双指针

信智学院　陈琰宏

教学内容

01 归并指向两个序列

02 快排 指向一个序列

03

[2709] 最输出每个单词

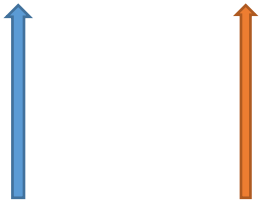
输入一个行简单英文句子，单词之间用空格分隔，没有缩写形式和其它特殊形式，请输出改句子中的每个单词。

输入样例
I am a student of Peking University

输出样例
I
am
a
student
of
Peking
University

双指针导入

I am a student of Peking University



i j

1 3 5 7 9 2 4 6 8 10

i j

双指针导入

```
for(int i=0;i<len;i++){
    while(j<i && check(i,j))
        j++;

    //每道题的具体逻辑
}
```

双指针算法的核心思想是把朴素的暴力写法o(n²)优化到o(n)。双指针写法看着两个循环，但两个指针总的移动次数不超过2n

```cpp
#include <bits/stdc++.h>
using namespace std;

int main()
{
    char str[1000];
    cin.getline(str,1000);
    int len=strlen(str);
    for(int i=0;i<len;i++){
        int j=i;
        while(j<len &&str[j] != ' ')j++;
        for(int k=i;k<j;k++)
            cout<<str[k];
        cout<<"\n";
        i=j;
    }
    return 0;
}
```

[1107] 最长连续不重复子序列

给定一个长度为n的整数序列，请找出最长的不包含重复的数的连续区间，输出它的长度。
第一行包含整数 n。$1 \leqslant n \leqslant 10^5$

第二行包含 n 个整数（均在 0~ 10^5范围内），表示整数序列。共一行，包含一个整数，表示最长的不包含重复的数的连续区间的长度

输入样例
5
1 2 2 3 5

输出样例
3

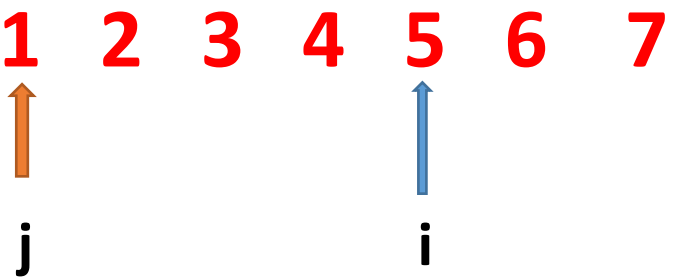
分析：纯暴力

用暴力法：对每个 i(终点） 和 j（起点） 都遍历一遍，对每个 i 和 j 都检查一下[i,j]间的数据是否满足给定的条件。这样的时间复杂度是O(n^2)；数据稍微大点就会超时。代码如下：

```
1   for (int i = 0; i < n; i++)
2           for (int j = 0; j <= i; j++)
3               if (check(v1,j, i) == 0)//检查 i 和 j 之间是否有重复的数字
4                   res = max(res, i - j + 1);
5
6     //check函数
7   int check(int  v1[], int l, int r)
8   {
9       for (int i = l+1; i <=r ; i++)
10          for (int j = l; j < i; j++)
11              if (v1[i] == v1[j])
12                  return 1;
13      return 0;
14  }
```

双指针方法1：

仔细考虑暴力法就会发现，暴力法在解题时有很多地方是重复计算了（ i 指针在 j 指针的后面，i是遍历的整个数组的，j 是遍历 0 到 i 的）：



比如 j = 1，i = 5，此时发现 i，j 是满足题解条件的；那么后面的 j = 2 到 5，就不用计算了，肯定是满足条件的。所以引出了双指针法：当发现 j = 1，i = 5满足题解条件，那就不用计算 j = 2 到 5，直接计算 j = 1，i = 6，如果不满足条件，那就计算 j = 2，i = 6，然后接着计算。这样就是 i 和 j 指针都是从前移到后，也就是计算2n次。时间复杂度是O(2n)。

双指针方法1：

```
for (int i = 0, j = 0; i < n; i++)
    {//
        while (j <= i)
        {
            if (check(v1, j, i) == 0)
            {//如果没有重复，i往前走
                res = max(res, i - j + 1);
                break;
            }
            else//有重复j++
                j++;
        }
    }
int check(int  v1[], int l, int r)
{//找得到重复的数返回1
    for (int i = l + 1; i <= r; i++)
        for (int j = l; j < i; j++)
        {
            if (v1[i] == v1[j])
                return 1;
        }
    return 0;
}
```

