UCPC 2025

전국 대학생 프로그래밍 대회 동아리 연합
여름 대회 2025

Finals

Official Problemset

본선 문제

2025년 7월 26일, 11:00 – 16:00

주최

전국 대학생 프로그래밍 대회 동아리 연합

후원

나정휘 (jhnah917)

정재헌(gravekper)

UCPC 2025

Finals - 본선대회

문제 목록

문제지에 있는 문제가 총 13문제가 맞는지 확인하시기 바랍니다.

모든 문제의 메모리 제한은 1GB로 동일합니다.

문제 A. 트리 이사

시간 제한 1초 메모리 제한 1024 MB

정점 N개와 간선 $N-1$개로 이루어진 트리 T가 주어진다. 정점은 1번부터 N번까지 번호가 매겨져 있고, 간선은 1번부터 $(N-1)$번까지 번호가 매겨져 있다.

T의 모든 정점을 D차원 정수 격자 $\mathbb{Z}^D$ 위의 점으로 이사시키고자 한다. 이때, 이사된 정점들의 위치는 다음 조건을 만족해야 한다.

- i번 정점이 이사된 위치를 $(x_{i,1}, x_{i,2}, \cdots, x_{i,D})$라 하자. $x_{i,j}$는 정수이다.
- 트리에서 i번 정점과 j번 정점의 거리 $\text{dist}(i,j)$를 i번 정점에서 j번 정점으로 가는 경로에 있는 간선의 수로 정의하자.
- 모든 $1 \le i < j \le N$에 대해서 $|x_{i,1} - x_{j,1}| + |x_{i,2} - x_{j,2}| + \cdots + |x_{i,D} - x_{j,D}| = \text{dist}(i,j)$여야 한다.

트리를 이사할 수 있는 최소 차원 D를 구하고, 조건을 만족하도록 정점들을 이사시키자.

입력

첫째 줄에 트리의 정점 수 N이 주어진다. $(2 \le N \le 2000)$

이후 $N-1$개의 줄에 걸쳐, 그중 i번째 줄에는 트리의 i번 간선이 잇는 두 정점 번호가 공백으로 구분되어 주어진다.

출력

첫째 줄에 트리를 이사할 수 있는 최소 차원 D를 출력한다.

이후 N개의 줄에 걸쳐, 그중 i번째 줄에 D개의 정수 $x_{i,j}$를 공백으로 구분해 출력한다. $(1 \le j \le D)$

$x_{i,j}$는 i번 정점이 이사되는 격자점의 j번째 좌표를 의미하며 $-10^9 \le x_{i,j} \le 10^9$이어야 한다.

트리를 여러 방법으로 이사시킬 수 있는 경우 그중 아무 것이나 출력한다.

입출력 예시

표준 입력(stdin)	표준 출력(stdout)
5 1 2 1 3 1 4 1 5	2 0 0 -1 0 1 0 0 -1 0 1

문제 B. 중력 발전소

시간 제한 3 초 메모리 제한 1024 MB

중력 발전소란 중력의 힘을 이용하여 에너지를 생산하는 발전소이다. 중력 발전소는 수직으로 배열된 $N+1$ 개의 공간으로 구성된다. 각 공간에는 0번부터 N번까지 차례대로 번호가 매겨져 있다. 이때 가장 위에 위치한 공간이 0번, 가장 아래에 위치한 공간이 N번이다.

중력 발전소를 가동시키면, 0번 공간에 공을 놓은 후 중력의 힘으로 공을 N번 공간까지 떨어뜨린다. 인접한 두 공간 사이에는 떨어지는 공을 감지할 수 있는 고리형 발전기가 있다. 구체적으로, $1 \le i \le N$에 대해 공이 $i-1$번 공간에서 i번 공간으로 떨어질 경우 두 공간 사이의 고리형 발전기가 이를 감지하여 A_i만큼의 에너지를 생산한다.

과학자들은 중력 발전소의 생산성을 높이기 위해 **반중력 장치**를 개발하였다. 중력 발전소에는 총 $\frac{N(N+1)}{2}$개의 반중력 장치가 존재하며, 각 장치는 $0 \le i < j \le N$을 만족하는 정수 쌍 (i, j)에 대응된다. (i, j)에 대응되는 반중력 장치를 공이 j번 공간에 있을 때 사용하면 중력을 거슬러 공을 i번 공간으로 이동시킬 수 있다. 고리형 발전기는 공을 감지할 때마다 독립적으로 에너지를 생산하기 때문에 반중력 장치를 사용하여 각 고리형 발전기가 여러 번 에너지를 생산하도록 설계할 수 있다.

다만 반중력 장치는 중력 발전소를 불안정하게 만들 수 있기 때문에 사용에 몇 가지 제약이 존재한다.

1. 반중력 장치를 사용할 때, 공이 거슬러 올라가는 공간의 개수마다 K만큼의 에너지를 소비한다. 즉, (i, j)에 대응되는 장치를 사용하면 $K \times (j-i)$ 만큼의 에너지를 소비한다.
2. 각 반중력 장치는 최대 한 번만 사용될 수 있다.
3. 반중력 장치를 사용하는 총 횟수는 최대 M번으로 제한되어 있다. 이때 M번을 모두 사용할 필요는 없다.

발전은 반드시 공을 0번 공간에 둔 상태에서 시작하고, 공이 N번 공간에 존재하는 상태에서 종료되어야 한다. 이때 공이 N번 공간에 있는 상태에서도 반중력 장치를 사용할 수 있다.

반중력 장치를 적절히 사용해 생산할 수 있는 총 에너지의 최댓값을 구하여라.

입력

첫째 줄에 세 정수 N, M, K가 공백으로 구분되어 주어진다. $(1 \le N \le 100\,000;\ 0 \le M \le \min(\frac{N(N+1)}{2}, 10^9);\ 0 \le K \le 10^9)$

둘째 줄에 N개의 정수 $A_1, A_2, \cdots, A_N$이 공백으로 구분되어 주어진다. $(1 \le A_i \le 10^9;\ \sum A_i \le 10^9)$

출력

생산할 수 있는 총 에너지의 최댓값을 출력한다.

입출력 예시

표준 입력(stdin)	표준 출력(stdout)
7 1 5 9 7 4 3 3 8 8	49
4 3 100 1 2 3 4	10
5 5 4 7 3 7 3 7	52

노트

첫 번째 예제의 경우 다음과 같은 방식으로 발전을 진행한다.

