```cpp
class Chuliu_Edmonds {
private:
  struct edge_t {
    int id;
    ll cost;
    stack<int> included_stk;
    bool operator<(const edge_t & another) const {
      return cost > another.cost;
    }
  };
  // edges are directed to the node itself
  struct node {
    int overnode; bool done; bool fin; priority_queue<edge_t> edges; edge_t from;
  };
  vector<node> nodes;
  int n, root;
  stack<int> stk;
  bool no_mca;
  int topnode(int k) {
    int a = k;
    while (nodes[a].overnode != -1) {
      a = nodes[a].overnode;
    }
    if (k != a) nodes[k].overnode = a;
    return a;
  }
  void contract(int s) {
    int a = s;
    priority_queue<edge_t> new_from_edges;
    int cnt = 0;
    do {
      a = topnode(a);
      while (nodes[a].edges.size()) {
        edge_t edge = nodes[a].edges.top();
        nodes[a].edges.pop();
        if (edge.id == nodes[a].from.id) continue;
        edge.cost -= nodes[a].from.cost;
        edge.included_stk.push(a);
        new_from_edges.push(edge);
      }
      nodes[a].overnode = nodes.size();
      a = nodes[a].from.id;
    } while (a != s);
    nodes.push_back({ -1, false, false, new_from_edges,{} });
  }
  void unfold() {
    while (stk.size()) {
      int a = stk.top(); stk.pop();
      if (a >= n) {
        int b = nodes[a].from.included_stk.top();
        ll d = nodes[b].from.cost;
        nodes[b].from = nodes[a].from;
        nodes[b].from.cost += d;
        nodes[b].from.included_stk.pop();
      }
      else nodes[a].fin = true;
    }
  }
public:
  Chuliu_Edmonds(const vvi &lst, const vvll &cst, int start) {
    n = lst.size();
    nodes.resize(n);
    Loop(i, n) nodes[i] = { -1, false, false, priority_queue<edge_t>(),{} };
    Loop(i, n) {
      Loop(j, lst[i].size()) {
        nodes[lst[i][j]].edges.push({ i, cst[i][j], stack<int>() });
      }
    }
    root = start;
    no_mca = false;
    nodes[root].fin = nodes[root].done = true;
```

```
72      Loop(i, n) {
73        if (!nodes[i].fin) {
74          int a = i;
75          nodes[a].done = true;
76          stk.push(a);
77          do {
78            int b;
79            do {
80              if (nodes[a].edges.empty()) { no_mca = true; return; }
81              nodes[a].from = nodes[a].edges.top(); nodes[a].edges.pop();
82              b = nodes[a].from.id;
83            } while (topnode(a) == topnode(b));
84            if (nodes[b].fin) unfold();
85            else if (nodes[b].done) {
86              contract(b);
87              stk.push(nodes.size() - 1);
88              a = nodes.size() - 1;
89            }
90            else {
91              nodes[b].done = true;
92              stk.push(b);
93              a = b;
94            }
95          } while (stk.size());
96        }
97      }
98      return;
99    }
100   vector<P> get_tree_idpair() {
101     if (no_mca) return{};
102     vector<P> ret;
103     Loop(i, n) {
104       if (i != root) ret.push_back({ nodes[i].from.id, i });
105     }
106     return ret;
107   }
108   ll get_weight() {
109     if (no_mca) return -1;
110     ll ret = 0;
111     Loop(i, n) {
112       if (i != root) ret += nodes[i].from.cost;
113     }
114     return ret;
115   }
116 };
```