AGGRCOW - Aggressive cows

#binary-search

Farmer John has built a new long barn, with N (2 <= N <= 100,000) stalls. The stalls are located along a straight line at positions x1,...,xN (0 <= xi <= 1,000,000,000).

His C (2 <= C <= N) cows don't like this barn layout and become aggressive towards each other once put into a stall. To prevent the cows from hurting each other, FJ wants to assign the cows to the stalls, such that the minimum distance between any two of them is as large as possible. What is the largest minimum distance?

Input

t – the number of test cases, then *t* test cases follows.
* Line 1: Two space-separated integers: N and C
* Lines 2..N+1: Line i+1 contains an integer stall location, xi

Output

For each test case output one integer: the largest minimum distance.

Example

Input:

```
1

5 3

1

2

8

4

9
```

Output:

```
3
```

Output details:

FJ can put his 3 cows in the stalls at positions 1, 4 and 8,
resulting in a minimum distance of 3.

```cpp
#include<bits/stdc++.h>
#define ll long long int
#define maxm 1e5+2
#define f(i, in, n) for(long long int i=in; i<n; i++)
using namespace std;

vector<ll> xor_of(300001);
vector<ll> seg_tree(maxm);

// Building a Segment Tree
ll buildSegmentTree(vector<ll> v, ll i, ll l, ll r) {
    if (l==r) {
        seg_tree[i]=v[l];
        return v[l];
    }
    ll mid=(l+r)/2;
    seg_tree[i]=buildSegmentTree(v, 2*i+1, l, mid)+buildSegmentTree(v, 2*i+2,
mid+1, r);
    return seg_tree[i];
}

// To get sum from Range Query
ll getSum(ll index, ll st, ll en, ll l, ll r) {
    if (st>r or en<l) return 0;
    if (l<=st and en<=r) return seg_tree[index];
    ll mid=(st+en)/2;
    return getSum(2*index+1, st, mid, l, r)+getSum(2*index+2, mid+1, en, l, r)
;
}

// Update Queries
void updateRange(ll index, ll pos, ll diff, ll st, ll en) {
    if (st>pos or en<pos) return;
    seg_tree[index]+=diff;
    if (st!=en) {
        ll mid=(st+en)/2;
        updateRange(2*index+1, pos, diff, st, mid);
        updateRange(2*index+2, pos, diff, mid+1, en);
    }
}

int power(long long x, unsigned int y, int p) {
    int res = 1;     // Initialize result

    x = x % p; // Update x if it is more than or
               // equal to p

    if (x == 0) return 0; // In case x is divisible by p;
    while (y > 0)
    {
        // If y is odd, multiply x with result
        if (y & 1)
            res = (res*x) % p;

        // y must be even now
        y = y>>1; // y = y/2
        x = (x*x) % p;
    }
    return res;
}
class Point {
public:
    int x, y;
    Point(int a=0, int b=0):x(a),y(b) {}
};

//utility function to find GCD of two numbers
// GCD of a and b
int gcd(int a, int b)
{
    if (b == 0)
       return a;
    return gcd(b, a%b);
}

// Finds the no. of Integral points between
// two given points.
int getBoundaryCount(Point p,Point q)
{
    // Check if line parallel to axes
    if (p.x==q.x)
        return abs(p.y - q.y) - 1;
    if (p.y == q.y)
        return abs(p.x - q.x) - 1;

    return gcd(abs(p.x-q.x), abs(p.y-q.y)) - 1;
}

// Returns count of points inside the triangle
int getInternalCount(Point &p, Point &q, Point &r)
{
    // 3 extra integer points for the vertices
    int BoundaryPoints = getBoundaryCount(p, q) +
                         getBoundaryCount(p, r) +
                         getBoundaryCount(q, r) + 3;
    // Calculate 2*A for the triangle
    int doubleArea = abs(p.x*(q.y - r.y) + q.x*(r.y - p.y)  +
                         r.x*(p.y - q.y));

    // Use Pick's theorem to calculate the no. of Interior points
    return (doubleArea - BoundaryPoints + 2)/2;
}

vector<long long> nextSmallerElement(long long arr[], int n) {
        // Your code here
    stack<long long> s;
    vector<long long> ans1(n);
    for (int i=n-1; i>=0; i--) {
        while (!s.empty() and arr[i]<=arr[s.top()]) {
            s.pop();
        }
