.



E. 반전 수와 쿼리

1부터 N까지의 수가 한 번씩 등장하는 수열 $P = \{1, 2, \cdots, N\}$이 주어진다.

수열 P에 대해, $i < j$ 이면서 $P_i > P_j$를 만족하는 순서쌍 (i, j)의 개수를 P의 **반전 수**라고 정의하자.

이때, 다음 쿼리를 수행하는 프로그램을 작성하시오.

- 1 l r : P_l와 P_r를 교환한다.
- 2 l r : P_l에서 P_r 사이의 수를 뒤집는다. 뒤집은 뒤의 수열은 다음과 같다.

$$\{P_1, \cdots, P_{l-1}, P_r, P_{r-1}, \cdots, P_{l+1}, P_l, P_{r+1}, \cdots, P_N\}$$

각각의 쿼리를 처리한 다음 수열의 반전 수를 2로 나눈 나머지를 출력한다.

입력 형식

첫째 줄에 수열의 길이 N과 쿼리의 개수 Q가 공백으로 구분되어 주어진다. ($2 \le N \le 10^9$, $1 \le Q \le 10^5$)

둘째 줄부터 Q개의 줄에 쿼리의 정보 a, l, r이 공백으로 구분되어 주어진다. ($1 \le a \le 2$, $1 \le l < r \le N$)

출력 형식

각 쿼리를 처리한 다음 수열의 반전 수를 2로 나눈 나머지를 한 줄에 하나씩 출력한다.

예제

표준 입력(stdin)	표준 출력(stdout)
3 4 2 1 3 1 1 2 2 1 3 1 2 3	1 0 1 0

첫 번째 쿼리를 처리하면 수열은 $\{3, 2, 1\}$이 되고 반전 수는 3이다.

두 번째 쿼리를 처리하면 수열은 $\{2, 3, 1\}$이 되고 반전 수는 2이다.

세 번째 쿼리를 처리하면 수열은 $\{1, 3, 2\}$가 되고 반전 수는 1이다.

네 번째 쿼리를 처리하면 수열은 $\{1, 2, 3\}$이 되고 반전 수는 0이다.

참고

- Python 사용자는 PyPy로 제출하는 것을 권장한다.

F. 시간딱딱충

준서는 약속이 있어 T초까지 선린랜드로 가야 한다. 현재 시각은 0초이고, 준서는 준비를 마친 상태로 집에서 언제 출발할지 고민하고 있다.

준서의 집과 선린랜드 사이에는 N개의 신호등이 있다. 선린랜드에 가기 위해서는 N개의 신호등을 차례로 건너야 한다.

준서는 약속 시간을 지키기 위해 신호등이 언제 켜지는지 조사했다. i번째 신호등의 주기는 A_i초이고, 각 주기의 첫 B_i초동안 켜져 있다. 신호등의 전원은 C_i초에 켜져 그때부터 작동을 시작한다. 횡단보도를 건너는 데에는 D_i초가 걸린다.

다시 말해, 어떤 음이 아닌 정수 X에 대해, 다음 두 조건을 만족하면 t초에 i번째 횡단보도를 건널 수 있다.

- $C_i + A_i \times X \leq t$
- $t + D_i \leq C_i + A_i \times X + B_i$

i번째 횡단보도 끝 지점에서 $i+1$번째 횡단보도 시작 지점까지 이동하는 데에는 E_i초가 걸린다. 편의상 E_0는 준서의 집에서 1번째 횡단보도 시작 지점까지 이동하는 데 걸리는 시간, E_N은 N번째 횡단보도 끝 지점에서 선린랜드까지 이동하는 데 걸리는 시간으로 정의한다.

준서는 낭비를 싫어하기 때문에 선린랜드에 도착하는 시간을 딱 T초로 맞추고 싶다. 이동 중에 쉬는 시간을 가지는 것 또한 낭비라고 생각하기 때문에, 집에서 출발하고 나서는 최대한 빨리 목적지에 도착해야 한다.

준서에게 출발 시각만 조절하여 약속 장소에 딱 T초에 도착할 수 있을지 알려주자.

입력 형식

첫째 줄에 테스트케이스의 개수 TC가 주어진다. ($1 \leq TC \leq 300\,000$)

각 테스트케이스는 $N+2$개의 줄로 구성되어 있다.

