班级 计算机203 学号 202007020625 姓名 於俊涛

**实验题目： 集合的表示与实现**

**一、概述**

应用需求：处理大规模数据；

设计目标：实现集合元素查找、插入、删除以及集合的交、并、差等运算；

关键算法：链表的创建与输出，交、并、查集的实现；

Union (); Intersection (); Difference ();

InitList (); DispList (); Find (); Insert (); Delete ();

实验方案；数据收集（生成随机数）、数据预处理（排序，集合确定性），数据储存（链表，二叉树），数据处理与分析，数据可视化（excel表格处理），数据应用；

主要结论：大规模数据处理需要考虑处理器时间，内存，硬盘空间和网络宽带等因素的限制，在分配这些资源时，将系统视为一个整体，可以将尽可能多的时间用于核心计算，其中Hadoop方法运用的最为广泛。

**二、问题分析**

1. 设计目标

结合大规模数据处理需求，分析大规模集合表示和运算所面临的挑战，运用数学、数据结构原理以及高级程序设计技术，通过查阅文献，设计不同的方案，实现集合元素查找、插入、删除以及集合的交、并、差等运算。

1. 数据对象、数据关系以及需要实现的主要操作

数据对象：随机数

数据关系：同属于一个集合

主要操作：1）结构体定义：链表；

2）函数声明：

Union (); Intersection (); Difference()；

集合的交，并，差；

InitList() DispList()；

链表的输入与输出

Find()；Insert()；Delete ();

查找，插入，删除元素；

3）函数调用；

1. 问题规模、算法设计、数据结构方案等不同因素对主要操作性能的影响

问题规模：规模大小与操作性能的成反比，即规模越大，性能越低，反之相反；

算法设计：实现集合的交并差运算时，采用随机数来构成集合的元素，将元素存储在链表里面，使用头插法，L1,L2,L3是三个链表的头结点，p、q、r是头尾指针，通过比较p，q结点数据的大小，较小的结点数据插入r的尾指针，以此循环，当p，q结点数据相同时只复制一个数据来继续插入，避免了集合元素的重复，降低了冗杂性；

数据结构方案：集合的交并差运算采用链表的数据结构，在数据存储和运算方面都比数组更优；

**三、设计和开发**

1. ADT设计

ADT{

数据对象：D = {a1 a2…ai|ai∈R}

数据关系：R = {<a(i-1), ai>|ai∈R}

基本操作： Union (LinkNode\* L1, LinkNode\* L2, LinkNode\*& L3);

Intersection (LinkNode\* L1, LinkNode\* L2, LinkNode\*& L3);

Difference (LinkNode\* L1, LinkNode\* L2, LinkNode\*& L3);

InitList (LinkNode\*& L);

DispList (LinkNode\* L);

LinkNode\* Find (ElemType a, LinkNode \*L);

Insert (ElemType a, LinkNode \*L);

Delete (ElemType a, LinkNode \*L);

}

2.存储方案

1. 链式存储

typedef struct LNode

{

int data;

struct LNode \*next;//存储后继节点位置的指针域，用next表示

}LinkNode;

优点：插入或删除元素时很方便，使用灵活，存储空间利用率高；

缺点：存储密度小时（<1），查找和修改需要遍历整个链表；

2）顺序存储

typedef struct

{

ElemType \*elem; *// 存储空间基址*

int length; *// 当前长度*

int listSize; *// 当前分配的存储容量*

}SeqList;

优点：无须为表示表中元素之间的逻辑关系而增加额外的存储空间。

可以快速地存取表中任意位置地元素

缺点：插入和删除操作需要移动大量元素；

当线性表长度变化较大时，难以确定存储空间的容量；

容易造成存储空间的”碎片“；

结论：因为数据规模大，线性表难以确定存储空间的容量，而链表使用灵活，存储空间利用率高，故选择链式存储结构；

3.关键算法

1.bool Insert(ElemType a, LinkNode \*L){ //将a插入链表L中

LinkNode \*p = (LinkNode \*)malloc(sizeof(LinkNode));//动态分配空间

p->data = a;

p->next = L; //将L接入尾指针

L = p;

return true;}

O（1）；

2.bool Delete(ElemType a, LinkNode \*L){ //删除链表L中的a元素

if(!L){ //当L为空

printf("already delete\n");

return true;

