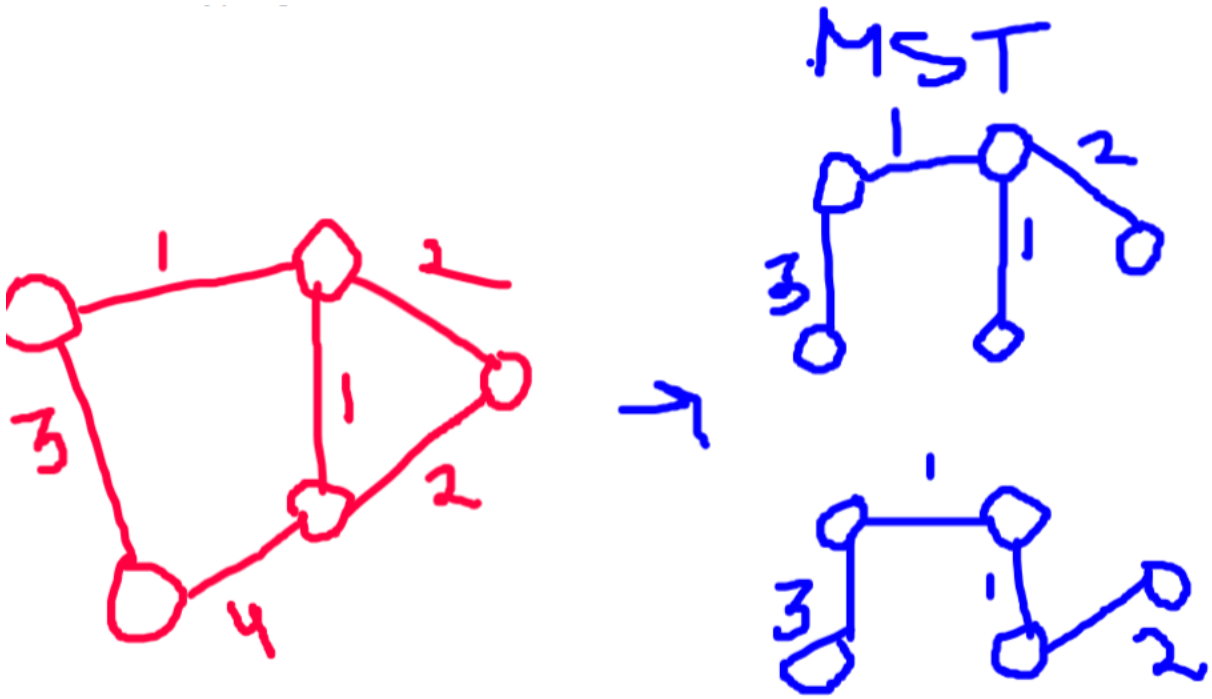
Minimum Spanning Tree

(Ref: https://www.hackerearth.com/practice/algorithms/graphs/minimum-spanning-tree/tutorial/)



Kruskal's Algorithm

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

using namespace std;

struct DSU{
    vector<int> par;
    vector<int> size;
    DSU(int s){
        par.resize(s);
        size.resize(s);
        for(int i=0;i<s;i++){
            par[i] = i;
            size[i] = 1;
        }
    }
    int find(int u){
        return (par[u]== u)?u : par[u] = find(par[u]);
```

```
    }
    bool add(int u, int v){
        u = find(u);
        v = find(v);
        if(u != v){
            if(size[u] < size[v]){
                swap(u, v);
            }
            size[u] += size[v];
            par[v] = u;
            return false;
        }
        return true;
    }
};
int main(){
    ios_base::sync_with_stdio(false); cin.tie(0); cout.tie(0);

    int n, m;
    cin>>n>>m;
    vector<vector<int>> edges;
    for(int i=0;i<m;i++){
        int u, v;
        cin>>u>>v;
        int w;
        cin>>w;
        u--; v--;
        edges.push_back({w, u, v});
    }
    sort(edges.begin(), edges.end());
    long long ans = 0;
    DSU d(n);
    for(auto e: edges){
        if(!d.add(e[1], e[2])){
            ans += e[0];
        }
    }
    cout<<ans<<"\n";
```

```
    return 0;
}
```

Prims Algorithm (using priority_queue)

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

using namespace std;

int main(){
    ios_base::sync_with_stdio(false); cin.tie(0); cout.tie(0);

    int n, m;
    cin>>n>>m;
    vector<vector<pair<int,int>>> g(n);
    for(int i=0;i<m;i++){
        int u, v;
        cin>>u>>v;
        int w;
        cin>>w;
        u--; v--;
        g[u].push_back({v, w});
        g[v].push_back({u, w});
    }
    priority_queue<pair<int,int>,vector<pair<int,int>>,
greater<pair<int,int>>> pq;
    const int inf = 1e9;
    vector<int> d(n, inf);
    vector<int> vis(n);
    pq.push({0, 0});
    long long ans = 0;
    while(pq.size()){
        pair<int,int> p = pq.top();
        pq.pop();
        if(vis[p.second]){
            continue;
        }
        vis[p.second] = true;
```

```
        ans += p.first;
        int node = p.second;
        for(auto e : g[node]){
            int v = e.first;
            int w= e.second;
            if(!vis[v] && d[v] > w){
                d[v] = w;
                pq.push({w, v});
            }
        }
    }
    cout<<ans<<"\n";
    return 0;
}
```

