**数据结构课程设计**

**----个人设计报告**

专 业： 软件工程

班 级： 卓越软件2203

姓 名： 周立成

学 号： 221310332

指导教师： 冯向阳

日 期： 2023年6月20日至2023年7月07日

**目录**

[1 课程设计目的 6](#_Toc138499355)

[2 任务完成情况 6](#_Toc138499356)

[3设计报告 7](#_Toc138499357)

[3.1 快速排序的应用 7](#_Toc138499358)

[3.1.1 题目以及要求 7](#_Toc138499359)

[3.1.2概要设计 7](#_Toc138499360)

[3.1.3 算法分析 7](#_Toc138499361)

[3.1.4使用说明: 9](#_Toc138499362)

[3.1.5 测试结果与分析 9](#_Toc138499363)

[3.2顺序表元素的快速删除 11](#_Toc138499364)

[3.2.1 题目以及要求 11](#_Toc138499365)

[3.2.2概要设计 11](#_Toc138499366)

[3.2.3 算法分析 12](#_Toc138499367)

[3.2.4使用说明: 13](#_Toc138499368)

[3.2.5 测试结果与分析 13](#_Toc138499369)

[3.3判断单链表的对称性 15](#_Toc138499370)

[3.3.1 题目以及要求 15](#_Toc138499371)

[3.2.2概要设计 15](#_Toc138499372)

[3.2.3 算法分析 15](#_Toc138499373)

[3.3.4使用说明: 16](#_Toc138499374)

[3.3.5 测试结果与分析 16](#_Toc138499375)

[3.4 搜索插入位置 19](#_Toc138499376)

[3.4.1题目及其要求 19](#_Toc138499377)

[3.4.2概要设计 19](#_Toc138499378)

[3.4.3算法分析 19](#_Toc138499379)

[3.4.4使用说明 20](#_Toc138499380)

[3.4.5测试结果与分析 20](#_Toc138499381)

[3.5有效的完全平方数 24](#_Toc138499382)

[3.5.1题目以及要求 24](#_Toc138499383)

[3.5.2概要设计 24](#_Toc138499384)

[3.5.3算法分析 24](#_Toc138499385)

[3.5.4使用说明 25](#_Toc138499386)

[3.5.5测试结果与分析 25](#_Toc138499387)

[3.6寻找比目标字母大的最小字母 27](#_Toc138499388)

[3.6.1题目以及要求 27](#_Toc138499389)

[3.6.2概要设计 27](#_Toc138499390)

[3.6.3算法分析 27](#_Toc138499391)

[3.6.4使用说明 28](#_Toc138499392)

[3.6.5测试结果与分析 28](#_Toc138499393)

[3.7矩阵中战斗力最弱的 K 行 30](#_Toc138499394)

[3.7.1题目以及要求 30](#_Toc138499395)

[3.7.2概要设计 30](#_Toc138499396)

[3.7.3算法分析: 30](#_Toc138499397)

[3.7.4使用说明: 30](#_Toc138499398)

[3.7.5测试结果与分析 31](#_Toc138499399)

[3.8找到和最大的长度为 K 的子序列 35](#_Toc138499400)

[3.8.1题目以及要求 35](#_Toc138499401)

[3.8.2算法设计 35](#_Toc138499402)

[3.8.3算法分析 35](#_Toc138499403)

[3.8.4使用说明 35](#_Toc138499404)

[测试结果与分析 36](#_Toc138499405)

[3.9无人机方阵 39](#_Toc138499406)

[3.9.1题目以及要求 39](#_Toc138499407)

[3.9.2概要设计 39](#_Toc138499408)

[3.9.3算法分析 39](#_Toc138499409)

[3.9.4使用说明 40](#_Toc138499410)

[3.9.5测试结果与分析 40](#_Toc138499411)

[3.10完成一半题目 46](#_Toc138499412)

[3.10.1题目描述 46](#_Toc138499413)

[3.10.2概要设计 46](#_Toc138499414)

[3.10.3算法分析 46](#_Toc138499415)

[3.10.4使用说明 46](#_Toc138499416)

[3.10.5测试结果与分析 47](#_Toc138499417)

[3.11 错误的集合 50](#_Toc138499418)

[3.11.1题目以及要求 50](#_Toc138499419)

[3.11.2概要设计 50](#_Toc138499420)

[3.11.3算法分析 50](#_Toc138499421)

[3.11.4使用说明 50](#_Toc138499422)

[3.11.5测试结果与分析 51](#_Toc138499423)

[3.12至少是其他数字两倍的最大数 53](#_Toc138499424)

[3.12.1题目以及要求 53](#_Toc138499425)

[3.12.2概要设计 53](#_Toc138499426)

[3.12.3算法分析 53](#_Toc138499427)

[3.12.4使用说明 53](#_Toc138499428)

[3.12.5测试结果与分析 54](#_Toc138499429)

[3.13 K 次取反后最大化的数组和 56](#_Toc138499430)

[3.13.1题目以及要求 56](#_Toc138499431)

[3.13.2概要设计 56](#_Toc138499432)

[3.13.3算法分析 56](#_Toc138499433)

[3.13.4使用说明 56](#_Toc138499434)

[3.13.5测试结果与分析 57](#_Toc138499435)

[3.14检查整数及其两倍数是否存在 61](#_Toc138499436)

[3.14.1题目以及要求 61](#_Toc138499437)

[3.14.2概要设计 61](#_Toc138499438)

[3.14.3算法分析 61](#_Toc138499439)

[3.14.4使用说明 61](#_Toc138499440)

[3.14.5测试结果与分析 62](#_Toc138499441)

[3.15数据流的第 K 大数值 64](#_Toc138499442)

[3.15.1题目以及要求 64](#_Toc138499443)

[3.15.2概要设计 64](#_Toc138499444)

[3.15.3算法分析 64](#_Toc138499445)

[3.15.4使用说明 64](#_Toc138499446)

[3.15.5测试结果与分析 65](#_Toc138499447)

## 1 课程设计目的

* 学习获取知识的方法；
* 提高发现问题、分析问题和解决实际问题的能力；
* 加强创新意识和创新精神；
* 掌握面向实际背景思考问题的方法。

## 2 任务完成情况

任务完成情况介绍，如2-1.（仅供参考，请根据实际完成情况填写）

**表2-1 任务完成情况表**

|  |  |  |
| --- | --- | --- |
| 任务序号 | 任务名称 | 完成情况 |
| 1 | 快速排序的应用 | AC |
| 2 | 顺序表元素的快速删除 | AC |
| 3 | 判断单链表的对称性 | AC |
| 4 | 有效的完全平方数 | AC |
| 5 | 寻找比目标字母大的最小字母 | AC |
| 6 | 寻找比目标字母大的最小字母 | AC |
| 7 | 矩阵中战斗力最弱的 K 行 | AC |
| 8 | 找到和最大的长度为 K 的子序列 | AC |
| 9 | 无人机方阵 | AC |
| 10 | 完成一半题目 | AC |
| 11 | 错误的集合 | AC |
| 12 | 至少是其他数字两倍的最大数 | AC |
| 13 | K 次取反后最大化的数组和 | AC |
| 14 | 检查整数及其两倍数是否存在 | AC |
| 15 | 数据流的第 K 大数值 | AC |

## 3设计报告

### 3.1 快速排序的应用

#### 3.1.1 题目以及要求

已知线性表按顺序存于内存，每个元素都是非零整数。在使用顺序表ADT的基础上，试设计基于快速排序的思想把所有值为负数的元素移到全部正数值元素前边的算法：例变为。

提示：要求重排*n*个元素且以顺序存储结构存储的线性表，使得所有值为负数的元素移到正数元素的前面。这可采用快速排序的思想来实现。只是比较的标准是元素是否为负数。因此枢轴元素的值为0(不是线性表中的元素)。基本思路是，先设置好上、下界和枢轴值(0)，然后执行一趟快速排序，即利用震荡交替法分别从线性表的两端查找正数和负数，找到后互相交换，直到上下界相遇。

参考函数原型：

template<class ElemType>

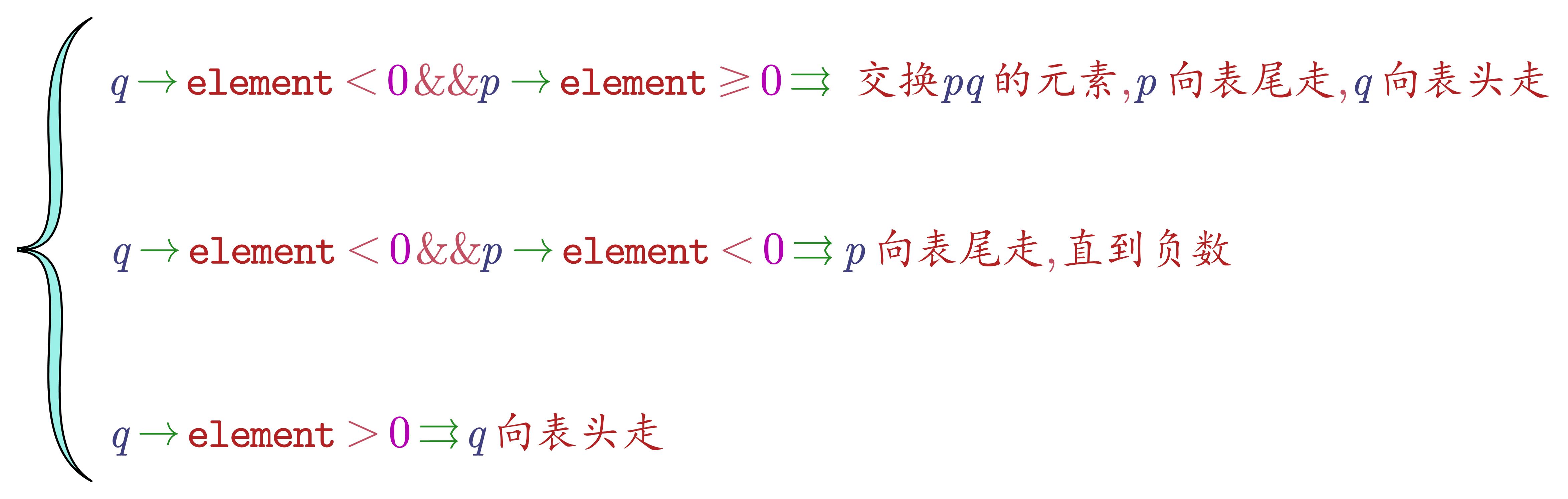
void Rearrange( SqList<ElemType> &A );

#### 3.1.2概要设计

* 设计双向链表deque\_chain<int>用来将输入数据转换成一个链表
* 为双向链表书写print函数用来按照OJ格式打印
* 设计Rearrange函数用来删除元素

#### 3.1.3 算法分析

考虑用两个指针*p,q*指向表头和表尾,按照快速排序的思想,会有如下情况的讨论:



**算法说明:**

void Rearrange(deque\_chain<int>&a)  
{  
 chainNode<int>\*p=a.get\_head()->next;  
 chainNode<int>\*q=a.get\_tail();  
 int l=0,h=a.get\_size()-1;  
 while(l<h)  
 {  
 if(q->element<0)  
 {  
 if(p->element>=0)  
 {  
 swap(p->element,q->element);  
 p=p->next;q=q->fa;  
 l++;h--;  
 a.print();  
 continue;  
 }  
 else  
 {  
 p=p->next; l++;  
 continue;  
 }  
 }else  
 {  
 q=q->fa;h--;  
 continue;  
 }  
 }  
}

通过双指针的方式,只需要循环一遍,即可调整整个链表;

**时间复杂度分析:**

由于只循环了一遍链表,故时间复杂度为;

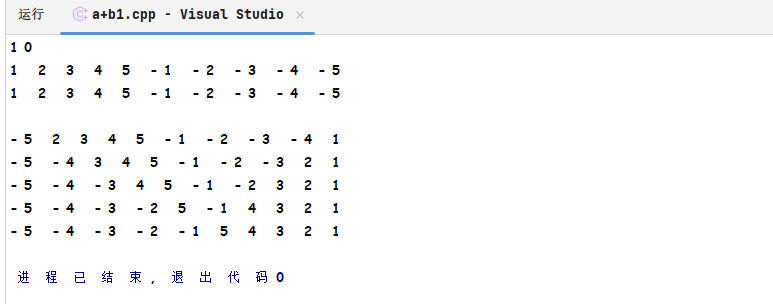
#### 3.1.4使用说明:

输入说明 :

第一行：顺序表A的长度

第二行：顺序表A的数据元素（数据元素之间以空格分隔）(按照题目要求,**数据元素必须是整数**)

**运行界面如下:在Clion2023.1,MVSC 17.0编译器的环境下可以正常运行**



#### 3.1.5 测试结果与分析

* **第一组数据:特殊情况,全是正数,只输出一列,运行正常**

IN:

10

1 2 3 4 5 6 7 8 9 10

OUT:

1 2 3 4 5 6 7 8 9 10

* **第二组数据:特殊情况,负数全在后面,需要调整至前面,假设一个表长为*n*的链表有*n*/2个负数,则应该输出1+*n*/2行,运行正常**

IN:

10

1 2 3 4 5 -1 -2 -3 -4 -5

OUT:

1 2 3 4 5 -1 -2 -3 -4 -5

-5 2 3 4 5 -1 -2 -3 -4 1

-5 -4 3 4 5 -1 -2 -3 2 1

-5 -4 -3 4 5 -1 -2 3 2 1

-5 -4 -3 -2 5 -1 4 3 2 1

-5 -4 -3 -2 -1 5 4 3 2 1

* **第三组数据,普通标准测试,正负交替,运行正常**

IN:

30

-1 -2 -3 -4 -5 1 2 1 2 3 -1 2 3 4 5 1 2 3 4 5 9 0 0 0 0 -1 -2 -3 -6 -3

OUT:

-1 -2 -3 -4 -5 1 2 1 2 3 -1 2 3 4 5 1 2 3 4 5 9 0 0 0 0 -1 -2 -3 -6 -3

-1 -2 -3 -4 -5 -3 2 1 2 3 -1 2 3 4 5 1 2 3 4 5 9 0 0 0 0 -1 -2 -3 -6 1

-1 -2 -3 -4 -5 -3 -6 1 2 3 -1 2 3 4 5 1 2 3 4 5 9 0 0 0 0 -1 -2 -3 2 1

-1 -2 -3 -4 -5 -3 -6 -3 2 3 -1 2 3 4 5 1 2 3 4 5 9 0 0 0 0 -1 -2 1 2 1

-1 -2 -3 -4 -5 -3 -6 -3 -2 3 -1 2 3 4 5 1 2 3 4 5 9 0 0 0 0 -1 2 1 2 1

-1 -2 -3 -4 -5 -3 -6 -3 -2 -1 -1 2 3 4 5 1 2 3 4 5 9 0 0 0 0 3 2 1 2 1

* **第四组数据,特殊数据,全是负数,运行正常**

IN:

5

-1 -2 -3 -4 -5

OUT:

-1 -2 -3 -4 -5

* **第五组数据,普通数据,数据量适中,运行正常**

IN:

17

2 -3 4 -5 6 -7 8 -9 10 -11 30 -12 40 -13 -14 40 23

OUT:

2 -3 4 -5 6 -7 8 -9 10 -11 30 -12 40 -13 -14 40 23

-14 -3 4 -5 6 -7 8 -9 10 -11 30 -12 40 -13 2 40 23

-14 -3 -13 -5 6 -7 8 -9 10 -11 30 -12 40 4 2 40 23

-14 -3 -13 -5 -12 -7 8 -9 10 -11 30 6 40 4 2 40 23

-14 -3 -13 -5 -12 -7 -11 -9 10 8 30 6 40 4 2 40 23

### 3.2顺序表元素的快速删除

#### 3.2.1 题目以及要求

已知长度为n的线性表A采用顺序存储结构，请在应用顺序表ADT的基础上，设计一时间复杂度为0(n)、空间复杂度为0(1)的算法，该算法删除线性表中所有值为item的数据元素。要求：线性表元素个数n很大，而值为item的数据元素个数很少，要求移动元素个数尽量少；删除后的数组元素与原数组元素不必保持顺序一致。

提示：算法参照快速排序的思想，使用两个指针，分别从左往右扫描找到需要被删除的元素、从右往左扫描找到不被删除的元素，然后交换。然后继续扫描。

参考函数原型：

//删除线性表中所有值为item的数据元素，时间复杂度为0(n)、空间复杂度为0(1)

template<class ElemType>

void DeleteItem( SqList<ElemType> &A, ElemType item );