1. 0번 공간의 공을 7번 공간까지 떨어뜨린다. 이때 총 $9+7+4+3+3+8+8=42$만큼의 에너지를 생산한다.
2. 반중력 장치를 사용하여 7번 공간의 공을 0번 공간으로 끌어올린다. 이때 $5\times7=35$만큼의 에너지를 소모한다.
3. 다시 0번 공간의 공을 7번 공간까지 떨어뜨린다. 1번 과정과 동일하게 42만큼의 에너지를 생산한다.

이때 총 $42-35+42=49$만큼의 에너지를 생산하며, 이보다 더 많이 생산할 수 없다.

두 번째 예제의 경우 반중력 장치를 사용하지 않는 것이 이득이며, 단순히 공을 떨어뜨리면 $1+2+3+4=10$만큼의 에너지를 생산할 수 있다.

문제 C. It's a Mod, Mod, Mod, Mod World 2

시간 제한 1초 메모리 제한 1024 MB

서로 다른 양의 정수 N개로 이루어진 집합 A가 주어진다. A의 부분집합과 1보다 큰 정수 K를 적절히 골라, 부분집합의 모든 원소를 K로 나눈 나머지가 서로 같게 하려고 한다.

고를 수 있는 부분집합의 최대 크기를 구하여라.

입력

첫째 줄에 집합 A의 크기 N이 주어진다. $(1 \le N \le 20000)$

둘째 줄에 A의 원소를 의미하는 N개의 정수 $A_1, A_2, \cdots, A_N$이 공백으로 구분되어 주어진다. $(1 \le A_i \le 10^9)$

A의 모든 원소는 서로 다르다.

출력

고를 수 있는 부분집합의 최대 크기를 출력한다.

입출력 예시

표준 입력(stdin)	표준 출력(stdout)
5 5 7 8 10 11	3

노트

A의 부분집합 $\{5, 8, 11\}$, $K = 3$을 골라 크기 3의 부분집합을 고를 수 있다.

크기 4 이상의 부분집합을 고를 수 없으므로, 고를 수 있는 부분집합의 최대 크기는 3이다.

문제 D. Wildcard and Query

시간 제한 2 초 메모리 제한 1024 MB

알파벳 소문자로 이루어진 문자열 S가 주어진다.
알파벳 소문자 및 "*"로 이루어진 문자열 T는 다음 조건을 만족할 때 S와 **매칭**된다.

- T의 각 "*"들을 각각 길이가 0 이상인 임의의 문자열로 대체했을 때 S를 만들 수 있다.

예를 들어 "a*b"는 "ab", "acb", "aabb" 등과 **매칭**되지만 "abc"와는 **매칭**되지 않는다.
만약 다음 조건을 만족한다면 **매칭**이 **유일**하다.

- T의 각 "*"를 문자열로 대체하여 S를 만드는 방법이 유일하다.

예를 들어 "a*b*c"는 "abc", "axbxc"와의 **매칭**은 **유일**하지만, "abbc"와의 **매칭**은 **유일**하지 않다. 첫 번째 "*"를 "b", 두 번째 "*"를 빈 문자열로 바꾸어도 되고, 첫 번째 "*"를 빈 문자열, 두 번째 "*"를 "b"로 바꾸어도 되기 때문이다.
S가 주어질 때 다음 쿼리에 답하는 프로그램을 작성하여라.

- 문자열 T가 주어지면 T가 S와 **매칭**되는지, **매칭**된다면 **유일**한지 여부를 출력한다.

입력

첫째 줄에 알파벳 소문자로 이루어진 문자열 S가 주어진다. $(1 \le |S| \le 300\,000)$
둘째 줄에 쿼리의 수 Q가 주어진다. $(1 \le Q \le 300\,000)$
이후 Q개의 줄에 걸쳐, 그중 i번째 줄에는 문자열 T_i가 주어진다. T_i는 알파벳 소문자 및 "*"로 이루어져 있다.
모든 쿼리에 대해 T_i들의 길이의 합은 300 000 이하이다.

출력

각 쿼리에 대한 답을 Q개의 줄에 걸쳐 순서대로 출력한다.
그중 i번째 줄에는 각 T_i에 대해서 S와 **매칭**되지 않는 경우 **0**, **매칭**되고 **유일**한 경우 **1**, 그 외의 경우 **2**를 출력한다.

입출력 예시

표준 입력(stdin)	표준 출력(stdout)
axbbc 3 abc a*c a*b*c	0 1 2

문제 E. 중간 뒤집기

시간 제한 2 초 메모리 제한 1024 MB

길이가 N인 수열 A에서, $1 \le i \le j \le N$에 대하여 $f(i,j)$를 A의 i번째부터 j번째 원소까지의 연속된 부분수열을 뒤집어서 얻어진 수열로 정의한다. 예를 들어, $A = [3,1,4,1,5]$라면 $f(2,3) = [3,4,1,1,5]$, $f(1,5) = [5,1,4,1,3]$, $f(1,1) = [3,1,4,1,5]$이다.

$f(i,j)$에서 i와 j를 정하는 경우의 수는 $\frac{N(N+1)}{2}$가지가 있다. 수열 A가 주어질 때, 모든 $f(i,j)$ 중 서로 다른 수열의 개수를 구하여라.

입력

첫째 줄에 수열 A의 길이 N이 주어진다. $(1 \le N \le 500\,000)$

둘째 줄에 A의 원소를 의미하는 N개의 정수 $A_1, A_2, \cdots, A_N$이 공백으로 구분되어 주어진다. $(1 \le A_i \le 10^9)$

출력

$f(i,j)$로 가능한 서로 다른 수열의 개수를 출력한다.

입출력 예시

표준 입력(stdin)	표준 출력(stdout)
4 3 1 4 1	6
1 20250726	1

노트

첫 번째 예제에서, 가능한 $f(i,j)$는 다음과 같다.

