6.并查集

并查集

AcWing

挑战程序设计竞赛

功能

1. 查询
2. 合并

优化

1. 路径压缩
2. 避免退化：合并时候rank(高度)小的指向rank(高度)大的(有路径压缩就不用了)

模板

```
//点分散时候用unordered_map<int, int>
int par[MAX_N]

//初始化
for(int i = 0; i < n; i++){
    par[i] = i;
}
//查询
int find(int x){
    if(par[x] == x) return x;
    par[x] = find(par[x]); //路径压缩
    return par[x];
}
//合并
void unite(int x, int y){
    par[find(y)] = find(x);
}
```

带size的模板

```
//每个点分散时候用unordered_map<int, int>
int par[MAX_N]
int size[MAX_N]

//初始化
for(int i = 0; i < n; i++){
    par[i] = i;
    //记录集合个数时候才用
    size[i] = 1;
}
//查询
int find(int x){
    if(par[x] == x) return x;
```

```
    par[x] = find(par[x]); //路径压缩
    return par[x];
}
//合并
void unite(int x, int y){
    //计算某个连通块内的数量
    //此处必须用parx,pary保存或者size计算在前，否则计算size时候，y的根节点已经被改变，加错了
    //而且size[x] += size[y] 和 y的根指向a合并，理清谁向谁合并

    //1.
    int parx = find(x), par = find(y);
    if(parx == pary) return
    par[pary] = parx;
    size[parx] += size[pary];

    //2.
    if(find(x) == find(y)) return;
    size[find(x)] += size[find(y)];
    par[find(y)] = find(x);
}
```

题目

- 990.等式方程的可满足性
- 剑指 Offer II 119. 最长连续序列（带size的并查集问题哦）

带权并查集

- 带权并查集-知乎

```
 //改xy数据类型，改更新路径方法

 //int[10000] p;
unordered_map<string, string> p; // p[x] 记录节点 x 的祖先节点
unordered_map<string, double> d; // d[x] 记录节点 x 到祖先节点的距离（即 x / root）

 string find(string x) {
    if(p== x) return x;
    else {
        string prePx = p[x];
        p= find(prePx);
        //更新路径
        d= d* d[prePx];
        return p[x];
    }
}

void merge(string x,string y, double w) {
    string px = find(x),py = find(y);
    if(px == py) return ;
    p[px] = py;
    //更新路径
    d[px] = d[y] * w / d[x];
}

double dist(string x,string y)
```

```
27 {
28     string px = find(x),py = find(y);
29     if(px != py) return -1;
30     //返回路径
31     else return d/ d[y];
32 }
```

- 剑指 Offer II 111. 计算除法