题意就是问你能否精确的删除m个数，使得数列中的众数只有一个并且保证值最大化。

//先二分可能出现的最大次数t，然后再从出现次数为t的数中选最大的

#include <bits/stdc++.h>

#include<cstring>

using namespace std;

#define ll long long

#define INF 1000000007

set<int> \_set;

map<int,int> \_map;

set<int>::iterator its;

map<int,int>::iterator itm;

int n,m;

bool pd(int x)//我们在二分的时候就只管判断这个次数可不可以，而先不用管出现次数为这个次数的数的大小

{

ll ans(0);

bool flag(0);

for(itm=\_map.begin();itm!=\_map.end();itm++)

{

if(itm->second==x)//第一个出现次数为x的数我们就保留它x的出现次数，之后在遇到出现次数为x的数就把它砍1，这样求出让x是最大次数要砍多少个数

if(flag==0)

{

flag=1;

}

else

ans+=1;

if(itm->second >x)

ans+=itm->second -x+1;

}

if(ans>m)return 0;

return 1;

}

int main()

{

// freopen("input.txt","r",stdin);

int T,maxx;

ll x;

cin>>T;

while(T--)

{

cin>>n>>m;

\_set.clear();

\_map.clear();

maxx=0;//存的是最大出现次数

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

{

cin>>x;

\_set.insert(x);

\_map[x]++;

maxx=max(maxx,\_map[x]);

}

//二分可能的出现次数

//如果出现次数是这个，那么其他大于这个出现次数的必须要砍

int l=1,r=maxx;

while(l<=r)

{

int mid=(l+r)>>1;

if(pd(mid))

r=mid-1;

else l=mid+1;

}

int ans=0;

for(its=\_set.begin();its!=\_set.end();its++)

if(\_map[\*its]>=l)

ans=max(ans,\*its);

if(ans==0)ans=-1;

cout<<ans<<"\n";

}

return 0;

}

法二：

题意就是问你能否精确的删除mm个数，使得数列中的众数只有一个并且保证值最大化。

如果数列中众数不止一个，那么我们去考虑对于出现次数相同的数，最少删除几个数可以使得它变为众数，我们设这个最少的删除数为minDeleteminDelete，如果此时minDelete<=mminDelete<=m，那么就满足条件，剩下的m−minDeletem−minDelete个数从剩余的数列里随便删即可。

#include <bits/stdc++.h>

#define ll long long

#define INF 0x3f3f3f3f

#define fi first

#define se second

using namespace std;

typedef pair<int, int> PAIR;

bool cmp\_by\_value(const PAIR& lhs, const PAIR& rhs) {

return lhs.second > rhs.second;

}

map<int,int> \_map;

map<int,int>::iterator it;

int main()

{

//freopen("input.txt","r",stdin);

int T,n,m,x;

scanf("%d",&T);

while(T--)

{

\_map.clear();

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

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

{

scanf("%d",&x);

\_map[x]++;

}

vector<PAIR> v(\_map.begin(),\_map.end());

sort(v.begin(),v.end(),cmp\_by\_value);//按照出现次数由大到小排序

int k=0,sum=0,mindel;

//k是目前为止出现次数大于等于i的数的个数

int ans(-1);

for(int i=0;i<v.size();i++)

{

while(k<v.size() && v[k].second==v[i].second)

sum+=v[k++].second;

mindel=sum-k\*(v[i].second-1)-1;//使得v[i].v变成唯一的众数需要删除的数的最小个数

//v[i].second-1是出现次数大于等于k的那些数的出现次数最少应该减少的v[i].second-1

//-1是去掉它自己

if(mindel<=m)

ans=max(ans,v[i].first);

}

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

}

return 0;

}