- $f(1,1) = f(2,2) = f(2,4) = f(3,3) = f(4,4) = [3,1,4,1]$
- $f(1,2) = [1,3,4,1]$
- $f(1,3) = [4,1,3,1]$
- $f(1,4) = [1,4,1,3]$
- $f(2,3) = [3,4,1,1]$
- $f(3,4) = [3,1,1,4]$

문제 F. ChannelTalk

시간 제한 4 초 메모리 제한 1024 MB

채널코퍼레이션의 커뮤니케이션 플랫폼 채널톡은 고객과의 실시간 소통을 통해 기업의 지속 가능한 성장을 돕는 올인원 AI 메신저이다. CRM 데이터와 AI를 활용해 상담 효율을 높이고 고객 경험을 개선하며, '고객이 답이다'라는 철학 아래 고객 중심의 서비스를 제공한다. 일본에서 업계 1위 수준의 점유율과 빠른 매출 성장을 이루며 아시아 시장을 넘어 미국 진출도 추진 중이다. 이러한 성과의 핵심은 '제품'에 있으며, 전체 직원 절반 이상이 개발자로 구성되어 하나의 우수한 제품 개발에 집중하고 있다.

채널코퍼레이션은 고객 의견을 수렴하기 위해 채널톡에서 여러 개의 **토론 채널**을 시범 운영하기로 하였다.

채널은 총 N개로, 1번부터 N번까지의 번호가 붙어 있다. 모든 채널은 같은 의견 주제를 다루며, i번 채널에는 처음에 해당 의견에 찬성하는 사람 A_i명과 반대하는 사람 B_i명이 있다. 또한 i번 채널의 최대 수용 인원은 C_i명이며, C_i는 짝수이다. (예외적으로 마지막 N번 채널은 수용 인원 제한이 없다.)

시스템 동작 중 가끔 새로운 참여자가 특정 채널에 들어온다. 이 참가자 역시 찬성 또는 반대 중 하나의 의견을 가지고 있다. 새로운 참가자가 들어올 때, 채널톡의 관리 규칙에 따라 다음과 같은 일이 일어난다.

- 새로운 참가자가 들어왔을 때 해당 채널의 인원수가 수용 한도를 넘지 않으면 그대로 채널에 머무른다.
- 해당 채널의 인원수가 정원을 초과했다면, 그 순간 해당 채널의 총 인원은 홀수가 된다. 이 경우 찬성 측과 반대 측 인원수 중 더 많은 쪽에서 한 명의 참가자가 자동으로 다음 번호의 채널로 이동된다. (이동된 사람의 의견 성향은 변하지 않는다.)
- 만약 이 과정으로 인해 다음 채널의 인원수가 수용 한도를 넘는다면 같은 과정이 반복된다. 이 과정은 모든 채널의 인원수가 수용 한도 이하가 될 때까지 반복된다. 특히, 마지막 N번 채널은 수용 인원 제한이 없기 때문에 이 과정은 언젠가는 끝나게 된다.

당신은 시범 운영을 돕기 위해, 새로운 참가자들이 토론 채널에 들어올 때 각 채널의 인원수를 빠르게 관리하는 프로그램을 작성해야 한다. 구체적으로, 다음과 같은 쿼리가 주어질 때, 해당 쿼리를 빠르게 처리해야 한다.

- 1 x v: x번 채널에 찬성 의견을 가진 사람 v명이 들어온다.
- 2 x v: x번 채널에 반대 의견을 가진 사람 v명이 들어온다.
- 3 x: x번 채널에 있는 사람 중 찬성 의견과 반대 의견을 가진 사람의 수를 각각 출력한다.

단, 1번과 2번 종류의 쿼리에서, v명의 사람들은 채널에 한 사람씩 차례로 들어가고, 이전 사람으로 인해 생긴 모든 이동 과정이 끝난 뒤 다음 사람이 들어온다고 생각한다.

입력

첫째 줄에 두 정수 N, Q가 공백으로 구분되어 주어진다. ($2 \le N \le 200\,000$; $1 \le Q \le 200\,000$)

이후 $N-1$개의 줄에 걸쳐, 그중 i번째 줄에는 세 정수 A_i, B_i, C_i가 공백으로 구분되어 주어진다. ($0 \le A_i, B_i \le 10^9$; $2 \le C_i \le 10^9$; $A_i + B_i \le C_i$; C_i는 짝수)

$N+1$번째 줄에는 두 정수 A_N, B_N이 공백으로 구분되어 주어진다. ($0 \le A_N, B_N \le 10^9$)

이후 Q개의 줄에 걸쳐, 그중 i번째 줄에는 i번 쿼리의 정보가 지문에서 안내된 형태로 주어진다. 모든 쿼리에 대해 $1 \le x \le N$, $1 \le v \le 10^9$이다.

3번 종류의 쿼리가 적어도 하나 주어짐이 보장된다.

출력

모든 3번 종류의 쿼리에 대해, 해당 채널에 있는 사람 중 찬성과 반대 의견을 가진 사람의 수를 순서대로 출력한다.

입출력 예시

표준 입력(stdin)	표준 출력(stdout)
4 5 2 1 4 0 3 6 2 0 2 2 4 1 2 2 3 2 1 2 4 3 2 3 4	2 3 3 3 5 4
2 3 0 0 4 0 0 2 1 6 3 1 3 2	0 4 0 2

문제 G. 자율 주행 프로그램 개발

시간 제한 2 초 메모리 제한 1024 MB

현대모비스는 자율주행, 인포테인먼트, 전동화 등 미래차 핵심 기술을 선도하는 글로벌 6위의 자동차 부품 기업이다. 현대모비스는 전통적인 하드웨어 중심의 자동차 부품 회사에서 소프트웨어(SW) 중심의 기술 기업으로 거듭나기 위해 연구개발 역량을 강화하고 있다.

특히 센서와 로직 등 자율주행에 특화된 융합 소프트웨어 분야 개발에 박차를 가하고 있으며, 빅데이터/인공지능을 연구개발에 적극 활용하고 있다. 앞으로도 현대모비스는 그동안 축적해 온 하드웨어 역량과 소프트웨어 기술의 시너지를 글로벌 시장으로 확대해 나갈 예정이다.

현대모비스의 자율 주행 시험장은 N개의 거점으로 구성되어 있다. 1번 거점이 출발지이며, 각 거점에는 왼쪽과 오른쪽의 두 방향으로 이루어진 갈림길이 있다. 각 갈림길에는 도로가 연결되었을 수도, 아닐 수도 있다. 도로가 연결된 경우 이 도로를 통해 다른 거점으로 이동할 수 있다. 도로의 개수는 정확히 $N-1$개이며 1번 거점에서 출발해서 다른 모든 거점에 도달할 수 있다. 즉, 자율 주행 시험장은 **1번 거점을 루트로 하는 이진 트리** 형태이다.

당신의 목표는 A번 거점에 있는 자동차를 B번 거점으로 이동시키기 위한 프로그램을 작성하는 것이다. 프로그램은 다음과 같은 세 개의 명령어로 구성된다.

