2022-2 HI-ARC 중급스터디

8주차. 최소 스패닝 트리

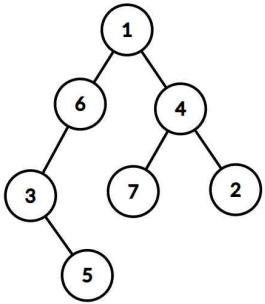
이지은 (leeju1013)

목차

트리 복습

- What?

- 무방향 비순환 연결 그래프



- 사이클이 없음
  : 임의의 서로 다른 두 정점을 잇는 경로가 유일함

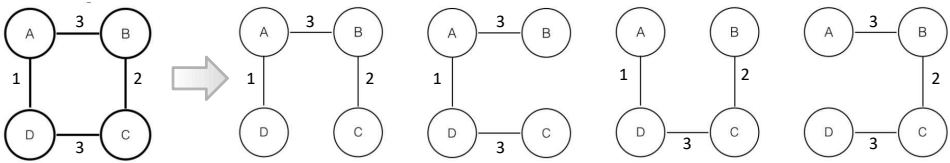
- 모든 정점(노드)이 이어져있음
  : 정점 개수 N개이면 간선 개수 N-1개

- 계층형 구조

0. 스패닝 트리(Spanning Tree)

- What?

: 무방향 그래프의 최소 연결 부분 그래프

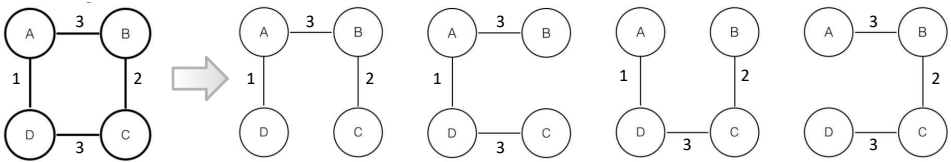


- 그래프의 모든 정점 N개를, 최소 간선 개수인 N-1개로 연결
- 사이클이 없음
- 유일하지 않을 수 있음
- 신장트리의 가중치 =간선 가중치들의 합

1. 최소 스패닝 트리(MST)

- What?

: 간선의 가중치 합이 최소인 스패닝 트리



- 유일하지 않을 수 있음

2. 크루스칼 알고리즘(Kruskal Algorithm)

- What?

: 탐욕법을 기반으로 간선을 추가해가면서 MST를 구하는 알고리즘

- How?

: 트리임을 만족하면서 가중치가 작은 간선을 차례로 추가함.
이때, 해당 간선을 추가해도 사이클이 생기지 않을 때만 연결!

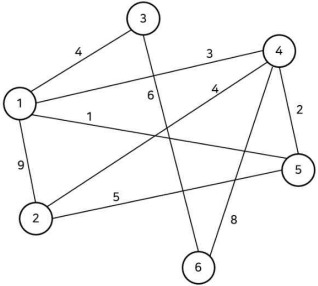
2. 크루스칼 알고리즘(Kruskal Algorithm)

- How?

1. 선택되지 않은 간선들 중 가중치가 최소인 간선을 선택한다.

2. 지금까지 구성된 스패닝 트리에 해당 간선을 추가해도 사이클이 발생하지 않는다면 스패닝 트리에 포함한다.

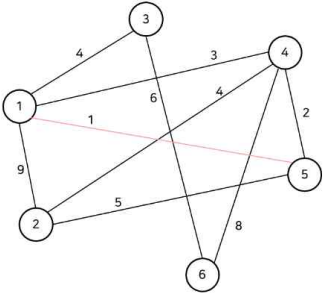
3. 총 (정점의 개수-1)개의 간선이 선택될 때까지 반복한다.



1. 선택되지 않은 간선들 중
   가중치가 최소인 간선을 선택한다.

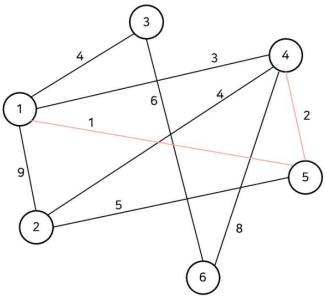
2. 지금까지 구성된 스패닝 트리에
   해당 간선을 추가해도
   사이클이 발생하지 않는다면
   스패닝 트리에 포함한다.

3. 총 (정점의 개수-1)개의 간선이
   선택될 때까지 반복한다.



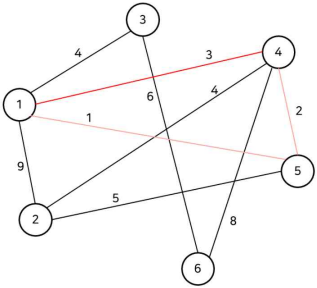
1, 5번 정점을 잇는
가중치 1인 간선이 선택됨.

