图

1.数连通分量(并查集)

PAT1013 Battle Over Cities

当删除其中一个顶点及其相关的边之后，计算出剩下的图的连通分量，那么增加的边就应该是求出的连通分量-1

> 法1:每次dfs前判断visit[i]==0;

```
void dfs(int s) {
    visit[s] = 1;
    for(int i = 1; i<=n; ++i)
        if(g[s][i]==1&&visit[i]==0)dfs(i);
}
...main:
for (i = 0; i < k; ++i) {
    sum = 0;cin >> m;
    fill(visit.begin(),visit.end(),0);
    visit[m] = 1;
    for (j = 1; j<=n; ++j)
        if (visit[j] == 0){dfs(j);sum++;}
    printf("%d\n", sum-1);
}
```

> 法2:数根结点father[v]==v;

```
while(K--){
    int v;int num=0;
    scanf("%d",&v);
    iota(father,father+N+1,0);//初始化并查集
    for(int i=1;i<N+1;++i)
        if(i!=v)for(int j:graph[i])
            if(j!=v)uni(i,j);
    for(int i=1;i<=N;++i)
        if(i!=v&&father[i]==i)
            ++num;
    printf("%d\n",num-1);
}
```

Leetcode947

在二维平面上将石头放置在一些整数坐标点上。每个坐标点上最多只能有一块石头。现在，一个**move**将会移除与网格上的另一块石头共享一列或一行的一个石头。

```

```

```
int removeStones(vector<vector<int>>& stones) {
    if(stones.size() <= 1) return 0;
    int res = stones.size(), len = stones.size();
    vector<int> p(len,-1);
    for(int i =0;i < len;i++){
        for(int j = i+1;j < len;j++){
            if(stones[j][0] == stones[i][0] || stones[j][1] == stones[i][1])
                u(p, i, j);
        }
    }
    for(auto e : p){
        if(e == -1) res--;
    }
    return res;
}
```

2.有关是对边dfs还是点dfs

PAT1034

> 给出多个人之间的通话长度，按照这些通话将他们分成若干个组，各个组的总权值是该组内所有通话长度之和，每个人的权值是其参与的所有通话长度之和。求组数和组内通话最长的

每个点至少有一个连线，故可以对边搜数量关系的运用:边权和=点权和/2

之后的操作--PAT2019春7-3 Telefraud Detection

> 电信诈骗判断嫌疑犯,若犯人之间通过话说明是一个团伙

```
for(i=1;i<=n;++i)
    if(sus[i]==1)
        for(j=i+1;j<=n;++j)
            if(sus[j]==1&&g[i][j]>0&&g[j][i]>0)uni(i,j);
w=1;
for(i=1;i<=n;++i){
    if(sus[i]==1){
        if(mp[findf(i)]==0){
            mp[findf(i)]=w;
            gang[w].push_back(i);
            w++;
        }//将联通分量转换为gang
        else gang[mp[findf(i)]].push_back(i);
    }
}
```

PAT1021

> 给定**N**个结点和**N-1**条边，问能否构成一棵树，如果能，则输出作为树的根节点时使得整棵树深度最大的结点，如果不能，输出这个图中有几个连通分量。

能否构成树，要么有>1个连通分量,要么有环

3.Hamiltonian Cycle

> **(1)**是否是**N+1**个点。 **(2)**除起点外，每个点是否只出现了**1**次 **(3)**经过的边是否存在 **(4)**起点是否等于终点

注意剔除重复点时要用set(往往隐含)

```
void check(int index) {
    int sum = 0, cnt, flag = 1;
    scanf("%d", &cnt);
    set<int> s;
    vector<int> v(cnt);
    for (int i = 0; i < cnt; i++) {
        scanf("%d", &v[i]);
        s.insert(v[i]);
    }
    for (int i = 0; i < cnt - 1; i++) {
        if(e[v[i]][v[i+1]] == 0) flag = 0;//3
        sum += e[v[i]][v[i+1]];
    }
    if (flag == 0)
    printf("Path %d: NA (Not a TS cycle)\n", index);
    else if(v[0]!=v[cnt-1]||s.size()!=n)//4,2
    printf("Path %d: %d (Not a TS cycle)\n", index, sum);
    else if(cnt != n + 1)printf("Path %d: %d (TS cycle)\n", index, sum);//1
    else printf("Path %d: %d (TS simple cycle)\n", index, sum);
}
```

4.dfs结构总结

```
void dfs(int x){
    temp.push_back(x);
    if (pre[x][0] == -1){
        ......
    }
    for(int i=0;i<pre[x].size();i++){
        dfs(pre[x][i]);
    }
    temp.pop_back();
}
```