        if (s.empty()) ans1[i]=n;
        else ans1[i]=s.top();
        s.push(i);
    }
    return ans1;
}

vector<long long> nextSmallerElementfromBegin(long long arr[], int n) {
    vector<long long> ans2(n);
    stack<long long> s1;
    for (int i=0; i<n; i++) {
        while (!s1.empty() and arr[i]<=arr[s1.top()]) {
            s1.pop();
        }
        if (s1.empty()) ans2[i]=0;
        else ans2[i]=s1.top();
        s1.push(i);
    }
    return ans2;
}

    long long getMaxArea(long long arr[], int n) {
        // Your code here
        vector<long long> aux1=nextSmallerElement(arr, n);
        vector<long long> aux2=nextSmallerElementfromBegin(arr, n);
        long long ans=0, area;
        for (int i=0; i<n-1; i++) {
            if (arr[i]>arr[i+1]) {
                long long width=aux1[i+1]-(i+1);
                area=arr[i+1]*(width+1);
                ans=max(ans, area);
```

```cpp
            }
            else {
                long long width=aux1[i]-i;
                area=arr[i]*width;
                ans=max(ans, area);
            }
        }
        ans=max(ans, arr[n-1]);
        ans=max(ans, arr[0]);
        for (int i=1; i<n; i++) {
            if (arr[i]<arr[i-1]) {
                long long width=i-aux2[i]+1;
                area=arr[i]*width;
                ans=max(ans, area);
            }
            else {
                long long width=i-aux2[i-1];
                area=arr[i-1]*(width+1);
                ans=max(ans, area);
            }
        }
        return ans;
    }

/*string decToBinary(int n) {
    // Size of an integer is assumed to be 32 bits
    string ans="";
    for (int i = 31; i >= 0; i--) {
        int k = n >> i;
        if (k & 1)
            ans+='1';
        else
            ans+='0';
    }
    return ans;
}*/

ll minCost(vector<ll> v, ll k, ll x) {
    sort(v.begin(), v.end());
    ll ans=0;
    while (v.size()>1) {
        ll n=v.size();
        if (k>0 and x<=v[n-1]+v[n-2]) {
            ans+=x;
            v.pop_back();
            v.pop_back();
            k--;
        }
        else {
            ans+=(v[n-1]+v[n-2]);
            v.pop_back();
            v.pop_back();
        }
    }
    if (v.size()==1) ans+=v[v.size()-1];
    return ans;
}

ll fun(string s, ll k) {
    unordered_map<char, ll> m;
    f(i, 0, s.length()) {
        int x=s[i]-'0';
        if (x==1 or (x!=2 and x%2==0) or x==9) return (ll)x;
        m[s[i]]++;
    }
    for (auto it : m) {
        if (it.second>1) {
            ll u=it.first-'0';
            return u+(10*u);
        }
    }
    f(i, 0, s.length()) {
        f(j, i+1, s.length()) {
            ll y=10*(s[i]-'0')+(s[j]-'0');
            if (y%2==0 or y%3==0 or y%5==0 or y%7==0) return y;
        }
    }
}

unsigned ll num(ll s, ll x) {
    ll z=(s-x)/2;
    ll ct=0;
    if (z<0 or (s-x)%2!=0) return 0;
    for (ll i=31; i>=0; i--) {
        if ((x&(1<<i))!=0 and (z&(1<<i))!=0) return 0;
        if ((x&(1<<i))!=0 and (z&(1<<i))==0) ct++;
    }
    if (z==0) ct--;
    ll ans=(1<<ct);
    return ans;
}

void seiv() {
    xor_of[0]=0;
    f(i, 1, xor_of.size()) {
        xor_of[i]=xor_of[i-1]^i;
    }
}

bool cows_placing(vector<ll> v, ll dis, ll cows) {
    ll last_pos=-1;
    f(i, 0, v.size()) {
        if (last_pos==-1 or (v[i]-last_pos)>=dis) {
            cows--;
            last_pos=v[i];
        }
        if (cows==0) break;
    }
    return cows==0;
}

int main() {
    int q;
    cin>>q;
    while (q--) {
       ll n, c;
       cin>>n>>c;
       vector<ll> v(n);
       f(i, 0, n) cin>>v[i];
       sort(v.begin(), v.end());
       ll l=0, h=1e9;
       while (h-l>1) {
           ll mid=(l+h)/2;
           if (cows_placing(v, mid, c)) l=mid;
           else h=mid-1;
       }
       if (cows_placing(v, h, c)) cout<<h<<endl;
       else cout<<l<<endl;
    }
    return 0;
}
```