테스트케이스의 첫째 줄에 정수 N, T가 공백으로 구분되어 주어진다. ($1 \leq N \leq 300\,000$, $1 \leq T \leq 10^9$)

테스트케이스의 둘째 줄부터 $N+1$번째 줄까지 N개의 줄에 걸쳐, i번째 줄에 정수 A_i, B_i, C_i, D_i가 공백으로 구분되어 주어진다. ($1 \leq A_i, B_i, D_i \leq 1\,000$, $0 \leq C_i \leq 1\,000$, $D_i \leq B_i < A_i$)

테스트케이스의 $N+2$번째 줄에 $N+1$개의 정수 $E_0, E_1, \cdots, E_N$이 공백으로 구분되어 주어진다. ($1 \leq E_i \leq 1\,000$)

모든 테스트케이스에서 N의 합은 $300\,000$을 넘지 않는다.

출력 형식

각 테스트케이스마다 출발 시각만 조절하여 약속 장소에 딱 T초에 도착할 수 있다면 “YES”, 아니면 “NO”를 출력한다.

예제

표준 입력(stdin)	표준 출력(stdout)
3 1 2 2 1 1 1 1 1 1 3 2 1 1 1 1 1 1 4 2 1 1 1 1 1	NO YES NO

참고

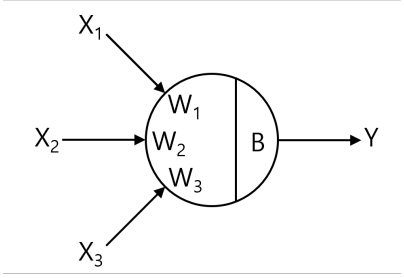
- Python 사용자는 PyPy로 제출하는 것을 권장한다.

G. 인공 신경망

2020년, 선린인터넷고등학교는 서울시 교육청에 의해 인공지능 분야 고등학교로 선정되었다.

정휘는 후배들이 지난 2년 동안 인공지능 교육을 잘 받았는지 확인하기 위해 신경망과 관련된 문제를 출제하기로 했다.

인공 신경망은 여러 개의 인공 신경으로 구성된 망 형태의 구조이다. i번째 인공 신경은 가중치 $W_{i,1}, W_{i,2}, \cdots, W_{i,C_i}$와 편향값 B_i를 갖고 있다. C_i개의 입력 데이터 $X_{i,1}, X_{i,2}, \cdots, X_{i,C_i}$를 받으면 $Y_i = X_{i,1}W_{i,1} + X_{i,2}W_{i,2} + \cdots + X_{i,C_i}W_{i,C_i} + B_i$를 계산해서 출력한다. 즉, $Y_i = \sum_{k=1}^{C_i} X_{i,k}W_{i,k} + B_i$를 출력한다. 아래 그림은 인공 신경의 구성을 그림으로 나타낸 것이다.

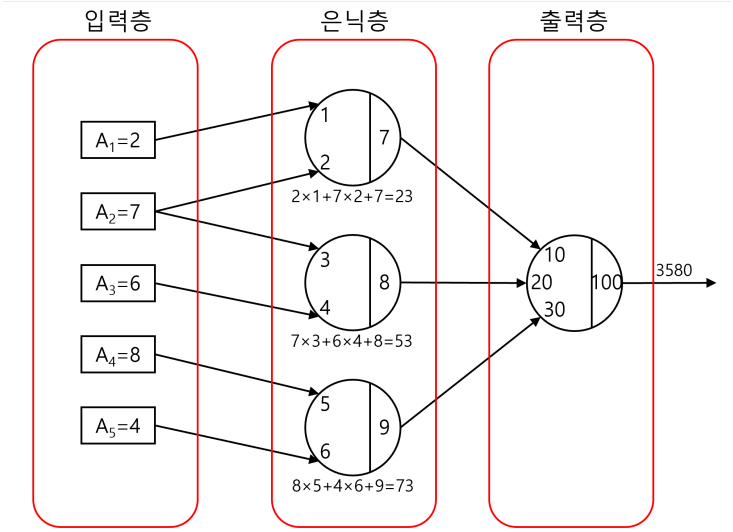


예를 들어 가중치가 $\{3, 1\}$, 편향값이 2인 인공 신경을 생각해보자. 입력이 $\{1, 2\}$이면 출력값은 $1 \times 3 + 2 \times 1 + 2 = 7$이다.

