호텔방문

[문제 설명]

신도시가 건설되고 있다.

이 도시는 N개의 지역으로 나뉘어져 있고, 각 지역마다 하나의 호텔이 건설된다.

호텔을 운영하는 회사들이 여러 개 존재하며 각각 자체 브랜드를 사용하여 호텔을 운영한다.

이 도시 내 각각의 호텔들은 위 회사들 중 하나에 속하게 된다.

도시는 성장하면서 지역간 도로가 건설되기도 하고, 호텔들을 소유한 회사 간에 합병도 일어난다.

호텔을 운영하는 회사들은 호텔 비즈니스를 성장시키기 위해

호텔에서 숙박을 무료로 할 수 있는 경품을 제공하기로 하였다.

경품에는 숙박할 수 있는 호텔의 브랜드 2개가 적혀있다.(브랜드A, 브랜드B)

경품 당첨자가 브랜드A에 속한 호텔들 중 하나와 브랜드B에 속한 호텔들 중 하나를 선택하면,

다음 사항을 무료로 제공한다.

- 선택한 브랜드A인 호텔에서 체크인 후 숙박
- 선택한 브랜드A인 호텔에서 선택된 브랜드B인 호텔로 이동
- 선택된 브랜드B인 호텔에서 숙박 후 체크아웃

당신은 경품 당첨자이다. 경품으로 선택할 수 있는 호텔의 브랜드A와 브랜드B가 주어진다.

당첨자가 여행을 위해 필요한 비용은 출발지에서 선택한 브랜드A의 호텔까지 이동하는 비용과

선택한 브랜드B의 호텔에서 출발지로 다시 돌아오는 비용 뿐이다.

이들 비용을 최소로 하는 호텔들을 선택하고 싶다. 이동 비용은 거리에 비례하기 때문에

최소 이동 거리를 구하면 비용을 최소화 시킬 수 있다.

void init(int N, int mBrands[])

각 테스트 케이스의 처음에 호출된다.

N개의 지역에 호텔이 건설되며, 각 호텔의 ID는 0부터 N-1가 부여된다.

그리고, 해당 호텔을 소유한 회사의 브랜드의 ID가 호텔 ID 순서대로 주어진다.

다시 말해 호텔 ID의 i인 경우 이 호텔의 브랜드의 ID는 mBrands[i]가 된다.

브랜드의 ID는 0 ~ 49로 주어진다.

초기에는 지역 간에 연결된 도로가 전혀 없다.

Parameters

N: 지역의 개수 (5 ≤ N ≤ 5,000)

mBrands[i]: 호텔 ID가 i인 호텔의 브랜드ID (0≤mBrands[i]≤49 , 0≤i≤N-1)

void connect(int mHotelA, int mHotelB, int mDistance)

호텔 mHotelA가 있는 지역과 호텔 mHotelB가 있는 지역을 연결하는 도로를 설치한다.

도로의 길이는 mDistance이다.

두 지역에는 이전에 설치된 도로가 없음을 보장한다.

mHotelA와 mHotelB가 다름을 보장한다.

설치된 도로는 양쪽 방향으로 이동이 모두 가능하다.

Parameters

mHotelA: 도로를 설치하는 지역의 호텔ID (0 ≤ mHotelA ≤ N-1)

mHotelB: 도로를 설치하는 지역의 호텔ID (0 ≤ mHotelB ≤ N-1)

mDistance: 도로의 길이 (1 ≤ mDistance ≤ 10)

int merge(int mHotelA, int mHotelB)

호텔 mHotelA가 속한 브랜드의 회사가 호텔 mHotelB가 속한 브랜드의 회사를 흡수 합병한다.

따라서 호텔 mHotelB가 속한 브랜드의 호텔들은 모두 mHotelA가 속한 브랜드로 변경된다.

합병 후, 호텔 mHotelA가 속한 브랜드를 가진 호텔의 개수를 반환한다.

mHotelA와 mHotelB가 다름을 보장한다.

mHotelA가 속한 브랜드와 mHotelB가 속한 브랜드가 동일한 경우 합병은 이루어지지 않고

해당 브랜드의 호텔 수만 반환한다.