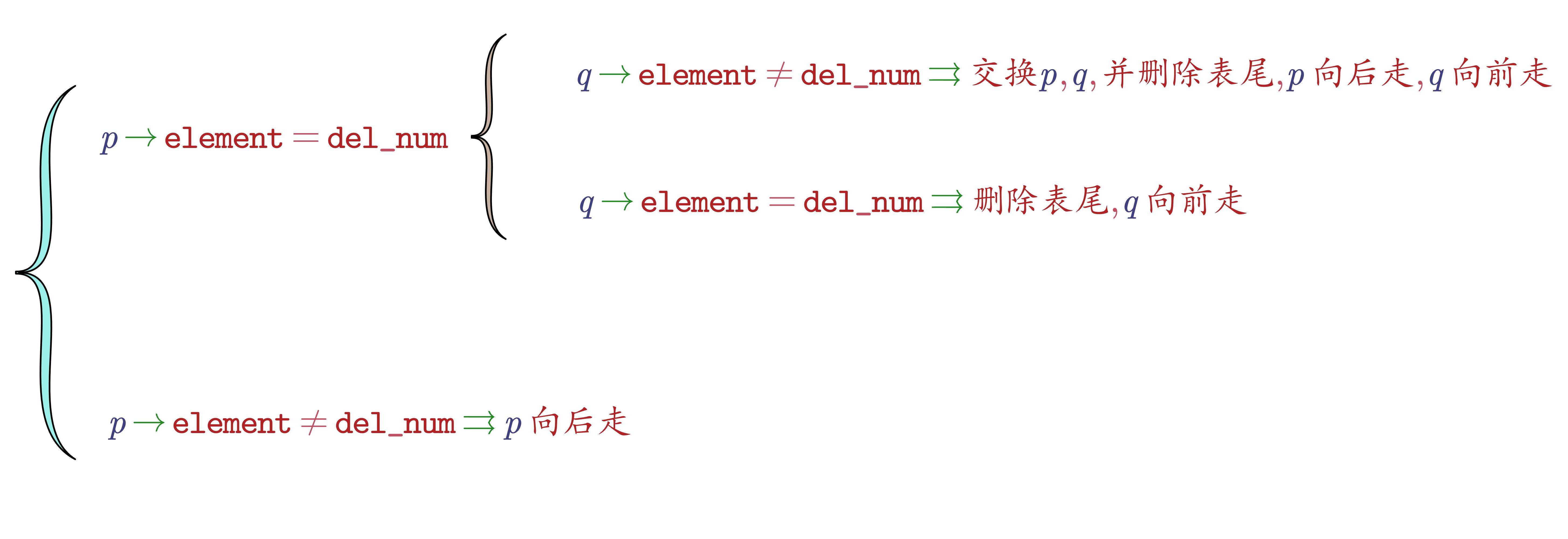
#### 3.2.2概要设计

* 设计双向链表deque\_chain<int>用来将输入数据转换成一个链表
* 为双向链表书写print函数用来按照OJ格式打印
* 设计DeleteItem函数用来删除元素

#### 3.2.3 算法分析

**算法说明:**

考虑用两个指针*p,q*指向表头和表尾,按照快速排序的思想,会有如下情况的讨论:



其次为链表类设计*pop\_back*函数便于快速删除尾部:

void pop\_back() {  
 if (this->listSize == 0) {  
 return;  
 } else {  
 listSize--;  
 if (tailNode == headNode->next) {  
 chainNode<T> \*p = headNode->next;  
 delete p;} else {  
 chainNode<T> \*g = tailNode;  
  
 tailNode->fa->next = **NULL**;  
 tailNode = tailNode->fa;  
 delete g;  
 }  
 }

核心代码部分:

void DeleteItem(deque\_chain<int> &a, int num) {  
 chainNode<int> \*p = a.get\_head()->next;  
 chainNode<int> \*q = a.get\_tail();  
 int l = 0, h = a.get\_size() - 1;  
 while (l < h) {  
 if (p->element == num) {  
 if (q->element != num) {  
 swap(q->element, p->element);  
 l++;  
 h--;  
 p = p->next;  
 q = q->fa;  
 a.pop\_back();  
  
 } else {  
  
 q = q->fa;  
 a.pop\_back();  
 h--;  
 }  
 } else {  
 p = p->next;  
 l++;  
 }  
 }  
 if (a.get\_tail()->element == num) {  
 a.pop\_back();  
 }  
}

通过双指针的方式,只需要循环一遍,即可调整整个链表,但是要注意的是,考虑到整个表仅表尾是被删除元素,或者整个链表全是需要删除的元素这样的情况,应该在循环结束再判一下表尾是不是需要删除的元素,如果是的化需要删除.

**时间复杂度分析:**

由于只循环了一遍链表,故时间复杂度为;

#### 3.2.4使用说明:

输入说明 :

第一行：顺序表A的长度

第二行：顺序表A的数据元素（数据元素之间以空格分隔）

第三行：待删除数据元素

**运行界面如下:在Clion2023.1,MVSC 17.0编译器的环境下可以正常运行**

表格

中度可信度描述已自动生成

#### 3.2.5 测试结果与分析

* 第一组数据:仅1个元素,需要被删除,返回空表,测试极端情况

IN:

1

1

1

OUT:

1

empty

* 第二组数据,全部元素都需要被删除,返回空表,属于特殊情况

IN:

10

1 1 1 1 1 1 1 1 1 1

1

OUT:

1 1 1 1 1 1 1 1 1 1

Empty

* 第三组数据,需要删除的元素都在末尾,考察有没有误删多余的元素

IN:

10

5 5 5 5 5 1 1 1 1 1

1

OUT:

5 5 5 5 5 1 1 1 1 1

5 5 5 5 5

* 第四组数据:正常的小规模数据测试

IN:

10

1 2 3 4 1 6 7 8 9 1

1

OUT:

1 2 3 4 1 6 7 8 9 1

9 2 3 4 8 6 7

* 第五组数据,正常的大规模数据测试

IN:

61

482 598 735 52 617 75 260 148 979 815 38 598 349 719 544 339 274 115 637 35 598 608 858 584 598 90 960 350 872 488 120 187 111 882 361 222 525 609 945 351 989 598 106 610 306 468 332 919 690 33 554 219 391 598 998 173 829 950 197 730 894

598

OUT:

482 598 735 52 617 75 260 148 979 815 38 598 349 719 544 339 274 115 637 35 598 608 858 584 598 90 960 350 872 488 120 187 111 882 361 222 525 609 945 351 989 598 106 610 306 468 332 919 690 33 554 219 391 598 998 173 829 950 197 730 894

482 894 735 52 617 75 260 148 979 815 38 730 349 719 544 339 274 115 637 35 197 608 858 584 950 90 960 350 872 488 120 187 111 882 361 222 525 609 945 351 989 829 106 610 306 468 332 919 690 33 554 219 391 173 998

* 第六组数据,特殊情况,没有要被删除的元素

IN:

5

926 587 525 261 694

547

OUT:

926 587 525 261 694

926 587 525 261 694

### 3.3判断单链表的对称性

#### 3.3.1 题目以及要求

设带头结点的单链表的头指针为head，结点结构由data和next两个域构成，其中data域为字符型。在使用单链表ADT的基础上，设计一算法判断该链表的前*n*个字符是否中心对称。例如 *x, xyx,  xyyx*都是中心对称。

参考函数原型：

//判断单链表的前n个结点是否中心对称

template<class ElemType>

bool Judge\_Symmetry( LinkList<ElemType> &L, int num );

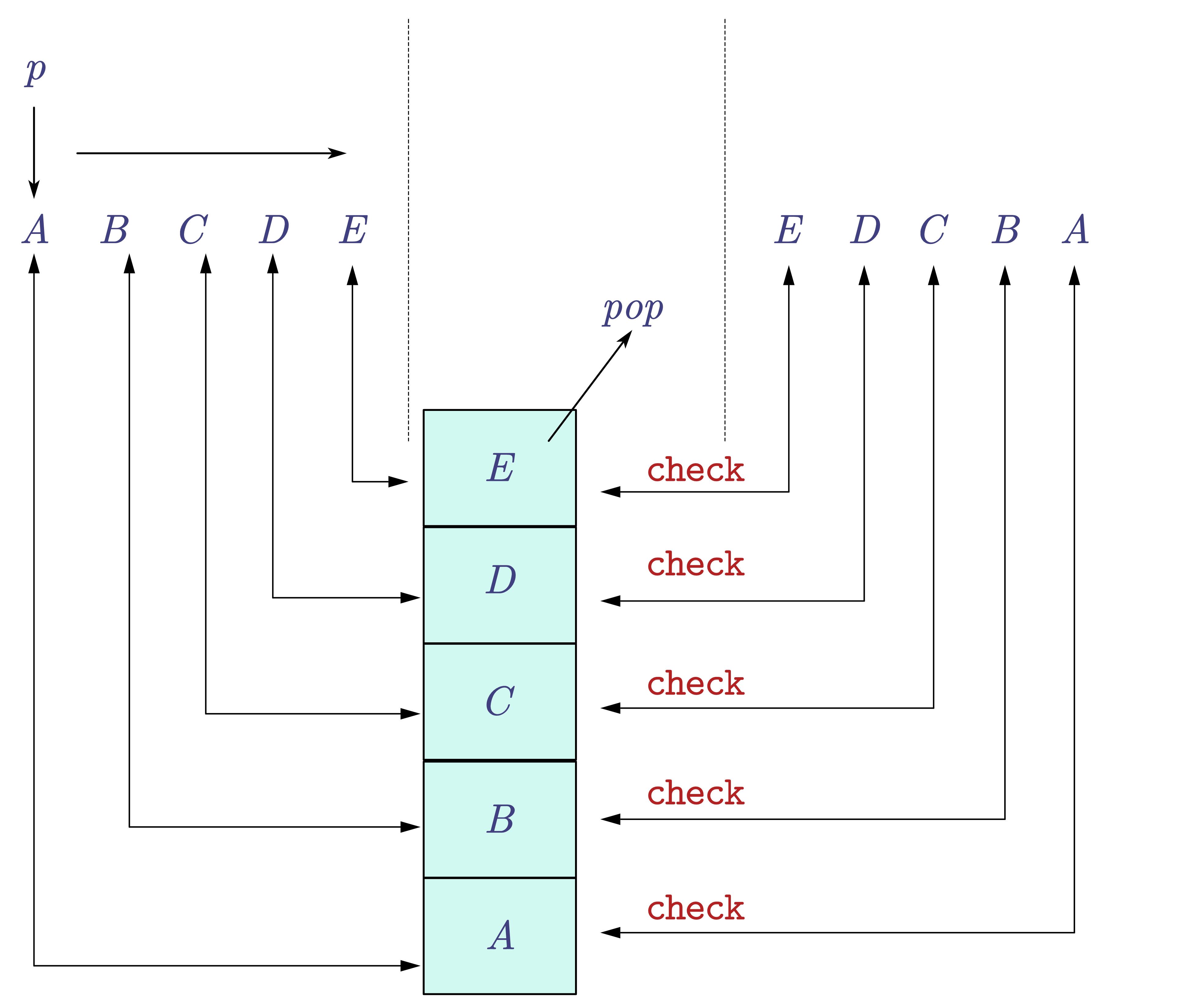
#### 3.2.2概要设计

用户函数bool solve\_oj\_3(int n, deque\_chain<T> &res)用来接收res链表**(虽然这里是双向链表类,但是我们在实现的时候并没有使用fa指针,所以也是符合题目要求的)**

#### 3.2.3 算法分析

**算法说明:**

先遍历到**待查找长度**的一半,并将其放入一个栈中,接着将后半部分依次与栈顶元素比较即可,如图:



**需要注意,如果查找长度为奇数,那么要先弹出一下栈顶元素.**

**时间复杂度分析:**

由于只遍历了一遍链表,故时间复杂度为

#### 3.3.4使用说明:

输入说明 :

第一行：顺序表A的数据元素的数据类型标记

第二行：待判断对称性的链表长度

第三行：单链表A的数据元素（数据元素之间以空格分隔）

注意：单链表A中的元素个数可能小于，也可能大于。如果元素个数小于，则判断整个单链表是否对称。

**运行界面如下:在Clion2023.1,MVSC 17.0编译器的环境下可以正常运行**



#### 3.3.5 测试结果与分析

本题测试流程图如下:



故严格来将共需要测试:组数据,但是为了简便,我们仅测试其中的几组:

* 第一组数据:err测试,通过

INPUT:

5

OUT:

err

* 第二组测试:INT类型,满足且,极端情况,仅一个元素,通过;

INPUT:

0

100

1

OUT:

1

true

* 第三组测试:char类型,满足,普通情况,对称,通过

INPUT:

2

5

a b c b a d s a w

OUT:

a->b->c->b->a->d->s->a->w

true

* 第四组测试:string类型,满足,普通情况,不对称,通过

INPUT:

3

6

HI FATHER HEY GIRL PARENT APPLE BANANA PEACH TEACHER DAD

OUT:

HI->FATHER->HEY->GIRL->PARENT->APPLE->BANANA->PEACH->TEACHER->DAD

false

* 第五组测试,double类型,满足，对称,通过

INPUT:

1

500

1.1 1.2 1.3 1.4 1.4 1.3 1.2 1.1

OUT:

1.1->1.2->1.3->1.4->1.4->1.3->1.2->1.1

true

* 第六组测试,int类型,满足,对称,大数据测试,通过

INPUT:

0

10000

55 44 48 50 68 36 57 68 29 64 57 69 20 39 54 32 39 47 34 55 53 52 42 20 59 70 68 21 66 40 59 38 35 30 31 30 32 25 68 32 28 57 21 64 38 45 67 64 33 70 59 22 46 54 69 47 27 50 36 67 44 50 42 48 69 56 20 20 56 69 48 42 50 44 67 36 50 27 47 69 54 46 22 59 70 33 64 67 45 38 64 21 57 28 32 68 25 32 30 31 30 35 38 59 40 66 21 68 70 59 20 42 52 53 55 34 47 39 32 54 39 20 69 57 64 29 68 57 36 68 50 48 44 55

OUT:

55->44->48->50->68->36->57->68->29->64->57->69->20->39->54->32->39->47->34->55->53->52->42->20->59->70->68->21->66->40->59->38->35->30->31->30->32->25->68->32->28->57->21->64->38->45->67->64->33->70->59->22->46->54->69->47->27->50->36->67->44->50->42->48->69->56->20->20->56->69->48->42->50->44->67->36->50->27->47->69->54->46->22->59->70->33->64->67->45->38->64->21->57->28->32->68->25->32->30->31->30->35->38->59->40->66->21->68->70->59->20->42->52->53->55->34->47->39->32->54->39->20->69->57->64->29->68->57->36->68->50->48->44->55

true

### 3.4 搜索插入位置

#### 3.4.1题目及其要求

给定一个排序数组和一个目标值，在数组中找到目标值，并返回其索引。如果目标值不存在于数组中，返回它将会被按顺序插入的位置。

请必须使用时间复杂度为的算法。

#### 3.4.2概要设计

设计函数通过**二分查找**的方式来找到插入位置

#### 3.4.3算法分析

**算法说明:**

本题核心代码如下:

**int Find(vector<int> &n, int num) {  
 int low = 0;  
 int high = n.size() - 1;  
 int mid = (low + high) / 2;  
 while (low < high) {  
 if (n[mid] == num) {  
 return mid;  
 } else {  
 if (n[mid] > num) {  
 high = mid - 1;  
 mid = (low + high) / 2;  
 } else {  
 low = mid + 1;  
 mid = (low + high) / 2;  
 }  
 }  
 }  
 return mid + 1;**

通过设置两个指针,分别指向表头和表尾,进行二分查找,最终找到的**mid**有两种可能即mid=num或者mid≠num但无论是那种情况,mid都是num在这个数组中最大下界的位序,返回mid+1即可,**但是仔细分析,会发现这样mid永远无法等于0,即如果有个元素是要插入在表头的,这个函数会返回错误的结果,所以可以特别判定一下如果这个插入的元素小于等于当前的表头,则应该直接输出0;**

**时间复杂度分析**

由于是使用二分查找的方式,故时间复杂度为

#### 3.4.4使用说明

输入说明 :

输入三行：

第一行输入一个整数n表示数组nums的长度。

第二行输入n个整数表示数组nums的元素。

第三行输入一个整数表示需要查找的目标值target.