정휘가 준비한 인공 신경망은 입력층, 은닉층, 출력층으로 구성되어 있다.

입력층은 N개의 입력 $A_1, A_2, \cdots, A_N$으로 구성되어 있고, 은닉층은 M개의 인공 신경으로 구성되어 있다. 은닉층의 i번째 인공 신경은 $A_{P_{i,1}}, A_{P_{i,2}} \cdots, A_{P_{i,C_i}}$를 입력으로 받아 출력값을 계산한다. 출력층은 1개의 인공 신경으로 구성되어 있고, 은닉층에 있는 $1, 2, \cdots, M$번째 인공 신경들의 출력값을 순서대로 입력으로 받아 출력값을 계산한다. 편의상 출력층의 인공 신경을 $M + 1$번째 인공 신경이라고 하자.

예를 들어 인공 신경들의 가중치가 $W = \{\{1, 2\}, \{3, 4\}, \{5, 6\}, \{10, 20, 30\}\}$, 편향값이 $B = \{7, 8, 9, 100\}$인 인공 신경망에서 은닉층의 인공 신경들이 입력 층의 $P = \{\{1, 2\}, \{2, 3\}, \{4, 5\}\}$ 번째 값을 입력으로 받는다고 하자. 입력으로 $A = \{2, 7, 6, 8, 4\}$이 주어지면 아래 그림처럼 나타낼 수 있다.



인공 신경망의 구성과 입력 데이터가 주어지면 신경망의 출력값을 구하는 프로그램을 작성해 보자.

입력 형식

첫째 줄에 입력층의 입력 크기 N, 은닉층의 인공 신경 개수 M, 출력값을 계산해야 하는 횟수 Q가 공백으로 구분되어 주어진다. ($1 \le N, M, Q \le 2\,000$)

둘째 줄부터 M번째 줄에 걸쳐 은닉층의 인공 신경의 정보가 한 줄에 하나씩 순서대로 주어진다. 인공 신경의 입력 개수 C_i, 입력 데이터 $P_{i,1}, P_{i,2}, \cdots, P_{i,C_i}$, 가중치 $W_{i,1}, W_{i,2}, \cdots, W_{i,C_i}$, 편향값 B_i가 차례로 공백으로 구분되어 주어진다. ($1 \le C_i \le N$, $1 \le P_{i,j} \le N$, 각 i마다 $P_{i,j}$는 서로 다름)

이어서 출력층의 인공 신경 정보가 한 줄에 걸쳐 주어진다. 가중치 $W_{M+1,1}, W_{M+1,2}, \cdots, W_{M+1,M}$, 편향값 B_{M+1}이 차례로 공백으로 구분되어 주어진다. ($1 \le W_{i,j}, B_i \le 100$)

다음 Q개의 줄에 걸쳐 여러분이 출력값을 구해야 하는 입력 데이터가 한 줄에 하나씩 주어진다. 각 줄은 N개의 정수 $A_1, A_2, \cdots, A_N$으로 구성되어 있다. ($1 \le A_i \le 100$)

입력으로 주어지는 수는 모두 정수이다.

출력 형식

Q개의 줄에 걸쳐 정답을 출력한다.

예제

표준 입력(stdin)	표준 출력(stdout)
5 3 2 2 1 2 1 2 7 2 2 3 3 4 8 2 4 5 5 6 9 10 20 30 100 2 7 6 8 4 1 2 3 4 5	3580 2510

참고

- Python 사용자는 PyPy로 제출하는 것을 권장한다.
- 정답이 32비트 정수 범위를 넘을 수 있다.

H. 최대 최소공배수

1부터 N까지의 수가 있다. 최소공배수가 최대가 되도록 서로 다른 3개의 수를 선택해 보자.

입력 형식

첫째 줄에 테스트케이스의 개수 T가 주어진다. ($1 \le T \le 1\,000$)

둘째 줄부터 T개의 줄에 각각 자연수 N이 주어진다. ($3 \le N \le 100\,000$)

출력 형식

각 테스트케이스마다, 최소공배수의 최댓값을 한 줄에 하나씩 차례대로 출력한다.

예제

표준 입력(stdin)	표준 출력(stdout)
2 3 4	6 12

$N = 3$인 경우, $1, 2, 3$을 선택하면 최소공배수는 6이다.