- L (좌회전): 자동차가 현재 거점의 왼쪽 갈림길에 연결된 도로를 통해 이동한다. 왼쪽 갈림길에 연결된 도로가 없을 경우 **에러**를 일으킨다.
- R (우회전): 자동차가 현재 거점의 오른쪽 갈림길에 연결된 도로를 통해 이동한다. 오른쪽 갈림길에 연결된 도로가 없을 경우 **에러**를 일으킨다.
- B (후진): 자동차가 후진해 1번 거점과 가까운 방향으로 연결된 도로를 통해 이동한다. 현재 1번 거점에 위치해 있을 경우 **에러**를 일으킨다.

단, 제어 시스템이 망가져 입력된 프로그램을 두 번 실행하게 되었다. 즉, 입력된 프로그램이 LRB라면 LRBLRB를 대신 실행한다.

두 번 실행하는 동안 **에러**가 발생하지 않으면서 A번 거점에 있는 자동차를 B번 거점으로 이동시키는 프로그램이 존재하는지 판별하고, 존재한다면 가장 짧은 프로그램을 구하여라.

입력

첫째 줄에 테스트 케이스의 개수 T가 주어진다. ($1 \le T \le 200\,000$)

각 테스트 케이스의 첫째 줄에는 거점의 수 N이 주어진다. ($2 \le N \le 200\,000$)

각 테스트 케이스의 둘째 줄에는 시작 거점과 도착 거점의 번호를 의미하는 두 정수 A, B가 공백으로 구분되어 주어진다. ($1 \le A, B \le N$; $A \ne B$)

각 테스트 케이스의 셋째 줄부터 N개의 줄에 걸쳐 두 정수 L_i, R_i가 공백으로 구분되어 주어진다.

L_i는 i번 거점의 왼쪽 갈림길에 도로가 연결되어 있다면 도로에 연결된 거점의 번호, 그렇지 않다면 0이다. R_i는 i번 거점의 오른쪽 갈림길에 도로가 연결되어 있다면 도로에 연결된 거점의 번호, 그렇지 않다면 0이다.

주어진 구조가 1번 거점을 루트로 하는 트리 형태임이 보장된다.

모든 테스트 케이스에 대해 N의 합은 1000000 이하이다.

출력

각 테스트 케이스에 대한 답을 T개의 줄에 걸쳐 순서대로 출력한다.

각 테스트 케이스에 대해, 조건을 만족하는 프로그램이 존재한다면 그중 가장 짧은 프로그램을 출력한다. 이때 가장 짧은 프로그램이 여러 개일 경우 그중 아무 것이나 출력한다. 만약 조건을 만족하는 프로그램이 존재하지 않는다면 대신 **ERROR**를 출력한다.

입출력 예시

표준 입력(stdin)	표준 출력(stdout)
2 7 7 3 2 3 4 5 6 0 7 0 0 0 0 0 0 0 7 6 2 2 3 4 5 6 0 7 0 0 0 0 0 0 0	BBR ERROR
2 12 3 6 12 7 0 9 0 0 2 0 6 8 0 0 0 0 0 11 0 0 0 4 0 3 10 5 12 1 5 12 7 0 9 0 0 2 0 6 8 0 0 0 0 0 11 0 0 0 4 0 3 10 5	BBBBLRL ERROR

문제 H. 시간선 통합

시간 제한 2 초 메모리 제한 1024 MB

"과거를 지배하는 자가 미래를 지배한다."

— 조지 오웰, '1984'

전 우주의 지성체로 구성된 UCPC (Universe Consistency Preservation Committee, 우주 균일성 유지 위원회)는 우주가 N개의 시간선으로 분기되어 있음을 발견했다. i번째 시간선은 1 이상 N 이하의 정수 시각 a_i를 가지며 각 시간선의 시각은 모두 다르다. 아직 각 시간선은 안정된 상태지만 언제든지 불안정해질 수 있으며, 불안정한 시간선은 전 우주에 인간의 인지 능력을 벗어난 악영향을 끼칠 수 있다. 최악의 경우 우주의 절멸까지 우려해야 하는 중대한 사태임에 따라, UCPC는 N개의 시간선을 현재 시각 t에 맞춰 하나로 통합해야 한다는 결론에 도달했다.

시간선 통합은 인접한 두 시간선을 시각의 최댓값 또는 최솟값을 가지는 하나의 시간선으로 합치는 과정이다. $1 \le i < N$일 때 i번째 시간선과 $i+1$번째 시간선은 인접해 있다. 시간선 통합은 우주적 규모의 비가역적 현상이기 때문에 극도로 신중히 결정되어야 하며, 두 시각의 최댓값을 선택하는 통합과 최솟값을 선택하는 통합의 수행 횟수에도 제약이 있다. 주어진 제약사항 속에서도 UCPC가 성공적으로 모든 시간선을 하나로 통합할 수 있을지 판단해보자.

입력

첫째 줄에 테스트 케이스의 개수 T가 주어진다. ($1 \le T \le 100\,000$)

각 테스트 케이스의 첫째 줄에는 시간선의 개수를 의미하는 정수 N, 현재 시각을 의미하는 정수 t, 가능한 최솟값 통합의 횟수를 의미하는 정수 a, 가능한 최댓값 통합의 횟수를 의미하는 정수 b가 공백으로 구분되어 주어진다. ($2 \le N \le 500\,000$; $1 \le t \le N$; $0 \le a, b < N$; $a + b \ge N - 1$)

각 테스트 케이스의 둘째 줄에는 N개의 정수 $A_1, A_2, \cdots, A_N$이 공백으로 구분되어 주어진다. A_i는 i번째 시간선의 시각을 의미한다. 각 A_i는 1 이상 N 이하의 정수이며 서로 다르다.

모든 테스트 케이스에 대해 N의 합은 500 000 이하이다.

출력

각 테스트 케이스에 대한 답을 순서대로 출력한다.

조건에 맞는 시간선 통합이 불가능하면, 첫째 줄에 `no`를 출력한다.

가능하다면, 첫째 줄에는 `yes`를 출력하고, 둘째 줄에는 `m`과 `M`으로만 구성된 길이 $N-1$의 문자열 S를 출력하며, 셋째 줄에는 $N-1$ 개의 정수 b_i를 공백으로 구분해 출력한다.

