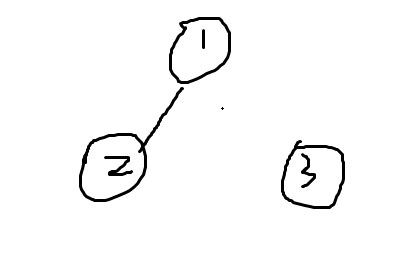
思路：对于入度最多的点即边最多的点，首先找到边最多的点，ans=边数。然后利用并查集求出区域块的个数s，对于除自己所在的区域块可以直接连一条边，ans=ans+s-1,然后再连一个点的话就需要消一条边以防止有环，ans=ans+k，还需要判断其不能大于n-1.

因为是不能有环的，所以n个点最多n-1条边

也就是说点的最大度数就是n-1

给你的m不一定是n-1，也就是说可能有

这种情况

M条边，可以看作

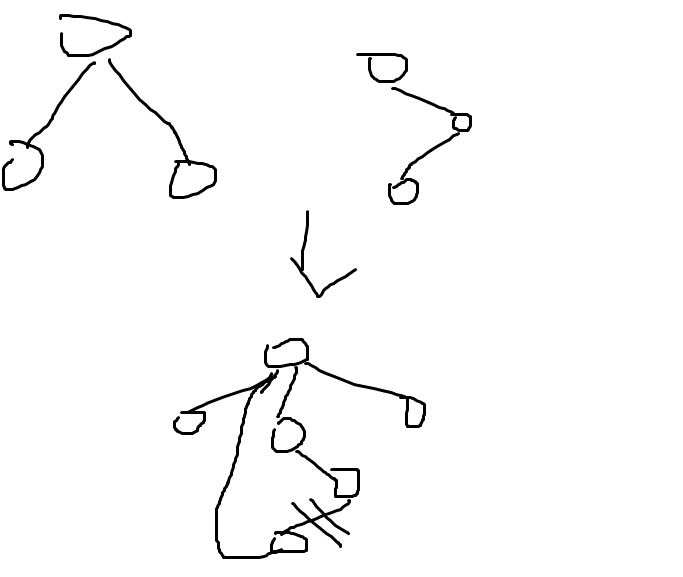
图中的m条边可以看作m个点与顶点相连了

总边数m-最大度数s就是除了和该点相连的边的其他边，该点和这些点相连会构成环

找有多少个联通块也就是有多少个图

我们可以把联通块都连一条边接到度数最大的点上

因为k可以删边嘛，而删了边之后只要不构成环就可以无限次连



像这样，将原本连在子联通块上的点删掉直接连到最大度的点上

最后要判断最大超过n-1了没有

因为你这么算有个缺陷，并没有看k的大小，万一k比m还大呢，所以最后要判断最大超过n-1了没有

#include <cstdio>

#include <cstring>

#include<iostream>

#include<algorithm>

using namespace std;

int fa[200010];

int degree[200010];

int Find(int x)

{

if(fa[x]==x)return x;

return fa[x]=Find(fa[x]);//是否需要带路径压缩还要考虑一下,有时带了会超时

//return Find(f[x]); //不路径压缩版

}

void Merge(int x,int y)//不按秩合并

{

x=Find(x);

y=Find(y);

if(x!=y)

{

fa[y]=x;

}

}

int main()

{

int T,n,k,m,x,y;

scanf("%d",&T);

while(T--)

{

scanf("%d%d%d",&n,&m,&k);

memset(fa,0,sizeof(fa));

memset(degree,0,sizeof(degree));

for(int i=1;i<=n;i++)

{

fa[i]=i;

}

for(int i=0;i<m;i++)

{

scanf("%d%d",&x,&y);

Merge (x,y);

degree[x]++;degree[y]++;

}

int num(0);

for(int i=0;i<n;i++)

if(fa[i]==i)

num++;

int cnt(0);

for(int i=0;i<n;i++)

cnt=max(cnt,degree[i]);

cnt+=num-1;

cnt+=k;

if(cnt>n-1)cnt=n-1;

printf("%d\n",cnt);

}

return 0;

}