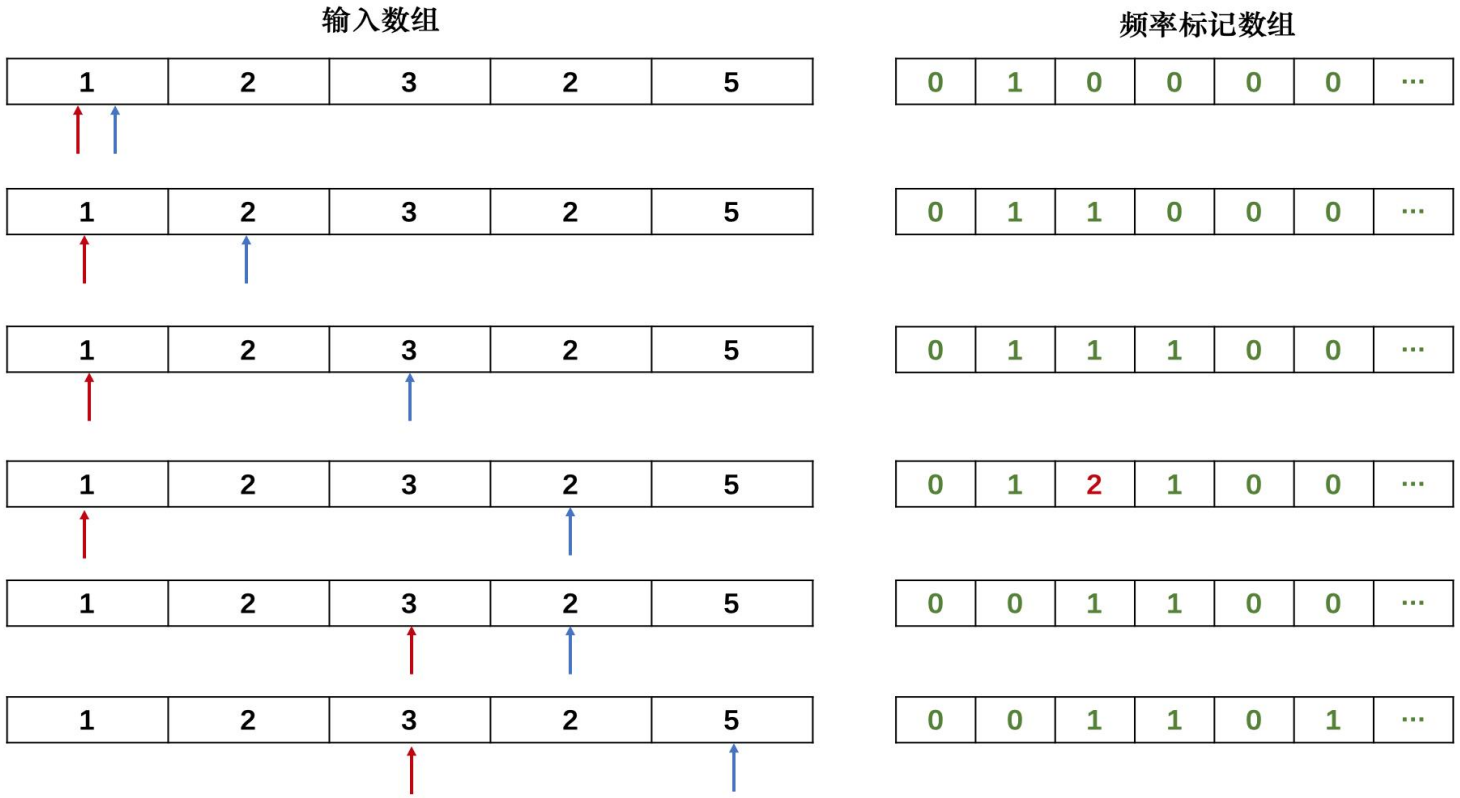
o(n²)->o(2n)

```
for (int i = 0; i < n; i++)
        for (int j = 0; j <= i; j++)
            if (check(v1,j, i) == 0)//检查
                res = max(res, i - j + 1);

  //check函数
int check(int  v1[], int l, int r)
{
    for (int i = l+1; i <=r ; i++)
        for (int j = l; j < i; j++)
            if (v1[i] == v1[j])
                return 1;
    return 0;
}
```

