문제 1. 장거리 버스

입력 파일:	`standard input`
출력 파일:	`standard output`
시간 제한:	2 seconds
메모리 제한:	256 megabytes

IOI나라에는 도시 I와 도시 O를 잇는 장거리버스가 달리고 있다. 버스의 내부에는 정수기가 있다. 탑승객은 정수기에서 물을 받아 마실 수 있다. 버스는 시간 0에 도시 I로 부터 출발하여, 시간 X에 도시 O로 도착한다. 버스의 경로 상에, 정수기에 물을 보충할 수 있는 장소가 N개 있다. 버스는 i 번째 $(1 \leq i \leq N)$ 장소에 시간 S_i에 도착 할 것이다.

처음에 정수기에는 물이 들어있지 않다. 버스가 출발하기 전에 정수기에 물을 채울 수 있다. 또한 버스가 물을 보충할 수 있는 장소에 도착 했을 때에 물을 채울 수 있다. 물은 버스가 어디있든 1리터를 채우는 데에 W엔이 든다.

도시 I에서 M명의 승객이 버스에 탑승한다. 승객들은 1번부터 M번까지의 번호가 붙어있다. 도시 I를 제외하고는 승객이 버스에 타지는 않는다. j번 $(1 \leq j \leq M)$ 승객은 시간 D_j에 1리터의 물을 마시고 싶어 한다. 또한, 물을 마시면 T초 후에 1리터의 물을 마시고 싶어 한다. 다른 말로, j번 승객은 $D_j + kT$ $(k = 0, 1, 2, \cdots)$ 시간에 물을 마시고 싶어 한다. 여기서, $1 \leq D_j < T$를 만족하며, 모든 승객에 대해 T는 모두 같은 값이다. 만약 승객이 물을 마시기를 원할 때 정수기에 물이 담겨있지 않다면, 승객은 버스에서 내리게 된다. 만약 j번 손님이 도시 O에 도착하기 전에 버스에서 내리게 된다면, 승객에게 C_j엔을 환불 해 줘야 한다.

버스기사도 역시 물을 마셔야 한다. 만약 물을 마시면, 그는 T초 후에 1리터의 물을 마시고 싶어 한다. 다른 말로, 버스기사는 시간 kT $(k = 0, 1, 2, \cdots)$ 시간에 물을 마시고 싶어 한다. 만약 버스기사가 물을 마시기를 원할 때 정수기에 물이 담겨있지 않다면, 버스는 더 이상 운행할 수 없다.

어떠한 두 사람도 물을 마셔야 하는 시간이 같지 않다. 또한, 버스가 도시 O나 물을 보충할 수 있는 장소에 도달 했을 때 물을 마셔야 하는 승객 혹은 버스기사가 있는 경우는 없다.

물을 보충할 수 있는 각 장소에서 물을 담는 양을 조절해서 도시 O까지 가는 도중에 물의 비용과 환불을 해 주는데 드는 비용의 합을 최소로 하고 싶다. 당신은 여행하는 도중 어디서 얼마나 물을 담아야 하는지를 결정해야 한다.

입력 형식

다음 정보가 표준 입력으로 주어진다.

- 첫째 줄에는 공백으로 구분된 다섯 정수 X, N, M, W, T가 주어진다. 이는 도시 O에 버스가 시간 X에 도착하며, 물을 보충할 수 있는 장소가 N개 있으며, M명의 손님이 버스에 타고 있으며, 물의 가격이 1리터당 W엔이며, 승객과 버스기사가 물을 마시는 간격이 T라는 의미이다.
- 다음 N개의 줄의 i 번째 $(1 \leq i \leq N)$ 줄에는 정수 S_i가 주어진다. 이는 버스가 i 번째 물을 보충할 수 있는 장소에 시간 S_i에 도착한다는 의미이다.
- 다음 M개의 줄의 j 번째 $(1 \leq j \leq M)$ 줄에는 공백으로 구분된 두 정수 D_j, C_j가 주어진다. 이는 j번 승객이 시간 D_j에 처음 물을 마시기를 원하며, 환불을 할 때 C_j엔을 환불 해 주어야 한다는 의미이다.

