

2020牛客国庆集训派对

Day 1

[比赛链接](#)

I Saba1000kg

题意

给定一张图 q 个询问，每次询问只考虑图中 s_i 个点构成的连通块数。数据保证 $\sum s_i \leq 10^5$

题解

考虑分块。当 $s_i \leq k$ 时暴力 $O(s_i^2)$ 枚举所有点对同时维护并查集 $s_i \geq k$ 时暴力 $O(m)$ 枚举所有边同时维护并查集。

从均摊复杂度考虑，消耗每点 $\sum s_i$ 的复杂度为 $O(\max(\frac{s_i^2}{s_i} (s_i \leq k), \frac{m}{s_i} (s_i \geq k)) \log s_i) = O(\max(k, \frac{m}{k}) \log k)$

取 $k = O(\sqrt{m})$ 时总时间复杂度为 $O(\sum s_i \sqrt{m} \log m)$

```
const int MAXN=1e5+5;
set<int>g[MAXN];
struct Edge{
    int u,v;
}edge[MAXN];
int p[MAXN],b[MAXN];
bool vis[MAXN];
int Find(int x){return x==p[x]?x:p[x]=Find(p[x]);}
void Merge(int x,int y){
    int xx=Find(x),yy=Find(y);
    if(xx!=yy)
        p[xx]=yy;
}
int main()
{
    int n=read_int(),m=read_int(),q=read_int(),limt=sqrt(m+10);
    _for(i,0,m){
        edge[i].u=read_int(),edge[i].v=read_int();
        g[edge[i].u].insert(edge[i].v);
        g[edge[i].v].insert(edge[i].u);
    }
    while(q--){
```

```
int sz=read_int(),ans=0;
_for(i,0,sz){
    b[i]=read_int();
    vis[b[i]]=true;
    p[b[i]]=b[i];
}
if(sz<limt){
    _for(i,0,sz)
        _for(j,i,sz){
            if(g[b[i]].count(b[j]))
                Merge(b[i],b[j]);
        }
}
else{
    _for(i,0,m){
        if(vis[edge[i].u]&&vis[edge[i].v])
            Merge(edge[i].u,edge[i].v);
    }
}
_for(i,0,sz){
    ans+=(Find(b[i])==b[i]);
    vis[b[i]]=false;
}
enter(ans);
}
return 0;
}
```

Day 4

[比赛链接](#)

B Arithmetic Progressions

题意

给定一个不可重集，求最大子集满足子集中数构成等差数列。

题解

先将不可重集排序得到序列 A 然后令 $\text{dp}(i,j)$ 表示等差数列最后一项为 a_j 且倒数第二项为 a_i 时的等差数列的长度。

考虑枚举 i 同时双指针 $k \leq j$ 查找满足 $a_k + a_j = a_i$ 的数，有状态转移 $\text{dp}(i,j) = \text{dp}(k,i) + 1$ 时间复杂度 $O(n^2)$

```
const int MAXN=5e3+5;
int a[MAXN],dp[MAXN][MAXN];
int main()
{
    int n=read_int();
    _for(i,0,n)a[i]=read_int();
    sort(a,a+n);
    _for(i,0,n)_for(j,i+1,n)dp[i][j]=2;
    int ans=2;
    _for(i,0,n){
        int pos1=i-1,pos2=i+1;
        while(pos1>=0&&pos2<n){
            if(a[pos1]+a[pos2]<(a[i]<<1))pos2++;
            else if(a[pos1]+a[pos2]>(a[i]<<1))pos1--;
            else{
                dp[i][pos2]=max(dp[i][pos2],dp[pos1][i]+1);
                ans=max(ans,dp[i][pos2]);
                pos1--;pos2++;
            }
        }
    }
    enter(ans);
    return 0;
}
```

From: <https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link: https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:contest:2020%E7%89%9B%E5%AE%A2%E5%9B%BD%E5%BA%86%E9%86%E8%AE%AD%E6%B4%BE%E5%AF%B9&rev=1601811816

Last update: 2020/10/04 19:43