提示:



为无重复元素的升序排列数组



**运行界面如下:在Clion2023.1,MVSC 17.0编译器的环境下可以正常运行**

图形用户界面, 文本, 应用程序

描述已自动生成

#### 3.4.5测试结果与分析

* 第一组数据:特殊情况,表中仅一个元素的插入,正确:

IN:

1

1

1

OUT:

0

* 第二组数据,特殊情况, 表中仅一个元素插入,但其不等于表内的元素,正确

IN:

1

1

2

OUT:

1

* 第三组数据:插入表头,正确

IN:

5

1 2 3 4 5

-1

OUT:

0

* 第四组数据:普通插入(元素在数组中),正确

IN:

10

1 2 3 4 5 6 7 8 9 10

5

OUT:

4

* 第五组数据,普通插入(元素不在数组中),正确

IN:

10

1 2 3 4 5 6 7 8 10 11

9

OUT:

8

* 第六组数据,特殊情况,要插入在表尾,正确

IN:

10

1 2 3 4 5 6 7 8 9 10

11

OUT:

10

* 第七组数据,随机数大样例,采用C++ std::random\_device以及uniform\_int\_distribution类以及mt19937\_64随机数引擎,来生成分布均匀的符合题目要求的真随机数,运行正常,**用时23ms符合二分查找的时间复杂度**

IN:

815

-9958 -9934 -9934 -9929 -9920 -9914 -9864 -9859 -9850

-9731 -9703 -9685 -9588 -9571 -9492 -9486 -9474 -9468 -9456

-9444 -9414 -9351 -9304 -9292 -9203 -9165 -9163 -9162 -9117

-9115 -9112 -9104 -9075 -9063 -9041 -9016 -8994 -8975 -8961

-8953 -8901 -8861 -8827 -8815 -8808 -8730 -8725 -8713 -8710

-8646 -8642 -8636 -8570 -8564 -8563 -8536 -8535 -8532 -8466

-8457 -8347 -8284 -8279 -8277 -8220 -8191 -8174 -8123 -8098

-8028 -8006 -7975 -7962 -7948 -7932 -7843 -7761 -7725 -7713

-7571 -7528 -7522 -7488 -7480 -7416 -7393 -7388 -7377 -7370

-7366 -7330 -7289 -7251 -7226 -7197 -7175 -7142 -7104 -7102

-7089 -7077 -7074 -7050 -6993 -6976 -6938 -6879 -6879 -6869

-6856 -6809 -6807 -6742 -6675 -6661 -6653 -6621 -6615 -6612

-6602 -6548 -6535 -6520 -6491 -6472 -6470 -6460 -6445 -6437

-6432 -6421 -6420 -6373 -6368 -6355 -6306 -6299 -6298 -6278

-6268 -6257 -6256 -6249 -6247 -6188 -6166 -6154 -6031 -6008

-6004 -5990 -5936 -5914 -5900 -5891 -5876 -5854 -5847 -5846

-5830 -5807 -5803 -5800 -5780 -5775 -5768 -5766 -5763 -5735

-5601 -5517 -5486 -5472 -5471 -5448 -5416 -5407 -5376 -5370

-5335 -5317 -5296 -5289 -5269 -5250 -5244 -5203 -5126 -5089

-5044 -5007 -4998 -4994 -4971 -4965 -4932 -4900 -4880 -4759

-4699 -4647 -4642 -4624 -4621 -4511 -4450 -4440 -4376 -4371

-4367 -4361 -4333 -4288 -4277 -4272 -4267 -4185 -4174 -4168

-4100 -4091 -4045 -3929 -3922 -3917 -3874 -3851 -3849 -3845

-3832 -3806 -3766 -3756 -3754 -3720 -3697 -3657 -3625 -3604

-3543 -3459 -3392 -3388 -3375 -3360 -3358 -3355 -3261 -3253

-3252 -3250 -3244 -3209 -3176 -3174 -3173 -3168 -3135 -3126

-3122 -3065 -3058 -3045 -3030 -3019 -3013 -3007 -3000 -2982

-2979 -2977 -2974 -2967 -2918 -2883 -2833 -2791 -2757 -2752

-2723 -2695 -2686 -2660 -2573 -2569 -2560 -2514 -2482 -2480

-2378 -2346 -2340 -2299 -2245 -2240 -2229 -2208 -2204 -2195

-2186 -2152 -2147 -2139 -2109 -2106 -2078 -2072 -2053 -2031

-2029 -2013 -1996 -1978 -1945 -1945 -1885 -1834 -1826 -1810

-1763 -1742 -1725 -1697 -1694 -1678 -1624 -1620 -1581 -1566

-1527 -1454 -1446 -1437 -1434 -1405 -1386 -1378 -1377 -1325

-1317 -1312 -1310 -1298 -1263 -1254 -1206 -1201 -1173 -1093

-1056 -1047 -1035 -1013 -989 -960 -960 -910 -910 -907

-893 -886 -865 -861 -857 -821 -788 -766 -750 -708

-707 -692 -684 -613 -603 -598 -548 -532 -530 -521

-521 -496 -480 -406 -406 -404 -233 -193 -168 -161

-147 -145 -143 -120 -32 -24 36 42 68 118

225 228 256 289 316 333 346 355 369 428

452 457 506 510 510 525 586 601 631 632

653 683 724 738 739 755 759 792 800 840

856 868 913 919 1021 1030 1071 1071 1125 1186

1262 1268 1272 1306 1361 1369 1397 1410 1421 1428

1452 1471 1475 1505 1556 1558 1586 1635 1664 1689

1725 1752 1777 1780 1858 1870 1902 1913 1946 1984

2073 2119 2225 2238 2247 2297 2322 2355 2376 2391

2393 2415 2480 2519 2531 2533 2557 2564 2580 2588

2604 2614 2640 2647 2650 2732 2772 2782 2789 2799

2821 2839 2910 2934 2949 2950 2956 3035 3047 3066

3071 3080 3101 3104 3106 3138 3146 3164 3165 3171

3173 3174 3181 3187 3204 3266 3312 3313 3315 3333

3373 3408 3410 3424 3450 3462 3544 3547 3555 3579

3620 3637 3642 3660 3700 3727 3790 3804 3822 3866

3869 3892 3897 3949 3966 3998 4023 4042 4057 4072

4084 4092 4112 4114 4216 4227 4228 4242 4251 4268

4299 4341 4346 4349 4382 4393 4423 4490 4490 4492

4518 4522 4540 4559 4576 4605 4641 4649 4695 4701

4745 4777 4796 4801 4814 4817 4873 4923 4933 4984

5056 5094 5106 5117 5142 5142 5165 5215 5231 5255

5294 5312 5368 5381 5401 5413 5422 5425 5472 5503

5536 5554 5566 5634 5636 5653 5653 5724 5752 5769

5776 5797 5813 5819 5830 5843 5860 5885 5945 5960

5986 5992 6021 6115 6129 6203 6214 6261 6276 6301

6304 6353 6397 6398 6458 6491 6508 6514 6516 6532

6545 6557 6578 6632 6642 6656 6706 6738 6770 6776

6815 6838 6849 6874 6881 6882 6891 6894 6896 6898

6932 6934 6940 6992 7038 7053 7067 7108 7163 7195

7196 7199 7210 7219 7227 7228 7235 7281 7291 7306

7318 7326 7351 7377 7385 7395 7423 7455 7468 7471

7491 7492 7500 7530 7551 7554 7572 7641 7643 7698

7712 7756 7757 7765 7806 7813 7830 7844 7891 7906

7925 7925 7978 7998 8019 8037 8050 8071 8117 8138

8154 8169 8170 8201 8214 8246 8298 8316 8383 8468

8493 8507 8508 8509 8520 8521 8521 8537 8554 8570

8572 8591 8596 8638 8746 8787 8853 8856 8863 8863

8878 8880 8889 8907 8948 9098 9104 9108 9119 9127

9136 9143 9207 9222 9242 9259 9267 9267 9274 9279

9281 9290 9309 9338 9339 9355 9371 9380 9409 9417

9422 9424 9440 9448 9467 9521 9543 9560 9608 9670

9829 9835 9866 9885 9910 9974

**3266**

OUT:

524

### 3.5有效的完全平方数

#### 3.5.1题目以及要求

给定一个 正整数 *num* ，编写一个函数，如果 *num* 是一个完全平方数，则返回 *true* ，否则返回 *false* 。

进阶：不要 使用任何内置的库函数，如  *sqrt* 。

#### 3.5.2概要设计

设计函数**void solve\_oj\_5()**利用二分查找的思想来判断是否正确

#### 3.5.3算法分析

本题核心代码如下:

**void solve\_oj\_5()  
{  
 int *num*;  
 cin>>*num*;  
 int *left* = 0, *right* = *num*;  
 while (*left* <= *right*) {  
 if(*left*==*right*&&*num*!= sqrt(*num*)\* sqrt(*num*)){break;}  
 long long *mid* = (*right* + *left*) / 2 ;  
 long long *square* = *mid* \* *mid*;  
 if (*square* < *num*) {*left* = *mid* + 1;  
 } else if (*square* > *num*) { *right* = *mid* - 1;  
 } else { cout<<"true";  
 return; }  
 }  
 cout<<"false";  
 return;  
}**

**算法分析:**

使用二分查找的思想,初始时left=0,right=num,**注意,这里如果left设为1会导致最后结果出现问题(TLE/WA的情况均有发生)**,在循环之中,不断通过二分找到mid的值,判断mid\*mid是否等于num,**注意,这里的square需要设置为long long ,否则会造成int 越界导致square变成负数,从而导致循环无法停止,造成TLE.**

**时间复杂度分析:**

由于依旧是二分查找的思想,故时间复杂度是的

#### 3.5.4使用说明

输入说明 :

输入一个整数num.

提示：

**运行界面如下:在Clion2023.1,MVSC 17.0编译器的环境下可以正常运行**

图形用户界面, 文本, 应用程序

描述已自动生成

#### 3.5.5测试结果与分析

* 第一组数据:特殊用例:1

IN:

1

OUT:

true

* 第二组数据:普通用例,完全平方数

IN:

4

OUT:

true

* 第三组数据:普通用例,一个非常接近完全平方数的数字

IN:

24

OUT:

False

* 第四组数据,大数检验

IN:

2147483648

OUT:

False

### 3.6寻找比目标字母大的最小字母

#### 3.6.1题目以及要求

给你一个排序后的字符列表 letters ，列表中只包含小写英文字母。另给出一个目标字母 target，请你寻找在这一有序列表里比目标字母大的最小字母。

在比较时，字母看成是依序循环出现的。即：如果目标字母大于等于letters中最后一个字符，则返回第0个字符。

#### 3.6.2概要设计

设计函数: **void solve\_oj\_6()通过二分查找的思想,找到最小的上界.**

#### 3.6.3算法分析

**本题核心代码如下:**

**if (*res*[*mid*] == *c*) {  
 if (*mid* + 1 >= *res*.size()) {cout << *res*[0];  
 } else {  
 for (int *i* = *mid*; *i* < *res*.size(); *i*++) {  
 if (*res*[*i*] > *c*) {cout << *res*[*i*];return;  
 }  
 }  
 }return;  
 } else {  
 if (*res*[*mid*] < *c*) {  
 *low* = *mid* + 1;*mid* = (*low* + *high*) / 2;}**

**else {*high* = *mid* - 1;*mid* = (*low* + *high*) / 2;}  
 }  
}  
if (*mid* + 1 >= *res*.size()&&res[mid]<=c) {cout << *res*[0];  
} else {  
 if (*res*[*mid*] > *c*) {cout << *res*[*mid*];  
 } else {  
 for (int *i* = *mid*; *i* < *res*.size(); *i*++) {  
 if (*res*[*i*] > *c*) {cout << *res*[*i*];return;  
 }}}**

由于是有序表,故可以通过二分查找定位到最小的上界,最后特别判定,如果是比表尾元素要大,则返回表头元素.

**时间复杂度分析:**

依旧是二分查找的思想,但是查找到mid若等于target时不知mid所在的值是否会重复出现,故需要做一个for循环遍历到第一个大于其的值,这个故时间复杂度是的

#### 3.6.4使用说明

输入说明 :

输入三行：

第一行输入一个整数n表示数组的长度。

第二行输入n个小写英文字母表示数组letters的元素。

第三行输入一个小写英文字母表示目标字符target.

提示：



    letters[i] 是一个小写字母

    letters 按非递减顺序排序

    letters 最少包含两个不同的字母

target 是一个小写字母

**运行界面如下:在Clion2023.1,MVSC 17.0编译器的环境下可以正常运行**

形状

中度可信度描述已自动生成

#### 3.6.5测试结果与分析

* 第一组数据:特殊数据,所有字母均一样,查找的字符大于这个字符

IN:

5

a a a a a

b

OUT:

a

* 第二组数据:特殊数据, 所有字母均一样,查找的字符等于这个字符

5

a a a a a

a

a

* 第三组数据,普通数据,无重复出现的元素,target在字符集中

IN:

10

a b c d e h i j k m

e

OUT:

H

* 第四组数据,普通数据,无重复出现的元素,target不在字符集中

IN:

10

a b c d e h i j k m

f

OUT:

h

* 第六组数据,大数据测试,target在字符集中

IN:

152

a a a a a a a a b b b c c c c c c c c d d d d d d d d d d d d e e e e e e f f f f f f g g h h h h h h h h h i i i i j j j j k k k k k k l l l l l l m m m m m m m m m n n n n n o o o o o p p q q q q q q q r r r r r r t t t t t t t t t u u v v v v v w w w x x x x x y y y y y y y y y y y y z z z z z z z z

i

OUT:

j

### 3.7矩阵中战斗力最弱的 K 行

#### 3.7.1题目以及要求

给你一个大小为的矩阵 mat，矩阵由若干军人和平民组成，分别用 1 和 0 表示。

请你返回矩阵中战斗力最弱的 *k* 行的索引（行号），按从最弱到最强排序。

如果第 *i* 行的军人数量少于第 *j* 行，或者两行军人数量相同但 *i* 小于 *j*，那么我们认为第 *i* 行的战斗力比第 *j* 行弱。

军人总是排在一行中的靠前位置，也就是说 1 总是出现在 0 之前。

#### 3.7.2概要设计

设计结构体**struct node**来存放值与下标,设置集合**set<node>op;**起到自动排序的作用

#### 3.7.3算法分析:

**算法说明:**

本题核心代码如下:

**for (int *i* = 0; *i* < *m*; *i*++) {  
 int *s* = 0;  
 for (int *j* = 0; *j* < *n*; *j*++) {  
 int *p* = 0;  
 cin >> *p*;  
 if (*p* == 1) {  
 *s*++;  
 }  
 }  
 op.insert({*i*, *s*});  
}**

输入的时候按行统计1出现的次数,打包成结构体插入*op*集合,最后直接输出输出集合中前*k*个结构体中的位序即可.

**时间复杂度分析:**

Set的每一次操作用时,一共有**次插入操作,总时间复杂度为

#### 3.7.4使用说明:

输入说明 :

输入若干行：

第一行输入两个整数m和n，表示矩阵的行数和列数。

之后m行，每行输入n个整数（0或1）表示矩阵的元素。

最后一行输入一个整数.

提示：



    矩阵的元素 不是 0 就是 1

**运行界面如下:在Clion2023.1,MVSC 17.0编译器的环境下可以正常运行**

图片包含 形状

描述已自动生成

#### 3.7.5测试结果与分析

* 第一组数据,特殊数据全是0,测试通过

IN:

3 3

0 0 0

0 0 0

0 0 0

3

OUT:

0 1 2

* 第二组数据,特殊数据,全是1,测试通过

IN:

3 3

1 1 1

1 1 1

1 1 1

2

OUT:

0 1

* 第三组数据,普通数据,但保证各排1和0的数量是不同的,测试通过

IN:

5 5

1 0 0 0 0

1 1 0 0 0

1 1 1 0 0

1 1 1 1 0

1 1 1 1 1

4

OUT:

0 1 2 3

* 第四组数据:普通数据,且某些排0和1的数量相同,且分布随机,测试通过

IN:

6 6

1 1 1 0 0 0

1 0 0 0 0 0

1 0 0 0 0 0

1 1 0 0 0 0

1 1 1 0 0 0

1 1 1 1 1 1

5

OUT:

1 2 3 0 4

* 第五组数据:普通数据,但不为方阵, 且某些排0和1的数量相同,且分布随机,测试通过

IN：

5 4

1 0 0 0

1 1 0 0

1 1 1 1

1 1 1 0

1 0 0 0

4

OUT:

0 4 1 3

* 第六组数据: 普通数据,但不为方阵, 且某些排0和1的数量相同,且分布随机,测试通过

IN:

7 8

1 1 1 0 0 0 0 0

1 1 1 1 0 0 0 0

1 1 1 1 1 1 1 1

1 1 1 0 0 0 0 0

1 0 0 0 0 0 0 0

1 1 0 0 0 0 0 0

1 1 1 1 1 1 1 1

6

OUT:

4 5 0 3 1 2

* 第七组数据,大样例测试,测试set插入的时间复杂度

IN:

50 40

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

30

OUT:

45 21 1 3 20 31 43 48 22 25 32 41 47 10 13 24 38 46 49 8 11 17 27 34 35 5 16 14 28 33

### 3.8找到和最大的长度为 K 的子序列

#### 3.8.1题目以及要求

给你一个整数数组 *nums* 和一个整数 *k* 。你需要找到 *nums* 中长度为 *k* 的子序列 ，且这个子序列的和最大 。

请你返回 任意 一个长度为 *k* 的整数子序列。

子序列定义为从一个数组里删除一些元素后，不改变剩下元素的顺序得到的数组。

#### 3.8.2算法设计

设计链表**list<int>jjj**用来起到快速删除的作用即可;

#### 3.8.3算法分析

**算法说明:**

由于要保留原有的位序,我们不能盲目的排序,而是需要把*n-k*个最小的元素删除,在STL中,我们可以使用**min\_element(list<int>.begin(),list<int>.end())**来定位到最小元素的迭代器.