출력 형식

표준 출력으로 한 개의 줄을 출력하여라. 출력은 최소 비용이다.

제한

- $1 \leq X \leq 1\ 000\ 000\ 000\ 000$.
- $1 \leq N \leq 200\ 000$.

- $1 \leq M \leq 200\ 000$.
- $1 \leq W \leq 1\ 000\ 000$.
- $1 \leq T \leq X$.
- $1 \leq S_i < X$ $(1 \leq i \leq N)$.
- $1 \leq D_j < T$ $(1 \leq j \leq M)$.
- $1 \leq C_j \leq 1\ 000\ 000\ 000$ $(1 \leq j \leq M)$.
- D_j $(1 \leq j \leq M)$ 는 서로 다르다.
- 버스가 도시 O나 물을 보충할 수 있는 장소에 도달 했을 때 물을 마셔야 하는 승객 혹은 버스기사가 있는 경우는 없다.

서브태스크 1 (16 점)

- $N \leq 8$
- $M \leq 8$

서브태스크 2 (30 점)

- $N \leq 100$
- $M \leq 100$

서브태스크 3 (25 점)

- $N \leq 2\ 000$
- $M \leq 2\ 000$

서브태스크 4 (29 점)

추가 제한조건이 없다.

예제

standard input	standard output
19 1 4 8 7 10 1 20 2 10 4 5 6 5	103

이 예제에서, 우리가 출발 전에 7리터의 물을 넣고, 물을 보충할 수 있는 장소에서 4리터의 물을 넣은 경우, 버스는 다음과 같이 운행된다:

1. 버스가 도시 I를 떠난다. 이 때, 정수기는 7리터의 물이 담겨있다.
2. 버스기사와 1, 2, 3, 4번 승객이 각각 시간 0, 1, 2, 4, 6에 물을 마신다. 정수기에 남은 물의 양은 2 리터이다.

3. 버스기사와 1번 승객이 각각 1리터의 물을 시간 7, 8에 마신다. 정수기에 남은 물의 양은 0리터이다.
4. 시간 9에, 2번 승객은 물을 마시고 싶어 하지만 정수기에 물이 없기 때문에 버스를 떠난다.
5. 시간 10에, 물을 보충할 수 있는 장소에서 4리터의 물을 정수기에 보충한다. 이 때, 정수기에는 4리터의 물이 담겨있다.
6. 3, 4번 승객, 버스기사, 1번 승객이 각각 시간 11, 13, 14, 15에 물을 마신다. 정수기에 남은 물의 양은 0리터이다.
7. 시간 18에, 3번 승객은 물을 마시고 싶어 하지만 정수기에 물이 없기 때문에 버스를 떠난다.
8. 시간 19에, 버스는 도시 O에 도착한다.

사용한 물의 총량은 11리터이다. 물값은 88엔이다. 2, 3번 승객을 환불 해 줄 때 사용한 돈의 양은 총 15엔이다. 총 사용한 돈의 양은 103엔이다.

103엔 보다 더 작은 양을 사용하여 버스를 운영하는 것은 불가능 하므로, 103엔을 출력한다.

standard input	standard output
105 3 5 9 10 59 68 71 4 71 6 32 7 29 3 62 2 35	547
1000000000000 1 1 1000000 6 999999259244 1 123456789	333333209997456789

문제 2. 긴 저택

입력 파일:	`standard input`
출력 파일:	`standard output`
시간 제한:	3 seconds
메모리 제한:	256 megabytes

JOI군의 집 근처에는 긴 저택이 있다. 이 저택은 동에서 서로 일렬로 N개의 방이 있고, 가장 동쪽 방부터 i 번째 방을 i번 방이라고 부르며, i번 ($1 \leq i \leq N-1$) 방과 $i+1$번 방을 잇는 통로가 있어서 양방향으로 오갈 수 있다. 방에서 통로로 들어가기 위해서는 열쇠가 필요하다. 각 열쇠에는 종류를 나타내기 위한 수가 하나씩 붙어 있다. 여러 열쇠에 같은 수가 붙어 있을 수도 있다.