이 간선을 추가해도
사이클이 발생하지 않으므로, 추가함.



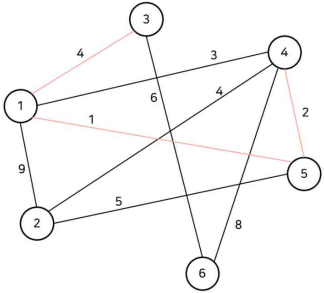
4, 5번 정점을 잇는
가중치 2인 간선이 선택됨.

이 간선을 추가해도
사이클이 발생하지 않으므로, 추가함.



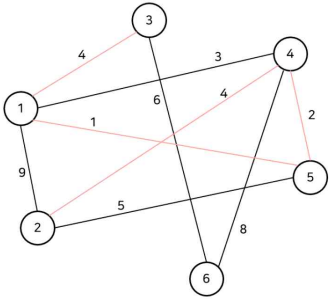
1, 4번 정점을 잇는
가중치 3인 간선이 선택됨.

이 간선을 추가하면
사이클이 발생하므로, 추가하지 않음.



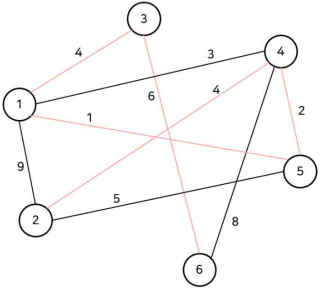
1, 3번 정점을 잇는
가중치 4인 간선이 선택됨.

이 간선을 추가해도
사이클이 발생하지 않으므로, 추가함.



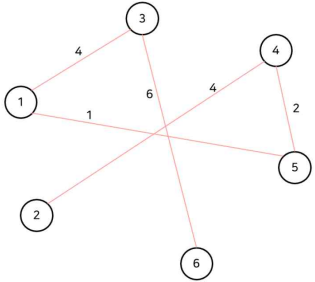
2, 4번 정점을 잇는
가중치 4인 간선이 선택됨.

이 간선을 추가해도
사이클이 발생하지 않으므로, 추가함.



3, 6번 정점을 잇는
가중치 6인 간선이 선택됨.

이 간선을 추가해도
사이클이 발생하지 않으므로, 추가함.



지금까지 5개의 간선을 뽑았고
(정점 개수-1) == 5 이므로 알고리즘 종료.

MST의 가중치는 뽑힌 간선들의 가중치 합인
1 + 2 + 4 + 4 + 6 = 17 임.

이 그래프에서 17보다 작은 가중치를
가지는 MST는 없음.

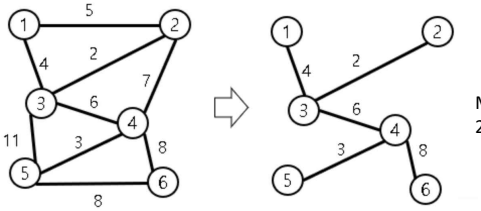
쉬는 시간

1. 최소 스패닝 트리
2. 크루스칼 알고리즘
   - 1922. 네트워크 연결
   - 21924. 도시건설

1922번 제출 맞힌 사람

네트워크 연결 성공

골드 IV



MST의 가중치 합
2 + 3 + 4 + 6 + 8 = 23

문제

도현이는 컴퓨터와 컴퓨터를 모두 연결하는 네트워크를 구축하려 한다. 하지만 아쉽게도 허브가 있지 않아 컴퓨터와 컴퓨터를 직접 연결하여야 한다. 그런데 모두가 자료를 공유하기 위해서는 모든 컴퓨터가 연결이 되어 있어야 한다. (a와 b가 연결이 되어 있다는 말은 a에서 b로의 경로가 존재한다는 것을 의미한다. a에서 b를 연결하는 선이 있고, b와 c를 연결하는 선이 있으면 a와 c는 연결이 되어 있다.)

그런데 이왕이면 컴퓨터를 연결하는 비용을 최소로 하여야 컴퓨터를 연결하는 비용 외에 다른 곳에 돈을 더 쓸 수 있을 것이다. 이제 각 컴퓨터를 연결하는데 필요한 비용이 주어졌을 때 모든 컴퓨터를 연결하는데 필요한 최소비용을 출력하라. 모든 컴퓨터를 연결할 수 없는 경우는 없다.

입력

첫째 줄에 컴퓨터의 수 N (1 ≤ N ≤ 1000)가 주어진다.

둘째 줄에는 연결할 수 있는 선의 수 M (1 ≤ M ≤ 100,000)가 주어진다.