故本题的核心代码如下:

**for(int *i*=0;*i*<*n*-*k*;*i*++)  
{  
 *jjj*.erase(min\_element(*jjj*.begin(),*jjj*.end()));  
}**

**时间复杂度分析:**

由于是使用链表这类数据结构,单次删除为,使用**min\_element**单次查找为,故总共时间复杂度为

而由于*k*小于*n*的特性,可以简写为

#### 3.8.4使用说明

输入说明 :

输入三行：

第一行为一个整数*n*代表数组*nums*的长度。

第二行输入*n*个整数代表数组*nums*的元素。

第三行为一个整数*k*。

提示：



**运行界面如下:在Clion2023.1,MVSC 17.0编译器的环境下可以正常运行**

形状

中度可信度描述已自动生成

#### 测试结果与分析

* 第一组数据:输入数据保证没有负数,特殊样例,测试通过

IN:

10

2 3 4 5 6 7 8 1 2 3

5

OUT:

4 5 6 7 8

* 第二组数据:输入数据保证没有正数,且保证无重复数据,特殊样例,测试通过

IN:

10

-1 -5 -10 -2 -3 -7 -6 -19 -20 -40

5

OUT:

-1 -5 -2 -3 -6

* 第三组数据,输入数据有正有负,但保证即无需调用删除操作,测试通过

IN:

10

-5 -4 -3 -1 -9 1 2 5 6 7

10

OUT:

-5 -4 -3 -1 -9 1 2 5 6 7

* 第四组数据,普通数据, 元素顺序随机,测试通过

IN:

10

-5 -6 -7 1 0 2 3 -8 9 10

5

OUT:

1 2 3 9 10

* 第五组数据:普通数据,元素顺序随机,测试通过

IN:

10

2 3 1 1 3 2 4 5 6 2

4

OUT:

3 4 5 6

* 第六组数据,特殊数据,所有元素全部相同,测试通过

IN：

10

1 1 1 1 1 1 1 1 1 1

1

OUT：

1

* 第七组数据,大样例输入,测试时间复杂度（当*n=500*时,用时25ms）测试通过

IN:

500

-719 -370 -2458 -8528 -9234 -7365 -5256 -5464 -7500 -9755 -6572 -4500 -4956 -5965 -3496 -3378 162 271 -3217 -7871 -313 -1256 -7577 -7686 -317 -7433 -9817 -1431 545 -9055 -1501 -8418 -4751 -2626 -2634 -4470 -2736 194 -145 -4828 -47 -5992 -3846 -8943 -7289 -7727 -6512 811 -5632 -3555 -7110 -8740 320 -6746 -6665 -2295 -5710 -1123 -3083 -4767 -5470 -5867 -2282 -361 -7177 -4868 -5175 -9022 -1326 -6173 -9445 -5145 -8463 254 -3247 -7873 -425 -6956 -6584 -4316 -1379 -7532 -3010 -6116 -1274 -9205 -5285 406 -3847 -3536 -1600 -8671 -8520 -5508 -1305 -4367 -5799 -2407 -7245 -9881 -2411 -8171 -847 -9685 326 -8284 -2509 -7006 676 723 -1173 -7368 -1986 -6027 -4423 -6374 -294 -4536 -3211 -7733 -5001 -5423 -597 -108 -5103 -6010 -2157 -4705 -9625 22 88 -6596 400 -5517 -2272 214 -6573 -3263 -5898 -2205 614 -1810 -1684 -2192 -7495 -2397 -206 -4264 -234 821 -1280 -6127 -1917 -713 -1145 -4 -8071 -4523 -1325 -881 -2318 -1737 -7151 -7476 -6019 -1411 -3871 -6279 -6464 -1398 277 -7034 -5922 -9426 -6644 -7570 -8751 -2204 -5427 -7252 -8320 -6416 -7343 -8420 -4332 -5075 -1088 -7201 -4450 -9983 685 -8676 -1221 -6459 -6407 374 -7866 -5002 -3964 -76 -4521 -5076 280 -7629 -1828 -8358 -4747 -1199 -7903 752 -2107 -6978 -2190 -9251 -8956 -6631 -8603 723 -5035 -2555 -4063 -6745 389 -3209 -9680 379 -6312 -8081 -8171 -4739 -4032 -2940 -103 -1120 -6791 -1174 -8835 -3415 508 -5449 -8522 -8148 -650 -7214 -7054 -4098 -7392 -633 -6778 -6328 -6374 -9978 -1726 -3257 -2967 -7822 -653 -2210 -8115 -5078 -1149 -1568 -6405 -793 -3032 -2842 -7094 -72 -3587 -637 -1725 -5374 -2480 -5235 -5041 -7342 -6473 -7621 -3608 -1192 -529 -4391 298 -3598 -5898 -2703 -1871 -5765 -653 -1085 -7420 -6036 -3531 -6136 -8402 -2103 681 -9513 -692 904 -4382 -2680 -2845 -1454 -9074 -9390 480 -766 -8422 -2854 -9406 -7199 -2922 81 -3305 436 -5591 -5885 -4426 -3539 -1685 -1608 -2546 -4710 -1270 -9350 -7790 -699 -7145 -4321 -9830 887 -6797 -8110 -5165 -6775 -8945 -5461 -800 -3757 -8604 -5048 -143 -9720 -2456 -9478 -4478 -4573 -1851 -2245 -6801 -6592 -5895 -8432 -9218 -3192 -4084 -49 -3152 -9861 913 -7467 -1182 -6052 -93 -8170 -1420 -2758 -4423 -7349 -9919 -9645 -4351 -9589 -7751 -2749 -5718 592 758 -3084 -7661 -5316 -2743 -9193 -1205 -2264 721 -5426 -666 -7090 -1321 -7904 -6079 -7196 -9734 -4246 -332 -1604 -7679 -7995 -907 -2790 -6059 -7128 -7369 -301 -3931 -500 -1718 -5121 -2647 -4195 -5651 -8045 -4168 -1648 -1712 -4895 -2855 -9487 -7273 -6021 -7479 -4808 -2056 -8112 -3692 -517 -6435 -9345 -2136 440 -2334 -6285 -9025 345 -7838 -8008 -4118 -6304 -698 278 -4914 99 -3221 -8770 -5164 -445 -1492 -4033 -8948 531 -7511 -8348 -359 -3459 -3490 -3756 -5715 -4192 -7956 -8460 335 -3396 -7029 -7736 -616 -3189 -1514 -1617 835 -8035 -6202 -8299 -5721 -385 -1494 -5170 -9082 -6132 374 -6683 -9026 -6468 -7639 -664 -5470 -4716 506 -4223 -4903 710 -290 631 -1629 -8494 -8403 -3779 -3133 697

300

OUT:

-719 -370 -2458 -5256 -5464 -4500 -4956 -3496 -3378 162 271 -3217 -313 -1256 -317 -1431 545 -1501 -4751 -2626 -2634 -4470 -2736 194 -145 -4828 -47 -3846 811 -3555 320 -2295 -1123 -3083 -4767 -5470 -2282 -361 -4868 -5175 -1326 -5145 254 -3247 -425 -4316 -1379 -3010 -1274 -5285 406 -3847 -3536 -1600 -1305 -4367 -2407 -2411 -847 326 -2509 676 723 -1173 -1986 -4423 -294 -4536 -3211 -5001 -5423 -597 -108 -5103 -2157 -4705 22 88 400 -2272 214 -3263 -2205 614 -1810 -1684 -2192 -2397 -206 -4264 -234 821 -1280 -1917 -713 -1145 -4 -4523 -1325 -881 -2318 -1737 -1411 -3871 -1398 277 -2204 -5427 -4332 -5075 -1088 -4450 685 -1221 374 -5002 -3964 -76 -4521 -5076 280 -1828 -4747 -1199 752 -2107 -2190 723 -5035 -2555 -4063 389 -3209 379 -4739 -4032 -2940 -103 -1120 -1174 -3415 508 -5449 -650 -4098 -633 -1726 -3257 -2967 -653 -2210 -5078 -1149 -1568 -793 -3032 -2842 -72 -3587 -637 -1725 -5374 -2480 -5235 -5041 -3608 -1192 -529 -4391 298 -3598 -2703 -1871 -653 -1085 -3531 -2103 681 -692 904 -4382 -2680 -2845 -1454 480 -766 -2854 -2922 81 -3305 436 -4426 -3539 -1685 -1608 -2546 -4710 -1270 -699 -4321 887 -5165 -5461 -800 -3757 -5048 -143 -2456 -4478 -4573 -1851 -2245 -3192 -4084 -49 -3152 913 -1182 -93 -1420 -2758 -4423 -4351 -2749 592 758 -3084 -5316 -2743 -1205 -2264 721 -5426 -666 -1321 -4246 -332 -1604 -907 -2790 -301 -3931 -500 -1718 -5121 -2647 -4195 -4168 -1648 -1712 -4895 -2855 -4808 -2056 -3692 -517 -2136 440 -2334 345 -4118 -698 278 -4914 99 -3221 -5164 -445 -1492 -4033 531 -359 -3459 -3490 -3756 -4192 335 -3396 -616 -3189 -1514 -1617 835 -385 -1494 -5170 374 -664 -5470 -4716 506 -4223 -4903 710 -290 631 -1629 -3779 -3133 697

### 3.9无人机方阵

#### 3.9.1题目以及要求

在 「挑战赛」 开幕式的压轴节目 「无人机方阵」中，每一架无人机展示一种灯光颜色。 无人机方阵通过两种操作进行颜色图案变换：

  (1)  调整无人机的位置布局

  (2)  切换无人机展示的灯光颜色

给定两个大小均为 *N\*M* 的二维数组 *source* 和 *target* 表示无人机方阵表演的两种颜色图案，由于无人机切换灯光颜色的耗能很大，请返回从 *source* 到 *target* 最少需要多少架无人机切换灯光颜色。

注意： 调整无人机的位置布局时无人机的位置可以随意变动。

#### 3.9.2概要设计

设计**int vis[10001]**进行哈希即可

#### 3.9.3算法分析

**算法说明:**

我们对第一个方阵进行哈希,将对应的**vis[*num*]++;**在输入第二个方阵数据时,我们只需要检查对应的***num***位置是否非零,如果非零,则无需变色,我们将该处的值-1即可,而如果为零则需要将答案***ans++***

核心代码如下:

**for(int *i*=0;*i*<*m*;*i*++)  
{  
 for(int *j*=0;*j*<*n*;*j*++)  
 {  
 int *num*;  
 cin>>*num*;  
 if(vis[*num*]>0)  
 {  
 vis[*num*]--;  
 }  
 else  
 {  
 *ans*++;  
 }  
 }  
}**

**时间复杂度分析:**

由于我们是边输入边计算,故如果除去输入的时间,整个过程是,若算上输入时间,则整个过程是

#### 3.9.4使用说明

输入说明 :

输入若干行：

第一行为两个整数*n*和*m，n*代表二维数组的行数，*m*代表二维数组的列数。

而后*n*行，每行输入*m*个整数，代表*source*数组的元素。

再*n*行，每行输入*m*个整数，代表*target*数组的元素。

提示：



**运行界面如下:在Clion2023.1,MVSC 17.0编译器的环境下可以正常运行**

图形用户界面, 文本, 应用程序

描述已自动生成

#### 3.9.5测试结果与分析

* 第一组数据:特殊用例,满足*m=n,*且第一排方阵与第二排方阵相同,测试通过

IN:

4 4

1 1 1 1

1 1 1 1

1 1 1 1

1 1 1 1

1 1 1 1

1 1 1 1

1 1 1 1

1 1 1 1

OUT:

0

* 第二组数据:特殊用例,满足且第一个方阵和第二个方阵的元素均不相等,测试通过

IN：

4 5

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 25

26 27 28 29 30

31 32 33 34 35

36 37 38 39 40

OUT：

20

* 第三组数据:普通用例,满足且同一个方阵不存在相同的元素,测试通过

IN：

4 4

1 3 4 2

5 6 8 7

13 16 10 9

78 64 56 90

78 90 64 1

75 2 3 4

9 10 17 18

23 24 25 26

OUT：

7

* 第四组数据普通用例,满足,且方阵中均存在相同的元素,测试通过

IN：

5 4

1 2 3 4

2 2 3 4

3 2 4 3

1 1 1 1

6 6 6 6

4 3 2 1

7 6 3 1

9 8 7 7

4 6 8 7

3 2 1 2

OUT:

7

* 第六组数据,边界条件,满足

IN：

1 1

1

2

OUT:

1

* 第七组数据,大数据测试，运行时间23ms,测试通过

IN:

35 34

7123 6828 7016 5782 4163 7028 6203 5585 846 9255 9699 5071 5796 9181 8064 2415 8177 4971 9148 6459 821 9754 4950 3678 8143 9537 6266 3075 9968 1286 507 5781 2571 282

5762 2498 3572 9848 7074 3765 6622 9589 5590 78 7202 9935 7950 3866 5092 2782 6875 2762 1554 6340 9223 991 6630 326 43 9704 7924 2443 5341 7635 4946 3355 2870 613

369 3801 1222 6555 8052 2546 1013 48 7151 9017 8917 3547 8089 8354 9098 2384 1377 6930 2680 3383 7806 1899 1863 5264 4867 9700 9248 5753 1128 895 9690 3425 6959 9038

3558 6384 6369 6731 2077 541 7726 574 3932 6682 2043 8865 7999 5246 2003 7664 1291 3710 1549 8314 6889 390 8282 5229 5835 4342 1797 2509 1331 9174 6659 1561 4834 6162

6532 6274 2780 9098 532 2144 6699 3929 7182 9990 6603 5547 7829 4222 2289 6686 5479 7021 7387 2916 6719 1664 4570 7496 3180 1327 5419 6073 4243 793 2922 3933 2018 824

4347 981 5101 1458 6041 7832 2916 7371 6987 5944 5771 5555 3478 6774 5968 8974 2364 156 1784 3785 4426 2680 4650 2479 5713 8614 8321 4773 5801 1961 2030 4487 41 865

9130 6589 8692 5401 8670 368 3492 6772 350 3081 1632 7622 2438 5666 171 8363 151 4440 4549 8081 1490 7506 8953 313 927 5004 9071 2796 2237 2594 197 908 4396 6460

4516 732 5285 79 729 202 2653 8041 8854 288 3680 1932 8793 5558 2367 5154 9977 4041 9967 4781 2051 409 2795 1059 2503 606 6503 859 8215 3302 6278 4411 6096 7747

9724 14 4524 9392 5230 9637 4163 6734 259 1114 3148 7476 1736 3144 9982 9600 3353 6227 7822 2646 4260 6826 8990 6616 3332 2414 1900 5521 5656 7347 6551 8911 374 9639

7336 2647 2622 1916 8085 2423 5866 6357 9252 6441 4739 602 9502 613 6906 3796 1982 8487 6194 2601 928 9174 1879 517 2574 295 8749 8964 8446 6311 9022 8135 5109 7196

9468 350 281 8952 1158 2591 7730 1596 4423 5712 7580 704 8639 8160 2485 1307 7270 5552 4041 4116 582 2898 6944 1550 4014 3380 5467 1621 4719 2201 5816 1883 8920 8064

2602 1946 7834 2871 3550 4177 3208 5542 6009 1409 4488 1131 8136 3635 5860 4428 8756 8046 5920 538 634 840 5294 2511 4558 2658 5881 1971 6457 6027 7199 9171 7173 4785

6777 2977 1950 137 8101 6980 8908 7301 9954 9472 1628 3558 2214 4045 9583 6762 9024 1209 1029 951 3908 8253 8482 8762 2435 9735 864 1341 2642 1701 7789 7298 1676 1320

9882 5174 935 2666 9082 8655 4367 3383 4399 7238 2728 2825 5069 4432 6460 8523 8984 6587 630 9364 595 841 9308 9308 9490 3723 5644 4583 1376 5467 7174 44 2983 1265

6661 9731 3297 6013 1823 9038 7828 4001 7385 6146 3614 3952 873 575 5880 3095 2547 1726 7733 6430 8262 1852 9933 9364 9773 9699 6398 1424 9549 6601 5327 4859 1380 4413