双指针方法2：

上面代码还是超时，因为check函数写的不好，循环太多，直接是暴力计算找重复数字的，显然不好。所以引出一个新的check方法：寻找是否有重复数字，可以用hash表。



双指针方法核心思路

1. 遍历数组a中的每一个元素a[i]，对于每一个i，找到j使得双指针[j, i]维护的是以a[i]结尾的最长连续不重复子序列，长度为i - j + 1。
2. 对于每一个i，如何确定j的位置：由于[j, i - 1]是前一步得到的最长连续不重复子序列，所以如果[j, i]中有重复元素，一定是a[i]，因此右移j直到a[i]不重复为止。（由于[j, i - 1]已经是前一步的最优解，此时j只可能右移以剔除重复元素a[i]，不可能左移增加元素，因此，j具有“单调性”、本题可用双指针降低复杂度）。
3. 用数组s记录子序列a[j~i]中各元素出现次数，遍历过程中对于每一个i有四步操作：cin元素a[i] -> 将a[i]出现次数s[a[i]]加1 -> 若a[i]重复则右移j（s[a[j]]要减1）-> 确定j及更新当前长度i - j + 1给r。

```
4   const int N = 100010;
5   int a[N], count[N];
6   int n, ans;
7   int main() {
8       scanf("%d", &n);
9       for (int i = 0; i < n; i++) {
10          scanf("%d", &a[i]);
11      }
12      for (int i = 0, j = 0; i < n; i++) {
13          count[a[i]] ++;
14          while (count[a[i]] > 1) {
15          //当a[i]重复时，先把a[j]次数减1，再右移j。
16          //只需考虑a[j]~a[i]间的元素出现的次数即可
17              count[a[j]] --;
18              j++;//右移j直到a[i]不重复为止
19          }
20          ans = max(ans, i - j + 1);
21      }
22      printf("%d", ans);
```

[5166] 数组元素的目标和

给定两个升序排序的有序数组 A 和 B，以及一个目标值 x。数组下标从 0 开始。 请你求出满足 A[i]+B[j]=x 的数对(i,j)。 数据保证有唯一解。

输入第一行包含三个整数n,m,x，分别表示 A 的长度，B 的长度以及目标值x。第二行包含 n 个整数，表示数组 A。第三行包含 m 个整数，表示数组 B。 数组长度不超过 10^5。同一数组内元素各不相同。1≤数组元素≤10^9

输出共一行，包含两个整数 i 和 j。

输入样例

```
4 5 6
1 2 4 7
3 4 6 8 9
```

输出样例

```
1 1
```

分析：暴力枚举

```
4   int a[100001],b[100001];
5   int main()
6   {
7       cin>>n>>m>>s;
8       for(int i=0;i<m;i++)
9           for(int j=0;j<n;j++)
10              if(a[i]+a[j]==s){
11                  cout<<i<<" "j<<endl;
12                  break;
13              }
```

分析:双指针

为了减少循环，采用了两个指针来保证能遍历符合要求的所有数；

i指针从A数组的头开始，此时a[i]最小；

j指针从B数组的尾开始，此时b[j]最大；

在i = 0 时，判断a[i] + b[j] 的值，如果大于x， j- -；

也就是说b[j]的值一直在减小；

分析:双指针

那么当a[i] + b[j] 的值第一次小于x时，j指针的左边的数与a[i]相加一定都小于x了；那么我们只能增大a[i]的值；于是i ++;

反复循环,如果a[i] + b[j] 大于x，就减小b[j]（相当于将j指针左移）反之，就增大a[i]（相当于i指针右移）

这样，一定保证了每一个符合要求的两个数都能加一遍，判断是不是和等于x。

```cpp
const int N = 100010;
int a[N], b[N];
int main()
{
    int n ,m ,x;
    scanf("%d%d%d", &n, &m, &x);
    for (int i = 0; i < n; i ++ ) scanf("%d", &a[i]);
    for (int j = 0; j < m; j ++ ) scanf("%d", &b[j]);
    for (int i = 0, j = m - 1; a[i] < x; i ++ )
    {
        while(a[i] + b[j] > x) j --;
            // 重要的地方在这!
        if (a[i] + b[j] == x)
        {
            printf("%d %d", i, j);
            break;
        }
    }
```

