Solution 2

1、证明：当一张图的所有边权互不相同时，最小生成树唯一

假设存在2个MST T_1, T_2，记a为最小的在一个MST上出现，另一个MST上未出现的边，不妨假设a在在T_2中出现，在T_1中未出现。将a加入到T_1中会出现一个环，因为T_1是MST，所以a一定是环上最大的边，并且，环上一定存在一条边$b < a$不在T_2中出现，与a是最小的在一个MST上出现，另一个MST上未出现的边矛盾。

2、**Fredman and Tarjan**算法

(a) 写出**Fredman and Tarjan**算法的伪代码

```
int FredmanTarjan(vector<vector<edge>> g, int m, int n) {
    if (pow(2, 2 * m / n) >= n) return Prim(g, m, n);
    vector<int> id(n, -1);
    int idx = 0;
    int res = 0;
    for (int i = 0; i < n; ++i) {
        if (~id[i]) continue;
        __gnu_pbds::priority_queue<pii, greater<pii>, thin_heap_tag> Q;
        Q.push({i, 0});
        vector<int> cc{i};
        id[i] = idx++;
        while (Q.size()) {
            auto[d, u] = Q.top(); Q.pop();
            if (~id[u] && id[u] != id[i]) {
                for (int v: cc) id[v] = id[u];
                --idx;
                break;
            }
            if (id[v] == id[i]) continue;
            res += d;
            id[u] = id[i];
            cc.emplace_back(u);
            for (auto[v, k]: g[u]) {
                if (id[v] != id[i]) {
                    Q.push({k, v});
                    if (Q.size() >= pow(2, 2 * m / n)) {
                        break;
                    }
                }
            }
            if (Q.size() >= pow(2, 2 * m / n)) break;
        }
    }
    vector<vector<edge>> gg(idx);
    for (int i = 0; i < n; ++i)
        for (auto[j, k]: g[i])
            if (id[i] != id[j])
                gg[id[i]].emplace_back(id[j]);
    return d + FredmanTarjan(gg, m, idx);
}
```

(b) 证明**Fredman and Tarjan**的时间复杂度是$O(m\log^* n)$

每一轮操作的复杂度是$O(m+n\log K)$。每次操作得到的连通分量的度数$\sum_{v\in C} d_v \ge K$（如果是因为邻居数量的条件终止显然成立，如果是因为合并两个连通分量，则至少有一个满足度数$\ge K$）。于是连通分量的数量$l \le \frac{2m}{K}$。每轮取$K_i := 2^{\frac{2m}{n_i}}$，则每轮复杂度$O(m)$。
$K_i \le \frac{2m}{n_{i+1}} = \log K_{i+1} \Rightarrow K_{i+1} \ge 2^{K_i}$，$log^* n$轮后，$K_i \ge n$。所以时间复杂度是$O(m\log^* n)$。

3、**Boruvka**算法

(a) 写出**Boruvka**算法中，用$O(m)$的时间进行一轮缩点（contraction）的伪代码：算法的输入是一张图$G=(V,E)$，其中每个点最小的边已经被标为蓝色，输出一张图$G'=(V',E')$，使得每个节点$v_C \in V'$对应原图中由蓝色边产生的连通分量C。图中没有重边和自环，如果G中连通分量C和C'间有边，则G'中对应的边的权重为这些边的最小值$(v_C, v_{C'}) := \min_{\{x,y\}\in E: x\in C, y\in C'} w_{xy}$。

```
void merge(vector<edge> &colored_edges, unordered_map<edge, int> edges,
vector<int> &id, int n) {
    for (auto[u, v]: colored_edges) {
        if (u > v) swap(u, v);
        id[v] = u;
    }
    for (int i = 1; i <= n; ++i) id[v] = id[id[v]];
    unordered_map<edge, int> new_edges;
    for (auto[x, w]: edges) {
        auto[u, v] = x;
        if (id[u] != id[v]) {
            if (!new_edges.count({u, v})) new_edges[{u, v}] = w;
            else new_edges[{u, v}] = min(new_edges[{u, v}], w);
        }
    }
    edges = move(new_edges);
}
```

