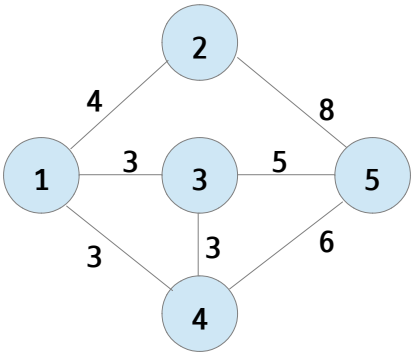
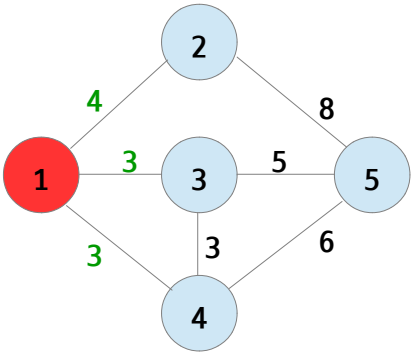
프림 알고리즘

sianke1991

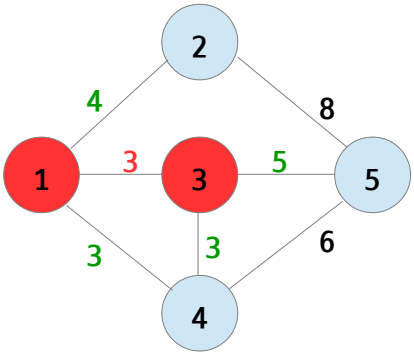
1. 주어진 연결그래프로 [간선의 가중치 합이 최소가 되게 하는 트리: 최소신장트리]를 찾는 방법중의 하나이다.
2. 임의의 노드를 최소신장트리(이하, MST)에 추가하고 매 단계마다 MST에 들어간 노드와 MST에 들어가지 않은 노드를 잇는 간선 중에서 가중치가 가장 작은 것(과 그 간선과 연결된 MST에 들어가지 않은 노드)을 MST에 추가한다. 이 과정을 (노드 개수-1) 만큼의 간선이 선택될 때 까지 반복한다.



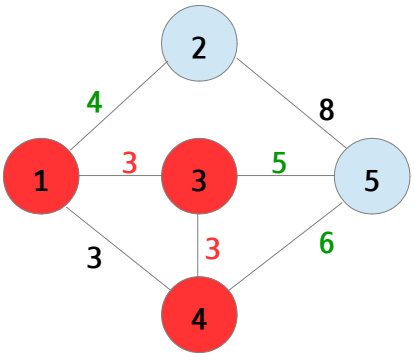

*이 연결그래프에 프림 알고리즘을 적용해 본다.*



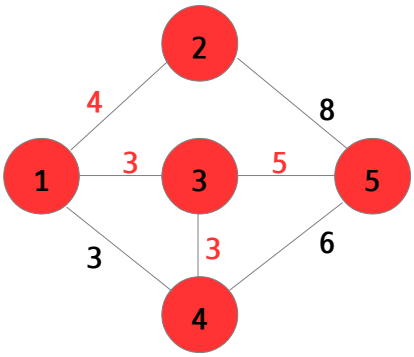

*노드 1을 MST에 추가하고 MST에 들어간 노드와 MST에 들어가지 않은 노드를 잇는 간선들을 살핀다. 이 중 가장 가중치가 낮은 (1, 3)을 MST에 추가하기로 한다.*




*노드 1과 3이 MST에 들어가 있고, MST에 추가할 간선 후보는 위 그림에서 녹색으로 표시되어 있다. 그 후보 중 가장 가중치가 낮은 (3, 4)를 MST에 추가하기로 한다.*




*이 과정을 간선 4개를 MST에 넣을 때까지 반복한다. (간선 (3, 4)를 MST에 넣으면서 동시에 간선(1, 4)는 더 이상 MST에 넣지 못하게 됨에 주의한다.)*




