길고양이

Anthony는 JOI 도시에 사는 개미이다. JOI 도시에는 0번 부터 $N-1$번까지 번호가 붙어있는 N개의 마을이 있다. Anthony는 0번 마을에 살고 있다. 또한, 0번 부터 $M-1$번까지 번호가 붙어 있는 M개의 도로가 있다. i번 $(0 \leq i \leq M-1)$ 도로는 마을 U_i와 V_i를 양방향으로 잇고 있다. 여러 개의 도로가 같은 쌍의 마을을 연결하는 경우는 없다. 어떤 마을에서도 다른 모든 마을까지 몇 개의 도로를 거치면 이동하는 것이 가능하다.

Anthony의 친구인 고양이 Catherine이 JOI 도시에 놀러 올 예정이다. Catherine은 도로의 정보를 모르기 때문에 길을 잃는 일이 잦았다. Anthony는 도로에 표식을 표시하기로 생각했다. 표식에는 0번부터 $A-1$번까지 번호가 붙어 있는 A 종류의 표식이 있다.

이제, Catherine은 JOI 도시의 한 마을에 도착했다. Catherine은 0번 마을을 제외한 마을에 있을 경우,

> (존재한다면) 직전에 지났던 도로를 제외하고, 지금 있는 마을에 인접한 도로 중 어떤 표식이 몇 개 있는가

를 셀 수 있고, 다음의 어떤 도로로 가는지 고르는 것이 가능하다. **직전에 지났던 도로를 제외하고는, 도로를 표식의 종류로만 구분할 수 있다.** 도로를 올바르게 고르는 것으로, 0번 마을까지 시간을 너무 많이 소비하지 않고 가고 싶다. 구체적으로는, 0번 마을까지 가는데 사용하는 도로의 최소 개수가 d 개인 경우, 0번 마을까지 최대 $d+B$ 개만 사용하여 도착하고 싶다.

도로의 정보가 주어졌을 때, 도로에 표식을 표시하는 Anthony의 전략을 구현한 프로그램과 도로의 정보를 받아서 길을 걷는 Catherine의 전략을 구현한 프로그램을 작성하여라.

구현 명세

당신은 파일 두 개를 제출해야 한다.

첫째 파일의 이름은 `Anthony.cpp`이다. 이 파일은 Anthony의 일을 나타내고, 다음 함수를 구현해야 한다. 또한, `Anthony.h`를 include해야 한다.

- `std::vector<int> Mark(int N, int M, int A, int B, std::vector<int> U, std::vector<int> V)`

  이 함수는 최초에 정확히 한 번 불린다.

  - 인자 `N`은 마을의 수 N을 나타낸다.
  - 인자 `M`은 도로의 수 A를 나타낸다.
  - 인자 `A`는 표식의 가짓수 A를 나타낸다.
  - 인자 `B`는 길을 걷는 횟수의 여유 B를 나타낸다.
  - 인자 `U`, `V`는 길이 M의 배열이다. `U[i]`와 `V[i]`는 i번 도로가 연결하는 두 마을의 번호 U_i, V_i를 의미한다. $(0 \leq i \leq M-1)$
  - 반환값 `x`는 길이 M의 배열이어야 한다. 길이가 M이 아닐 경우 **오답 [1]**이 된다. `x[`i`]`는 $(0 \leq i \leq M-1)$ 도로 i에 붙어있는 표식의 번호를 의미한다. $0 \leq$ `x[`i`]` $\leq A-1$을 만족해야 한다. $0 \leq$ `x[`i`]` $\leq A-1$이 아닐 경우 **오답 [2]**이 된다.

둘째 파일의 이름은 `Catherine.cpp`이다. 이 파일은 Catherine의 일을 나타내고, 다음 함수를 구현해야 한다. 또한, `Catherine.h`를 include해야 한다.

- `void Init(int A, int B)`

  이 함수는 최초에 정확히 한 번 불린다.

  - 인자 `A`는 표식의 가짓수 A를 나타낸다.

- 인자 B는 길을 걷는 횟수의 여유 B를 나타낸다.

- `int Move(std::vector<int> y)`

  이 함수는 Catherine이 0번 마을 이외의 마을을 방문할 때마다 불린다.

  - 인자 y는 배열이고, (존재한다면) 직전에 지났던 도로를 제외하고, 지금 있는 마을에 인접한 도로 중 j 번째 ($0 \leq j \leq A-1$) 표식이 y[j]개 있다는 것을 의미한다.
  - 반환값 z는 $-1 \leq$ z $\leq A-1$을 만족해야 한다. $-1 \leq$ z $\leq A-1$을 만족하지 않을 경우 **오답 [3]**이 된다. z$= -1$인 경우 직전에 왔던 도로로 돌아간다는 것을 의미하고, $0 \leq$ z $\leq A-1$인 경우 직전에 지났던 도로 이외의 z번 표식이 붙어있는 도로를 고른다는 것을 의미한다. 함수 Move가 처음으로 호출 될 때, z$= -1$인 경우 **오답 [4]**이 된다. $0 \leq$ z $\leq A-1$ 이고 y[z]$= 0$ 인 경우 **오답 [5]**이 된다.

  Catherine은 어떤 마을에서 직전에 지났던 도로 이외의 길을 고르는 경우 도로는 Move의 반환값에 해당하는 표식이 있는 도로 중 하나를 따라간다. **이 선택이 무작위가 아닐 수 있음에 유의하여라.**

  Catherine이 0번 마을에 $d+B$개의 도로를 사용한 이후 (즉, 함수 Move가 $d+B$ 번 호출 된 이후)에 도착하지 못할 경우, **오답 [6]**이 된다.