}

LinkNode \*p = L, \*q = L;

while(p){

if(p->data==a)

break;

q = p;

p = p->next; //p继续向后扫描L1

}

if(p){

q = p->next;

free(p);

}

printf("already delete\n");

return true;

}

O（1）/O（n）；

3.bool Union(LinkNode\* L1, LinkNode\* L2, LinkNode\*& L3) //L1、L2、L3为三个单链表的头结点

{

if (L1->next == NULL && L2->next == NULL) { //如果L1和L2中均没有数据

return false;

}

while (!L3) { //直到L3申请成功

L3 = (LinkNode\*)malloc(sizeof(LNode));

}

LinkNode\* p, \* q, \* r, \* s = NULL;

p = L1->next; //p为L1的工作指针

q = L2->next; //q为L2的工作指针

r = L3; //r为L3的尾指针

while (p && q) { //当L1和L2均不为空时

s = (LinkNode\*)malloc(sizeof(LNode));

//如果申请失败继续申请，直到申请成功，因为malloc可能申请失败

while (!s) {

s = (LinkNode\*)malloc(sizeof(LNode));

}

if (p->data < q->data) {

s->data = p->data; //如果此时p指向的data的值小，复制该结点的值

r->next = s; //将s接到尾指针后面

r = r->next; //尾指针后移

p = p->next; //p继续向后扫描L1

}

else if (q->data < p->data) {

s->data =q->data; //如果此时q指向的data的值小，复制该结点的值

r->next = s; //将s接到尾指针后面

r = r->next; //尾指针后移

q = q->next; //q继续向后扫描L2

}

else {

s->data = p->data; //如果两个值相等，复制其中一个结点的值

r->next = s; //将s接到尾指针后面

r = r->next; //尾指针后移

p = p->next; //p继续向后扫描L1

q = q->next; //q继续向后扫描L2

}

}

while (p) { //当p不为空时，将剩余结点接到尾指针后面

s = (LinkNode\*)malloc(sizeof(LNode));

while (!s) {

s = (LinkNode\*)malloc(sizeof(LNode));

}

s->data = p->data; //复制结点的值

r->next = s; //将s接到尾指针后面

r = r->next; //尾指针后移

p = p->next; //p继续向后扫描L1

}

//当q不为空时，将剩余结点接到尾指针后面

while (q) {

s = (LinkNode\*)malloc(sizeof(LNode));

while (!s) {

s = (LinkNode\*)malloc(sizeof(LNode));

}

s->data = q->data; //复制结点的值

r->next = s; //将s接到尾指针后面

r = r->next; //尾指针后移

q = q->next; //q继续向后扫描L2

}

r->next = NULL; //此时将尾指针后面置空，单链表L3建成

return true;

}

4．bool InitList(LinkNode\*& L) //建立单链表

{

while (!L) {

L = (LinkNode\*)malloc(sizeof(LNode));

}

L->next = NULL;

int i, n;

LinkNode\* p = NULL, \* q = NULL;

q = L;

printf("请输入数据规模：\n");

scanf("%d", &n);

srand((int)time(0));

for (i = 0; i < n; i++) {

while (!p) {

p = (LinkNode\*)malloc(sizeof(LNode));

}

p->data=rand();

q->next = p;

q = p;

p = q->next = NULL;

}

return true;

}

4.实现总结

关键难点：生成不重复的随机数；

集合的并运算需要剔除两个集合中重复出现的一共元素

问题：集合具有互异性，每个元素只能出现一次，所以生成的随机数不能重复；

解决思路：随机数生成函数rand（）位置放置不对；

方法：将rand（）函数放置于for循坏之前即可；

srand((int)time(0));

for (i = 0; i < n; i++) {

while (!p) {

p = (LinkNode\*)malloc(sizeof(LNode));

