题目要求：<34-P1638-逛画展-题目要求.docx>

## 分析

尺取法算法文档链接：<尺取法.docx>

首先注意到，必须看到所有的画师的画才行，所以可以考虑维护一个区间内，每个画师有多少画，可以用一个数组来维护

第一个循环中，把区间右端点不断右移，并维护该画的画师出现次数，出现新的画师把计数器加一，直到看到所有画师的画，然后循环判断左端点的画师是否出现一次以上，

这时候应用贪心思想，只要出现过一次以上，那么左端点就一定可以弹出，因为这个点的存在只会让区间更长，而不会让区间包含画师更多，我们只要保证这个区间有每个画师的画就行了，

这时用两个变量来记录当前区间左右端点作为初始答案

然后第二次循环，继续循环剩下的画，

每次将右端点+1，维护画的画师出现次数，并不断弹出没必要存在的左端点，因为在第二个循环里任何时刻的区间都保证满足能看到所有画师的画，所以应该在循环里面判断当前区间是否比已知答案的区间更短，如果更短，更新答案

我的代码区间是从零计的，所以输出时+1

虽然循环嵌套，但是因为每个点都只会最多出入一次区间，所以复杂度O(n)，只是常数略大

## 分析2

类似滑动窗口的贪心。

考虑暴力。枚举右端点，贪心找到一个最优的左端点，判断区间长度是否更优，更新答案。

尝试将这个暴力优化。

考虑右端点从1开始一步一步往右边推。我们增加一个数组pos记录大师最晚出现的位置。显然每次右端点往右移一位，就会有一个大师的pos被更新。

若左端点在上一个右端点的最优解为l，我们考虑左端点上的画是k。若l<pos[k]，是不是这个画根本就没有必要看下去了？因为这个画在后面能够看到，而这个画又处于最左边，可以选择不看，于是此时我们应该把l往右移一位，显然这是符合贪心策略的。但若l==pos[k]，l显然不能右移，因为这样不符合包含所有种类的画的条件。

于是我们便可以在每次右端点向右移一位，再贪心地更新左端点的位置来找到左端点的最优解，这就是一个优化的暴力。然后每次就判断区间长度是否更优就行了。

一些细节： 首先这个区间里面必须包含所有的画。一开始把pos初始化为-1，则如果左端点的画仅出现过一次，那么左端点不会更新。当所有的pos都不为-1的时候，才可以统计答案。

1. #include<bits/stdc++.h>
2. using namespace std;
3. const int N=1e6+10,M=2e3+10;
4. int mlen=0x7fffffff,ml,mr; *// 记录答案*
5. int pos[M]; *// 记录种类画最后一次出现的位置*
6. int pic[N],l=1,cnt; *// 画、左端点和已经包含了多少种类的画*
7. int main(){
8. ios::sync\_with\_stdio(false);
9. int n,m;
10. cin>>n>>m;
11. int na;
12. memset(pos,-1,sizeof(pos));
13. for(int i=1;i<=n;i++){
14. cin>>pic[i];
15. if(pos[pic[i]]==-1) cnt++; *// 如果没有出现，统计+1*
16. pos[pic[i]]=i; *// 更新位置*
17. while(l!=i && l<pos[pic[l]]) l++; *// 更新左端点*
18. if(cnt==m && i-l+1<mlen)
19. mlen=i-l+1,ml=l,mr=i;
20. *// 如果已经包含了所有种类的画，而且区间较之前更小，更新答案*
21. }
22. cout<<ml<<' '<<mr<<endl;
23. }

## 尺取法

###尺取法###

\*\*思路：\*\*首先这是尺取法的板子题，我们假设从第s幅画开始看，为了覆盖所有大师的画需要看到t。这样的话可以知道如果从s+1开始看的话，那么必须看到t‘>=t幅画为止。so，我们从区间的最开头把s取出之后，s这幅画所属的大师出现次数就要减1,如果此时这个大师出现次数为0了，在同一个大师再次出现之前，不停将区间末尾t向后推进即可。每次在区间末尾追加画t时将t所属的大师出现次数加1,这样就完成了下一个区间上各个大师出现次数的更新，重复这一操作便能在O(nlogn)的时间复杂度下解决这道题。

\*\*注意：\*\*本题数据很大，建议读入优化，但m很小，可以直接计数，如果m很大可以离散或者用stl的map。

1. #include<bits/stdc++.h>
2. using namespace std;
3. inline int gi()
4. {
5. int a=0;char x=getchar();bool f=0;
6. while((x<'0'||x>'9')&&x!='-')x=getchar();
7. if(x=='-')x=getchar(),f=1;
8. while(x>='0'&&x<='9')a=a\*10+x-48,x=getchar();
9. return f?-a:a;
10. }
11. int countt[2005];
12. int n,m,a[1000005];
13. int main()
14. {
15. n=gi(),m=gi();
16. for(int i=0;i<n;i++)a[i]=gi();
17. int s=0,t=0,num=0,res=n,l=1,r=n;
18. for(;;){
19. while(t<n&&num<m){
20. if(countt[a[t++]]++==0)num++;
21. }
22. if(num<m)break;
23. if(res>t-s){res=t-s;l=s+1;r=t;}
24. if(--countt[a[s++]]==0)num--;
25. }
26. cout<<l<<' '<<r;
27. return 0;
28. }

