第十五周实验报告

沈家成

1137 唯一生成树

问题转化

题目中要求补全为完全图，同时最小生成树唯一。这是我们接触过的最小生成树算法的逆向问题，可以先正向思考。

使用 Kruskal 算法生成最小树时，一段回路中一定是选择最小的几条边，因而补全完全图时，所加边的权值，应该是首尾结点在最小生成树中单边最大权值加一，否则 Kruskal 算法就会选择这条新加入的边，与最小生成树唯一矛盾。

因此问题就转化为了，连接每两个结点的最小生成树路径中，最大的边权值是多少？

排序

如果在生成每条边时，都沿着路径找最大权值，是$O(|V^2|)$的时间复杂度，显然会超时，因此要考虑如何减少重复的运算。

既然每次都要找最大的权值，那么先把最小生成树中的权值进行排序，就可以节省很多后续的工作。选用快速排序，是$O(ElogE)$的时间复杂度，可以接受。

接下来就是要充分利用排序的便利。首先，可以想象每个点之间都是分离的。然后，从最小权值的边开始，一对一连接即可。

之后，再连接第二小权值的边，可能刚好是二对一连接，那么当前的权值肯定是最大的，这样就把排序的结果用上了。

接下来的问题就是，要连几条边？从集合的角度考虑，一方的两个点要和另一方的一个点连接，所以总共要连$2 \times 1 = 2$条边。再减去已经连的最小生成树路径，共有$2 \times 1 - 1 = 1$条边。

推广到一般情况，每次 p 对 q 连接，连接的边数是$pq - 1$，每条边的权值是在最小生成树中当前次序的权值v加一，因此需要累加的权值之和为$(pq - 1) \times (v + 1)$。

并查集

还有一个问题没有解决，那就是如何得知每个连通域有多少元素？涉及到集合的问题，一般使用并查集解决。稍作修改，并查集就可以储存每个集合的元素数。

```
void join(int x, int y) {
    int fx = find(x), fy = find(y);
    if (fx != fy) {
        ctr[fy] += ctr[fx];
        prev[fx] = fy;
    }
}
```

设置 ctr 数组表示每个集合的元素个数，在合并时，个数累加到根元素。以后，只要访问根元素，就可以知道有多少元素了。

1283 Mixture

问题本质

想要获得更高的危险系数，就要尽可能多地发生反应。有如下几种情况：

- 要是一种物质和其他物质都不发生反应，那么就可以直接忽略

- 要是一群物质之间，两两之间存在反应，那么第一次随便放入一种物质，之后放可以发生反应，再放可以与之前所有发生反应的，那么就可以保证每次危险系数都相乘，一定是最大的危险系数
- 要是有两群如上的物质，但是裙之间都不反应，那么完全可以把他们分开，只是第二群物质的起始危险系数变了

如果用结点表示物质，用边相连表示可以发生反应，那么问题的本质就变成了求有多少个连通分量。

如果只有一个连通分量，那么每次都可以发生反应，危险系数就是2^{n-1}

如果有两个连通分量，说明有一次无法发生反应，危险系数就是2^{n-2}

推广之，如果有 p 个连通分量，就有 p 次无法发生反应，危险系数为2^{n-p}

并查集

由于我们只关心连通分量的个数，因此使用并查集就足以，同时可以获得很高的效率。

并查集的查找以及合并操作不再多说，统计连通分量个数的代码如下：

```
int contourCount() {
    int ctr = 0;
    for (int i = 1; i <= size; ++i) {
        if (prev[i] == i)
            ++ctr;
    }
    return ctr;
}
```

利用根节点在并查集中的前继为其本身，统计根结点的个数，就获得了连通分量的个数。

大数

n 最大可以到 1000，但是 2^1000 显然是 long long 也不能表示的，因此还需要设计一个类来输出大数。

计算 2^n时，注意进位即可：

```
BigTwo (int power) {
    num[0] = 1;
    length = 1;
    int len;
    int carry;
    int tmp;
    for (int i = 0; i < power; ++i) {
        carry = 0;
        len = length;
        for (int j = 0; j < len; ++j) {
            tmp = num[j] * 2 + carry;
            num[j] = tmp % 10;
            carry = tmp / 10;
        }
        if (carry != 0) {
            num[length] = carry;
            ++length;
        }
    }
}
```

为了便于最后输出，要设置一个 length 变量表示数位，同时也可以减少运算量，避免高位0的乘法。