i번 방 혹은 $i+1$번 방으로 부터 두 방을 잇는 통로로 들어가기 위해서는 수 C_i가 붙어 있는 열쇠가 필요하다.

i번 방에는 B_i개의 열쇠가 있다. 이 열쇠에는 각각 수 $A_{i,j}$ ($1 \leq j \leq B_i$) 가 붙어있다. 만약 JOI군이 방에 들어가면, 그는 방에 있는 모든 열쇠를 집을 것이다. 그 이후에는 집은 열쇠들을 통로를 출입하는데 사용할 수 있다.

JOI군은 열쇠를 원하는 횟수 만큼 사용할 수 있다. 가끔 JOI군은 같은 수가 붙어 있는 열쇠를 여러개 가지고 있는 경우도 있다. 하지만, 그 열쇠가 하나 있는 것과 특별히 다른 점은 없다.

JOI군이 길을 잃는 경우를 방지하기 위해서, 다음 종류의 질문에 대답하는 프로그램을 작성하려고 한다.

- 만약 JOI군이 어떠한 열쇠도 가지지 않고 x번 방에서 시작하여, y번 방으로 이동할 수 있을까?

JOI군을 대신하여 이 문제를 답할 수 있는 프로그램을 작성하여 주자.

입력 형식

다음 정보가 표준 입력으로 주어진다.

- 첫째 줄에는 정수 N이 주어진다. 이는 저택에 있는 방의 갯수이다.
- 둘째 줄에는 $N-1$개의 공백으로 구분된 정수 C_1, C_2, $\cdots$, C_{N-1}이 주어진다. 이는 우리가 i번 방과 $i+1$번 방을 잇는 통로를 오가기 위해서 수 C_i가 붙은 열쇠가 필요하다는 의미이다.
- 다음 N개의 줄의 i 번째 ($1 \leq i \leq N$) 줄에는 정수 B_i와, B_i개의 공백으로 구분된 정수 $A_{i,1}$, $A_{i,2}$, $\cdots$, A_{i,B_i}가 주어진다. 이는 i번 방에 B_i개의 열쇠가 있으며, 열쇠에 각각 수 $A_{i,j}$ ($1 \leq j \leq B_i$) 가 붙어있다는 의미이다.
- 다음 줄에는, 질문의 갯수 Q가 주어진다.
- 다음 Q개의 줄의 k 번째 ($1 \leq k \leq Q$) 줄에는 공백으로 구분된 두 정수 X_k, Y_k가 주어진다. 이는 k 번째 질문이 JOI군이 어떠한 열쇠도 가지지 않고 방 X_k에서 시작하여, 방 Y_k로 이동할 수 있는지를 묻는다는 의미이다.

출력 형식

표준 출력으로 Q개의 줄을 출력하여라. Q개의 줄의 k 번째 ($1 \leq k \leq Q$) 줄은 JOI군이 어떠한 열쇠도 가지지 않고 X_k번 방에서 시작하여, Y_k번 방으로 이동할 수 있으면 `YES`, 아니면 `NO`여야 한다.

제한

- $2 \leq N \leq 500\ 000$.
- $1 \leq Q \leq 500\ 000$.
- $1 \leq B_1 + B_2 + \cdots + B_N \leq 500\ 000$.

- $1 \leq B_i \leq N$ $(1 \leq i \leq N)$.
- $1 \leq C_i \leq N$ $(1 \leq i \leq N-1)$.
- $1 \leq A_{i,j} \leq N$ $(1 \leq i \leq N, 1 \leq j \leq B_i)$.
- B_i개의 정수 $A_{i,1}, A_{i,2}, \cdots, A_{i,B_i}$는 서로 다르다.
- $1 \leq X_k \leq N$ $(1 \leq k \leq Q)$.
- $1 \leq Y_k \leq N$ $(1 \leq k \leq Q)$.
- $X_k \neq Y_k$ $(1 \leq k \leq Q)$.

