통행료

일본의 도시들은 고속도로 망으로 연결되어 있다. 이 고속도로 망은 N 개의 도시를 M 개의 고속도로가 잇는 형태로 되어 있다. 각각의 고속도로는 한 쌍의 서로 다른 두 도시를 잇는다. 임의의 두 도시를 잇는 고속도로는 최대 하나이다. 도시는 0부터 $N-1$ 까지 숫자로 표현하고, 고속도로는 0부터 $M-1$ 까지 숫자로 표현한다. 모든 고속도로는 양방향으로 통행할 수 있다. 한 도시에서 어떤 다른 도시든지 한 개 이상의 고속도로를 타고 여행할 수 있다.

각 고속도로마다 통행료가 있다. 고속도로의 통행료는 **교통 상황** 에 따라 결정된다. 각 고속도로의 교통 상황은 **한산**하거나 **혼잡**하거나 둘 중 하나이다. 각 고속도로의 통행료는, 이 고속도로가 한산하면, A 엔 (일본 화폐 단위), 이 고속도로가 혼잡하면 B 엔이다. $A < B$라는 것은 보장된다. A와 B의 값은 당신이 알고 있다는데 유의하시오.

당신은 모든 고속도로의 교통 상황을 정해주면 도시 S에서 도시 T로 ($S \neq T$) 여행하는데 드는 통행료 총액의 최소값을 구해주는 기계를 갖고 있다.

그러나 이 기계는 완벽하지 않다. S와 T가 무엇인지는 고정되어 있고 (즉, 기계 안에 쓰여 있어서 바꿀 수 없다.) 당신은 이 값을 모른다. 당신은 S와 T가 무엇인지 알고 싶다. 이를 위해서, 기계에 여러 가지 교통 상황을 주고, 이 때 계산된 통행료 총액의 최소값들을 가지고 S와 T 값을 추론해내려 한다. 교통 상황을 정해서 기계에 전달하는 일은 비용이 많이 들기 때문에, 이 기계를 작은 횟수로 사용하고 싶다.

Implementation details

다음 함수를 구현하시오.

```
find_pair(int N, int[] U, int[] V, int A, int B)
```

- `N`: 도시의 수
- `U`,`V`: 길이 M인 배열인데, M 은 도시들을 잇는 고속도로의 갯수다. 모든 i ($0 \leq i \leq M-1$) 에 대해서, 고속도로 i는 도시 `U[i]` 와 도시 `V[i]` 를 연결한다.
- `A`: 교통 상황이 한산할 때 고속도로 하나를 지나가는 통행료
- `B`: 교통 상황이 혼잡할 때 고속도로 하나를 지나가는 통행료
- 이 함수는 각 테스트케이스마다 정확히 한 번 호출된다.
- M의 값은 배열의 길이이며, 구현 명세에서 알려준 방법을 통해서 얻을 수 있다는데 주의하라.

함수 `find_pair`는 다음 함수를 호출할 수 있다.

```
int64 ask(int[] w)
```

- `w` 의 길이는 M이어야 한다. 배열 `w`는 교통 상황을 표현한다.
- 모든 i $(0 \leq i \leq M-1)$에 대해서 , `w[i]`는 고속도로 i의 교통 상황 정보이다. `w[i]`의 값은 0 아니면 1이다.
  - `w[i] = 0`이면 고속도로 i의 교통 상황은 한산하다.
  - `w[i] = 1`이면 고속도로 i의 교통 상황은 혼잡하다.
- 이 함수의 리턴값은 교통 상황이 `w`로 주어졌을 때 도시 S에서 도시 T로 여행하는데 드는 통행료의 총합의 최소값을 리턴한다.
- 이 함수는 최대 100 번 호출될 수 있다 (매 테스트케이스마다).

`find_pair`는 답을 보고하기 위해서 다음 함수를 호출해야 한다.

```
answer(int s, int t)
```

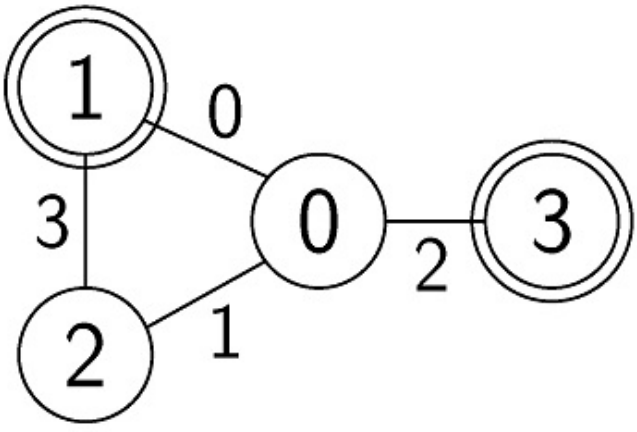
- `s` 와 `t` 는 S 와 T의 쌍이어야 한다 (순서는 상관없다).
- 이 함수는 정확히 한 번 호출되어야 한다.

만약 위 조건이 만족되지 않으면 당신의 프로그램은 **Wrong Answer**로 판정된다. 그렇지 않으면, 당신의 프로그램은 **Accepted**로 판정되며, 당신의 점수는 `ask` 함수의 호출 횟수에 따라 판정된다. (Subtasks 항목을 참조하시오)

Example

$N = 4, M = 4, U = [0, 0, 0, 1], V = [1, 2, 3, 2], A = 1, B = 3, S = 1, T = 3$이라고 하자.