9250 2511 999 460 2934 7616 2801 7018 1919 8613 9099 4690 7686 823 6790 7630 6728 1659 6275 913 5042 6278 7919 1223 2185 4686 9890 6097 9232 4658 3273 5394 9594 942

2554 8146 2596 8079 3639 7922 8074 1218 5108 430 4794 8175 5267 1962 6281 3510 5009 428 8038 3956 5244 5784 5098 3804 4730 2299 5569 3835 2448 7414 8476 1568 8874 5226

6781 3125 3769 8172 6896 9956 3426 1582 6305 7533 2597 9937 3304 3301 3563 3590 5404 6667 6674 9999 6345 2430 8888 6515 3980 7869 7536 8875 8600 9763 3134 1443 9540 9035

8741 9632 6858 1499 3014 5646 3153 3116 8564 4832 5370 9831 560 3117 6530 2732 2633 3053 2646 5173 3079 4798 1224 4565 4477 1356 4200 3695 246 8518 8713 6913 9763 2353

1445 7676 5029 5289 3137 6954 8721 3337 3095 1867 2914 3354 6084 9197 7139 3336 6308 1359 8822 4243 9722 3394 1641 5970 9803 9154 7258 1392 5494 6617 7774 3936 1050 2794

7427 2963 6349 8687 2754 5636 7611 7078 281 7234 1793 7131 4090 4422 7034 8613 7808 6490 9395 2381 4473 5782 172 1368 6304 4783 1316 4927 198 5633 1457 3387 8928 3600

5308 5170 5496 8078 1012 4714 6650 7296 9454 1631 5481 8314 6533 3709 5162 7891 6056 9745 8581 7923 4291 6171 4730 8601 5039 9516 3095 6720 3480 3105 5839 7066 5354 3032

1271 4799 96 7621 883 7269 5603 4294 8451 719 6032 7042 3381 8094 633 5805 9142 2387 2175 495 9435 7922 4462 7900 1042 6759 2104 3714 4479 6470 5956 9755 246 1606

3320 6952 7514 5542 1035 7021 6213 3062 3685 267 5715 8953 3068 6153 9492 4785 4964 4291 2272 8809 9692 5334 1513 5371 6273 6817 6168 7411 6596 7132 8216 1498 5877 1987

7425 5848 6542 9004 1473 2559 1288 607 1347 343 6759 605 1668 7672 5206 3552 7437 3349 8796 363 8131 5941 1804 3264 931 3498 6461 6603 1460 2660 4860 1394 1634 5481

7611 912 9409 7179 8522 6755 9914 3245 4292 4916 1488 223 9172 3865 4050 5742 6133 3088 7492 2308 7526 75 220 5377 6631 3763 425 3781 6766 301 1456 6845 7001 3798

4138 9112 9850 1251 2254 5210 7344 5620 4288 7851 23 555 7865 9150 913 8783 8807 5160 841 2709 7968 2546 8540 8919 373 5866 3373 2937 4050 9841 4083 7989 8011 9978

3917 6983 4282 2854 9000 2718 3300 6587 48 694 2735 3587 5947 8622 7088 2522 3140 8109 7434 8689 3408 3713 3852 8180 8010 2800 1998 2691 4498 3689 2531 1186 2144 9124

6846 6572 7409 3609 6935 9550 2493 9459 9165 2226 749 4148 8967 9690 5073 1764 8954 3160 9215 7565 2832 8449 4245 8196 1227 5239 1031 2484 7448 154 3238 725 2331 4091

7645 7882 1858 6748 3509 5634 7913 8221 8499 4006 3675 5895 7361 3804 4381 3474 3545 449 623 2125 6430 2413 9767 5631 6378 5616 6893 6989 3846 6480 4640 7878 4542 7678

403 420 6459 8963 1940 2180 9367 6088 6625 7581 9191 8908 7723 7579 5347 9669 9059 2086 7701 9661 6566 8397 3037 3736 8639 3454 3706 4351 9543 9794 7265 1690 5131 3695

3716 5517 6375 4580 5516 5234 738 9519 6861 3977 6663 9318 1046 8992 6398 5548 7990 1621 1969 1537 7690 8602 422 3776 655 8340 2796 5262 2776 4763 7392 6140 6820 6154

8289 8653 4912 3937 9425 9940 1434 9675 1636 5882 7766 8345 3784 6594 8630 4143 7524 6982 5793 6511 7518 9879 4794 7576 3804 599 9522 3422 8291 7308 5140 9283 4239 3804

4282 8651 4278 540 1383 4872 3199 626 3059 2163 5292 2971 7407 1188 103 2296 174 7598 6190 205 136 5498 7532 1550 4040 6780 5939 269 1887 4303 4333 3170 9993 2125

1845 867 4364 5862 333 5313 7089 9088 8928 6079 3196 2835 5171 3071 1422 8496 8490 6829 3126 3617 9640 6261 9947 3664 9229 1247 7339 6566 8983 1531 7689 9063 4868 9448

650 3505 1973 7350 7049 6553 1191 338 4627 4345 3205 4724 9376 9998 1250 1315 9411 8072 1345 5065 3084 8476 7052 4434 9129 9370 7161 6471 5940 8085 4772 5291 7612 3242

9477 1839 2092 1326 3162 6004 8149 8999 129 1718 8947 1733 2376 1537 9259 6095 3505 7875 7546 6334 4486 2835 3639 8415 2471 1684 1593 6230 2653 228 1549 2898 5643 5415

2835 9316 2225 5229 4040 8788 6408 1885 5728 4352 7816 2500 1335 219 2286 443 9220 4688 68 2914 8123 9574 7705 1575 7707 7984 2820 1430 929 2749 8319 5349 8713 9216

8972 5735 7480 8426 2346 7133 4825 7049 7333 3421 2383 7971 122 3021 2011 5840 8980 1356 9159 8575 4997 3815 3938 8697 8312 1512 9772 3089 8657 8729 3909 4326 3978 7586

4972 9132 5131 4995 5837 4706 2929 1380 3811 5516 9227 9288 1487 772 2710 3347 6548 5672 8778 2078 6240 7342 2041 8301 8222 266 7300 108 4331 7237 53 7719 2642 3540

5499 316 4668 671 9567 2661 8476 821 8995 5161 4504 566 4641 5610 4781 5878 4023 9364 2774 8457 4203 9880 7808 5597 3050 5139 9448 4394 270 5471 2987 4840 2854 2668

3959 8464 5780 9245 1110 127 8478 8916 5541 568 2301 4285 4399 6158 7269 705 348 535 4203 6789 3892 7784 8244 9672 119 5258 6106 6138 5375 9746 6537 3982 7867 4907

4818 2715 7380 4363 7877 1120 1535 2318 6258 2722 6990 2482 5029 7621 5409 945 4371 5705 2431 4768 9979 5476 1495 3655 4202 6836 979 2963 5253 1947 6814 4315 3092 9101

6509 2829 6176 1521 4969 576 5762 6039 3505 7065 6831 67 8159 1723 9127 1191 7619 3325 7472 5570 9179 2495 5062 6599 669 1615 9984 383 5990 9460 3549 9389 1844 9875

5549 5856 7818 8846 2091 5783 3157 8200 6434 7595 5544 289 5107 2841 3902 3175 5092 4324 5344 8267 7110 9912 657 2074 3663 3915 3026 6169 3635 9263 3870 3534 1025 5556

9225 2916 6290 271 8406 2767 1846 5579 6081 4578 1844 7510 4345 4699 4786 6483 2558 5362 8847 8577 1796 1289 9510 8109 9730 1633 6393 3842 2202 5702 3827 7079 3884 734

9292 633 5149 4244 8996 6872 2000 871 9236 9156 1052 4818 8581 9353 6057 973 7944 1735 4746 5076 9923 1850 1546 7982 49 7644 8445 2883 6847 8794 1059 7536 4603 2031

7154 62 5131 1326 9574 2818 5314 5623 2619 2720 9316 599 5243 939 2994 8456 5781 271 4706 1962 1788 4481 8557 8444 5260 7731 9481 9138 5584 2050 9302 3634 6817 1104

3724 156 6452 1004 7219 7029 3186 156 6234 1013 3959 35 7813 9163 3668 6721 4840 6181 5143 9358 5221 7518 3449 2849 7714 1811 49 1302 8531 1153 4420 9432 5339 1646

501 3537 2941 7564 3445 1555 3888 587 3875 2864 1434 8774 2627 3608 2580 9682 5971 3883 7383 2222 1513 6123 3468 486 9930 3401 4481 9112 2866 3284 9190 1539 751 1079

594 1405 1199 1233 3128 7246 1615 896 3748 234 641 1695 9920 9899 9847 5266 48 221 6703 4968 1734 7780 2807 6860 9395 2918 1244 7482 9194 1140 1739 6145 7187 7912

3874 1974 7088 8026 1350 6004 7570 3665 5185 7354 4045 4630 3688 7840 1350 9742 9993 9577 4816 1913 6739 7753 340 3276 6232 9661 1777 7247 8091 2169 1207 9818 1796 5548

3242 8459 2477 2038 5574 1129 2906 8727 5107 6492 7323 156 791 2954 3461 637 749 1473 430 7170 5944 7832 4868 8011 6629 9506 3105 7966 7620 1902 6211 5803 4174 6012

9556 537 1733 6078 2806 7687 9707 5049 9087 3111 4374 2928 9354 3746 9862 239 8558 8795 3088 3248 928 8828 3540 7798 9 5163 7653 7420 6341 9242 2358 5312 4208 111

4872 5285 6715 6718 9619 3240 3160 3844 9676 7991 9263 6516 6169 5217 7450 6271 3930 7113 7282 2247 3009 9730 2170 1199 1430 894 912 2392 7279 2566 6523 2431 319 2651

7238 1493 2359 5743 3998 1248 1081 6021 9684 1256 4602 2706 9192 6548 2286 6591 2027 5924 3235 5589 7939 4120 3601 8706 8157 8624 5748 1470 7154 2066 6648 899 3828 1824

467 5286 1075 4154 5271 2787 1583 8316 8661 6215 8639 4700 5010 2854 6857 2340 4386 1362 6524 6014 4696 4309 741 5792 4661 9676 5720 3307 6527 3842 486 6426 7433 9208

4476 636 2608 1376 8185 4186 9709 1050 6814 1380 3893 1919 5739 8579 7852 7081 2108 5401 6154 6737 2034 6738 3735 1684 7700 2762 491 4903 8285 4933 575 1947 5934 7467

4860 2431 6795 7905 856 1301 507 8814 872 5718 5590 4808 435 1654 3575 8754 1736 2950 6671 7922 7395 6249 4732 9437 1729 9694 2241 9803 348 1047 400 6532 9303 9817

4691 2034 8940 7083 7243 6135 2961 8672 4898 370 532 8600 9155 4585 1597 8052 6181 298 3557 3509 269 6837 7498 4771 913 5115 8317 8502 8130 4879 1093 271 3418 8576

7565 9532 5318 2658 2516 562 2528 5565 369 5917 6766 6469 2358 9297 1994 2176 4223 4732 7527 3129 7317 8086 1241 1380 3285 8098 7566 9991 6600 6187 6531 8369 347 8234

5063 8519 8462 9662 9504 6603 4932 3368 7733 1140 7437 2751 2437 2904 4929 9464 1933 421 7777 2151 2348 6911 668 2861 1376 49 342 119 8764 3158 7961 9577 1709 735

732 5972 9971 8771 1934 4373 7809 4602 340 7191 3629 4195 5822 5193 5053 5474 3949 5554 8218 2607 488 957 1651 2260 634 8338 1869 8126 6815 7876 2847 3448 1117 8394

7553 7824 9464 8675 4156 1207 9200 1763 6206 8376 7467 8294 8531 9337 4835 6293 5668 983 5394 4182 1920 8690 4832 8302 9755 8816 4940 4802 4799 523 7258 8085 5270 5639

5421 6388 3070 1460 4847 740 3909 1417 8640 4461 7076 8950 8399 6781 1852 6408 4232 1088 5406 4124 413 8541 8052 2873 8517 7972 1602 6380 6715 3827 4329 5098 1567 3368

795 1101 4403 6232 1150 6561 385 854 7049 7453 1316 8830 7714 9121 5683 1482 5399 307 4209 5065 5448 8706 274 7382 9864 7329 6452 3503 3145 7967 4338 954 143 8161

3305 6628 3856 6367 1598 4618 2719 1163 836 5589 9315 8033 2321 6168 6966 8282 2392 711 5796 2024 5098 5157 3967 837 204 3286 6834 9023 3477 9552 168 4285 4140 6854

3787 541 6933 9838 5394 5271 1879 7968 7764 81 9945 4855 6334 805 772 4901 9646 5238 8730 8781 9799 4648 6857 4743 2431 7857 3656 9050 7985 1547 2246 7975 1392 8325

4368 8967 7529 967 6440 7330 3157 9359 6525 6780 6821 3979 9475 3460 6681 2119 5452 212 5231 2604 8106 6898 29 7362 8688 3219 1271 3160 9708 861 128 8367 5047 4445

795 8451 1502 8029 226 9468 9908 962 8474 5630 5864 3672 8481 5909 9510 8803 5205 6321 7494 1923 6571 3214 8013 8130 8202 2668 3769 9679 6042 1987 8987 9569 9196 4466

OUT：

1067

### 3.10完成一半题目

#### 3.10.1题目描述

有 *N* 位扣友参加了微软与力扣举办的「以扣会友」线下活动。主办方提供了道题目，整型数组 *questions* 中每个数字对应了每道题目所涉及的知识点类型。

若每位扣友选择不同的一题，请返回被选的 N 道题目至少包含多少种知识点类型。

#### 3.10.2概要设计

设计数组**int vis[10001];**用来对question元素进行哈希,函数的入口在main函数处.

#### 3.10.3算法分析

**算法说明:**

假设一共有个同学,将同一个*question*出现的频率从大到小排列的数组为,则如果在数据范围内,可以让*k*个同学中个同学去完成对应的题目,个同学其完成另一个题目……依次类推,最终得到最优的结果.

核心代码如下:

**int *k*=*n*/2;  
int *start*=0;  
while(*k*>0)  
{  
 *k*-=vis[*start*];  
 *ans*++;  
 *start*++;  
}**

时间复杂度说明,排序时间复杂度为,之后的遍历在最坏的情况下时间复杂度为,最好的情况下仅进行一次循环耗时,故本题时间复杂度为:



#### 3.10.4使用说明

输入说明 :

输入两行：

第一行为一个整数*n*代表数组*questions*的长度，，*N*为扣友的数量。

第二行输入*n*个整数代表数组*questions*的元素。

提示：



**运行界面如下:在Clion2023.1,MVSC 17.0编译器的环境下可以正常运行**



#### 3.10.5测试结果与分析

第一组数据:*question*数组已经按照降序排列,且保证没有重复出现的元素,测试通过

IN：

6

6 5 4 3 2 1

OUT:

3

第二组数据:特殊数据,*question*数组所有元素均相同,测试通过

IN：

10

1 1 1 1 1 1 1 1 1 1

OUT:

1

第三组数据:普通数据,*question*数组无序,但是不存在重复出现的元素,测试通过

IN：

10

20 30 40 23 43 51 12 19 67 39

OUT:

5

第四组数据:普通数据,*question*数组无序,且存在重复出现的元素,测试通过

IN：

10

1 3 1 1 4 5 6 7 2 3

OUT：

2

第五组数据：普通数据,*question*数组无序,且存在重复出现的元素,测试通过

IN:

8

1 2 3 4 1 2 3 4

OUT：

2

第六组数据：普通数据,*question*数组无序,且存在重复出现的元素,测试通过

IN:

15

10 9 7 2 1 7 7 6 1 4 4 1 6 10 8

OUT：

3

第七组数据:大数据测试,考察时间复杂度,测试通过

IN：

819

913 95 821 204 685 580 753 777 485 921 567 459 669 963 425 757 446 920 678 43 891 201 118 484 292 756 743 865 527 778 408 263 53 839 241 157 540 712 887 227 572 444 254 937 228 760 404 193 122 365 197 295 961 293 17 402 233 770 664 170 185 966 470 12 780 617 348 892 285 461 6 829 708 972 935 727 829 942 172 180 691 661 392 642 307 245 822 873 511 706 570 305 789 571 425 179 903 343 483 929 687 966 127 659 593 439 80 45 7 961 337 526 540 973 270 216 234 63 375 693 973 238 622 95 523 624 304 555 967 568 995 998 472 775 230 678 202 308 204 397 124 350 557 274 593 869 926 499 779 897 878 368 801 761 625 316 493 208 505 880 35 658 54 89 433 549 1 812 331 283 900 855 775 106 643 721 450 909 562 455 176 169 209 185 474 409 754 726 547 477 75 874 693 307 311 746 172 656 807 358 304 220 672 310 627 154 726 74 12 979 35 230 876 175 145 799 449 184 518 143 213 253 930 970 597 644 28 875 1 788 160 732 541 288 422 728 404 852 926 269 16 220 895 959 835 195 152 171 505 536 848 67 536 633 570 470 513 363 3 521 421 330 490 287 333 667 288 490 852 774 240 972 738 312 388 215 806 18 478 351 542 623 242 886 54 222 979 812 156 987 662 661 916 495 949 604 656 455 197 335 785 611 844 257 401 954 426 856 63 250 847 553 594 984 874 539 799 18 619 241 155 551 846 396 437 354 572 570 412 888 244 227 191 632 670 946 908 735 41 684 407 243 994 846 636 144 961 57 780 311 457 706 672 912 622 907 661 764 886 570 856 64 976 344 93 33 583 568 91 273 566 708 285 980 776 243 795 58 289 25 609 883 894 431 296 836 921 670 372 86 713 65 997 32 378 428 735 943 255 555 279 584 701 720 985 604 71 407 250 527 463 570 254 223 195 956 716 654 450 711 847 739 148 839 340 181 192 848 691 583 485 55 826 185 754 618 936 115 493 826 566 608 943 33 713 638 834 109 575 433 891 412 931 458 583 441 804 248 719 429 472 61 463 586 352 301 927 824 313 681 323 69 432 345 382 153 668 861 531 39 645 790 146 963 468 688 450 762 669 199 467 245 878 116 405 707 114 821 269 648 136 6 241 247 622 722 438 990 809 42 554 710 576 538 625 434 467 504 865 833 696 831 417 494 863 527 244 835 971 488 33 266 785 16 381 821 329 995 380 946 545 493 323 563 90 277 260 903 884 576 856 121 975 1000 212 466 622 190 323 978 662 812 530 913 611 760 985 132 752 79 163 846 605 815 89 884 797 461 158 815 704 925 399 140 979 5 261 572 472 351 807 514 539 114 423 36 189 224 20 954 709 253 761 218 633 730 247 106 195 580 669 156 415 304 714 322 688 286 284 664 864 805 94 187 129 876 755 724 352 297 257 288 331 996 512 638 68 74 368 238 417 206 549 505 121 870 754 446 893 884 369 203 24 275 699 798 548 457 826 949 127 93 756 785 680 655 450 641 400 622 501 988 686 984 869 148 620 853 874 673 556 191 867 199 454 34 442 378 862 454 639 652 258 88 914 763 115 770 951 160 711 851 795 716 442 574 820 421 203 833 524 791 907 753 299 250 935 814 251 524 14 473 81 233 361 221 95 869 917 575 715 9 354 153 449 842 721 65 487 226 847 952 203 522 206 902 522 793 6 295 368 486 845 207 163 736 605 962 13 857 81 611 204 688 815 910 166 670 201 52 891 306 158 865 815 430 478 43 400 226 959 852 49 887 326 885 562 566 303 174 567 966 309 207 681 969 372 231 705 912 169 951 419 777 620 245 37 604 675 468 47 622 964 697 108 818 837 607 281

OUT:

177

### 3.11 错误的集合

#### 3.11.1题目以及要求

集合 *s* 包含从 1 到 *n* 的整数，并且按升序排列。不幸的是，因为数据错误，导致集合里面某一个数字复制了成了集合里面的另外一个数字的值，导致集合 丢失了一个数字 并且有一个数字重复 。

给定一个数组 *nums* 代表集合 *S* 发生错误后的结果。

请你找出重复出现的整数，再找到丢失的整数。

#### 3.11.2概要设计

设计数组**int vis[10001],变量index,来定位与找到缺失的元素,函数入口在main函数中**

#### 3.11.3算法分析

**算法分析**

将每一个输入的*num*映射到*vis*数组,显而易见的,若一个*num*出现了两边,那么对应的*vis[num]*必然大于1,这个数就是重复出现的数字;最后从1-*n*扫描*vis*数组如果这个位序对应的值为0,那么这个数一定是未出现的数字.

核心代码如下:

**for (int *i* = 0; *i* < *n*; *i*++) {  
 int *num*;cin >> *num*;  
 if (vis[*num*] == 1) {  
 *index\_* = *num*;}  
 vis[*num*] = 1;}  
cout << *index\_* << " ";  
for (int *i* = 1; *i* <= *n*; *i*++) {  
 if (vis[*i*] == 0) {cout << *i* << " ";  
 return 0;}}**

**时间复杂度分析:**

仅需遍历一遍哈希表,故时间复杂度为

#### 3.11.4使用说明

输入说明 :

输入两行：

第一行为一个整数*n*代表数组*nums*的长度。

第二行输入*n*个整数代表数组*nums*的元素。

提示：





**运行界面如下:在Clion2023.1,MVSC 17.0编译器的环境下可以正常运行**

图片包含 背景图案

描述已自动生成

#### 3.11.5测试结果与分析

* 第一组数据:特殊情况,确实的数字为1

IN:

5

3 2 3 4 5 6 7 8 9 10

OUT：

3 1

* 第二组数据:简单情况,缺失的数字和重复的数字相邻

IN：

10

1 2 3 3 5 6 7 8 9 10

OUT:

3 4

* 第三组数据:普通情况,但缺失的数字和重复的数字不相邻

IN:

15

1 2 3 5 4 5 6 7 8 9 10 11 12 13 14 15

OUT:

5 15

* 第四组数据:特殊情况,缺失的数字在末端

IN:

10

1 2 3 4 5 6 7 8 9 9

OUT:

9 10

* 第五组数据:特殊情况

IN：

2

1 1

OUT:

1 2

* 第六组数据,普通情况

IN:

10

1 3 3 4 5 6 7 8 9 10

OUT:

3 2

* 第七组数据:大数据检验

IN:

500

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 500 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500

OUT:

500 17

### 3.12至少是其他数字两倍的最大数

#### 3.12.1题目以及要求

给你一个整数数组 *nums* ，其中总是存在 唯一的 一个最大整数 。

请你找出数组中的最大元素并检查它是否 至少是数组中每个其他数字的两倍 。如果是，则返回 最大元素的下标 ，否则返回 -1 。

#### 3.12.2概要设计

设计结构体**struct node**用来存放位序和值,同时为其重载小于运算符易于后面比较,设计集合**set<node>x**易于自动排序.

#### 3.12.3算法分析

**算法说明：**

按照题意,只要集合中最大的数是第二大的数的两倍以及以上,则一定是其余数的两倍及以上.另外需要特别判定如果集合中只有一个元素,则直接输出0即可.

核心代码如下:

**if(x.size()==1)  
{  
 cout<<"0";  
 return 0;  
}  
auto *it*=x.begin();  
*it*++;  
if(x.begin()->num>=*it*->num\*2)  
{  
 cout<<x.begin()->index\_;  
}  
else  
{  
 cout<<"-1";  
}**

**时间复杂度分析:**

*Set*单次插入,一共执行*n*次插入操作,故总复杂度为

#### 3.12.4使用说明

输入说明 :

输入两行：

第一行输入一个整数*n*表示数组*nums*的长度。

第二行输入*n*个整数表示数组的元素。

提示：



*nums* 中的最大元素是唯一的

**运行界面如下:在Clion2023.1,MVSC 17.0编译器的环境下可以正常运行**

图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成

#### 3.12.5测试结果与分析

* 第一组数据:特殊情况,仅一个元素,测试正确

IN:

1

1

OUT:

0

* 第二组数据:普通情况,但保证数组中的元素按照顺序排列,测试正确

IN:

10

1 3 5 7 9 11 15 17 19 21 23

OUT:

-1

* 第三组数据:普通情况,数组元素无序,但有一个元素是其余元素的至少两倍,测试正确

IN:

10

1 2 4 5 3 9 2 1 10 100

OUT:

8

* 第四组数据:普通情况,数组元素无序,没有一个元素是其余所有元素的两倍,测试正确

IN:

10

37 72 18 49 28 10 56 57 28 4

OUT:

-1

* 第五组数据:普通数据,*n*=20,测试正确

IN:

20

12 90 57 42 27 76 35 28 58 38 15 27 14 94 93 11 48 13 27 81

OUT:

-1

* 第六组数据:普通数据*n*=50,测试正确

IN:

50

76 28 29 81 93 29 11 92 76 37 98 97 7 74 60 84 52 5 82 18 81 73 53 49 87 89 59 11 34 75 51 75 7 0 87 21 74 69 90 9 37 13 53 39 67 68 45 25 70 13

OUT:

-1

* 第七组数据:普通数据,*n=*70,测试正确

IN:

70

37 52 34 85 77 57 43 76 49 99 91 68 52 57 47 48 74 62 8 9 74 76 86 17 29 60 100 91 98 20 16 50 37 95 59 65 32 67 38 97 4 23 6 18 61 76 90 4 91 83 33 31 14 39 64 45 28 68 59 9 78 80 38 6 53 99 69 87 31 62

OUT:

-1

### 3.13 K 次取反后最大化的数组和

#### 3.13.1题目以及要求

给你一个整数数组 *nums* 和一个整数 k ，按以下方法修改该数组：

选择某个下标 *i* 并将 *nums*[*i*] 替换为 *-nums*[*i*] 。

重复这个过程恰好 *k* 次,可以多次选择同一个下标 *i* 。

以这种方式修改数组后，返回数组 可能的最大和 。

#### 3.13.2概要设计

设计**vector<int>*s***用来存放元素,函数的入口在main处

#### 3.13.3算法分析

**算法说明:**

假设升序排序完的数组为一共有个负数出现在的位置,进行*k*次取反,根据贪心的思想,一定是先将小的负数取反,则这个时候讨论:若,则只需要将前个数组取反即可,若,则将前个负数取反后还剩下次机会,此时再将数组排序,循环对第一个最小的元素取反即可,更快的,若,则第一个元素仍然是,若,则将第一个元素取反后整个数组相加即可;

**时间复杂度说明:**

总共进行两次排序需要，相加使用的函数需要的时间复杂度,故总时间复杂度为,若是使用*set*集合来进行操作,则前次插入为，后*k*次取反后插入需要,加上求和需要的时间,故是会略快一些.

#### 3.13.4使用说明

输入说明 :

输入三行：

第一行输入一个整数*n*表示数组*nums*的长度。

第二行输入*n*个整数表示数组的元素。

第三行输入一个整数表示*k*.

提示：

****

**运行界面如下:在Clion2023.1,MVSC 17.0编译器的环境下可以正常运行**



#### 3.13.5测试结果与分析

* 第一组数据:简单数据满足,测试通过

IN:

2

-5 5

50

OUT:

0

* 第二组数据:普通数据,但保证,测试通过

IN:

10

2 3 4 5 6 10 19 20 63 12

5

OUT:

140

* 第三组数据:普通数据,中含有负数与正数,但保证负数的个数小于*k*,*k*为偶数

IN:

15

-4 -10 -2 -1 0 1 2 3 4 5 67 89 100 100 100 100

OUT:

488

* 第四组数据:普通数据,中含有负数与正数,且出现位置随机,但保证负数的个数小于*k*,*k*为奇数

IN:

20

-41 22 -65 -2 -74 40 -47 85 -18 -28 -87 2 -16 -14 -39 -7 10 2 -51 -88

7

OUT:

490

* 第五组数据: 普通数据,中含有负数与正数,且出现位置随机,但保证负数的个数大于*k*,*k*为奇数

IN:

20

-2 5 15 23 -90 -36 29 -46 90 24 23 -59 -100 -99 -91 43 62 35 20 -17

5

OUT:

707

* 第六组数据: 普通数据,中含有负数与正数,且出现位置随机,但保证负数的个数大于*k*,*k*为偶数

IN:

20

66 -5 37 79 -45 52 -99 28 -86 -12 74 48 -99 70 98 -20 43 -60 29 24

4

OUT:

910

* 第七组数据:大随机数据

IN:

2000

27 -69 -47 -67 -9 -83 -42 83 -86 -23 -80 1 91 -30 -26 39 -89 -11 -99 8 21 -74 -80 -89 -26 -9 -22 -65 0 -15 9 96 87 -40 16 28 -21 -43 -90 94 -45 36 -42 16 -52 25 -50 43 -57 22 -55 -26 -31 -51 -77 -6 -18 47 -64 54 46 40 53 4 -37 6 -61 40 -90 29 -53 10 -7 23 -21 -89 -56 72 -20 73 9 -19 71 -34 63 80 -13 -97 8 66 -57 62 73 -27 49 100 87 -28 -73 52 53 13 -32 -99 -22 -87 -76 -11 0 57 81 -32 -35 -9 -84 91 87 -25 86 -79 -17 -33 -73 -82 92 30 -56 -43 -30 -22 23 -29 46 57 70 -52 -87 55 -24 77 -89 -51 13 -82 14 -79 72 -39 -3 -44 -91 -90 22 -91 46 46 82 -98 -66 11 -82 97 -24 -51 4 -38 -98 -78 -40 77 -37 -1 1 -71 -73 -66 32 -31 81 -54 7 47 -96 -82 85 86 -41 61 -56 -85 43 96 -80 -36 8 -81 -70 -90 51 100 -5 -25 -61 28 76 2 -18 -64 -92 -83 -14 -90 -26 43 -87 64 -86 -47 53 10 61 -56 -33 83 -47 -99 -27 -56 3 5 -41 -8 -86 91 63 -60 84 7 -68 94 -68 -35 30 5 27 -93 -91 41 -27 4 -89 -36 37 10 -55 9 -12 -22 86 -64 -54 34 87 55 -81 23 80 51 -17 -73 54 68 16 5 95 44 69 -5 -77 -3 10 -56 -11 -67 -85 30 46 -23 24 36 71 -83 16 -19 -32 33 -50 -92 -99 -63 47 -33 -22 -26 -11 15 -66 -55 100 -85 39 -79 -68 -91 -22 -76 62 75 -97 -70 40 -37 -78 -30 57 -5 29 22 -63 20 59 -58 -11 -79 37 19 57 17 89 -24 72 25 -18 88 -68 -62 25 39 -66 86 -5 81 31 0 30 -1 -38 -92 91 -69 -8 -64 -56 50 -2 13 -62 -26 -12 -69 -52 97 -83 32 15 -89 48 79 -4 -48 57 -6 61 65 -36 13 1 84 -56 59 -15 -47 -71 -26 -77 98 44 -99 53 -65 93 48 95 37 34 -85 99 58 5 41 12 70 -21 85 81 -98 -85 -98 -51 -90 -96 8 -86 65 20 -32 82 46 69 -70 99 -36 54 40 -65 24 70 -70 -51 73 72 85 16 60 -89 99 -41 61 4 13 -27 57 4 22 -70 93 97 5 -25 20 24 13 -62 -37 59 37 -91 -54 -88 21 -33 -52 28 -41 49 -49 -12 -35 58 -42 35 -72 -15 -7 -63 83 -3 11 -16 43 6 18 53 -93 10 -69 11 51 16 -6 -100 -25 -24 86 66 55 69 -15 -7 94 -4 77 -21 44 92 -60 25 56 -21 27 88 -67 75 13 47 89 -43 -53 -20 -80 20 23 60 -33 26 -13 99 -95 87 17 -46 70 13 23 20 -31 -43 -44 -27 -100 -12 -94 -10 -7 -27 -43 -78 69 -25 -27 95 -3 -50 -7 -97 -92 -80 1 63 -43 26 6 68 -2 53 28 89 -22 19 18 99 7 -96 46 -93 54 -26 54 -62 3 -22 -44 14 68 79 87 75 46 -85 -65 -35 93 19 -21 -98 34 -28 -18 43 2 -55 -91 -65 -30 -87 10 83 -73 33 -51 -43 -76 -81 75 20 13 53 -44 40 51 -73 -63 95 -49 96 98 -70 77 -23 -31 48 73 -99 -66 3 63 2 -94 -56 75 58 75 -89 -68 90 -25 -13 71 -57 58 -99 89 -63 -3 18 96 -38 97 -90 99 3 88 95 -62 37 21 55 61 -29 -63 25 -23 -17 -11 58 20 -78 82 -98 -93 92 80 51 53 25 -50 -24 -74 48 64 99 14 -44 82 -39 -35 -65 -34 76 88 -62 88 20 43 -99 90 -17 -49 -64 27 28 1 -89 84 76 72 99 -70 72 75 67 66 -66 -43 -69 -31 -23 64 30 8 -81 -29 -43 -15 -1 64 -38 -64 54 33 -44 59 -43 41 -79 27 72 83 -40 84 72 -45 64 33 -81 -7 -21 91 26 90 -59 38 -91 -90 65 -77 62 -38 -46 -92 -14 -19 -72 -8 30 81 -33 -98 -65 74 77 -94 0 -34 85 44 44 91 93 -22 -70 11 95 43 88 77 -4 74 40 -56 44 24 75 99 24 84 -90 -44 5 69 -70 -37 -69 -11 32 -20 -17 51 83 -57 88 36 45 10 76 91 34 93 66 -65 96 26 -32 81 72 -59 98 -79 -4 8 -23 22 40 37 -50 65 -50 81 -73 78 -37 15 -74 59 -40 -65 86 -78 23 -29 0 -59 -100 73 86 84 1 24 44 -32 1 50 -54 -62 -15 -61 15 -12 12 -17 -95 75 56 -87 -16 12 -55 92 29 89 25 -29 -92 69 26 -21 43 24 47 50 -18 26 -19 100 67 -53 -12 -87 -38 40 -65 -83 91 73 55 -59 -80 36 -27 62 65 28 25 -3 99 55 -9 93 -50 -5 -60 -64 -80 -81 30 75 -58 -26 47 10 -7 50 78 19 26 -52 -95 15 40 -98 25 -53 16 -23 -22 27 -30 35 -61 -53 -73 -9 7 57 61 -97 14 -75 -19 100 -48 33 -71 -21 0 -79 -46 -58 82 82 51 -29 32 71 -13 -25 21 -64 28 72 95 26 -96 17 65 96 77 -61 -58 -6 25 83 -61 86 -87 44 43 -52 -10 -24 -55 76 -70 -85 92 -87 -27 -94 -83 29 -87 65 25 -35 -13 -85 -97 -74 10 -45 -9 -80 -48 61 -100 96 -100 -27 7 87 31 -25 -53 18 -94 51 -8 31 -41 -52 -97 -54 -51 -81 49 -49 89 89 55 -46 13 -74 80 -19 15 -69 -71 62 61 37 10 -70 -57 88 44 47 -86 -30 -49 24 90 10 -96 -65 6 -57 -8 -34 36 -20 -60 19 50 -30 92 -60 -87 -53 -43 -47 -48 62 -28 44 51 -5 -65 42 42 -28 30 -22 53 -79 -93 89 -65 50 29 -95 -47 77 -3 -17 -28 -73 -1 90 32 -54 44 -10 89 -99 -26 67 31 87 -94 81 -60 -42 -20 -34 67 62 -46 -51 -6 45 30 57 -50 -32 4 69 62 -76 63 68 -13 60 -76 -52 85 -31 59 -5 -52 66 68 6 97 -32 84 -77 -27 -28 -100 -38 -30 7 34 -68 24 82 31 -27 85 -74 -98 28 -6 20 -5 -32 61 2 14 95 27 -50 52 -72 -60 -27 -83 37 -99 9 -88 -27 37 -65 -32 19 9 39 -81 18 8 33 -51 51 44 69 -34 58 13 78 48 -8 83 -8 62 -51 52 -25 2 -20 -59 -40 -97 -80 -43 -97 20 34 -81 9 -12 -70 60 -81 10 -57 96 5 33 -55 -91 -35 -83 0 -84 -60 -84 -64 -9 -90 66 -46 7 -48 35 54 89 -77 98 57 -52 99 -13 -45 39 51 -19 36 88 -13 70 -21 -79 -65 92 77 -82 -33 13 67 48 -50 35 18 30 62 -69 7 64 84 -52 -84 94 -94 -31 1 21 80 -52 75 -33 78 -12 -16 -45 76 54 11 -100 -62 -61 94 -54 -82 23 49 27 64 -97 1 44 62 -37 96 -62 52 30 -35 -87 -74 45 69 60 -35 -55 53 -22 -30 20 -6 96 49 -41 4 54 -37 63 83 -31 -67 -18 -96 -34 -85 -81 91 -67 86 77 1 -80 54 86 -53 39 11 -13 84 -22 13 81 88 -88 60 -48 29 27 29 -100 -13 42 -11 24 65 100 44 -50 83 19 21 48 20 30 -93 -94 -82 5 -65 -64 69 -68 1 29 -12 -63 86 -50 -5 -64 21 -58 -16 12 87 46 20 38 -75 -2 69 43 70 16 22 29 -61 69 -44 -67 -88 -1 -14 92 -68 87 71 65 -62 -16 -8 38 -9 -93 33 67 -30 -45 -3 4 -37 -43 63 -98 46 -65 -42 33 -21 4 -72 -21 -81 35 49 66 -66 -62 -1 -10 -14 51 14 -85 -91 -14 48 -10 30 84 -31 -93 -100 -65 39 55 -69 42 -2 35 94 -31 22 63 -17 -9 -47 75 74 3 96 50 94 48 -83 92 -62 30 -67 14 -85 -67 2 63 89 80 90 -89 4 -4 -16 -34 -76 -32 50 -54 39 -77 87 -57 51 80 12 48 -90 84 -79 6 6 3 18 15 62 -84 -91 -2 6 78 33 -68 2 -77 -70 93 -27 60 1 -3 -58 31 15 -3 79 -4 98 99 81 46 32 81 89 -5 99 -67 70 59 3 -82 58 -86 15 20 -9 -1 -5 3 -24 56 6 -28 -41 -15 42 -76 -57 85 99 43 64 47 -100 52 -75 50 49 36 -75 39 -74 92 62 10 21 -40 78 5 -86 17 -28 -74 12 -75 63 67 69 -98 87 -73 -56 12 39 24 74 51 -18 -88 22 -82 -77 94 -8 -28 54 26 -38 47 40 -63 -50 -63 13 20 36 30 -19 -64 -63 -80 -12 5 85 -88 -89 99 -25 -96 -42 -74 -20 36 47 -7 -67 99 31 -49 -12 54 22 -13 51 24 99 -25 -59 72 -41 -92 -85 98 80 81 42 -81 -62 -44 -27 13 -38 33 63 -47 -54 20 -18 32 67 98 25 -99 18 -1 77 47 71 -10 -42 12 90 95 1 -32 -3 82 -96 -100 35 78 -4 -58 50 -52 24 -24 48 -2 2 -42 67 1 15 2 59 30 59 -54 48 19 73 -100 -70 -58 -72 -78 72 63 45 45 -56 -38 8 71 25 37 -37 50 4 -95 51 2 -15 69 -80 15 91 -8 62 -16 -89 5 -77 50 -67 51 46 -59 -13 -16 10 -14 -26 49 100 67 80 35 -38 -31 26 10 19 38 -1 -60 -39 -79 -76 29 8 -65 -53 21 1 -32 -55 -7 99 2 -21 56 -28 48 -99 63 -82 100 -1 82 53 -53 -93 49 -81 -42 8 -25 -100 34 71 22 26 -10 -97 76 86 -56 -76 67 1 20 11 -15 -12 -89 -99 28 -31 0 23 -14 60 -58 -96 -33 -90 -29 87 3 -12 35 -70 10 2 76 27 64 31 -60 -41 -89 -41 46 -63 70 69 10 72 -78 42 26 -75 -74 40 -89 -49 -17 -4 71 -33 12 -87 63 -62 -23 74 -74 -83 50 -54 71 -1 97 -48 37 -42 -53 -73 -82 -9 61 56 6 71 -79 66 -85 -86 67 21 -18 27 48 38 71 -72 -89 -70 18 -11 51 -85 -17 -79 8 -81 6 94 -25 76 -59 59 31 -57 59 -2 31 89 -24 -19 52 84 30 86 -4 58 88 -69 59 -60 21 32

10000

OUT:

100442

### 3.14检查整数及其两倍数是否存在

#### 3.14.1题目以及要求

给你一个整数数组 *arr*，请你检查是否存在两个整数 N 和 M，满足 N 是 M 的两倍（即，）。

更正式地，检查是否存在两个下标 *i* 和 *j* 满足：



#### 3.14.2概要设计

设计哈希表**int vis[10001]**,函数入口在main函数处

#### 3.14.3算法分析

**算法说明:**

由于输入数据存在负数,故进行哈希的时候统一对每个数+2000避免发生负数无法哈希的情况,在进行寻找时,对于每个*i*,我们只需要确认与是否同时存在即可.

核心代码如下:

**for (int *i* = -1001; *i* <= 1001; *i*++) {  
 if (vis[*i* + 2000] == 1 && vis[2 \* *i* + 2000] == 1) {  
 cout << "true" << endl;  
 return 0;  
 }  
}**

**时间复杂度分析:**

映射到哈希表的过程是遍历的步数是常数,即数组的数据范围是固定的是常数,故时间复杂度为

#### 3.14.4使用说明

输入说明 :

输入两行：

第一行输入一个整数*n*表示数组*arr*的长度。

第二行输入*n*个整数表示数组的元素。

提示：



**运行界面如下:在Clion2023.1,MVSC 17.0编译器的环境下可以正常运行**

图片包含 图形用户界面

描述已自动生成

#### 3.14.5测试结果与分析

* 第一组数据:只有两个元素,且全为正数,测试通过

IN:

2

2 4

OUT:

true

* 第二组数据:数据保证不含有负数,且元素有序排列,测试通过

IN:

10

1 2 4 8 16 32 64 128 256 512 1024

OUT:

true

* 第三组数据:数据集含有负数,且随机排列

IN:

10

-4 -2 -1 0 4 2 -4 8 10 1

OUT:

True

* 第四组数据,全0校验

IN:

10

0 0 0 0 0 0 0 0 0 0

OUT:

True

* 第五组数据,边界校验

IN:

2

500 1000

OUT:

true

* 第六组数据:随机校验,*n*=100

IN:

100

456 682 -739 -382 54 560 -954 -419 824 -968 559 -774 372 -384 -207 395 -172 -459 5 -651 761 -535 -632 791 636 -286 164 -379 665 -333 607 -898 -797 291 -385 -726 830 -621 -923 679 138 -736 -642 -186 141 161 -171 88 769 -48 548 75 -196 974 -232 82 806 242 653 454 -328 -776 -249 -558 -26 -645 -537 -627 762 642 474 10 -699 -7 988 485 -262 -48 -8 -251 632 564 -70 -623 -959 -550 -80 663 270 966 -444 -545 -431 406 186 691 466 585 317 724

OUT:

True

* 第七组数据:随机校验,*n*=500

IN:

500

94 240 236 -758 89 675 -637 -408 -968 -541 -831 -137 801 655 -765 -559 -707 -129 -427 -780 -476 -369 -583 -240 -303 -667 907 -677 853 545 187 -40 989 963 -644 554 -533 391 -174 -925 473 166 923 824 -332 -399 268 -928 825 814 -965 739 -236 -58 872 -971 -223 797 35 -925 615 428 5 563 687 656 4 792 -566 270 708 -94 223 464 -519 -4 -692 420 904 -486 82 673 -210 480 -157 -303 -5 -237 761 353 -470 521 930 -501 -620 -917 495 -765 -317 -5 -904 -789 584 -592 503 -349 -742 -512 -96 -562 539 152 935 -928 -895 210 711 28 642 -881 -487 517 474 -971 621 -832 -512 796 -14 406 -168 -660 -390 406 898 530 240 -881 -891 458 -630 -433 -905 813 122 259 -843 885 -163 949 -176 -186 950 -33 61 -254 803 662 -795 421 775 -617 194 140 617 687 949 -407 -115 344 -248 -768 -949 -136 -796 -302 -881 875 -15 524 377 -55 -960 200 439 831 -517 -903 825 -427 -28 -132 967 63 -597 137 -607 65 568 -599 -703 120 -666 71 1 -920 -158 -794 -588 646 999 -560 -531 -406 929 -391 -916 691 677 963 -820 -981 488 -820 978 -401 -367 -740 -136 -499 258 698 962 994 -915 443 526 -887 985 -916 818 61 -573 -598 838 645 -305 824 157 920 -843 250 125 -37 918 147 -984 982 -628 -537 -222 724 -501 826 160 -310 -962 651 584 75 158 -553 950 -211 36 189 762 -445 957 359 206 -769 -629 996 -177 -694 412 -989 462 -17 232 -451 559 -16 -17 899 -696 48 932 278 -434 721 -159 -802 -144 -958 -546 -863 871 955 -221 186 -231 -569 -773 28 -462 -361 -552 311 -20 -298 603 -504 -730 213 -865 -475 -380 65 -829 -102 57 56 -399 535 -263 -971 -360 9 53 -80 -369 190 -366 -353 329 -859 -287 -424 -148 -252 -813 -764 -77 -713 -581 385 199 -980 851 825 632 463 -64 -797 -886 566 -819 -520 609 -996 -938 -40 18 428 975 840 44 -946 -350 41 294 160 732 563 -577 -896 -199 -822 212 984 554 -981 -952 -251 -708 -738 95 -189 779 -575 -909 -910 915 -64 393 -126 -283 122 -502 633 -789 983 -482 -379 -489 707 -762 -713 526 -122 614 -507 -995 994 692 482 -419 70 -200 90 333 -65 191 -663 627 323 551 750 92 537 -104 -534 -345 -752 -771 -998 946 27 -297 -4 554 177 -912 -11 -736 235 843 455 214 -379 -135 65 798 -715 -176 802 -476 770 219 -595 -413 -631 379 15 -69 37 717 -365 907 -545 -370 -210 231 639 -296 965 466 852 -915 -712 -195 -347 164 -703 -250 -96 -299 807

OUT:

true

### 3.15数据流的第 K 大数值

#### 3.15.1题目以及要求

设计一个找到数据流中第 *k* 大元素的类（*class*）。注意是排序后的第 *k* 大元素，不是第 *k* 个不同的元素。

请实现 *KthLargest* 类：

*KthLargest(int k, int*[] *nums)* 使用整数 *k* 和整数流 *nums* 初始化对象。

*int add(int val)* 将 *val* 插入数据流 *nums* 后，返回当前数据流中第 *k* 大的元素。

#### 3.15.2概要设计

设计大根堆储存介质***priority\_queue<int,vector<int>,greater<int>>*** 用来存放数组中的元素,设计函数*int add(int val)* 将 *val* 插入数据流 *nums* 后，返回当前数据流中第 *k* 大的元素。

#### 3.15.3算法分析

**算法说明:**

不用等待全部元素输入后再初始化大根堆,考虑在输入时,让元素直接进入大根堆,加入元素后,若元素个数大于*k*,则弹出一个元素,这样保证在堆顶的元素就是第*k*大的,简图如下,设初始时*k*=3:



**时间复杂度说明**:

由于是使用堆这种数据结构初始化时间复杂度,之后每一次插入用时

#### 3.15.4使用说明

输入说明 :

输入若干行：

首行输入两个整数*k*和*n*，*k*代表题中的*k*，*n*代表用来初始化对象的*nums*数组的长度。

第二行输入*n*个整数代表用来初始化对象的数组*nums*的元素。

后面若干行输入 *add* 指令，每个*add*后面跟一个整数表示*val*。

提示：



    最多调用 add 方法  次

    题目数据保证，在查找第 *k* 大元素时，数组中至少有 *k* 个元素

**运行界面如下:在Clion2023.1,MVSC 17.0编译器的环境下可以正常运行**

图片包含 形状

描述已自动生成

#### 3.15.5测试结果与分析

* 第一组数据:边界情况,测试通过

IN:

1 0

add 1

add 2

add 3

add 4

add 1

add 8

add 9

add 10

OUT:

1

2

3

4

4

8

9

10

* 第二组数据,边界情况,测试通过

IN:

1 5

2 3 4 5 1

add 2

add 4

add 1

add -1

add -2

add -10

add 10

add 99

add 88

OUT:

5

5

5

5

5

5

10

99

99

* 第三组数据:普通情况,但保证输入的元素是有序的,测试通过

IN:

5 6

4 6 7 10 13 15

add 25

add 26

add 30

add 35

add 40

add 100

add 101

OUT:

7

10

13

15

25

26

30

* 第四组数据:普通情况,元素无序且含有负数,测试通过

IN:

7 10

-975 824 666 -326 669 200 358 911 -694 282

add 1

add 4

add 8

add 1000

add 39

add 68

add-4

add -5

OUT:

200

200

200

282

282

282

282

* 第五组数据:普通情况,元素无序且含有负数,测试通过

IN:

6 15

-58 869 373 413 855 498 -829 -415 -394 639 -363 -505 738 -189 610

add 1

add 0

add 2

add -1

add -2

add -4

add -100

add 100

add 200

add 400

OUT：

498

498

498

498

498

498

498

498

498

498

* 第六组数据: 普通情况,元素无序且含有负数,测试通过

IN:

6 15

7 10 -7 -3 -6 1 -9 1 -10 7 -6 -10 -7 2 -3

add 1

add 0

add 2

add -1

add -2

add -4

add -100

add 100

add 200

add 400

OUT:

1

1

1

1

1

1

1

2

2

7

* 第七组数据:大样例检验,测试时间复杂度,进行1000次*add*调用25ms

IN:

