A:
注意到 x, y 范围为 1 ~ 1e9，内存只给了 256M，显然直接开数组记录次数 是会 爆内存的.
然后发现 n 最大为 1e6，那么显然复杂度 n^2 级别的 每次都搜一遍数组 是会 超时 的.
学习了 map，应当想到，实现本题的加操作，和查询操作，都是 log 复杂度，空间也十分优秀，足以通过.

```
#include <bits/stdc++.h>

using namespace std;

int main() {

	map<int, int> q;
	int n;
	cin >> n;
	while (n--) {
		int x, y;
		cin >> x >> y;
		q[x]++;
		// 用 endl 的 活该超时阿
		cout << q[y] << '\n';
	}
	return 0;
}
```

B:
使用 multiset，并学会迭代器的使用，这题就很显然了吧.
第二种方法是，每次只考虑前 11 大小的数，不断更新，好像看到有同学这么写了，想法很好

```
#include <bits/stdc++.h>

using namespace std;

int main() {
	int n;
	cin >> n;
	multiset<int> q;
	for (int i = 0; i < n; i++) {
		int x;
		cin >> x;
		q.insert(x);
	}
	int m;
```

```
    cin >> m;
    while (m--) {
        int x;
        cin >> x;
        q.insert(x);
        auto t = q.begin();
        for (int i = 0; i < 9; i++) {
            t = next(t);
        }
        cout << *t << '\n';
    }
    return 0;
}
```

C:
基础 DP，考虑到第 i 个数的时候直接往回找最优的转移路线即可，n 最大为 1000，n^2 复杂度足以通过

```
#include <bits/stdc++.h>

using namespace std;

int main() {
    int n, k;
    cin >> n >> k;
    vector<int> d(n), a(n);
    for (int i = 0; i < n; i++) {
        cin >> a[i];
        if (i) d[i] = 1e9;
        for (int j = 0; j < i; j++) {
            if (abs(a[i] - a[j]) <= k) d[i] = min(d[i], d[j] + 1);
        }
    }
    if (d[n - 1] > 1e8) d[n - 1] = -1;
    cout << d[n - 1] << '\n';
    return 0;
}
```

D:
本题考察并查集的基本操作。操作 1 是合并操作，操作 2 是查询操作，操作 3 是查询当前有多少集合，对于操作 3，显然每当成功合并一次集合，集合数就会减 1，用一个变量记录即可。时间复杂度 O(m*logn)

```
#include<bits/stdc++.h>
```

```
using namespace std;
const int N=1e5+5;
int n,m;
struct DSU
{
	int gp,fa[N],sz[N];
	int find(int x)
	{
		while(x!=fa[x]) x=fa[x]=fa[fa[x]];
		return x;
	}
	bool con(int x,int y)
	{
		return (find(x)==find(y));
	}
	bool merge(int x,int y)
	{
		if(con(x,y)) return 0;
		x=find(x),y=find(y);
		if(sz[x]>sz[y]) swap(x,y);
		sz[y]+=sz[x];fa[x]=y;gp--;
		return 1;
	}
	int qry(int x)
	{
		return sz[find(x)];
	}
	void init(int n)
	{
		gp=n;
		for(int i=1;i<=n;i++)
		{
			sz[i]=1;
			fa[i]=i;
		}
	}
}dd;
main()
{
	ios::sync_with_stdio(false);
	cin.tie(nullptr);
	cin>>n>>m;
	dd.init(n);
	for(int i=1;i<=m;i++)
```

```
    {
        int op,x,y;
        cin>>op;
        if(op==1)
        {
            cin>>x>>y;
            dd.merge(x,y);
        }
        else if(op==2)
        {
            cin>>x;
            cout<<dd.qry(x)<<'\n';
        }
        else cout<<dd.gp<<'\n';
    }
}
```

推荐学习并查集的文章：https://zhuanlan.zhihu.com/p/93647900

E:

本题正解是线段树。对于操作 1，是一个单点修改操作，而对于操作 2，考虑到所维护的颜色数量非常少，对于查询的区间之间可以用桶暴力维护，时间复杂度为 O(20*m*logn)，题解用了 bitset 优化，复杂度在原来基础上再除 32。

```
#include<bits/stdc++.h>
using namespace std;
#define int long long
const int N=1e5+5;
const int ll=0,rr=1e9;
int dp[N],n,m;
int a[N],b[N];
int root=1,tot=1;
struct node
{
    int ls,rs;
    bitset<25>a;
}nd[N<<2];
struct T
{
    void pushup(int rt)
    {
        int ls=nd[rt].ls,rs=nd[rt].rs;
        nd[rt].a=nd[ls].a|nd[rs].a;
    }
    void upd(int pos,int v,int &rt=root,int l=ll,int r=rr)
    {
```

```
        if(!rt) rt=++tot;
        if(l==r)
        {
            nd[rt].a.reset();
            nd[rt].a[v]=1;
            return;
        }
        int mid=(l+r)>>1;
        if(pos<=mid) upd(pos,v,nd[rt].ls,l,mid);
        if(pos>mid) upd(pos,v,nd[rt].rs,mid+1,r);
        pushup(rt);
    }
    bitset<25> qry(int L,int R,int rt=root,int l=ll,int r=rr)
    {
        if(l>=L&&r<=R)
        {
            return nd[rt].a;
        }
        int mid=(l+r)>>1;
        bitset<25>res;
        if(L<=mid) res|=qry(L,R,nd[rt].ls,l,mid);
        if(R>mid) res|=qry(L,R,nd[rt].rs,mid+1,r);
        return res;
    }
}tr;
main()
{
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cin>>n>>m;
    for(int i=1;i<=n;i++)
    {
        int x;cin>>x;
        tr.upd(i,x);
    }
    for(int i=1;i<=m;i++)
    {
        int op,l,r,x;
        cin>>op;
        if(op==1)
        {
            cin>>l>>x;
            tr.upd(l,x);
        }
```

```
            else
            {
                cin>>l>>r;
                cout<<tr.qry(l,r).count()<<'\n';
            }
        }
}
```

推荐学习线段树的文章：https://zhuanlan.zhihu.com/p/106118909