백준 1956번 - 운동

시간제한	메모리 제한	제출	정답	맞은 사람	정답 비율
2초	192MB	10649	4347	3278	44.059%

문제

V개의 마을와 E개의 도로로 구성되어 있는 도시가 있다. 도로는 마을과 마을 사이에 놓여 있으며, 일방 통행 도로이다. 마을에는 편의상 1번부터 V번까지 번호가 매겨져 있다고 하자.

당신은 도로를 따라 운동을 하기 위한 경로를 찾으려고 한다. 운동을 한 후에는 다시 시작점으로 돌아오는 것이 좋기 때문에, 우리는 사이클을 찾기를 원한다. 단, 당신은 운동을 매우 귀찮아하므로, 사이클을 이루는 도로의 길이의 합이 최소가 되도록 찾으려고 한다.

도로의 정보가 주어졌을 때, 도로의 길이의 합이 가장 작은 사이클을 찾는 프로그램을 작성하시오. 두 마을을 왕복하는 경우도 사이클에 포함됨에 주의한다.

입력

첫째 줄에 V와 E가 빈칸을 사이에 두고 주어진다. ($2 \le V \le 400$, $0 \le E \le V(V-1)$) 다음 E개의 줄에는 각각 세 개의 정수 a, b, c가 주어진다. a번 마을에서 b번 마을로 가는 거리가 c인 도로가 있다는 의미이다. ($a \rightarrow b$임에 주의) 거리는 10,000 이하의 자연수이다. (a, b) 쌍이 같은 도로가 여러 번 주어지지 않는다.

출력

첫째 줄에 최소 사이클의 도로 길이의 합을 출력한다. 운동 경로를 찾는 것이 불가능한 경우에는 -1을 출력한다.

예제 입력1

```
3 4
1 2 1
3 2 1
1 3 5
2 3 2
```

예제 출력1

```
3
```

출처

출처

알고리즘 분류

- 그래프 이론
- 플로이드-와샬

접근 방법

문제를 잘 읽어보면, 임의의 2개의 노드 u, v 에 대하여 사이클의 길이를 구하는 것이다. (모든 마을을 방문하지 않아도 된다.)

이 문제를 Naive하게 생각해보자면, 임의의 노드 u, v에 대해 $u \rightarrow v \rightarrow u$ 로 가는 모든 쌍 끼리의 거리를 구하는 것으로 생각할 수 있다.

V 의 범위가 $2 \leq V \leq 400$ 이고 E의 범위가 $0 \leq E \leq V(V-1)$ 이므로 $O(V^3)$ 의 복잡도를 가지는 플로이드 와샬로 접근할 수 있다.

각각의 E에 부가된 cost는 10,000 이하이므로, int overflow에 대한 추가적인 고려도 필요하다.

소스코드

```cpp
#define FASTIO cin.tie(0)->sync_with_stdio(false), cout.tie(0)
////////////////////////////////////////////////////////////////////
#include <bits/stdc++.h>
using namespace std;
int V, E, dist[405][405];
vector<vector<pair<int,int>>> adj;
int main(void){
    FASTIO;
////////////////////////////////////////////////////////////////////
    cin >> V >> E;
    adj.resize(V+1);

    for(int i=0; i<405; i++){
        for(int j=0; j<405; j++) {
            dist[i][j] = 2e9;
```

```
        }
    }

    for(int i=0; i<E; i++){
        int a, b, c;
        cin >> a >> b >> c;
        adj[a].push_back({c, b});
        dist[a][b] = c;
    }

    for(int k=1; k<=V; k++){
        for(int i=1; i<=V; i++){
            for(int j=1; j<=V; j++){
                if(dist[i][k] != 2e9 && dist[k][j] != 2e9){
                    dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j]);
                }
            }
        }
    }

    int ans = 2e9;
    for(int st = 1; st <= V; st++){
        for(int dst = 1; dst <= V; dst++){
            if(dist[st][dst] != 2e9 && dist[dst][st] != 2e9){
                ans = min(ans, dist[st][dst] + dist[dst][st]);
            }
        }
    }
    if(ans != 2e9)
        cout << ans ;
    else
        cout << -1;
    return 0;
}
```