(b) **Boruvka**算法每轮会把节点的数量减少一个常数倍，但是边的数量呢？构造一张n个节点m条边的图，使得在经过$\Omega(\log n)$轮缩点后的图G_i边数依然是$\Omega(m)$，即便清除过重边和自环。

将n个点均分分成$\sqrt{n}$组，每一组内记编号为$1,2,\dots,\sqrt{n}$，假设$\sqrt{n}$为2的整次幂。组内i号点向$i+lowbit(i)$连一条长度为$lowbit(i)$的边（$lowbit(i) = \text{i在2进制下的最低位1} = i\& - i$）。组间每组的1号点向其他所有组的1号点连一条长度为∞的边。总边数$\Theta(n)$。前$\log(\sqrt{n})$轮，会先在组内进行合并，组间的$\Theta(n)$保持不变，总边数为$\Omega(n)$。

(c) 证明：运行$\log\log n$轮**Boruvka**算法后，使用斐波那契堆（Fibonacci Heap）实现的**Prim**算法求MST的时间复杂度是$O(m\log\log n)$。

前$\log\log n$轮，**Boruvka**算法每轮复杂度$O(m)$，总复杂度$O(m\log\log m)$。$\log\log n$轮后，点数为$O(n*(\frac{1}{2})^{\log\log n}) = O(\frac{n}{\log n})$，斐波那契堆实现的**Prim**算法复杂度为
$O(m+n'\log n') = O(m+\frac{n}{\log n}(\log(n)-\log\log n)) = O(m+n)$

4^*（**挑战问题**：可2-3位同学组队回答，但在提交的答案中需写明每位同学的**具体**贡献）、**Yao's** $O(m\log\log n)$的MST算法：（主要思想：在**Boruvka**算法中，每一轮我们都需要扫描所有的边，而这种重复的扫描应该尽力避免）

(a) 假设每个节点的邻接表已经按照边的权重升序排好。修改**Boruvka**算法使它的复杂度变为$O(m+n\log n)$

每次只检查每个节点最小的边，没用的边直接从领接表里删除。$O(\log n)$轮，检查次数最多$O(m+n\log n)$次。

(b) 给定一个长度为N的序列和一个参数k，给出一个$O(N\log k)$的算法将序列分为k个组$g_1, g_2, \cdots, g_k$，每个组的元素个数最多$\lceil \frac{N}{k} \rceil$，并且所有$g_i$中的元素均小于$g_{i+1}$中的元素$\forall 1 \le i < k$

先$O(n)$确认中位数$p_{k/2}$小于中位数的放左边，大于中位数的放右边，然后分别递归左右区间，直到确定$k-1$个pivots，根据pivots分组。每层复杂度总共$O(n)$，一共$\log k$层，总复杂度$O(n\log k)$。

(c) 假设每个节点的领接表已经按照(b)中的算法排好序。修改(a)的算法使它的复杂度变为$O(m + \frac{m}{k}\log n + n\log n)$

领接表被分为k组，每个节点每次暴力检查当前组的d_i/k个元素，检查完的组直接从领接表里删除。$O(\log n)$轮，检查次数最多$O(m + \frac{m}{k}\log n + n\log n)$次。

(d) 令$k = \log n$，证明：若$m \ge n\log n$，则(c)中的算法复杂度为$O(m\log\log n)$。并给出一种算法，使得对于任意m，都能达到$O(m\log\log n)$。（提示：你可能需要先运行几轮**Boruvka**算法）

$m \ge n\log n$，代入即可得证。

先运行$\log\log n$轮**Boruvka**算法，剩下的图节点数$O(n') = O(\frac{n}{\log n})$，再取$k = \log n'$，代入即可得证，证明类似第3题的(c)。