서브태스크 1 (5 점)

- $N \leq 5\ 000$
- $Q \leq 5\ 000$
- $1 \leq B_1 + B_2 + \cdots + B_N \leq 5\ 000$.

서브태스크 2 (5 점)

- $N \leq 5\ 000$
- $1 \leq B_1 + B_2 + \cdots + B_N \leq 5\ 000$.

서브태스크 3 (15 점)

- $N \leq 100\ 000$
- $C_i \leq 20$ $(1 \leq i \leq N-1)$.
- $1 \leq A_{i,j} \leq 20$ $(1 \leq i \leq N, 1 \leq j \leq B_i)$.

서브태스크 4 (75 점)

추가 제한조건이 없다.

예제

standard input	standard output
5 1 2 3 4 2 2 3 1 1 1 1 1 3 1 4 4 2 4 4 2 1 5 5 3	YES NO NO YES

- 첫째 질문에서, JOI군이 방을 2, 1, 2, 3, 4번 방 순서로 방문한다면, 4번 방에 도착할 수 있다.
- 둘째 질문에서, JOI군은 3, 4번 방 밖에 방문할 수 없다. 1과 3이 붙어 있는 열쇠 밖에 얻을 수 없으므로, 2번 방에 들어가지 못한다.
- 셋째 질문에서, JOI군은 4번 방에서 5번 방으로 가기 위한 종류 4의 열쇠를 얻을 수가 없기 때문에, 5번 방에 들어가지 못한다.
- 넷째 질문에서, JOI군이 방을 5, 4, 3번 방 순서로 방문한다면, 4번 방에 도착할 수 있다.

standard input	standard output
5 2 3 1 3 1 3 1 2 1 1 1 3 1 2 4 1 3 3 1 4 3 2 5	NO YES NO YES
7 6 3 4 1 2 5 1 1 1 5 1 1 1 1 2 2 3 1 4 1 6 3 4 1 5 3 4 7	YES NO YES

문제 3. 자연공원

시간 제한: 2 seconds
메모리 제한: 256 megabytes

JOI섬은 섬 전체가 자연공원으로 지정된 관광지이다.

JOI섬에는 N개의 광장과 몇개의 도로가 있다. 광장에는 0번부터 $N-1$번까지 번호가 붙어있다. 도로는 섬 내에 서로 다른 2개의 광장을 잇고, 양방향으로 이동가능하다. 어떤 광장에 대해서도 이 광장에 직접 연결된 도로는 최대 7개이다. 서로 다른 두 광장에 대해서, 두 광장를 잇는 도로는 최대 한 개 존재한다. 몇 개의 도로를 거치면, 한 광장에서 부터 다른 광장으로 갈 수 있다.

당신과 당신의 친구 IOI양은 JOI섬을 탐험할 것이다. 효율적인 탐사를 위해, JOI섬의 구조를 파악 할 필요가 있다. 섬에는 위험한 야생동물이 많아 위험하므로, 운동신경이 좋은 IOI양이 도시를 탐색하고 IOI양의 보고를 받아 당신이 섬의 구조를 파악하게 되었다.

당신은 IOI양에게 두 광장의 번호 A, B와 경유가능한 광장을 몇개 지정하여, 광장 A부터 광장 B까지 지정된 광장만 경유하여 이동하는 것이 가능한가 라는 질문을 한다. IOI양은 질문의 내용을 따라 섬을 탐색하여 결과를 보고한다.

조사에 긴 시간이 사용되면 위험하기 때문에, 질문 횟수는 45 000번 이내여야 한다.

구현 명세

당신은 섬의 구조를 파악하는 방법이 구현된 하나의 파일을 작성해야 한다. 이 파일은 `park.h`를 include해야 한다.

이 파일에는, 다음 함수가 작성되어 있어야 한다.

- `void Detect(int T, int N)`

  이 함수는 한 번만 불린다.

  - 인자 `T`는 서브태스크의 번호, `N`은 광장의 갯수를 의미한다.

  프로그램 안에서 다음의 함수를 호출하여, JOI섬의 구조를 출력해야 한다.