S_i가 `m`이면 i번째 통합은 최솟값 통합이며, `M`이면 최댓값 통합이다. b_i는 i번째 통합에 b_i번째 시간선과 b_i+1번째 시간선을 통합했음을 의미한다. 조건에 부합하는 시간선 통합 방법이 여러 가지 있으면 그중 아무 것이나 출력한다.

i번째 시간선 통합을 할 때는 시간선이 총 $N-i+1$개 남아 있으므로, $1 \le b_i \le N-i$를 충족해야 함에 유의한다.

입출력 예시

표준 입력(stdin)	표준 출력(stdout)
2 4 2 2 3 1 4 2 3 3 3 2 0 3 1 2	yes mMm 2 1 1 no

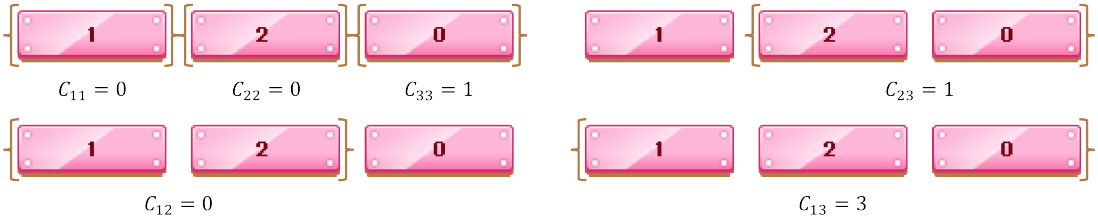
문제 I. 골드리치의 비밀 금고

시간 제한 2 초 메모리 제한 1024 MB

골드리치는 메이플 용사들을 위해 새로운 비밀 금고 이벤트를 열었다. 메이플 용사는 각자 금고에 비밀번호를 입력해 이벤트에 응모하게 된다. 기존의 황금 금고, 다이아 금고 이벤트에서는 응모된 비밀번호 중 중복되지 않는 작은 수를 입력한 용사들에게 상품을 지급했다.

골드리치는 이번 루비 금고 이벤트에서 큰돈을 들여 모든 용사에게 같은 개수의 상품을 지급할 계획이다. 대신 각 용사가 받을 상품 개수는 용사들이 입력하는 비밀번호에 따라 달라진다. 상품 개수를 결정하는 방식은 아래와 같다.

- 각 용사 i는 정수 A_i를 비밀번호로 입력한다. $(0 \le A_i \le 10^9;\ 1 \le i \le N)$
- $[A_l, A_{l+1}, \cdots, A_{r-1}, A_r]$에서 존재하지 않는 가장 작은 음이 아닌 정수를 C_{lr}라고 한다. $(1 \le l \le r \le N)$
- 가능한 C_{lr} 중 존재하지 않는 가장 작은 음이 아닌 정수가 각 용사가 받을 상품 개수가 된다.



$$\{C_{11}, C_{22}, C_{33}, C_{12}, C_{23}, C_{13}\} = \{0, 0, 1, 0, 1, 3\} \rightarrow 2$$

메이플 용사들은 단합하여 최대한 많은 상품을 받으려 했지만, 용사의 수가 너무나 많은 까닭에 의견을 모으지 못한 채로 비밀번호를 모두 입력해 버리고 말았다.

메이플 용사로서 루비 금고 이벤트에 참가한 팬텀 역시 더 많은 상품을 원하고 있었다. 비밀번호가 모두 입력된 것을 본 팬텀은, 메이플 월드 최고의 괴도답게 비밀번호를 조작해서 받게 될 상품 수를 최대한 크게 만드는 작전을 세웠다.

다른 용사가 입력한 비밀번호를 바꾸는 것은 팬텀에게는 쉬운 일이지만, 금방 발각될 위험이 있기 때문에 입력된 비밀번호 $A_1, A_2, \cdots, A_N$의 순서만 바꾸려고 한다.

상품 개수가 최대가 되도록 팬텀이 재배열한 순서를 구해 보자.

입력

첫째 줄에 이벤트에 응모한 메이플 용사의 인원 수 N이 주어진다. $(1 \le N \le 1\,000\,000)$

둘째 줄에 N개의 정수 $A_1, A_2, \cdots, A_N$이 공백으로 구분되어 주어진다. $(0 \le A_i \le 10^9)$

출력

첫째 줄에 만들 수 있는 최대 상품 개수를 출력한다.

둘째 줄에 그때의 재배열된 $A_1, A_2, \cdots, A_N$을 공백으로 구분하여 출력한다. 가능한 답이 여러 개라면 그중 아무 것이나 출력한다.

입출력 예시

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

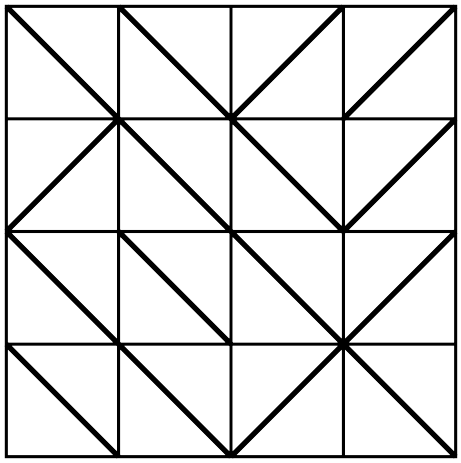
문제 J. 격자 조각 자르기

시간 제한 1초 메모리 제한 1024 MB

격자 아티스트 브루는 $N \times M$ 크기의 격자를 가지고 있다. 격자의 위에서부터 r번째 행, 왼쪽에서부터 c번째 열의 칸을 (r, c)와 같이 표기한다. 브루는 격자를 여러 조각으로 자르려고 한다.

격자를 자를 때는 NM개의 모든 격자칸에 대해 각각 자를 방향을 정해야 한다. 각 칸을 자르는 방향은 두 대각선 방향 중 하나여야 한다.