100 500

-787 263 227 -689 -695 -339 884 514 474 -106 435 439 834 755 225 -657 534 761 -556 -961 -875 -226 276 877 -754 868 328 369 374 -230 447 -989 -846 -556 -612 478 54 -81 -787 -780 510 631 736 -536 648 -605 -859 509 444 -872 73 -773 194 -955 643 -403 -82 0 695 542 641 543 198 328 -63 964 363 -641 -689 402 117 -900 585 -570 -12 797 314 -792 456 -63 -399 558 781 -70 444 -335 -231 -548 324 -879 667 386 909 954 -412 724 154 -79 753 282 -973 -88 72 -709 174 501 -480 22 -78 563 -755 168 636 323 -189 11 879 -716 441 -982 595 147 7 542 -86 -816 -541 -902 451 -701 243 -629 486 227 -126 -181 -613 386 743 644 -874 -410 -878 -634 149 -158 20 -567 -16 -310 838 -30 -525 554 161 323 584 217 224 -400 360 24 -293 954 -859 -247 -762 441 505 507 -612 381 346 339 -538 991 641 877 -275 -556 690 -573 736 220 312 87 -403 -854 750 677 -778 992 119 372 609 536 -150 -736 -210 -196 -254 -776 -219 803 241 -686 184 -50 283 -225 -492 -473 960 -267 977 -542 928 61 651 -83 832 -314 834 338 0 -242 247 -405 -465 636 757 -523 45 367 544 -486 459 214 477 574 777 -321 -295 -523 99 -201 -867 -925 49 -670 -63 255 -883 -276 327 -603 -156 -700 153 -973 218 -823 -659 -648 -113 -151 749 203 -94 689 -892 -442 946 -536 85 -761 -914 191 0 -934 329 -826 -335 -970 958 407 -733 374 -621 -481 73 429 -752 912 -39 472 593 925 -795 -472 650 436 380 -578 -400 -262 -851 963 -425 558 233 832 8 -741 64 603 -949 730 -761 176 -878 37 -460 568 180 -212 5 230 510 857 303 320 832 -129 251 99 -996 -608 58 481 99 912 -665 -205 -798 -54 58 119 -477 -620 705 945 -867 -245 422 -262 894 75 -306 -757 -186 676 151 823 -692 181 835 770 766 484 -18 393 -120 -855 293 753 229 -438 -31 855 -844 298 414 283 149 20 -656 445 -714 112 617 754 35 -482 565 389 -335 -131 231 618 743 -469 676 153 840 55 -613 478 -432 246 -613 -640 364 260 130 308 -788 955 -578 113 19 -20 655 -915 638 -586 -147 -77 -492 -130 -88 -660 786 686 383 966 -368 -581 -606 787 -986 -686 -542 239 -626 286 321 -828 -678 188 823 715 589 530 -561 600 -883 -634 983 -6 -566 -980 387 251 545 171 -982 -937 766 -875 973 945 -13 -41 -848 924 -754 -314 -126 -480 333 -238 720 -302 -82 -532 -290 -362 -594 -8 -989 288 -738 331 -513 -188 -938 -353 390 -579

add 579

add 514

add 283

add -727

add -660

add -685

add -380

add 683

add 988

add 715

add -262

add -485

add -400

add 467

add -631

add -996

add -898

add -855

add 137

add -720

add -235

add -980

add 990

add 825

add -358

add -754

add 900

add -490

add -698

add -358

add -256

add 652

add 783

add -428

add 101

add 364

add -675

add 816

add 719

add 899

add 134

add 695

add -795

add -143

add -679

add 124

add 608

add -595

add 813

add 917

add 955

add -326

add 397

add -339

add -296

add -129

add -185

add -791

add 959

add -140

add 379

add -529

add 285

add -420

add -278

add -682

add 882

add -503

add 342

add -236

add 623

add -332

add -432

add -864

add 825

add 717

add 545

add -69

add -246

add -417

add 222

add -187

add -560

add -366

add -915

add -759

add -378

add 585

add -989

add -958

add 960

add 127

add -329

add -338

add -672

add -572

add 517

add -159

add -66

add 289

add -365

add 15

add 232

add -374

add 415

add 419

add -31

add -832

add 474

add 297

add 502

add -621

add -713

add -104

add 620

add -974

add 655

add 371

add -40

add 238

add -725

add -479

add 621

add -708

add -309

add -744

add -115

add -293

add -430

add -353

add 589

add -890

add -875

add 696

add -376

add -521

add 855

add 671

add -573

add -687

add 659

add -68

add -977

add -337

add -461

add -373

add -552

add -704

add -780

add 462

add 877

add -315

add -827

add 20

add -928

add -789

add 38

add -131

add 882

add -119

add -707

add 140

add -840

add 619

add 682

add 144

add 447

add 104

add -372

add -866

add 463

add -514

add 405

add -701

add 748

add -54

add -786

add -13

add -557

add 773

add -888

add 778

add 303

add -354

add -70

add -950

add -7

add -605

add 232

add -61

add -30

add -864

add -1000

add -130

add -775

add 173

add -992

add -781

add 967

add -193

add 232

add -151

add -427

add -358

add 771

add -505

add -736

add -272

add -175

add -356

add -813

add -820

add -463

add 143

add 202

add 496

add -220

add -965

add -372

add 803

add 167

add -846

add -445

add -543

add 219

add -556

add 17

add -920

add -648

add -562

add 925

add 637

add -811

add -149

add -997

add 730

add 501

add 184

add 238

add 108

add -792

add -555

add -932

add -93

add 161

add 493

add 612

add 892

add 399

add 969

add -242

add -951

add -157

add 638

add -611

add -308

add -4

add -3

add -902

add -138

add 764

add 4

add 93

add -983

add 273

add 999

add 59

add -657

add -541

add -269

add -945

add 392

add -153

add 129

add 478

add 658

add -255

add -610

add 579

add 358

add 0

add -940

add 672

add 648

add -943

add -461

add 354

add 896

add -694

add 523

add -788

add 597

add 574

add -725

add 696

add -179

add -588

add -274

add 443

add 80

add 954

add 328

add 429

add 356

add -409

add 998

add 435

add 836

add 320

add 743

add -550

add -696

add -513

add 755

add 580

add -330

add 615

add -904

add -721

add -682

add 828

add 28

add -537

add 279

add 781

add 2

add 978

add 570

add 810

add 382

add 716

add 25

add -506

add 580

add -820

add -283

add 25

add -155

add 742

add 352

add 682

add -672

add 856

add -180

add -840

add 664

add -829

add 468

add 214

add -493

add -952

add 146

add 629

add -659

add -560

add -506

add 360

add -848

add -653

add -677

add 731

add -788

add -653

add -218

add -227

add 600

add -681

add 584

add -512

add 481

add -752

add -162

add 401

add 516

add -992

add -639

add 136

add -654

add 128

add -875

add 412

add -27

add 513

add -307

add 797

add -674

add -816

add -815

add -301

add 471

add 645

add 249

add 536

add -440

add 74

add 791

add 812

add 803

add 717

add -229

add 469

add 211

add 141

add 832

add -690

add 308

add -817

add -638

add 121

add -929

add 631

add -488

add -145

add -728

add 557

add -334

add 77

add -768

add 165

add -692

add 851

add 94

add -454

add 389

add 362

add -102

add -300

add 303

add -932

add 766

add 364

add 214

add 101

add 804

add -939

add -614

add -941

add 442

add 758

add -268

add 195

add 581

add 176

add -344

add -99

add -523

add 942

add 890

add 493

add 601

add -266

add 689

add -816

add 719

add -873

add -561

add 156

add 53

add 902

add -684

add 287

add 401

add -276

add 367

add -273

add 400

add -322

add -359

add 946

add 805

add 224

add 222

add 990

add -295

add -705

add -553

add -814

add 507

add -380

add 443

add 467

add 270

add 603

add -401

add 346

add 698

add 941

add -66

add 499

add -887

add 961

add 46

add -960

add 126

add 591

add -69

add 433

add 878

add -1

add 986

add 66

add 728

add -965

add -704

add 118

add -731

add -839

add 17

add -14

add -170

add -695

add 898

add -715

add -305

add 410

add 129

add -363

add 136

add 118

add 663

add -445

add 335

add -927

add 570

add -628

add 894

add 966

add 221

add -833

add -664

add 766

add 976

add -575

add -184

add 500

add -73

add 738

add -838

add 497

add -58

add 939

add -163

add 39

add -328

add 422

add 760

add 810

add 994

add -548

add -66

add 298

add 586

add 207

add -587

add 677

add -824

add 462

add 873

add -91

add -461

add -858

add 416

add 696

add -944

add 366

add -568

add -220

add -177

add -481

add -955

add 348

add 465

add -55

add -500

add -326

add 475

add -486

add 888

add -342

add -854

add 120

add -834

add -444

add 546

add 560

add -409

add 21

add -493

add 227

add -57

add 966

add -43

add 945

add 269

add -337

add 913

add 838

add -310

add 45

add 474

add 677

add -158

add -305

add -518

add 928

add 369

add -119

add 990

add 991

add -265

add -245

add 967

add 789

add -389

add -748

add -642

add -699

add -508

add -422

add -309

add -893

add 254

add -134

add -1000

add -290

add 634

add -308

add -376

add 182

add -406

add -871

add 467

add -393

add -806

add 940

add 206

add -955

add 240

add 538

add -708

add 962

add 992

add -677

add 127

add 789

add 284

add 323

add -233

add -388

add -985

add -190

add -421

add -200

add -890

add 902

add 181

add -338

add -768

add 951

add -391

add -125

add -748

add -623

add -194

add -424

add 342

add 766

add -361

add 485

add -544

add 846

add 357

add -435

add 539

add 555

add 31

add 798

add -36

add -188

add 735

add -596

add -70

add -426

add -652

add -583

add -44

add 30

add 697

add -422

add 613

add 653

add 254

add -207

add 588

add 982

add -799

add 251

add 914

add -787

add -578

add 168

add 525

add 973

add 971

add -599

add -256

add 987

add -945

add -537

add 25

add -655

add -143

add 891

add 220

add 767

add 858

add 770

add -812

add -867

add -389

add 381

add 364

add 722

add -976

add -343

add -7

add 396

add -185

add 639

add 469

add 579

add -666

add -294

add -5

add 223

add 287

add -522

add -93

add -676

add 858

add -206

add -802

add -539

add -790

add -424

add 307

add 956

add -183

add -197

add -646

add 13

add 723

add -159

add -194

add -353

add -543

add 503

add -605

add 339

add 245

add -406

add -881

add -198

add 359

add -976

add -319

add 706

add -321

add 690

add 211

add -507

add -300

add 228

add -672

add 441

add 599

add 96

add -467

add 436

add 3

add -929

add 281

add 565

add -642

add -375

add -810

add 194

add 843

add 246

add -112

add -711

add 33

add 733

add -512

add -547

add -828

add -708

add -557

add -118

add -240

add 898

add -405

add 79

add -484

add 442

add 774

add -502

add -495

add 747

add -939

add 193

add 312

add -958

add 736

add 138

add -817

add -774

add -53

add -98

add -97

add 321

add 762

add -420

add 228

add 772

add 112

add 102

add -333

add -697

add -684

add -968

add 879

add 930

add -891

add 411

add 323

add 599

add 689

add 117

add -171

add 286

add -535

add -829

add 984

add 392

add 264

add 513

add -116

add -900

add 151

add -206

add -793

add 678

add 890

add 145

add 510

add 672

add -920

add -686

add 813

add 807

add 822

add -747

add -178

add -539

add -365

add 586

add -600

add -548

add -944

add -648

add -607

add -60

add 717

add 960

add -405

add 869

add 477

add -161

add -419

add 729

add -646

add -685

add 741

add -664

add 752

add 150

add 207

add 513

add 486

add -875

add 282

add 277

add -313

add -109

add 686

add -979

add 115

add -794

add -383

add 998

add -969

add 930

add 495

add 398

add -700

add 250

add 858

add 89

add 58

add 50

add -135

add -874

add 652

add -516

add 475

add 190

add 26

add -952

add 689

add 941

add -916

add 596

add 98

add -730

add -303

add -726

add -400

add 960

add -591

add -927

add -428

add 666

add -944

add -210

add -474

add -248

add 945

add 440

add 128

add 460

add -77

add -772

add 382

add -579

add -403

add -415

add 787

add -10

add -865

add 68

add 570

add -164

add 957

add 50

add 39

add -619

add -749

add 925

add 164

add 840

add 468

add -206

add 680

add -847

add -123

add 278

add 194

add 413

add 237

add -11

add -520

add 544

add 467

add 506

add 869

add -860

add 841

add 244

add -85

add -75

add 479

add -544

add -645

add -422

add 60

add 190

add 782

add 13

add 425

add 938

add -81

add -806

add 872

add 787

add 570

add 118

add 618

add -31

add 268

OUT:

565

565

565

565

565

565

565

568

574

579

579

579

579

579

579

579

579

579

579

579

579

579

584

585

585

585

589

589

589

589

589

593

595

595

595

595

595

600

603

609

609

617

617

617

617

617

617

617

618

631

636

636

636

636

636

636

636

636

636

636

636

636

636

636

636

636

638

638

638

638

638

638

638

638

641

641

641

641

641

641

641

641

641

641

641

641

641

641

641

641

643

643

643

643

643

643

643

643

643

643

643

643

643

643

643

643

643

643

643

643

643

643

643

643

643

643

644

644

644

644

644

644

644

644

644

644

644

644

644

644

644

644

644

648

648

648

650

651

651

651

652

652

652

652

652

652

652

652

652

652

655

655

655

655

655

655

655

655

655

655

655

655

655

655

659

659

659

659

659

659

659

659

659

659

667

667

667

667

667

671

671

676

676

676

676

676

676

676

676

676

676

676

676

676

676

676

676

676

676

676

676

676

676

676

677

677

677

677

677

677

677

677

677

677

677

677

677

677

677

682

682

682

682

682

682

682

682

682

682

682

683

683

683

683

683

686

686

686

686

686

686

686

686

686

686

686

686

689

689

690

690

690

690

690

690

690

690

690

690

690

695

695

695

695

695

695

695

695

695

695

695

695

695

695

695

695

695

695

695

695

695

695

695

695

695

695

695

696

696

696

696

696

696

696

696

696

696

696

696

696

705

705

705

705

705

715

715

715

715

717

717

717

717

719

719

719

719

719

719

719

720

720

720

720

724

724

730

730

730

730

730

730

730

730

730

730

730

730

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

736

742

742

742

742

742

742

742

742

742

742

742

743

743

743

743

743

743

743

743

748

748

748

748

748

748

748

748

748

748

748

748

748

748

748

748

748

749

749

749

749

749

749

749

749

749

750

750

750

750

753

753

753

753

753

753

753

753

753

753

753

753

753

754

755

755

755

755

755

755

755

755

755

755

755

755

755

755

755

755

755

755

755

755

755

757

758

758

758

761

761

761

761

761

761

761

761

761

761

761

761

761

761

764

764

764

764

766

766

766

766

766

766

766

766

766

766

766

766

766

766

766

766

766

766

766

766

766

770

770

770

770

770

770

770

770

770

770

770

770

770

770

771

773

773

773

773

773

777

777

777

777

777

777

777

777

777

778

778

778

778

778

778

781

781

781

781

781

781

781

781

781

781

781

783

783

783

783

783

783

783

783

783

783

783

783

783

783

783

783

783

783

783

783

786

786

786

786

786

786

786

786

786

786

786

786

786

787

787

791

791

791

797

797

797

797

797

797

797

797

797

803

803

803

803

803

803

803

804

804

804

804

804

804

804

804

804

804

804

804

804

804

804

804

804

804

804

804

804

804

804

805

805

805

805

805

805

810

810

810

810

810

810

810

810

810

810

810

810

810

810

812

812

812

812

813

813

813

813

813

813

813

813

813

813

813

813

816

816

816

816

816

816

816

816

816

816

816

816

816

816

816

816

816

816

816

816

816

816

816

816

823

823

823

823

823

823

823

823

825

825

825

825

828

828

828

828

828

828

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

832

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

834

835

836

836

836

836

836

836

836

836

836

836

836

838

838

838

838

838

838

838

838

838

838

838

838

838

838

838

838

838

838

838

838

838

838

838

838

838

838

838

838

838

838

838

840

840

843

843

843

843

843

843

843

843

843

843

843

843

843

843

843

843

843

843

843

843

843

843

843

843

846

846

851

851

851

851

851

855

855

855

855

855

855

855

855

855

855

855

855

855

855

855

855

855

855

855

855

855

856

856

856

856

856

856

856

856

856

857

857

857

857

857

857

857

857

857

857

857

857

857

857

857

857

858

858

858

858

858

858

858

858

858

858

858

858

858

858

858

858

858

858

858

858

858

858

858

858

858

858

858

858

858

858

858

858

858

858

858

858

858

868

868

868

869

869

869

869

869

869

869