참고 사항

- 당신의 프로그램은 내부에서 사용할 목적으로 함수나 전역변수를 사용할 수 있다. 제출된 파일들은 같이 컴파일되어 하나의 실행 파일이 된다. 모든 글로벌 변수나 함수는 충돌을 피하기 위해 이름이 없는 namespace에 구현되어야 한다. 채점 될 때는, Anthony와 Catherine에 해당하는 두 프로세스로 나누어서 실행될 것이다. Anthony의 프로세스와 Catherine의 프로세스는 전역변수를 공유할 수 없다.
- 당신의 프로그램은 표준 입출력을 사용해서는 안 된다. 당신의 프로그램은 어떠한 방법으로도 다른 파일에 접근해서는 안 된다. 단, 당신의 프로그램은 디버그 목적으로 표준 에러출력에 출력할 수 있다.

당신은 대회 홈페이지의 아카이브에서 프로그램을 테스트하기 위한 목적의 샘플 그레이더를 받을 수 있다. 아카이브는 당신의 프로그램의 예제 소스 또한 첨부되어 있다. 샘플 그레이더는 파일 grader.cpp 이다. 당신의 프로그램을 테스트하기 위해서, grader.cpp, Anthony.cpp, Catherine.cpp, Anthony.h, Catherine.h를 같은 디렉토리 안에 놓고, 컴파일하기 위해 다음 커맨드를 실행하여라.

- `g++ -std=gnu++14 -O2 -o grader grader.cpp Anthony.cpp Catherine.cpp`

컴파일이 성공적이면, 파일 grader가 생성된다.

실제 그레이더와 샘플 그레이더는 다름에 주의하여라. 샘플 그레이더는 하나의 프로세스에서 실행되며, 입력을 표준 입력으로부터 받고, 출력을 표준 출력에 출력한다.

입력 형식

샘플 그레이더는 표준 입력에서 다음과 같은 형식으로 입력받는다.

$N\ M\ A\ B\ S$

$U_0\ V_0$

$\vdots$

$U_{M-1}\ V_{M-1}$

여기서 S는 Catherine이 처음으로 도착하는 마을의 번호이다.

출력 형식

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

- 오답 [1], [2], [3], [4] 혹은 [5]로 판단 된 경우, 오답의 종류를 "`Wrong Answer [1]`"과 같은 형식으로 출력한다.
- Catherine이 0번 마을에 $N + B$번 이동 이내에 도착하지 못한 경우, "`Wrong Answer; Number of moves > N+B`"를 출력한다.
- 아닌 경우, 이동 횟수(`Move`의 호출 횟수)를 "`Number of moves = 4`"와 같은 형식으로 출력한다. 샘플 그레이더는 오답 [6] 인지 아닌지를 검사하지 않는다.

프로그램이 다양한 오답의 종류에 속해 있으면, 샘플 그레이더는 그중 하나만 출력할 것이다.

샘플 그레이더에서, Catherine이 자기가 직전에 지났던 도로 이외의 도로를 선택 할 때, 선택한 도로는 `Move`의 반환값에 해당하는 표식이 있는 도로 중 하나가 동일한 확률로 선택되며, 이는 주어진 시드에 따른 유사난수생성기에 의해 결정된다. 다른 시드로 프로그램을 실행하고 싶은 경우, 그레이더를 다음과 같은 형식으로 실행하여라:

```
./grader 2020
```

여기서, 첫 번째 인자는 유사난수생성기의 시드로 사용되는 정수이다.

제한

- $2 \leq N \leq 20\ 000$.
- $1 \leq M \leq 20\ 000$.
- $1 \leq S \leq N - 1$.
- $0 \leq U_i < V_i \leq N - 1$ $(0 \leq i \leq M - 1)$.
- $(U_i, V_i) \neq (U_j, V_j)$ $(0 \leq i < j \leq M - 1)$.
- 어떤 마을에서도 다른 모든 마을까지 몇 개의 도로를 거치면 이동하는 것이 가능하다.

서브태스크 1 (2 점)

- $A = 4$.
- $B = 0$.
- $M = N - 1$.

서브태스크 2 (2 점)

- $A = 4$.
- $B = 0$.

서브태스크 3 (2 점)

- $A = 3$.
- $B = 0$.
- $M = N - 1$.

서브태스크 4 (9 점)

- $A = 3$.
- $B = 0$.

서브태스크 5 (5 점)

- $A = 2$.
- $B = 2N$.
- $M = N - 1$.
- $6 \leq N \leq 500$.

서브태스크 6 (71 점)

- $A = 2$.
- $B = 12$.
- $M = N - 1$.

서브태스크 5 (9 점)

- $A = 2$.
- $B = 6$.
- $M = N - 1$.

예제

이 함수는 그레이더의 예제 입력과 해당하는 함수 호출을 보여준다.

예제 입력
7 6 2 6 1 0 2 0 4 1 2 1 3 1 5 4 6

Anthony		Catherine	
호출	반환값	호출	반환값
`Mark(7,6,2,6,[0,0,1,1,1,4],[2,4,2,3,5,6])`	`[1,0,0,1,0,1]`		
		`Init(2, 6)`	
		`Move([2, 1])`	`0`
		`Move([0, 0])`	`-1`
		`Move([1, 1])`	`0`
		`Move([0, 1])`	`1`

이 예제 입력에서, Catherine은 1, 5, 1, 2, 0 순으로 마을을 방문한다. $d = 2$이고, Catherine은 4번 움직였다.

이 입력은 서브태스크 7의 조건을 만족한다.

대회 홈페이지의 아카이브에서 받을 수 있는 파일 중, `sample-02.txt`는 서브태스크 4의 조건을 만족한다.