## 双指针

【思路】

双指针

枚举起点a

终点b（尾指针）累加

直到可以看到每一位名师的画作

然后比较记录的区间长度

如果这个区间小那就a，b替代之前记录下来的

怎么判断每一位名师的画作是否被看过？

用一个桶，

如果这位名师的画作加入进来那就在他的桶上累加

如果是从0变为1的一个桶那就是新加入了一位画师

所以计数器累加（计数器用来计数目前可以看到画师画作的数量）

如果是一个从1变为0的桶

那就是少看到一位画师的画作

所以这个时候要计数器累减

【原理】

Two pointers就是两个指针在序列上移动。（注意这里的指针不是C/C++中保存地址的指针变量，就是两个序列的下表，比如i,j,k）

我们先来看一道题

给定s和一个长度为n的正整数序列A。求出最短的子区间长度，满足这个子区间的和不小于s.

例：n = 5, s = 11, A = (1,2,3,4,5). (3, 4 ,5)的和为12，不小于11且最短。

n <= 10 ^ 7

暴力做法？ 枚举左右端点。

O(n^2), 无法通过。

怎么优化？

还记得二分法吗？

二分所求区间长度k，然后跑一遍整个序列求出所有长度为k的区间的和，看看有没有超过给定的s。

复杂度O(n log n)。但还是不够优。

Two pointers

用two pointers可以做到O(n).

我们取两个指针i, j,最开始都放在1位置。

这两个指针会夹住区间a[i,j]。我们现在可以通过移动这两个指针来表示任意一段区间。

指针移动时，区间和的变化可以快速得到。用一个变量记录当前两指针范围内所有数的和即可。

注意到序列中只包含正数，那么区间越长其元素和就越大。

开始时，我们将右指针j向右一直移动，直到区间和大于s，然后用此时的区间长度更新答案。此时我们得到的是区间左端点为i时最短的符合条件的区间。

然后我们将i右移一格，此时可能区间和变小而不满足条件，那么我们继续将j从之前位置向右移动，直到区间和大于s。

反复进行，将得到整个题目的答案。

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

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

scanf("%d", a + i);

int l = 1, r = 1, su = a[1];

while(l <= n)

{

while(su < s && r <= n)

r++, su += a[r];

if(su >= s) ans = min(ans, r - l + 1);

su -= a[l];

l++;

}

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

注意到两个指针均只会右移，所以每个最多移动n次, 所以时间复杂度是O(n)

我们再来看一道题

给定n,问将n拆成连续的整数的平方和的方案。

输出这些方案。 n <= 10 ^ 14.

例如，2030 = 21 ^ 2 + 22 ^ 2 + 23 ^ 2 + 24 ^2 = 25 ^ 2 + 26 ^ 2 + 27 ^ 2。

提示

既然n<=10^14,我们只需要考虑<=10^7的数的平方。 算出数组a，其中a[i]=i^2;

现在我们只需要在a上面取子区间，使得区间和恰为n.把上面的代码稍稍修改一下就行了。

[Two pointers](https://www.cnblogs.com/forever97/category/909925.html)

【回溯】

首先我们需要一个数组存入输入的数据，然后由于wangjy要看遍每个画家的至少一幅画，我们还需要一个check数组来存入当前i,j之间每个画家画的个数。

最后还需要判断是否满足条件，我们当然可以遍历check数组，但是M小还可以，大了则是一个不小的开支，仔细思考一下我们可以开一个cnt变量，每当check[j]由0变到1，便++，每当check[i]由1变到0,便--.

1. #include<cstdio>
2. int check[2001], paint[1000001];
3. int main()
4. {
5. int n, m, cnt = 0, minn = 1000001, j = 0, i = 1;
6. scanf("%d%d", &n, &m);
7. for(int i = 1; i <= n; i++)
8. scanf("%d", paint + i);
9. int a = 1, b = n;
10. while(i <= n)
11. {
12. while(cnt < m && j < n) {
13. j++;
14. if(check[paint[j]] == 0) cnt++;
15. check[paint[j]]++;
16. }
17. check[paint[i]]--;
18. if(check[paint[i]] == 0) cnt--;
19. i++;
20. if(j - i + 1 < minn && cnt == m) a = i, b= j, minn = j - i + 1;
21. }
22. printf("%d %d", a, b);
23. return 0;
24. }

## 我的代码：

附上文件路径：<34-P1638-逛画展.cpp>

1. #include<stdio.h>
2. int n,m,sum,check[2050];
3. int main(void)
4. {
5. scanf("%d%d",&n,&m);
6. int pic[n+1];
7. int right=1,left=1,ansl=1,ans=n,ansr=n;
8. for(int i=1;i<=n;i++) scanf("%d",&pic[i]);
9. while(1)
10. {
11. while(right<=n&&sum<m)
12. if(check[pic[right++]]++==0) sum++;
13. if(sum<m)break;
14. if(ans>(right-left))
15. {
16. ans=right-left;
17. ansr=right-1;
18. ansl=left;
19. }
20. if(--check[pic[left++]]==0) sum--;
21. }
22. printf("%d %d",ansl,ansr);
23. return 0;
24. }