  - `void Answer(int A, int B)`
    이 함수의 호출 횟수는, JOI섬의 도로의 갯수와 같아야 한다.
    * 인자 `A`, `B`는 `A`번 광장과 `B`번 광장 사이에 도로가 있다는 것을 의미한다.

    함수 인자는 다음과 같은 조건을 만족해야 한다:
    * `A`, `B`는 $0 \leq$`A`$<$`B`$\leq N-1$을 만족해야 한다. 이 조건을 만족하지 않으면, **오답 [1]**이 된다.
    * 함수가 인자 (`A`, `B`)로 호출되었으면, `A`번 광장과 `B`번 광장을 잇는 도로가 있어야 한다. 이 조건을 만족하지 않으면, **오답 [2]**이 된다.
    * 이 함수는 같은 인자 (`A`, `B`)로 두 번 이상 호출되면 안된다. 이 조건을 만족하지 않으면, **오답 [3]**이 된다.

  또한, 당신의 프로그램은 다음 함수를 호출 할 수 있다:

- `int Ask(int A, int B, int Place[])` 이 함수는 IOI양에게 질문을 하는 데에 쓰인다.

  - `Place`는 경유할 수 있는 광장들의 배열을 가리키는 포인터이다. 각 i $(0 \leq i \leq N-1)$에 대해, `Place`$[i]=1$이면, i번 광장을 경유할 수 있다는 의미이고, `Place`$[i]=0$이면, i번 광장을 경유할 수 없다는 의미이다.
  - 이 함수의 반환값은 `A`번 광장에서 `B`번 광장으로 배열 `Place[]` 에 주어진 광장 만으로 이동할 수 있으면 1이고, 아니면 0이다.

  함수 인자는 다음과 같은 조건을 만족해야 한다:

- $0 \leq$ A $<$ B $\leq N-1$
- $0 \leq$ Place[i] ≤ 1 $(0 \leq i \leq N-1)$
- Place[A] $= 1$
- Place[B] $= 1$

위 조건들을 만족하지 않으면, **오답** [4]이 된다. 하지만, 배열 Place[]의 길이가 N이 아닌 경우에 이 함수의 동작은 보장되지 않는다.

함수 Ask는 45 000번 초과로 호출되어서는 안된다. 만약 초과하는 경우, **오답** [5]이 된다.

함수가 종료되었을 때, Answer의 호출로 출력되지 않은 도로가 있는 경우에는 **오답** [6]이 된다.

당신의 프로그램은 내부에서 사용할 목적으로 함수나 전역변수를 사용할 수 있다. 당신의 프로그램은 표준 입출력을 사용해서는 안된다. 당신의 프로그램은 어떠한 방법으로도 다른 파일에 접근해서는 안된다.

당신은 대회 홈페이지의 아카이브에서 프로그램을 테스트 하기 위한 목적의 샘플 그레이더를 받을 수 있다. 아카이브는 당신의 프로그램의 예제 소스 또한 첨부되어 있다. 샘플 그레이더는 파일 grader.c 혹은 grader.cpp이다. 당신의 프로그램이 park.c 혹은, park.cpp 인 경우 다음 커맨드로 컴파일할 수 있다.

- C `g++ -std=c11 -O2 -o grader grader.c park.c -lm`
- C++ `g++ -std=c++14 -O2 -o grader grader.cpp park.cpp`

컴파일이 성공적이면, 파일 grader가 생성된다.

실제 그레이더와 샘플 그레이더는 다름에 주의하여라. 샘플 그레이더는 하나의 프로세스에서 실행 되며, 입력을 표준 입력으로 부터 받고, 출력을 표준 출력에 출력한다.

입력 형식

샘플 그레이더는 다음 형식으로 표준 입력으로 부터 데이터를 입력받는다.

- 첫 번째 줄에 서브태스크의 번호 T가 주어진다.
- 두 번째 줄에 광장의 갯수 N이 주어진다.
- 세 번째 줄에 도로의 갯수 M이 주어진다.
- 다음 M개의 줄의 i 번째 $(1 \leq i \leq M)$ 줄 에는 공백으로 구분된 두 정수 A_i, B_i가 주어진다. 이는, 광장 A_i와 광장 B_i를 잇는 도로가 있어서, 양방향으로 오갈 수 있다는 의미이다.