总结:双指针

在写出暴力的方法的同时，看看看是否存在单调性，如果存在可以考虑双指针优化。

分析：二分

```
10      for(int i = 0;i < n;i ++) scanf("%d",&a[i]);
11      for(int i = 0;i < m;i ++) scanf("%d",&b[i]);
12      int j = 0;
13      for(int i = 0;i < n;i ++)
14      {
15          if(a[i] < x)
16          {
17              int l = 0,r = m - 1;
18              int c = x - a[i];
19              while(l < r)
20              {
21                  int mid = (l + r) / 2;
22                  if(c > b[mid]) l = mid + 1;
23                  else r = mid;
24              }
25              if(c == b[r]) printf("%d %d",i,r);
26          }
27      }
```

[5167] 判断子序列

给定一个长度为 n 的整数序列 a1,a2,…,an 以及一个长度为 m 的整数序列b1,b2,…,bm。

请你判断 a 序列是否为 b 序列的子序列。

子序列指序列的一部分项按原有次序排列而得的序列，例如序列 {a1,a3,a5} 是序列 {a1,a2,a3,a4,a5} 的一个子序列。

输入第一行包含两个整数 n,m。第二行包含 n 个整数，表示 a1,a2,…,an。第三行包含 m 个整数，表示 b1,b2,…,bm。 $1 \leqslant n \leqslant m \leqslant 10^5$, $-10^9 \leqslant ai,bi \leqslant 10^9$

输出如果 a 序列是 b 序列的子序列，输出一行 Yes。否则，输出 No。

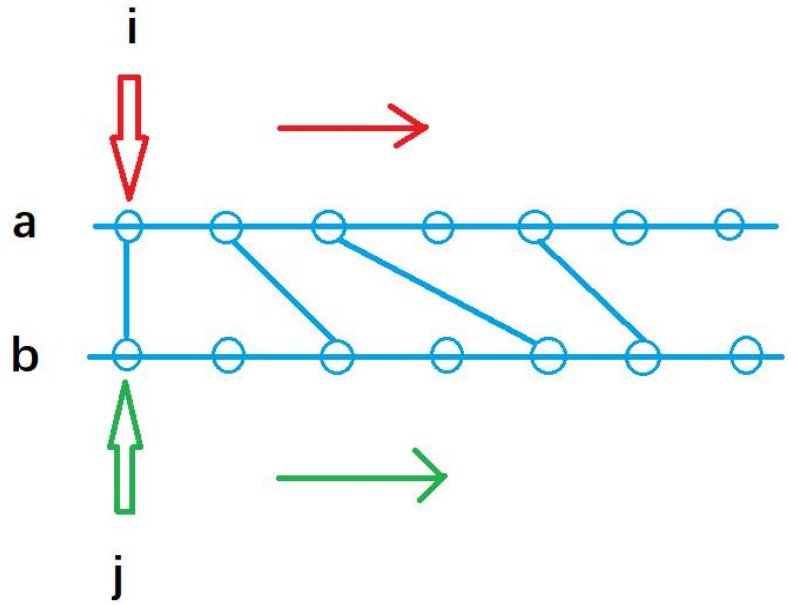
[5167] 判断子序列

输入样例
3 5
1 3 5
1 2 3 4 5

输出样例 Yes

1. j指针用来扫描整个b数组，i指针用来扫描a数组。若发现a[i]==b[j]，则让i指针后移一位。
2. 整个过程中，j指针不断后移，而i指针只有当匹配成功时才后移一位，若最后若i==n，则说明匹配成功。

