```cpp
// https://toph.co/p/battle-of-endor
#include<bits/stdc++.h>
using namespace std;
const int N  = 10005;
const int RT = 100;
int n,k,a[N],last[N],lft[N],rgt[N],ans[100005];
int validGap[N],cnt[N],strength[N];
bool active[N];
set<int>res;
struct query{
    int l,r,id;
    bool operator < (const query& q) const {
        if(l / RT != q.l / RT) return l < q.l;
        return ((l / RT) & 1) ? r > q.r : r < q.r;
    }
};

void Add(int idx){
    int v = a[idx];
    if(active[v]){
        strength[cnt[v]]--;
        if(strength[cnt[v]]==0) res.erase(cnt[v]);

        cnt[v]++;

        strength[cnt[v]]++;
        if(strength[cnt[v]]==1) res.insert(cnt[v]);
    }else{
        cnt[v]++;
        if(validGap[v]>0){
            active[v] = true;
            strength[cnt[v]]++;
            if(strength[cnt[v]]==1) res.insert(cnt[v]);
        }
    }
}
void Remove(int idx){
    int v = a[idx];
    if(active[v]){
        strength[cnt[v]]--;
        if(strength[cnt[v]]==0) res.erase(cnt[v]);

        cnt[v]--;

        if(validGap[v]>0){
            strength[cnt[v]]++;
            if(strength[cnt[v]]==1) res.insert(cnt[v]);
        }else{
            active[v] = false;
        }
    }else{
        cnt[v]--;
    }
}
int main(){
    int tt;scanf("%d",&tt);
    for(int ks=1; ks<=tt; ks++){
        scanf("%d%d",&n,&k);
        for(int i=1; i<=n; i++)scanf("%d",&a[i]);

```

```
61          for(int i=0; i<=n; i++)last[i]=0;
62          for(int i=1; i<=n; i++){
63              lft[i] = last[a[i]];
64              last[a[i]] = i;
65          }
66
67          for(int i=0; i<=n; i++)last[i]=n+1;
68          for(int i=n; i>=1; i--){
69              rgt[i] = last[a[i]];
70              last[a[i]] = i;
71          }
72
73          int q; scanf("%d",&q);
74          //RT = sqrt((n*n)/q);
75          vector<query>queries;
76          for(int i=1; i<=q; i++){
77              int l,r; scanf("%d%d",&l,&r);
78              if(l>r)swap(l,r);
79              queries.push_back({l,r,i});
80          }
81          sort(queries.begin(), queries.end());
82
83          res.clear();
84          for (int i=0; i<=n; i++){
85              cnt[i] = 0;
86              active[i] = false;
87              validGap[i] = 0;
88              strength[i] = 0;
89          }
90
91          int l=1,r=0;
92          for(query q : queries){
93              while(r<q.r) {
94                  ++r;
95                  if(lft[r]>=l && r-lft[r]<=k) validGap[a[r]]++;
96                  Add(r);
97              }
98              while(l>q.l) {
99                  --l;
100                 if(rgt[l]<=r && rgt[l]-l<=k) validGap[a[l]]++;
101                 Add(l);
102             }
103             while(r>q.r) {
104                 if(lft[r]>=l && r-lft[r]<=k) validGap[a[r]]--;
105                 Remove(r);
106                 r--;
107             }
108             while(l<q.l) {
109                 if(rgt[l]<=r && rgt[l]-l<=k) validGap[a[l]]--;
110                 Remove(l);
111                 l++;
112             }
113
114             if((int)res.size()==0) ans[q.id] = 0;
115             else ans[q.id] = strength[*res.rbegin()];
116         }
117
118         for(int i=1; i<=q; i++) printf("%d\n",ans[i]);
119     }
120     return 0;
121 }
```

Instead of forming a grid formation they formed a single line. The position of each trooper lies between **1** and **n** and each position has exactly one trooper. There are several groups of imperial troopers and each group has a uniform of unique color. Moreover, each of the troopers has a newly designed blaster rifle, the F-13D Blaster. This rifle can be dangerous because of its powerful energy beam and long range shooting capability.

There are some limitations though. The rifle must be connected to a power supply. The imperial troopers have a light weight device attached to their uniform which can generate huge amount of power. Different groups have different types of devices. To activate all the power supply devices of a group, at least one pair of troops of that group needs to have distance no more than **k** between them.

Somehow Han knows this information. To make a perfect plan for destroying the bunker, Han needs to know how many strongest groups are there if we consider the troopers in devices within range **[l, r]** only. **A strongest group is the largest group with active power supply**.

As the only programmer of the rebel army it's your turn to help Han destroying the bunker to deactivate the shield. May the force be with you.

Input

Input starts with a positive integer **T (1 ≤ T ≤ 10)**, denoting the number of test cases. Each test case starts with two positive integers **n (1 ≤ n ≤ 10^4)** and **k (1 ≤ k ≤ n)**, denoting the number of troopers and the minimum distance to activate the power supply devices. Following line will have ***n*** integers denoting the color of each trooper's uniform **(1 ≤ c_i≤ n)**. Here, c_i is the color of the trooper positioned at **i**. Next line will have a positive integer **q (1 ≤ q ≤ 10^5)**, denoting the number of queries. Each of the next ***q*** lines will have two positive integer **l** and **r (1 ≤ l ≤ r ≤ n)**, denoting the range.

Output

For each query output the number of strongest groups in a single line.

Samples

Input	Output
2 5 1 1 1 2 2 1 3 1 1 1 2 2 5 7 2 3 1 2 1 2 3 2 5 2 4 3 5 2 5 1 6 2 7	0 1 1 1 1 2 2 1

Data set is huge **(around 10 MB)**. Please use faster I/O.