いろはちゃんコンテスト解説

Day2-D:楽しすぎる家庭菜園

@Pro_ktmr

注意

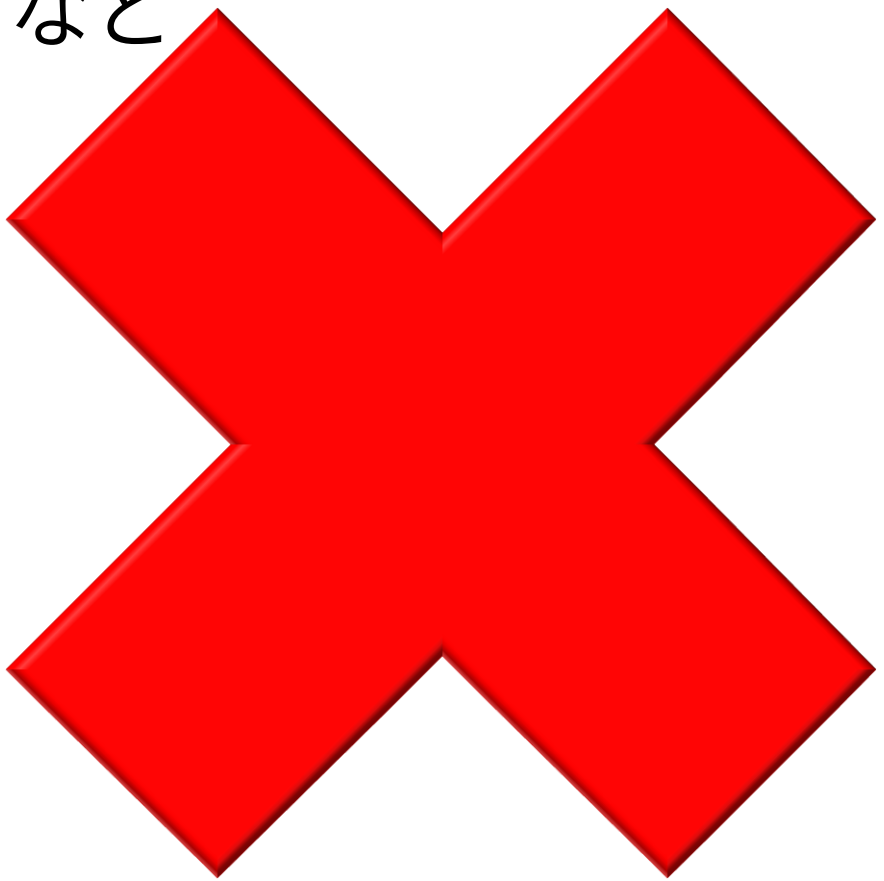
- 内容の正確性には十分注意していますが、間違った情報が含まれることもあります。何かお気づきのことがあればご連絡ください。
- このスライドのサンプルコードはC++で実装されています。

この問題の狙い

- フローが使えるか　など

この問題の狙い

- フローが使えるか　など



問題文が途中までかなりフローっぽい

この問題の狙い

- 最小全域木を求められるか　など

問題概要

- N頂点M辺の連結な無向グラフがある
- 辺にはバラバラのコストが定まっている

- 連結な状態を保って辺の本数を最小化
- 辺のコストの総和を最大化

- $N \leqq 200000$　$M \leqq 400000$

考察

- 連結な状態を保って辺の本数を最小化すると・・・木になる！
- いわゆる全域木を求める問題

考察

- 連結な状態を保って辺の本数を最小化すると・・・木になる！
- いわゆる全域木を求める問題

- 辺のコストを最大化・・・

考察

- 連結な状態を保って辺の本数を最小化すると・・・木になる！
- いわゆる全域木を求める問題

- 辺のコストを最大化・・・最小化の逆とも考えられる
- よって、最小全域木を求める問題！(のちょっと変なアレンジ)

余談

- どうして最小全域木という言葉はあるのに最大全域木という言葉は一般に使われないか
- 普通コストの和を最大化したければわざわざ木にする必要がないからだと思います、たぶん

最小全域木の求め方

- この解説ではクラスカル法を紹介する

- UnionFindを用意する(連結判定用)
- 辺をコストの降順でソートする
- 辺をコスト大きい方から見ていって、頂点A_iとB_iが非連結ならその辺を採用する
  連結ならその辺は採用しない

- こんなに簡単な貪欲法でOK(証明は検索してください)

解法① クラスカル法(逆順ver) O(M log M)

```
#include"bits/stdc++.h"
using namespace std;

struct UnionFind{
private:
        vector<int> parent;
        vector<int> rank;
public:
        void init(int N){
                parent.clear();
                rank.clear();
                for(int i=0; i<N; i++) parent.push_back(i);
                for(int i=0; i<N; i++) rank.push_back(0);
        }
        int root(int a){
                if(parent[a] == a) return a;
                return parent[a] = root(parent[a]);
        }
        void unite(int a, int b){
                int rootA = root(a);
                int rootB = root(b);
                if(rootA == rootB) return;
                if(rank[rootA] < rank[rootB]) parent[rootA] = rootB;
                else{
                        parent[rootB] = rootA;
                        if(rank[rootA] == rank[rootB]) rank[rootA]++;
                }
        }
        bool same(int a, int b){
                return root(a) == root(b);
        }
};
```

```
struct Edge{
        int a, b, i;
        long long c;
        bool operator<(const Edge& e) const {
                return (c < e.c);
        }
};

int N, M;
vector<Edge> v;
UnionFind uf;
vector<int> ans;
int main(){
        scanf("%d%d", &N, &M);
        v.resize(M);
        for(int i=0; i<M; i++){
                scanf("%d%d%d", &v[i].a, &v[i].b, &v[i].c);
                v[i].i = i+1;
        }
        sort(v.begin(), v.end());
        uf.init(N+1);
        for(int i=M-1; i>=0; i--){
                if(!uf.same(v[i].a, v[i].b)){
                        uf.unite(v[i].a, v[i].b);
                        ans.push_back(v[i].i);
                }
        }
        sort(ans.begin(), ans.end());
        for(int i=0; i<N-1; i++) printf("%d¥n", ans[i]);
}
```

Thank you for reading!