물류허브 22.10.01

[문제 설명]

여러 도시를 연결하는 단방향의 도로 N개가 운송 비용과 함께 주어진다.

이 도시 중에서, 한 곳에 물류 허브를 설치했을 때, 총 운송 비용을 계산하고자 한다.

총 운송 비용은 각 도시에서 허브 도시까지 왕복에 필요한 최소 비용을 모두 합한 값이다.

허브 도시의 운송 비용은 0이다.

그리고 도로는 단방향이기 때문에 허브 도시까지 가는 최소 비용과 돌아오는 최소 비용은 다를 수 있다.

1->5 35 , 1->4 15

2->3 32

3->2 46, 3->5 23

4->1 24, 4->5 13

5->4 31, 5->3 30, 5->2 47

1 1->4->5->4->1 : 83

2 2->3->5->2 : 102

3 3->5->3 : 53

4 4->5->4 : 44

5 0

예를 들어, [Fig. 1]과 같이 도시와 도로의 정보가 주어진 경우를 살펴 보자.

원 안의 숫자는 도시를 나타내는 번호이고, 화살표는 각 도시를 연결하는 단방향의 도로이고 그 위의 숫자가 운송 비용이다.

5번 도시에 허브를 설치할 경우 각 도시에서 허브 도시까지 왕복에 필요한 최소 비용은 [Fig. 1]의 표와 같다.

따라서, 총 운송 비용은 83 + 102 + 53 + 44 = 282가 된다.

※ 아래 API 설명을 참조하여 각 함수를 구현하라.

int init(int N, int sCity[], int eCity[], int mCost[])

각 테스트 케이스의 처음에 호출된다.

N개의 도로 정보가 주어진다. 각 도로의 출발 도시와 도착 도시, 운송 비용이 주어진다.

도로 정보로 주어지는 도시의 총 개수를 반환한다.

단방향 도로이기 때문에 출발 도시에서 도착 도시로만 갈 수 있다.

출발 도시와 도착 도시의 순서쌍이 동일한 도로는 없다.

출발 도시와 도착 도시가 서로 같은 경우는 없다.

Parameters

N: 도로의 개수 (10 ≤ N ≤ 1400)

(0 ≤ i < N)인 모든 i에 대해,

sCity[i]: 도로 i의 출발 도시 (1 ≤ sCity[i] ≤ 1,000,000,000)

eCity[i]: 도로 i의 도착 도시 (1 ≤ eCity[i] ≤ 1,000,000,000)

mCost[i]: 도로 i의 운송 비용 (1 ≤ mCost[i] ≤ 100)

Returns

도시의 총 개수를 반환한다.

void add(int sCity, int eCity, int mCost)

출발 도시가 sCity이고, 도착 도시가 eCity이고, 운송 비용이 mCost인 도로를 추가한다.

init()에 없던 새로운 도시는 주어지지 않는다.

sCity에서 eCity로 가는 도로가 이미 존재하는 경우는 입력으로 주어지지 않는다.

Parameters

sCity: 도로 i의 출발 도시 (1 ≤ sCity ≤ 1,000,000,000)

eCity: 도로 i의 도착 도시 (1 ≤ eCity ≤ 1,000,000,000)

mCost: 도로 i의 운송 비용 (1 ≤ mCost ≤ 100)

int cost(int mHub)

mHub 도시에 물류 허브를 설치할 경우, 총 운송 비용을 계산하여 반환한다.

mHub 도시의 운송 비용은 0으로 계산한다.

각 도시에서 mHub 도시까지 왕복이 불가능한 경우는 입력으로 주어지지 않는다.

Parameters

mHub: 허브를 설치할 도시 (1 ≤ mHub ≤ 1,000,000,000)

Returns

총 운송 비용, 다시 말해 각 도시에서 허브 도시까지 왕복에 필요한 최소 비용을 모두 합한 값을 반환한다.

[제약사항]

1. 각 테스트 케이스 시작 시 init() 함수가 호출된다.

2. 각 테스트 케이스에서 도시의 최대 개수는 600 이하이다.

3. 각 테스트 케이스에서 모든 함수의 호출 횟수 총합은 50 이하이다.

```
#include <stdio.h>

#define MAX_N 1400

#define CMD_INIT 1

#define CMD_ADD 2

#define CMD_COST 3


static bool run() {

	int q;

	scanf("%d", &q);


	int n;

	int sCityArr[MAX_N], eCityArr[MAX_N], mCostArr[MAX_N];

	int sCity, eCity, mCost, mHub;

	int cmd, ans, ret = 0;

	bool okay = false;


	for (int i = 0; i < q; ++i) {

		scanf("%d", &cmd);

		switch (cmd) {

		case CMD_INIT:

			okay = true;

			scanf("%d", &n);

			for (int j = 0; j < n; ++j) {

				scanf("%d %d %d", &sCityArr[j], &eCityArr[j], &mCostArr[j]);
```

```
            }
            scanf("%d", &ans);
            ret = init(n, sCityArr, eCityArr, mCostArr);

            if (ans != ret)
                okay = false;
            break;
        case CMD_ADD:
            scanf("%d %d %d", &sCity, &eCity, &mCost);
            add(sCity, eCity, mCost);
            break;
        case CMD_COST:
            scanf("%d %d", &mHub, &ans);
            ret = cost(mHub);
            if (ans != ret)
                okay = false;
            break;
        default:
            okay = false;
            break;
        }
    }
    return okay;
}

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;
}
```

25 100

6

1 10

3 2 46

1 4 15

5 3 30

5 4 31

3 5 23

5 2 47

1 5 35

4 1 24

2 3 32

4 5 13

5

3 5 282

2 5 1 24

3 5 251

2 2 1 11

3 4 266

10

1 40

654583776 406932607 68

51550460 406932607 83

109335179 51550460 86

449466925 794471794 81

551188311 51550460 76

12346 109335179 11

654583776 293799193 46

654583776 12346 1

551188311 109335179 84

51550460 551188311 8

51550460 293799193 52

229283574 109335179 28

551188311 406932607 24

406932607 51550460 76

51550460 449466925 98

51550460 109335179 92

109335179 229283574 6

12346 654583776 96

293799193 51550460 24

406932607 229283574 82

794471794 449466925 45

229283574 406932607 72

12346 551188311 47

293799193 654583776 42

406932607 654583776 5

406932607 12346 8

551188311 229283574 55

654583776 449466925 54

109335179 12346 57

449466925 654583776 57

551188311 654583776 77

406932607 551188311 9

551188311 12346 29

229283574 551188311 33

109335179 551188311 71

12346 406932607 41

293799193 12346 55

12346 293799193 81

654583776 551188311 13

449466925 51550460 11

10

3 551188311 830

2 654583776 109335179 15

3 449466925 1203

2 12346 449466925 73

3 406932607 821

2 229283574 293799193 17

3 449466925 1170

2 794471794 109335179 71

3 449466925 1170