}

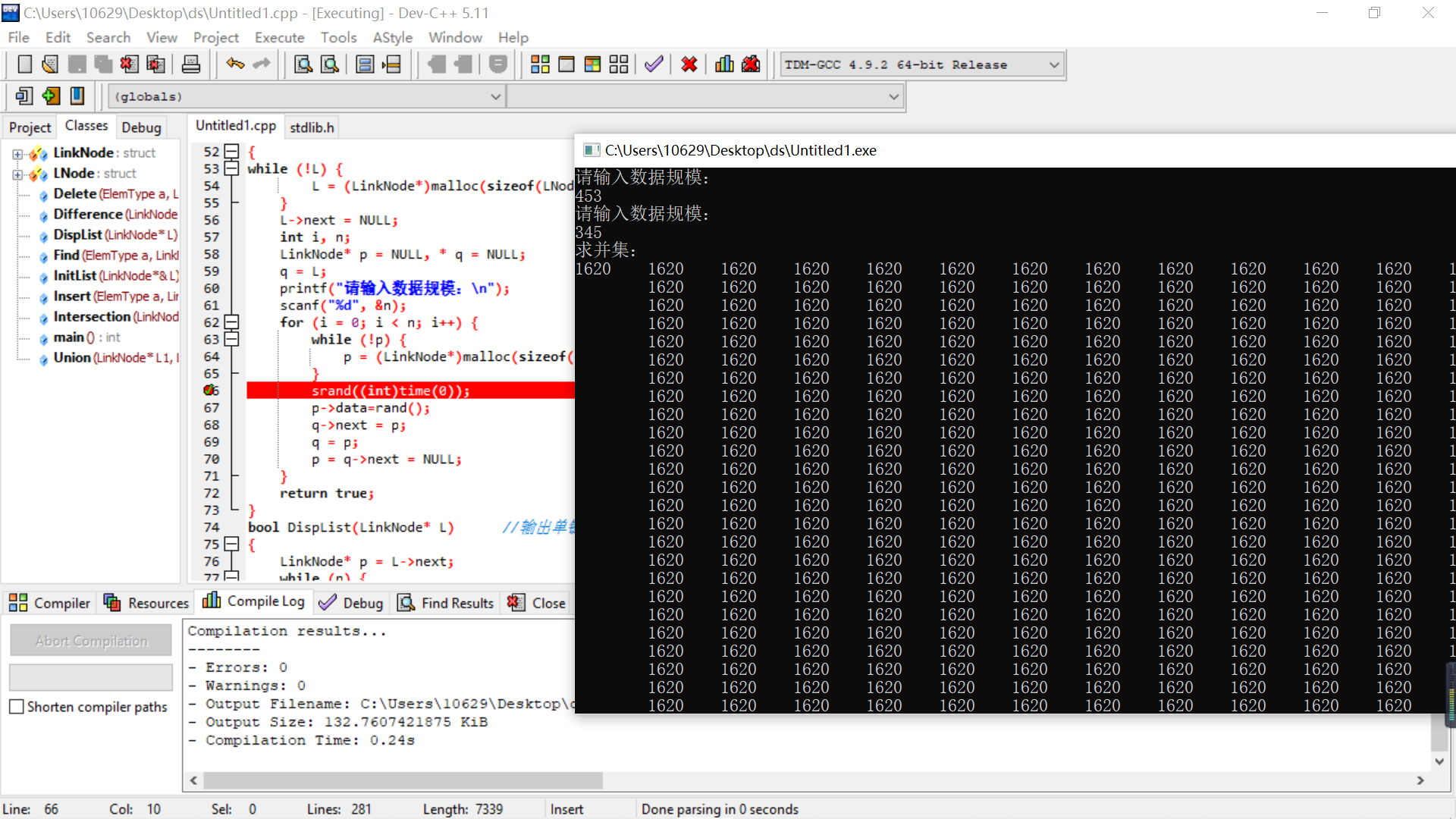
p->data=rand();