*이렇게 간선 4개가 MST에 들어가 있으면 알고리즘이 종료된다.*

3. MST에 들어갈 수 있는 간선 후보 중 최소의 것을 찾기 위해 매 단계마다 정렬을 하는 것은 매우 비효율적이다. 이를 해결하기 위해 우선순위 큐를 사용한다. 매번 노드을 MST에 추가하면서 그 노드에서 뻗어나가는 간선들을 우선순위 큐에 넣는다. 우선순위 큐는 자신이 담고 있는 간선 중에서 그 가중치가 가장 낮은 것을 뽑아낼 수 있다. 단, 노드를 하나 둘 MST에 넣으면서 MST에 들어갈 수 있는 간선 후보 중 그 자격을 잃어버리기도 한다. 따라서 우선순위 큐에서 MST에 추가할 간선 후보를 뽑되, 그 간선 후보가 MST에 들어갈 수 있는지 그 자격을 재점검해야 한다. (해당 간선 후보가 MST에 이미 추가된 노드에 연결되는 간선이라면 그 간선 후보는 MST에 포함될 자격을 잃은 것이다.)

```cpp
#include <bits/stdc++.h>

using namespace std;

int main() {
    int numNodes, numEdges;
    cin >> numNodes >> numEdges;

    vector<pair<int, int>> adjList[10001]; //인접리스트 {비용, 도착노드}

    for (int i=0; i<numEdges; i++) {
        int nodeA, nodeB, weight;
        cin >> nodeA >> nodeB >> weight;
        adjList[nodeA].push_back({weight, nodeB});
        adjList[nodeB].push_back({weight, nodeA});
    }


    //tuple<int, int, int> {비용, 출발노드, 도착노드}
    //greater<tuple<int, int, int>> 설정을 넣음으로써 최소힙을 사용하도록 만든다.
    priority_queue< tuple<int, int, int>, vector<tuple<int, int, int>>, greater<tuple<int, int, int>> > pq;
    //해당 노드가 MST에 포함되었는지 여부 (이 boolean 배열은 각 원소의 값을 false로 초기화 해야 한다.)
    bool isNodeAlreadySelected[10001];
    for (int i=0; i<10001; i++) isNodeAlreadySelected[i] = false; //초기화 전의 배열은 쓰레기 값이 들어있다.
    int numSelectedEdges = 0; //MST에 포함된 간선 개수
    long long sumWeight = 0; //MST에 포함된 간선의 가중치 합

    isNodeAlreadySelected[1] = true; //1번 노드를 MST에 추가하고-
    for (pair<int, int> e:adjList[1]) { //1번 노드에서 나가는 간선들을 MST에 추가한다.
        pq.push({e.first, 1, e.second});
    }

    while (numSelectedEdges<numNodes-1) { //MST에 (노드-1) 개 만큼의 간선이 들어가면 알고리즘이 종료된다.
        tuple<int, int, int> selectedEdge = pq.top();
        pq.pop();
        if (isNodeAlreadySelected[get<2>(selectedEdge)]) continue;
        //우선순위 큐에서 뽑은 [MST에 추가될 간선 후보]가 그 자격을 잃은 경우 그 간선은 버리고 다음 간선 후보를 뽑는다.
        sumWeight += get<0>(selectedEdge); //MST에 넣을 간선의 가중치를 더하고
        isNodeAlreadySelected[get<2>(selectedEdge)] = true; //MST에 넣을 간선의 도착지점 노드를 MST에 넣고
        numSelectedEdges++; //MST에 포함된 간선 수도 1 증가시킨다.
        for (pair<int, int> e:adjList[get<2>(selectedEdge)]) {
        //MST에 추가로 들어간 노드에서 출발하는 간선들을 우선순위 큐에 넣는다.
            pq.push({e.first, get<2>(selectedEdge), e.second});
        }
    }
    cout << sumWeight;
    return 0;
}
```