셋째 줄부터 M+2번째 줄까지 총 M개의 줄에 각 컴퓨터를 연결하는데 드는 비용이 주어진다. 이 비용의 정보는 세 개의 정수로 주어지는데, 만약에 a b c 가 주어져 있다고 하면 a컴퓨터와 b컴퓨터를 연결하는데 비용이 c (1 ≤ c ≤ 10,000) 만큼 든다는 것을 의미한다. a와 b는 같을 수도 있다.

출력

모든 컴퓨터를 연결하는데 필요한 최소비용을 첫째 줄에 출력한다.

예제 입력 1

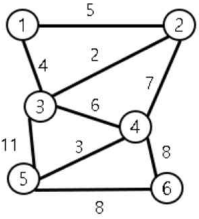
```
6
9
1 2 5
1 3 4
2 3 2
2 4 7
3 4 6
3 5 11
4 5 3
4 6 8
5 6 8
```

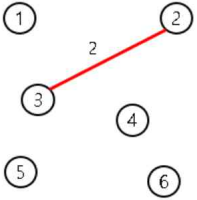
예제 출력 1

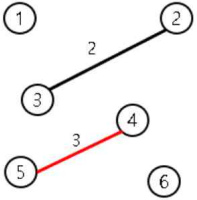
```
23
```

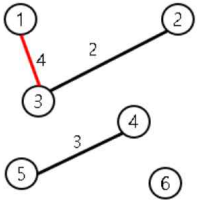
예제 입력 1

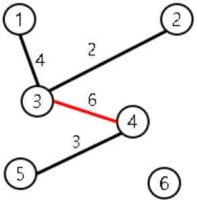
```
6
9
1 2 5
1 3 4
2 3 2
2 4 7
3 4 6
3 5 11
4 5 3
4 6 8
5 6 8
```

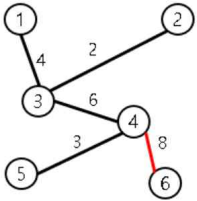


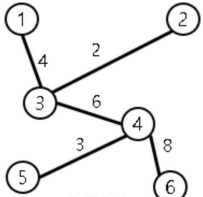












예제 출력 1

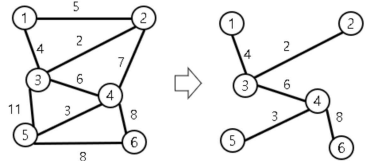
```
23
```

정점 개수가 V, 간선 개수가 E일 때
시간복잡도는 **O(ElogE)**

```cpp
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;
typedef pair<int,int> pii;

vector<pair<int, pii>> v;
int par[1001],n,m, ans;

int find(int num){
    if(num==par[num]) return num;
    return par[num] = find(par[num]); //경로압축
}
void unite(int cost, int a, int b){
    a=find(a), b=find(b);

    if(a==b) return; // 이미 같은 집합이면 사이클 발생. 간선 추가 불가능

    par[b]=a; // 다른 집합이면 간선 추가 가능
    ans+=cost; // MST 가중치 합 업데이트
}
```

```cpp
int main() {
    ios_base::sync_with_stdio(false); cin.tie(NULL);

    cin>>n>>m;
    for (int i = 0; i <= n; i++) par[i] = i;          // O(V)

    for (int i = 0; i < m; i++) {                     // O(E)
        int x, y, z;
        cin >> x >> y >> z;
        v.push_back({z, {x, y}}); // { 가중치z , { x,y } }
    }
    sort(v.begin(), v.end()); //가중치 기준 오름차순 정렬   // O(ElogE)

    for (auto &k:v) {                                 // O(E) *
        unite(k.first, k.second.first, k.second.second); //  O(1)
    }

    cout << ans; // 최소 비용(MST의 가중치)
    return 0;
}
```

MST의 가중치 합
2 + 3 + 4 + 6 + 8 = 23



도시 건설

성공

골드 IV

시간 제한	메모리 제한	제출	정답	맞힌 사람
1 초	512 MB	1623	727	567

문제

채완이는 신도시에 건물 사이를 잇는 양방향 도로를 만들려는 공사 계획을 세웠다.

공사 계획을 검토하면서 비용이 생각보다 많이 드는 것을 확인했다.

채완이는 공사하는 데 드는 비용을 아끼려고 한다.

모든 건물이 도로를 통해 연결되도록 최소한의 도로를 만들려고 한다.

예제 입력 3

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

예제 출력 3

```
-1
```

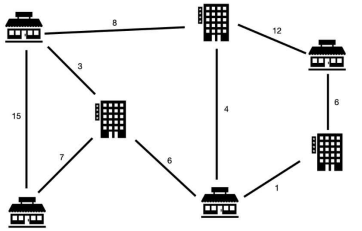
입력

첫 번째 줄에 건물의 개수 N $(3 \le N \le 10^5)$와 도로의 개수 M $(2 \le M \le min(\frac{N(N-1)}{2}, 5 \times 10^5))$가 주어진다.