그레이더는 `find_pair(4, [0, 0, 0, 1], [1, 2, 3, 2], 1, 3)`를 호출한다.



위 그림에서, i가 쓰여진 에지는 고속도로 i에 대응한다. 다음 순서대로 `ask` 함수를 호출했을 때 각각에 대응하는 리턴 값은 다음과 같다.

Call	Return
`ask([0, 0, 0, 0])`	2
`ask([0, 1, 1, 0])`	4
`ask([1, 0, 1, 0])`	5
`ask([1, 1, 1, 1])`	6

`ask([0, 0, 0, 0])`가 호출되면, 모든 고속도로의 교통 상황은 한산하며 이 때 각 고속도로의 통행료는 1이다. $S = 1$에서 $T = 3$로 이동하는 가장 통행료가 적은 경로는 $1 \to 0 \to 3$이다. 이 때 이 경로의 통행료 총합은 2이다. 따라서 이 함수의 리턴값은 2이다.

정답을 보고하기 위해서, 함수 `find_pair`는 `answer(1, 3)` 또는 `answer(3, 1)`를 호출해야 한다.

압축된 첨부 패키지 파일의 `sample-01-in.txt`는 이 예제에 대응한다. 다른 입출력 예제도 이 패키지에 포함되어 있다.

Constraints

- $2 \le N \le 90\,000$
- $1 \le M \le 130\,000$
- $1 \le A < B \le 1\,000\,000\,000$
- 모든 $0 \le i \le M - 1$에 대해
  - $0 \le U[i] \le N - 1$
  - $0 \le V[i] \le N - 1$
  - $U[i] \ne V[i]$
- $(U[i], V[i]) \ne (U[j], V[j])$ 이고 $(U[i], V[i]) \ne (V[j], U[j])$ $(0 \le i < j \le M - 1)$
- 어떤 두 도시를 고르더라도 한 도시에서 다른 도시로 하나 이상의 고속도로를 따라서 이동할 수 있다.
- $0 \le S \le N - 1$
- $0 \le T \le N - 1$
- $S \ne T$

이 문제에서 그레이더는 적응적이지 않다 (NOT adaptive). 이것은 S와 T가 그레이더의 수행 초기에 고정되어서 당신이 요청하는 쿼리에 따라 바뀌지 않는다는 것을 의미한다.

Subtasks

1. (5 points) S, T 중 하나는 0, $N \le 100$, $M = N - 1$
2. (7 points) S, T 중 하나는 0, $M = N - 1$
3. (6 points) $M = N - 1$, $U[i] = i$, $V[i] = i + 1$ $(0 \le i \le M - 1)$
4. (33 points) $M = N - 1$
5. (18 points) $A = 1$, $B = 2$
6. (31 points) 추가적인 제약 조건이 없다.

당신의 프로그램이 **Accepted**로 판정되고 `ask`함수를 X번 호출하였다고 하자. 그렇다면 이 테스트케이스에서 당신의 점수 P는 서브태스크 번호에 따라 다음과 같이 계산된다.

- Subtask 1. $P = 5$.
- Subtask 2. 만약 $X \leq 60$이면, $P = 7$. 그렇지 않으면 $P = 0$.
- Subtask 3. 만약 $X \leq 60$이면, $P = 6$. 그렇지 않으면 $P = 0$.
- Subtask 4. 만약 $X \leq 60$이면, $P = 33$. 그렇지 않으면 $P = 0$.
- Subtask 5. 만약 $X \leq 52$이면, $P = 18$. 그렇지 않으면 $P = 0$.
- Subtask 6.
  - 만약 $X \leq 50$이면, $P = 31$.
  - 만약 $51 \leq X \leq 52$이면, $P = 21$.
  - 만약 $53 \leq X$이면, $P = 0$.

각 subtask의 점수는 그 subtask의 테스트 케이스들에 대한 점수 중 최소값임에 주의하자.

Sample grader

샘플 그레이더는 다음 형식으로 입력을 받는다.

- line 1: $N\ M\ A\ B\ S\ T$
- line $2 + i$ $(0 \leq i \leq M - 1)$: $U[i]\ V[i]$

프로그램이 **Accepted**로 판정되면, 샘플 그레이더는 `Accepted : q` 를 출력한다. 여기서, `q`는 함수 `ask`의 호출 횟수이다.

프로그램이 **Wrong Answer**로 판정되면, 샘플 그레이더는 `Wrong : MSG` 로 출력한다. 여기서, `MSG`는 다음 중 하나이다.

- `answered not exactly once`: 함수 `answer`가 정확히 한번 호출되지 않았다.
- `w is invalid`: `ask`에 주어진 `w`의 길이가 M가 아니거나 `w[i]`의 값이 0 또는 1 이 아닌 i가 있다 $(0 \leq i \leq M - 1)$.
- `more than 100 calls to ask`: 함수 `ask` 가 100 번보다 많이 호출되었다.
- `{s, t} is wrong`: 함수 `answer`의 리턴값 `s`와 `t`가 정답이 아니다.