整个过程中j指针不断扫描b数组并且向后移动，相当于不断给i指针所指向的a数组创建匹配的机会，只有匹配成功时i指针才会向后移动一位，当i==n时，说明全部匹配成功。



j指针用来扫描整个b数组，i指针用来扫描a数组。若发现a[i]==b[j]，则让i指针后移一位。
整个过程中，j指针不断后移，而i指针只有当匹配成功时才后移一位，若最后i==n，则说明匹配成功。

```
int i = 0;
for(int j = 0;j < m;j++)
{
   if(i < n && a[i] == b[j]) i++;
}
if(i == n) puts("Yes");
else puts("NO");
```

```
4   const int N=1e5+10;
5   int a[N],b[N];
6   int main()
7   {
8       int n,m;
9       scanf("%d%d",&n,&m);
10      for(int i = 0;i < n; i++)
11          scanf("%d",&a[i]);
12      for(int j = 0;j < m; j++)
13          scanf("%d",&b[j]);
14      int i = 0;
15      for(int j = 0;j < m; j++)
16      {
17          if(i < n&&a[i] == b[j])  i++;
18      }
19      if(i == n) puts("Yes");
20      else puts("No");
21      return 0;
22  }
```

[6686] 和为S的连续正数序列

输入一个非负整数 S，打印出所有和为 S 的连续正数序列（至少含有两个数）。

例如输入 15，由于 1+2+3+4+5=4+5+6=7+8=15，所以结果打印出 3 个连续序列 1~ 5、4~ 6和 7~ 8。

输入一个非负整数 S,0≤S≤100000 。输出所有和为 S 的连续正数序列（至少含有两个数）

输入

15

输出

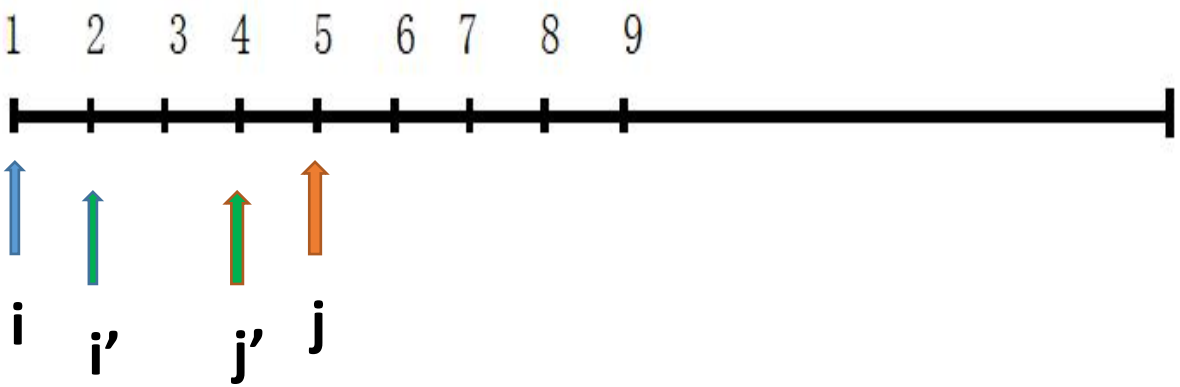
1 2 3 4 5

4 5 6

7 8

分析

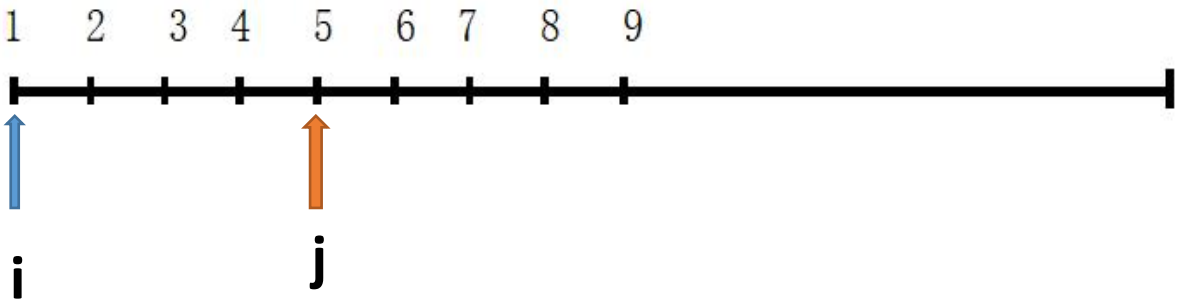
题目要求一个连续区间的值[i,j]，使得的[i,i+1,..,j]的和等于x



当 i=1时，可以求出sum=1+2+3+4+5=15。此时i++,去求下一个符合要求的i值。这时的j呢？

j可能项左走吗？显然不能！因为如果往左走[i,j]将包含[i’，j’]，因此j必然是往右走，符合单调性。

分析



1. 设置两个指针i和j，分别指向连续正数序列的起始和终止
2. 用s表示当前连续正数序列的和，即s=i+(i+1)+…+j
3. 以i递增的方式遍历整个序列(1到n)，代表查找以i开头的时候结尾j应该是多少。当s<sum说明j应该往后移动，当s=sum说明满足题意，当s>sum说明向后走即可。
4. 注意上述遍历过程中，s=sum的情况下不需要把j往前移动，原因是当进入下一个循环前s-=i，即(i+1)到j的和肯定小于sum。

```
3  int main(){
4      int x,s;
5      cin>>x;
6      for(int i=1,j=1,s=1;i<=x ;i++){
7          //以i递增的方式遍历整个序列(1到n)
8          while(s<x){
9              j++;
10             s+=j;//
11         }
12         if(s==x && j>i){
13             for(int k=i;k<=j;k++){
14                 printf("%d ",k);
15             }
16             printf("\n");
17         }
18         s=s-i;//操作x,i要往前走
19
20     }
```

分析2：前缀和+二分

首先求前缀和，枚举每一个前缀和sum[i]，在1~i 之间二分另外一个前缀和sum[mid]，使得sum[i] - sum[mid] == target。

