实验四 最小生成树的求解

实验内容

1.下载boruvka.py代码，阅读该代码并参照课件理解其过程（40 分）

•代码解释

2.补全boruvka算法代码，使其能够正常工作（30 分）

3.生成不同规模的 **有向有权**图，运行boruvka算法，记录并分析不同规模图的时间性能（30 分）

Ø实验目的

- 实验内容
- 实验步骤与结果
  - 代码思路解释（40 分）
  - 补全的代码（30 分）
  - 性能分析（30 分）
- 总结

Boruvka算法编码方式：

> 我们维护图中所有连通块，然后遍历所有的点和边，
>
> 找到每一个连通块和其他连通块相连的最小的一条边，
>
> 然后把连通块合并起来，重复这个操作，直到剩下一整个连通块，
>
> 最开始状态是每个点是一个单独的连通块。
>
> 复杂度是(n+m)longn，因为每次都会合并两个连通块，整个程序进行log次操作就会完成，每次操作的复杂度是n+m的。

代码(Code)

```
#include<bits/stdc++.h>
const int N = 200010;

std::vector<int> mn[2];
std::vector<std::pair<int,int>> edge[N];

struct Boruvka{
    int n;
    std::vector<int> fa;
    Boruvka(int _n):n(_n){
        fa.resize(n+1,0);
        for(int i=0;i<=n;++i) fa[i]=i;
    }
    int find(int x){return fa[x]==x?fa[x]:fa[x]=find(fa[x]);}
```

```
    bool join(int x,int y){
        int px=find(x),py=find(y);
        if(px!=py){
            fa[px]=py;
            return true;
        }
        return false;
    }
};

void solve(){
    auto clear=[&](int n)->void{
        for(int i=0;i<=n;++i) edge[i].clear();
    };
    clear(N-1);
    //========读入图
    int n,m;
    std::cout<<"请输入图的点数n和边数m:\n";
    std::cin >> n >> m;
    std::cout<<"请输入图的每一条边:\n";
    for(int i=1;i<=m;++i){
        int u,v,w;
        std::cin >> u >> v >> w;
        edge[u].push_back(std::make_pair(v,w));
        edge[v].push_back(std::make_pair(u,w));
    }
    //==================================
    Boruvka T(n+1);
    mn[0].resize(n+1,0x3f3f3f3f);mn[1].resize(n+1);

    int ans=0;//最小生成树的路径和
    while(true){
        mn[0].resize(n+1,0x3f3f3f3f);
        bool flag=0;
        for(int i=1;i<=n;++i){
            for(auto v:edge[i]){
                int ver=v.first, w=v.second;
                if(T.find(i)!=T.find(ver))
                    if(mn[0][T.find(i)]>w)
                        mn[0][T.find(i)] = w, mn[1][T.find(i)] = T.find(ver);
            }
        }
        for(int i=1;i<=n;++i){
            if(mn[0][i]!=mn[0][0]&&T.find(i)!=T.find(mn[1][i])){
                flag=true;
                ans += mn[0][i];
                T.join(i, mn[1][i]);
            }
        }
        // std::cout<<"flag:"<<flag<<std::endl;
        if(!flag) break;
    }
    for(int i=1;i<n;++i){
        if(T.find(i)!=T.find(i+1)){
            std::cout << "本图不连通,请重新输入!\n";
            return ;
        }
    }
```

```
    std::cout<<"最小生成树的路径和:"<<ans<<"\n";
}

signed main(){
    solve();
    return 0;
}
```

效率测试

对于一个5000个点5e5条边的图来说，经过测试，此算法效率较为优异：