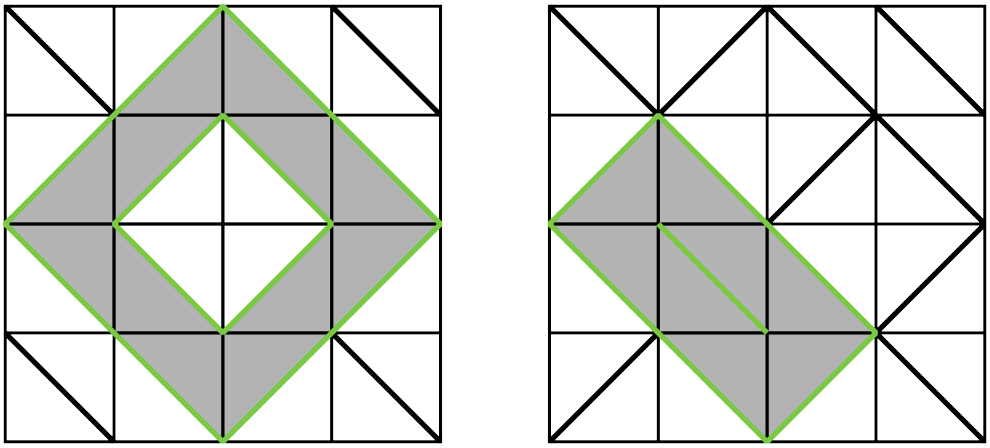
아래 그림은 4×4 크기의 격자판을 자르는 한 가지 예시를 나타낸다.



위의 그림과 같이 자를 경우, 이 격자판은 총 9개의 조각으로 나뉘어진다.

다음 조건을 만족하는 조각을 **아름다운 조각**이라고 하자.

- 조각을 적당히 회전해, 해당 조각의 경계를 구성하는 모든 선분이 x축 또는 y축과 평행하도록 할 수 있어야 한다.
    - 조각을 자르는 방법에 따라, 조각의 내부에 구멍이나 잘린 흔적이 있을 수 있다. 이러한 경우, 해당 부분을 이루는 선분들 역시 모두 x축 또는 y축과 평행하도록 조각을 회전할 수 있어야 한다. 아래 그림에서 왼쪽은 내부에 구멍을, 오른쪽은 내부에 잘린 흔적을 포함하는 아름다운 조각의 예시이다. (해당 조각은 초록색으로 표시되어 있다.)



브루는 격자를 더 아름답게 자르는 방법에 대해 연구하던 중, 격자에 있는 **변**에 관심을 가지기 시작했다. 변의 정의는 아래와 같다.

- 위아래로 인접한 두 격자칸 사이에 존재하는 길이 1의 선분을 **가로변**이라고 한다.
- 좌우로 인접한 두 격자칸 사이에 존재하는 길이 1의 선분을 **세로변**이라고 한다.
- 가로변과 세로변을 통틀어 **변**이라고 한다.

정의에 따라, $N \times M$ 크기의 격자에는 총 $(N-1)M$개의 가로변과 $N(M-1)$개의 세로변이 존재함을 알 수 있다.

브루는 NM개의 칸 중 일부 칸은 자를 방향을 이미 정했지만, 나머지 칸들은 아직 자를 방향을 정하지 않았다. 또한, 브루는 특정한 K개의 변이 아름다운 조각에 속하기를 원한다. (해당 변들이 같은 조각에 속할 필요는 없다.)

위의 조건을 만족하도록 격자를 자르는 방법이 존재하는지 알아내고, 만약 존재한다면 그러한 방법을 하나 찾아보자.

입력

첫째 줄에 세 정수 N, M, K가 공백으로 구분되어 주어진다. ($2 \le N \le 50$; $2 \le M \le 50$; $0 \le K \le (N-1)M + N(M-1)$)

이후 N개의 줄에 걸쳐, 그중 r번째 줄에는 브루가 각 칸을 자를 방향을 나타내는 M개의 문자 C_{r1}, C_{r2}, $\cdots$, C_{rc}가 주어진다. C_{rc}는 `'/'`, `'\'`, `'.'` 중 하나이다.

- C_{rc}가 `'/'` 또는 `'\'`인 경우 격자칸 (r, c)를 자를 방향을 나타낸다.
- C_{rc}가 `'.'`인 경우 격자칸 (r, c)를 자를 방향을 아직 정하지 않았다는 것을 의미한다.

이후 K개의 줄에 걸쳐, 그중 i번째 줄에는 아름다운 조각에 포함되어야 하는 i번째 변을 나타내는 세 정수 d_i, a_i, b_i가 공백으로 구분되어 주어진다. K개의 변은 서로 다르다. ($0 \le d_i \le 1$; $1 \le a_i \le N - (1 - d_i)$; $1 \le b_i \le M - d_i$)

- 만약 i번째 변이 가로변일 경우, $d_i = 0$이고, 해당 가로변의 위쪽에 존재하는 격자칸이 (a_i, b_i)이다.
- 만약 i번째 변이 세로변일 경우, $d_i = 1$이고, 해당 세로변의 왼쪽에 존재하는 격자칸이 (a_i, b_i)이다.

출력

입력으로 주어진 K개의 변이 모두 아름다운 조각에 속하도록 격자를 자르는 방법이 존재한다면, 첫째 줄에 `"YES"`를 출력한다.

다음 N개의 줄에는 격자를 자르는 방법을 출력한다. 각 줄에는 M개의 문자를 출력하며, 각 문자는 `'/'` 또는 `'\'` 중 하나여야 한다.

만약 그러한 방법이 존재하지 않는다면 첫째 줄에 `"NO"`를 출력한다.

입출력 예시

표준 입력(stdin)	표준 출력(stdout)
4 4 2 ..// .\./ .\.. \./\ 0 3 3 1 3 1	YES \\// /\\/ \\\/ \\/\
2 3 2 .\. ... 1 2 1 1 2 2	NO

문제 K. 자동 광고 배치 시스템

시간 제한 5 초 메모리 제한 1024 MB

Moloco는 전 세계의 다양한 회사들이 온라인 광고를 더 잘할 수 있도록 돕는 광고 기술 회사이다. Moloco의 기술을 사용하면 인터넷 사용자들이 더 관련성 높은 광고를 볼 수 있게 되고, 광고주들은 효과적으로 광고를 할 수 있다.

Moloco의 자동 광고 배치 시스템은 대기 중인 광고 요청들을 실시간으로 묶어 처리하여 효율을 최적화한다. 각 광고 요청에는 그 요청을 처리할 때의 비용을 나타내는 양의 정수 값이 정해져 있다. 총 N개의 광고 요청이 순서대로 대기열에 주어졌을 때, 시스템은 다음 과정을 반복하여 모든 요청을 소화한다.

- 현재 대기열의 앞쪽에서 최대 3개의 광고 요청을 확인한 다음, 그 중 2개의 요청을 선택하여 동시에 처리하고 제거한다. 이때 발생하는 비용은 선택된 두 요청 중 작지 않은 값이다.
- 만약 대기열에 단 하나의 요청만 남아 있다면, 그 요청을 단독 처리하고 제거한다. 이때의 비용은 그 요청의 값이 된다.