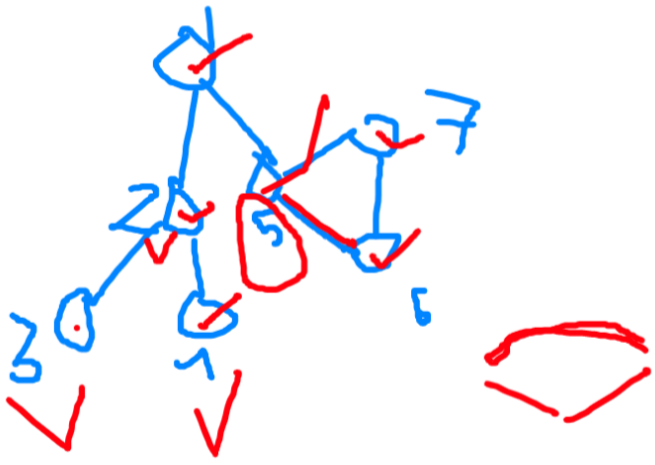
Using Set Ref: (https://usaco.guide/gold/mst?lang=cpp)
V vertices and E edges
Time complexity - Kruskal - E Log V(dsu add) + E Log E(sorting) = E log E
Time Complexity- Prims - E log E

Q. n Vertices undirected graph , with n edges such that graph is one connected component
for every i from 1 to n {ei edge connects ai , bi with wi weight}
find minimum xor spanning tree.

Let S be the xor of all edges weight. As there are n edges there will be only one cycle. We can only remove edges if it is a part of cycle.
Xor of all edges after removing only one edge with weight e_w is
(S xor e_w)
So find the cycle and for every edge in the cycle try removing it and minimize ans with current xor of remaining.

Pseudo code:

```
stack of pair st; // pair.first = node pair.second = weight of the edge
int ans; // final answer
int S; // Xor of all weights
bool dfs(int node, int from){
        vis[node] = true;
        for(auto [to, w]: g[node]){
                if(to== from) continue;
                if(!vis[to]){
                        st.push({to, w});
                        if(dfs(node, from){
                                return true;
                        }
                        st.pop();
                }else {
                        ans = min(ans, S^w);
                        while(st.top().first != to){
                                int e_w = st.top().second;
                                ans = min(ans, S^e_w);
                                st.pop();
                        }
                        return true;
                }
        }
        return false;
}
```



Optional:- Bridge Finding Algorithm:
Note: This is not aplicable for previous question with m edges.
https://cp-algorithms.com/graph/bridge-searching.html
https://codeforces.com/blog/entry/68138

MST+1 ABC: https://atcoder.jp/contests/abc235/tasks/abc235_e
GCD And MST: https://codeforces.com/problemset/problem/1513/D

CheckSum KickStart Round A 2021 :
https://codingcompetitions.withgoogle.com/kickstart/round/0000000000436140/000000000068c2c3

Bipartite Graph Check using DFS

Building Teams CSES: https://cses.fi/problemset/task/1668

https://codeforces.com/contest/1627/problem/D
5*2*5 , 5*2*3 , 5*3*5
5*2 , 5*3
5
1e6
cnt[a[i]] = 1
d = iterate(1e6-1 … 1)
g =0
j = d , 2*d , 3*d ,,...
cur += cnt[j]
if(cnt[j] == 1)
g = gcd(j, g)

```
if(cur >=2 && g== d ){
        cnt[d] = 1;
```

}

20 40
10

Topological Sort

(Ref: https://cp-algorithms.com/graph/topological-sort.html)

Course Schedule CSES: https://cses.fi/problemset/task/1679
Fox and Names: https://codeforces.com/problemset/problem/510/C
Longest Flight Route CSES: https://cses.fi/problemset/task/1680
Game Routes CSES: https://cses.fi/problemset/task/1681

Strongly Connected Component

(Ref: https://cp-algorithms.com/graph/strongly-connected-components.html)

https://www.hackerearth.com/practice/algorithms/graphs/strongly-connected-components/tutorial/
https://www.codechef.com/problems/MCO16405
Resources (Kosaraju's Algorithm and Tarjan's Algorithm):
https://usaco.guide/adv/SCC?lang=cpp