출력 형식

프로그램이 정상적으로 종료되었다면, 샘플 그레이더는 다음과 같은 정보를 표준 출력에 출력한다. (따옴표는 출력하지 않는다.)

- 프로그램이 정답으로 판단 된 경우에는, "`Accepted.`"를 출력한다.
- 오답으로 판단 된 경우, 오답의 종류를 "`Wrong Answer [1]`"과 같은 형식으로 출력하고, 프로그램이 종료된다.

프로그램이 다양한 오답의 종류에 속해 있을 경우, 샘플 그레이더는 그 중 하나만 출력 할 것이다.

제한

T, N, M의 의미에 대해서는 입력 형식을 참고하여라.

- $1 \leq T \leq 5$.

- $2 \leq N \leq 1\ 400$.
- $1 \leq M \leq 1\ 500$.
- 어떤 광장에 대해서도 이 광장에 직접 연결된 도로는 최대 7개이다.
- 몇 개의 도로를 거치면, 한 광장에서 부터 다른 광장으로 갈 수 있다.
- 서로 다른 두 광장에 대해서, 두 광장를 잇는 도로는 최대 한 개 존재한다.

서브태스크 1 (10 점)

- $T = 1$.
- $N \leq 250$.

서브태스크 2 (10 점)

- $T = 2$.
- $M = N - 1$.
- 0번과 $N - 1$번 광장에 대해, 다른 광장으로 가는 도로가 정확히 한 개 존재한다. 다른 모든 장소에 대해서는, 다른 광장으로 가는 도로가 정확히 두 개 존재한다.

서브태스크 3 (27 점)

- $T = 3$.
- $M = N - 1$.
- 모든 i $(0 \leq i \leq N - 1)$에 대해, 0번 광장으로 부터 i번 광장으로 까지 최대 8개의 다른 광장을 거치면 오갈 수 있다.

서브태스크 4 (30 점)

- $T = 4$.
- $M = N - 1$.

서브태스크 5 (23 점)

- $T = 5$.

예제

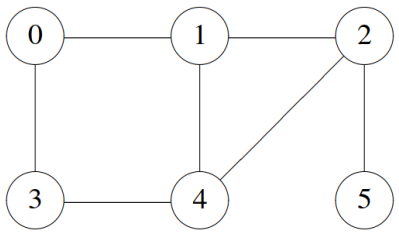
예제 입력과 이에 해당하는 함수 호출을 보여준다.

예제 입력	예제 함수 호출	
	호출	반환값
1 6 7 0 1 0 3 1 2 1 4 2 4 2 5 3 4	`Ask(3, 5, {0,0,1,1,1,1})`	1
	`Answer(2, 4)`	
	`Answer(3, 5)`	
	`Answer(3, 4)`	
	`Ask(0, 4, {1,0,1,0,1,0})`	0
	`Answer(0, 1)`	
	`Answer(0, 3)`	
	`Answer(1, 4)`	
	`Answer(1, 2)`	

이 함수 호출이 의미가 없을 수 있다.

이 함수에서, 함수 `Detect`는 인자 `T`= 1, `N`= 6으로 호출되었다.

이 예제에서, JOI섬의 구조는 다음과 같다.



JOI섬의 구조

원과 숫자는 광장과 그 번호를 의미한다. 선분은 도로를 의미한다.

- 첫 번째 `Ask` 함수 호출은 3번 광장부터 5번 광장까지 2, 3, 4, 5번 광장만 경유하여 갈 수 있는지를 물어보았다. 가능 하기 때문에, 함수 `Ask`는 1을 반환한다.
- 두 번째 `Ask` 함수 호출은 0번 광장부터 4번 광장까지 0, 2, 4번 광장만 경유하여 갈 수 있는지를 물어보았다. 불가능 하기 때문에, 함수 `Ask`는 0을 반환한다.