시스템이 모든 광고 요청을 처리하기 위해 필요한 총 비용의 최솟값을 구하시오.

입력

첫째 줄에 테스트 케이스의 개수 T가 주어진다. ($1 \le T \le 50\,000$)

각 테스트 케이스의 첫째 줄에는 광고 요청의 개수 N이 주어진다. ($1 \le N \le 1\,000\,000$)

각 테스트 케이스의 둘째 줄에는 각 광고 요청의 비용을 의미하는 N개의 정수 $A_1, A_2, \cdots, A_N$이 공백으로 구분되어 주어진다. ($1 \le A_i \le 10^9$)

모든 테스트 케이스에 대해 N의 합은 $1\,000\,000$ 이하이다.

출력

각 테스트 케이스에 대해 모든 광고 요청을 처리하기 위해 필요한 총 비용의 최솟값을 T개의 줄에 걸쳐 순서대로 출력한다.

입출력 예시

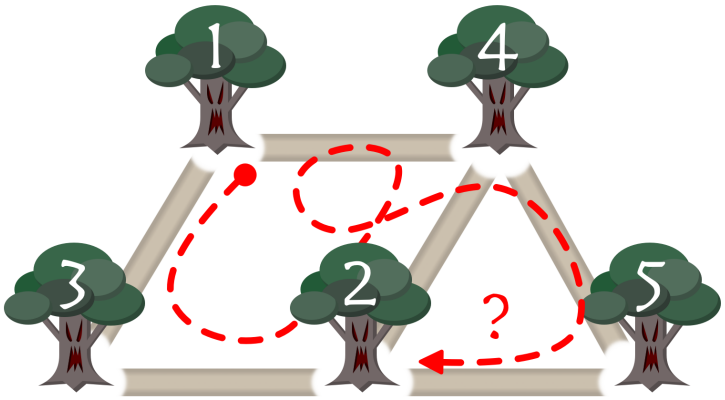
표준 입력(stdin)	표준 출력(stdout)
2 5 1 6 3 2 5 2 314 1592	11 1592

문제 L. 망각의 최장 경로

시간 제한 2 초 메모리 제한 1024 MB

표석이는 영생을 얻을 수 있다는 마녀의 물약을 구하기 위해 저주받은 숲으로 탐험을 떠났다.

저주받은 숲에는 1번부터 N번까지 번호가 붙은 N개의 나무가 있고, 나무와 나무 사이를 양방향으로 잇는 오솔길이 있다. 표석이는 S번 나무 아래에서 출발해서 오솔길을 따라 E번 나무 아래에 있는 마녀의 오두막까지 도달하는 길을 찾고자 한다.



저주받은 숲에는 기억을 잃게 만드는 저주가 걸려 있어, 한 번 발을 들인 탐험가는 평생을 헤매다가 끝내 자신조차 잃어버리게 된다는 전설이 전해진다. 탐험가가 i번 나무 아래에 도달한다면, i보다 작은 번호의 나무에 방문했던 기억은 흔적도 없이 사라진다.

표석이는 길을 잃지 않기 위해 한 가지 원칙을 세웠다.

> "한 번 방문한 나무는 다시 방문하지 않는다."

저주로 인해 방문했던 나무를 기억하지 못하고 다시 방문할 수도 있지만, 방문했던 사실을 기억하는 나무에는 절대 다시 발을 들이지 않을 것이다.

> .....
> 영겁의 시간이 기억 속에서 한순간에 스쳐 간다.
> 표석이는 마녀의 오두막 앞에 선, 낯선 노인을 발견하고 곧 그것이 자신임을 알아챈다.
> 그토록 갈망했던 영생은, 이제 낡은 육신 안에선 축복이 아닌 저주에 지나지 않으리라.
> 그저 묻고 싶은 것은, 얼마나 많은 세월이 덧없이 흘러가 버렸는가 하는 것뿐이다.

표석이는 지금 E번 나무 아래에 서 있다. 지금까지 방문한 나무들 중에서, 표석이가 기억하는 나무들의 번호가 주어진다. 이때, 지금까지 표석이가 오솔길을 따라 이동한 횟수로 가능한 최댓값을 구하여라. 단, 표석이는 S번이나 E번 나무를 여러 번 방문했을 수 있다. (저주로 인해 S번 나무에서 출발했다는 사실을 잊거나, E번 나무에서 마녀의 오두막을 발견하지 못하고 지나칠 수 있다.)

입력

첫째 줄에 정수 N, M, S, E가 공백으로 구분되어 주어진다.

- N은 저주받은 숲의 나무의 수, M은 오솔길의 수를 나타낸다. ($2 \le N \le 100\,000$; $1 \le M \le 300\,000$)

- S는 표석이가 출발하는 나무의 번호, E는 마녀의 오두막이 있는 나무의 번호를 나타낸다. $(1 \le S, E \le N)$

다음 M개의 줄에 오솔길이 잇는 두 나무의 번호 a, b가 공백으로 구분되어 주어진다. $(1 \le a, b \le N;\ a \ne b)$

- 중복된 오솔길은 주어지지 않는다.
- 모든 나무는 하나 이상의 오솔길을 통해 다른 모든 나무와 이어져 있음이 보장된다.

다음 줄에 표석이가 기억하는 방문한 나무의 개수 K가 주어진다. $(1 \le K \le N)$

다음 줄에 표석이가 기억하는 방문한 나무들의 번호를 의미하는 K개의 정수가 공백으로 구분되어 주어진다. 번호는 감소하는 순으로 주어지며, 마지막 번호는 항상 E이다.

출력

표석이가 마녀의 오두막에 도달하기까지 오솔길을 따라 이동한 횟수의 최댓값을 출력한다. 이동 경로는 S번 나무에서 출발하여 E번 나무까지 도달하는 경로여야 하며, 표석이의 기억과 일관되어야 한다.

만약 답이 10^{18} 이상이라면 대신 `eternity`를 출력한다.

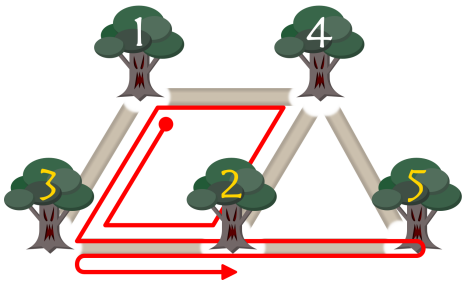
만약 조건에 맞는 경로가 존재하지 않으면 대신 `impossible`을 출력한다.