두 번째 줄 부터 $M+1$줄까지 건물의 번호 a, b $(1 \le a, b \le N, a \ne b)$와 두 건물 사이 도로를 만들 때 드는 비용 $c(1 \le c \le 10^6)$가 주어진다. 같은 쌍의 건물을 연결하는 두 도로는 주어지지 않는다.

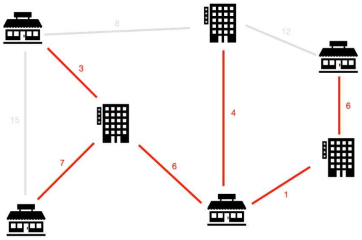
출력

예산을 얼마나 절약 할 수 있는지 출력한다. 만약 모든 건물이 연결되어 있지 않는다면 -1을 출력한다.



예제 입력 1

```
7 9
1 2 15
2 3 7
1 3 3
1 4 8
3 5 6
4 5 4
4 6 12
5 7 1
6 7 6
```



예제 출력 1

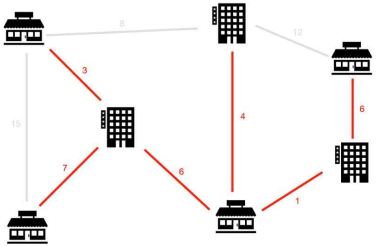
```
35
```

그림에 있는 도로를 다 설치할 때 드는 비용은 62이다. 모든 건물을 연결하는 도로만 만드는 비용은 27로 절약하는 비용은 35이다.

채완이는 도로가 너무 많아 절약되는 금액을 계산하는 데 어려움을 겪고 있다.

채완이를 대신해 얼마나 절약이 되는지 계산해주자.

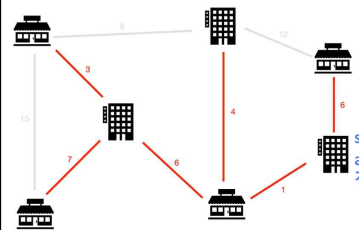
```cpp
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
typedef long long ll;
typedef pair<int,int> pii;

vector<pair<int, pii>> v;
int par[1000001], visited[1000001];
int n,m;
ll ans,sum;
vector<int> adj[1000001]; //dfs할때 이용
```



그림에 있는 도로를 다 설치할 때 드는 비용은 620이다. 모든 건물을 연결하는 도로만 만드는 비용은 27로 절약하는 비용은 350이다.

```cpp
int find(int num){
    if(num==par[num]) return num;
    return par[num] = find(par[num]); //경로압축
}
void unite(int cost, int a, int b){
    a=find(a), b=find(b);

    if(a==b) return; //이미 같은 집합이면 사이클 발생하므로 간선 추가 불가능

    par[b]=a; //다른 집합이면 간선 추가 가능
    ans+=cost; // MST 가중치 합 업데이트
}

void dfs(int cur){
    visited[cur]=1;
    for(auto &next:adj[cur]){
        if(!visited[next]) dfs(next);
    }
}
```

```cpp
int main(){
    ios_base::sync_with_stdio(false); cin.tie(NULL);

    cin>>n>>m;
    ans=0, sum=0;
    for (int i = 0; i <=n; i++) par[i] = i;

    for (int i = 0; i < m; i++) {
        int x, y, z;
        cin >> x >> y >> z;
        v.push_back({z, {x, y}}); //{ 가중치z , { x,y } }
        sum+=z; //모든 간선의 가중치 다 합친거

        adj[x].push_back(y);
        adj[y].push_back(x);
    }
    sort(v.begin(), v.end()); //가중치 기준 오름차순 정렬
```



sum : 모든 간선의 가중치 합
ans : MST의 가중치 합
정답 : sum-ans

```cpp
    dfs(1); //모든 도시가 연결되어 있는지 판단 위해 탐색
    for(int i=1; i<=n; i++){
        if(!visited[i]){
            //방문하지 않은 건물이 있다면 모든 도시가 연결 될 수 없다는 의미
            cout<<-1;
            return 0;
        }
    }

    for (auto &k:v) {
        unite(k.first, k.second.first, k.second.second);
    }
    cout << sum-ans << '\n';
    //최소 비용(ans)이 아니라 절약 할 수 있는 최대 비용(sum-ans)
    return 0;
}
```

감사합니다

- 필수 문제
  - 1922. 네트워크 연결
  - 21924. 도시 건설
- 연습 문제
  - 1197. 최소 스패닝 트리
  - 16938. 행성 연결
  - 6497. 전력난
  - 1647 도시 분할 계획
  - 4386 별자리만들기
  - 10423. 전기가 부족해
- 강의를 마무리하며..