Parameters

mHotelA: 합병할 브랜드를 가진 호텔ID (0 ≤ mHotelA ≤ N-1)

mHotelB: 합병될 브랜드를 가진 호텔ID (0 ≤ mHotelB ≤ N-1)

Return

mHotelA가 속한 브랜드를 가진 호텔의 개수

int move(int mStart, int mBrandA, int mBrandB)

경품에 당첨되어 투숙할 브랜드 mBrandA와 mBrandB가 주어진다.

호텔ID가 mStart인 호텔이 위치한 지역에서 당첨자는 출발한다.

가장 이동 비용이 적게 드는 해당 브랜드의 호텔들을 찾은 후,

비용을 발생시키는 이동거리의 최소 합을 반환한다.

mBrandA와 mBrandB는 동일할 수 있으나, 선택되는 호텔들은 모두 달라야 한다.

선택되는 호텔들의 지역은 mStart가 있는 지역과 달라야 한다.

위에서 언급한 조건들을 만족하지 않아 이동거리의 최소 합을 구할 수 없는 경우는 주어지지 않는다.

Parameters

mStart: 경품 당첨자의 출발할 지역에 있는 호텔ID (0 ≤ mStart ≤ N-1)

mBrandA: 투숙할 첫번째 호텔 브랜드 (0 ≤ mBrandA ≤ 49)

mBrandB: 투숙할 두번째 호텔 브랜드 (0 ≤ mBrandB ≤ 49)

Return

비용을 발생시키는 이동거리의 최소 합

[제약사항]

1. 각 테스트 케이스 시작 시 init() 함수가 호출된다.
2. 각 테스트 케이스에서 connect() 함수의 호출 횟수는 10,000 이하이다.
3. 각 테스트 케이스에서 merge() 함수의 호출 횟수는 100 이하이다.
4. 각 테스트 케이스에서 move() 함수의 호출 횟수는 1,000 이하이다.

```
// *** main.cpp ***
#ifndef _CRT_SECURE_NO_WARNINGS
#define _CRT_SECURE_NO_WARNINGS
#endif

#include <stdio.h>

void init(int N, int mBrands[]);
void connect(int mHotelA, int mHotelB, int mDistance);
int merge(int mHotelA, int mHotelB);
int move(int mStart, int mBrandA, int mBrandB);

#define MAX_N 5000

#define CMD_INIT 100
#define CMD_CONNECT 200
#define CMD_MERGE 300
#define CMD_MOVE 400

static bool run()
{
	int query_num;
	scanf("%d", &query_num);

	int ans;
	bool ok = false;
```

```
for (int q = 0; q < query_num; q++)
{
	int query;
	scanf("%d", &query);
	if (query == CMD_INIT)
	{
		int N;
		int mBrands[MAX_N];
		scanf("%d", &N);
		for (int i = 0; i < N; i++) {
			scanf("%d", &mBrands[i]);
		}
		init(N, mBrands);
		ok = true;
	}
	else if (query == CMD_CONNECT)
	{
		int mHotelA, mHotelB, mDistance;
		scanf("%d %d %d", &mHotelA, &mHotelB, &mDistance);
		connect(mHotelA, mHotelB, mDistance);
	}
	else if (query == CMD_MERGE)
	{
		int mHotelA, mHotelB;
		scanf("%d %d", &mHotelA, &mHotelB);
```

```
            int ret = merge(mHotelA, mHotelB);
            scanf("%d", &ans);
            if (ans != ret)
            {
                ok = false;
            }
        }
        else if (query == CMD_MOVE)
        {
            int mStart, mBrandA, mBrandB;
            scanf("%d %d %d", &mStart, &mBrandA, &mBrandB);
            int ret = move(mStart, mBrandA, mBrandB);
            scanf("%d", &ans);
            if (ans != ret)
            {
                ok = false;
            }
        }
    }
    return ok;
}

int main()
{
    setbuf(stdout, NULL);
    freopen("input.txt", "r", stdin);
```

```
	int T, MARK;
	scanf("%d %d", &T, &MARK);

	for (int tc = 1; tc <= T; tc++)
	{
		int score = run() ? MARK : 0;
		printf("#%d %d₩n", tc, score);
	}
	return 0;
}
```