입출력 예시

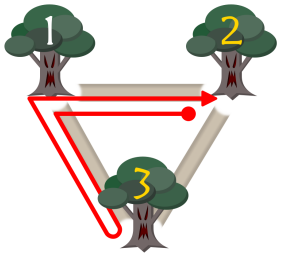
표준 입력(stdin)	표준 출력(stdout)
5 6 1 2 1 3 3 2 2 4 4 1 2 5 4 5 3 5 3 2	10
3 3 2 2 1 3 3 2 2 1 2 3 2	4
4 4 4 1 1 3 3 2 2 4 4 1 4 4 3 2 1	impossible
60 59 1 1 1 2 1 3 1 4 ... 1 58 1 59 1 60 60 60 59 58 ... 3 2 1	eternity

노트

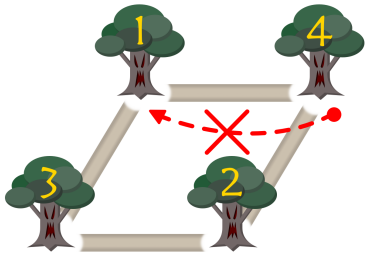
첫 번째 예제에서, 순서대로 $1, 3, 2, 4, 1, 3, 2, 5, 2, 3, 2$번 나무를 방문하는 것이 가장 이동 횟수가 많은 방법이다.



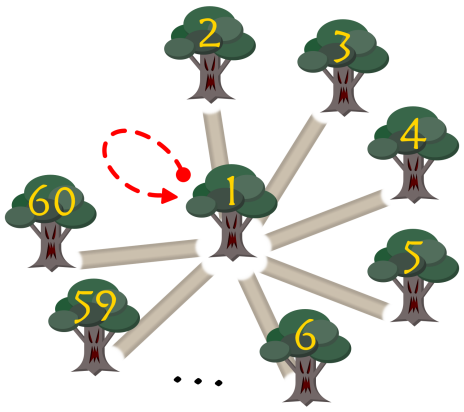
두 번째 예제에서, 순서대로 $2, 1, 3, 1, 2$번 나무를 방문하는 것이 가장 이동 횟수가 많은 방법이다.



세 번째 예제에서, 1번 나무에 도달하려면 반드시 3번이나 4번 나무를 거쳐야 하므로, 2번 나무를 방문한 사실을 기억하는 것이 불가능하다.



네 번째 예제는 각 오솔길이 1번 나무와 $2, 3, 4, \ldots, 60$번 나무를 이으며, 모든 나무를 방문한 기억이 존재하는 예제이다. 가능한 이동 횟수의 최댓값이 10^{18} 이상이므로 `eternity`를 출력한다.



문제 M. Only Shallow

시간 제한 1초 메모리 제한 1024 MB

Jump Trading은 전 세계에 흩어진 서버들로 이루어진 네트워크를 통해 초단타 트레이딩(HFT)을 하고자 한다.

네트워크는 N개의 서버와 M개의 통신 링크로 구성되어 있다. 각 서버는 1번부터 N번까지의 번호로, 각 통신 링크는 1번부터 M번까지의 번호로 구분된다. 네트워크의 서로 다른 두 서버 사이에는 거래 정보를 전송할 수 있는 통신 링크가 최대 하나 존재하며, 특이하게도 통신 링크를 통한 전송은 단방향으로만 이루어질 수 있다. 초기에는 모든 통신 링크의 방향이 정해져 있지 않으며, 방향을 무시할 경우 모든 서버는 하나 이상의 통신 링크를 통해 직간접적으로 연결되어 있다.

초단타 트레이딩에서 네트워크 지연을 최소화하는 것은 중요하다. 이를 위해 Jump Trading은 현재 방향이 설정되어 있지 않은 통신 링크들의 방향을 적절히 설정해 전파 깊이가 얕은 네트워크를 구성하고자 한다. 구성된 네트워크는 다음 조건을 만족해야 한다.

- 모든 서버에 대해, 하나 이상의 통신 링크를 거쳐 도달할 수 있는 다른 서버의 수가 2개 이하이다.

모든 통신 링크들의 방향을 적절히 설정해 조건을 만족하는 네트워크를 만들 수 있는지 판별하고, 가능하다면 구성하여라.

입력

첫째 줄에 테스트 케이스의 수 T가 주어진다. ($1 \le T \le 100$)

각 테스트 케이스의 첫째 줄에는 서버의 수 N과 통신 링크의 수 M이 공백으로 구분되어 주어진다. ($2 \le N \le 1000$; $1 \le M \le 2000$)

이후 M개의 줄에 걸쳐, 그 중 i번째 줄에는 i번째 통신 링크가 연결하는 두 서버의 번호 U_i, V_i가 공백으로 구분되어 주어진다. ($1 \le U_i, V_i \le N$; $U_i \neq V_i$)

서로 다른 두 서버를 직접 연결하는 통신 링크는 최대 하나이며, 방향을 무시할 경우 모든 서버는 하나 이상의 통신 링크를 통해 직간접적으로 연결되어 있다.

출력

각 테스트 케이스에 대한 답을 T개의 줄에 걸쳐 순서대로 출력한다.

각 테스트 케이스에 대해, 만약 문제의 조건을 만족하는 네트워크를 구성할 수 없다면 -1을 출력한다. 문제의 조건을 만족하는 네트워크를 구성할 수 있다면 M개의 수를 공백으로 구분해 출력한다. i번째 수는 i번째 통신 링크의 방향이 정방향($U_i \to V_i$) 이라면 0, 역방향($V_i \to U_i$) 이라면 1이다. 문제의 조건을 만족하는 네트워크가 여러 가지라면 그중 아무것이나 출력한다.

입출력 예시

표준 입력(stdin)	표준 출력(stdout)
3 3 3 1 2 2 3 3 1 4 4 1 2 2 3 3 4 4 1 5 5 1 2 2 3 3 4 4 5 5 1	1 1 1 0 1 0 1 -1
1 4 5 1 2 1 3 1 4 2 4 3 4	1 1 0 0 0
1 12 12 1 2 2 3 3 1 1 4 1 5 2 6 2 7 1 8 8 9 9 10 10 11 10 12	1 1 0 1 1 1 1 1 0 1 1 1