$N = 4$인 경우, $2, 3, 4$를 선택하면 최소공배수는 12이다.

참고

- 정답이 32비트 정수 범위를 넘을 수 있다.

I. 최장 최장 증가 부분 수열

정휘는 정수로 구성된 $N \times N$ 크기의 배열의 가장 왼쪽 위 칸에서 가장 오른쪽 아래 칸까지 최단 경로로 이동하려고 한다. 한 칸에서 다른 칸으로 이동할 때 서로 변을 공유하는 칸으로만 이동할 수 있다.

정휘는 배열에서 이동하면서 만난 정수들을 순서대로 모아서 성현이에게 선물로 주려고 한다. 성현이는 증가하는 부분 수열의 길이가 긴 수열을 좋아하기 때문에, 정휘는 최장 증가 부분 수열의 길이가 최대인 수열을 만들어서 주려고 한다.

정휘는 문제를 만들어야 하기 때문에 경로를 찾을 여유가 없어서 여러분들에게 도움을 요청했다. $N \times N$ 크기의 배열이 주어지면 만들 수 있는 수열 중 최장 증가 부분 수열의 길이의 최댓값을 대신 구해보자.

입력 형식

첫째 줄에 N이 주어진다. $(1 \le N \le 100)$

이어 N개의 줄에, 각각 N개의 정수가 공백을 사이에 두고 주어진다. 각 정수는 1 이상 10 000 이하다.

출력 형식

최장 증가 부분 수열의 길이의 최댓값을 출력한다.

예제

표준 입력(stdin)	**표준 출력**(stdout)
3 1 2 3 3 2 1 3 4 5	4
3 1 1 1 1 1 1 1 1 1	1

첫 번째 예시에서 가능한 방법으로 $\{1, 2, 3, 1, 5\}, \{1, 2, 2, 4, 5\}, \{1, 3, 2, 4, 5\}, \{1, 3, 3, 4, 5\}$ 등이 있다.

참고

- Python 사용자는 PyPy로 제출하는 것을 권장한다.
- **부분 수열**이란 주어진 수열에서 1개 이상의 원소를 골라 원래 순서대로 나열하여 얻은 수열을 말한다.
- **증가하는 부분 수열**이란 맨 처음 원소를 제외한 모든 원소가 바로 전 원소보다 큰 수열을 말한다. 다시 말해, 길이가 N인 부분 수열 A가 있을 때, $A_{i-1} < A_i (2 \le i \le N)$을 만족하면 A는 증가하는 부분 수열이다.
- **최장 증가 부분 수열**이란 주어진 수열의 증가하는 부분 수열 중 길이가 가장 긴 수열을 의미한다.

J. 행성 정렬

행성 정렬은 행성들이 일직선으로 정렬된 것처럼 보이는 현상이다. 최근 지구에서도 18년 만에 행성 정렬을 관측할 수 있었다.

평행세계의 준서가 살고 있는 지구에서는 N개의 행성을 관측할 수 있다. 준서는 얼마나 기다려야 N개의 행성이 일렬로 나열되는 순간을 볼 수 있을지 궁금해졌다.

하늘을 열심히 관찰한 결과, 준서는 다음 사실들을 알 수 있었다.

- N개의 행성이 일렬로 나열되는 순간이 존재한다.
- 행성 정렬의 주기는 10^9초 이하이다.
- $1, 2, 3$번째 행성은 T_1초마다 일렬로 나열된다.
- $2, 3, 4$번째 행성은 T_2초마다 일렬로 나열된다.
- $\dots$
- $N-2, N-1, N$번째 행성은 T_{N-2}초마다 일렬로 나열된다.

준서를 위해 행성 정렬의 주기를 구해주자.

입력 형식

첫째 줄에 정렬되길 바라는 행성의 개수 N이 주어진다. $(3 \le N \le 100\,000)$

둘째 줄에 행성이 일렬로 나열되는 주기를 나타내는 정수 $T_1, T_2, \cdots, T_{N-2}$가 공백으로 구분되어 주어진다. $(1 \le T_i \le 100\,000)$

출력 형식

행성 정렬의 주기를 출력한다. 행성 정렬의 주기는 10^9초 이하이다.

예제

표준 입력(stdin)	표준 출력(stdout)
5 1 2 3	6