[8240] 一排奶牛

农夫约翰的 N 头奶牛排成一排。
每头奶牛都用一个整数品种 ID 标识，队列中第 i 头奶牛的 ID 为 Bi。
约翰认为如果有一大段连续的奶牛都具有相同的品种 ID，他的奶牛就会更加的引人注目。
为了创造这样的连续段，约翰决定选取一个特定品种 ID，并从队列中剔除所有具有此 ID 的奶牛。
请帮助约翰确定，他通过这样做，能够获得的具有相同品种 ID 的最大奶牛连续段的长度。

输入第一行包含整数 N。 接下来 N 行，每行包含一个 Bi。1≤N≤1000,0≤Bi≤106,不含所有奶牛品种都相同的数据。

输出具有相同品种 ID 的最大奶牛连续段的长度。

输入样例	输出样例
9 2 7 3 7 7 3 7 5 7	4

最初队列中奶牛的品种 ID 依次为
2,7,3,7,7,3,7,5,7
我们去掉所有品种 ID 为 3 的奶牛，
剩下的奶牛的品种 ID 依次为
2,7,7,7,7,5,7
最大的具有相同品种 ID 的奶牛连续段的长度为 4。

分析：枚举

因为n<=1000,直接枚举就可以，做题时根据数据范围选择比较好写，不容易出错的方法最好。

```
#include<bits/stdc++.h>
using namespace std;

const int N=1010;
int n;
int a[N];

int main() {
    cin>>n;
    for(int i=0;i<n;i++)
        cin>>a[i];
    int res=0;
    int res=0;
    for(int i=0;i<n;i++){
        //枚举删除每一个a[i]
        int cnt=0,last=-1;
        //cnt表示当前长度
        //last 表示计入当前长度的元素
        for(int j=0;j<n;j++){
            if(a[j]!=a[i]){
                if(a[j]==last)
                    cnt++;//计数
                else{
                    last=a[j];
                    cnt=1;//重新计数
                }
                res=max(res,cnt);
            }

        }
    }
    cout<<res;
```

分析：枚举

因为n<=1000,直接枚举就可以，做题时根据数据范围选择比较好写，不容易出错的方法最好。

```
#include<bits/stdc++.h>
using namespace std;

const int N=1010;
int n;
int a[N];

int main() {
    cin>>n;
    for(int i=0;i<n;i++)
        cin>>a[i];
    int res=0;
    int res=0;
    for(int i=0;i<n;i++){
        //枚举删除每一个a[i]
        int cnt=0,last=-1;
        //cnt表示当前长度
        //last 表示计入当前长度的元素
        for(int j=0;j<n;j++){
            if(a[j]!=a[i]){
                if(a[j]==last)
                    cnt++;//计数
                else{
                    last=a[j];
                    cnt=1;//重新计数
                }
                res=max(res,cnt);
            }

        }
    }
    cout<<res;
```

分析

分析

3745. 牛的学术圈 I

8146: Diamond Collector
钻石收藏家

8214: Cow baseball 奶牛棒球

今天的课程结束啦……

下课了…
同学们再见！