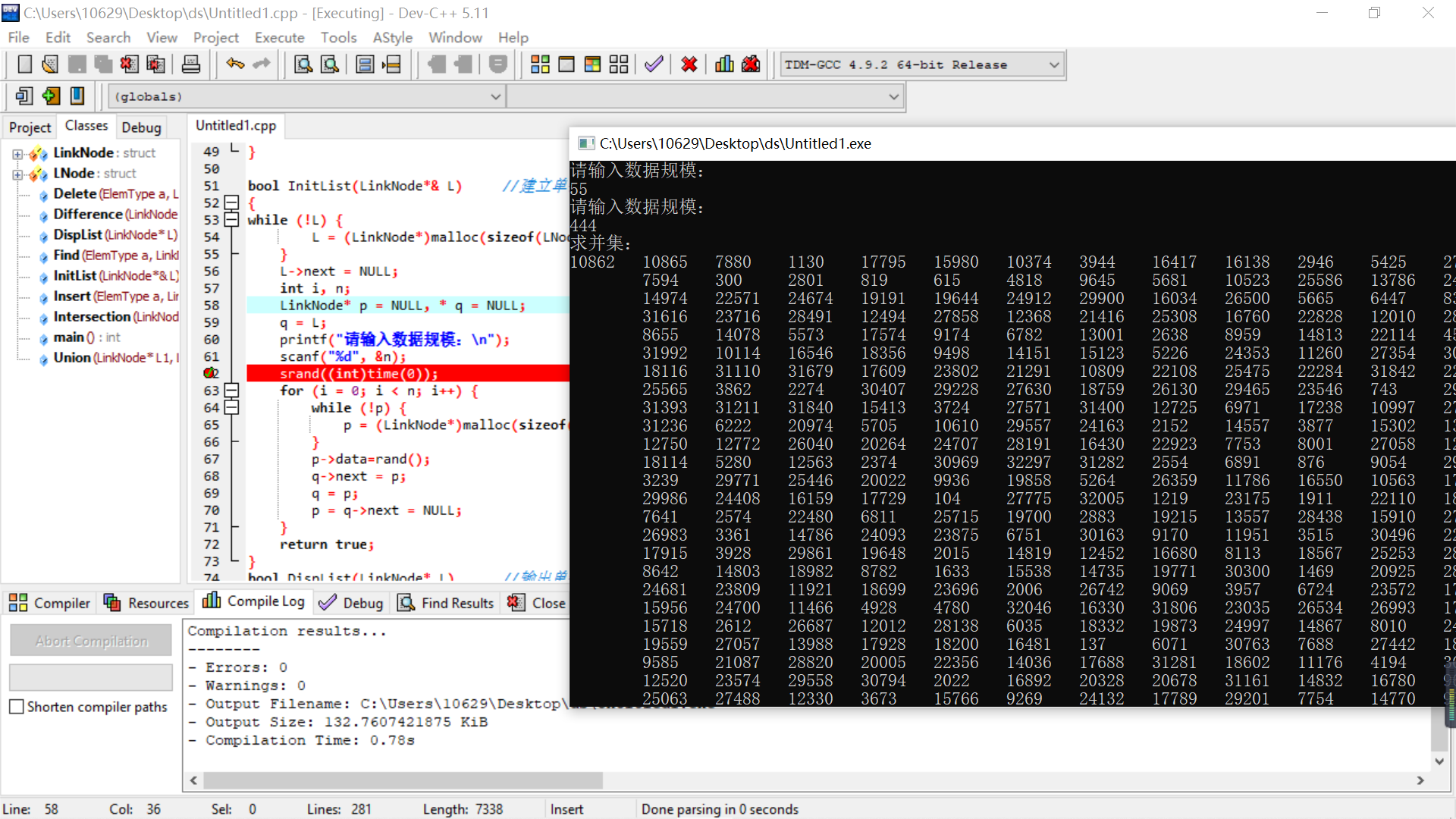
q->next = p;

q = p;

p = q->next = NULL;

}





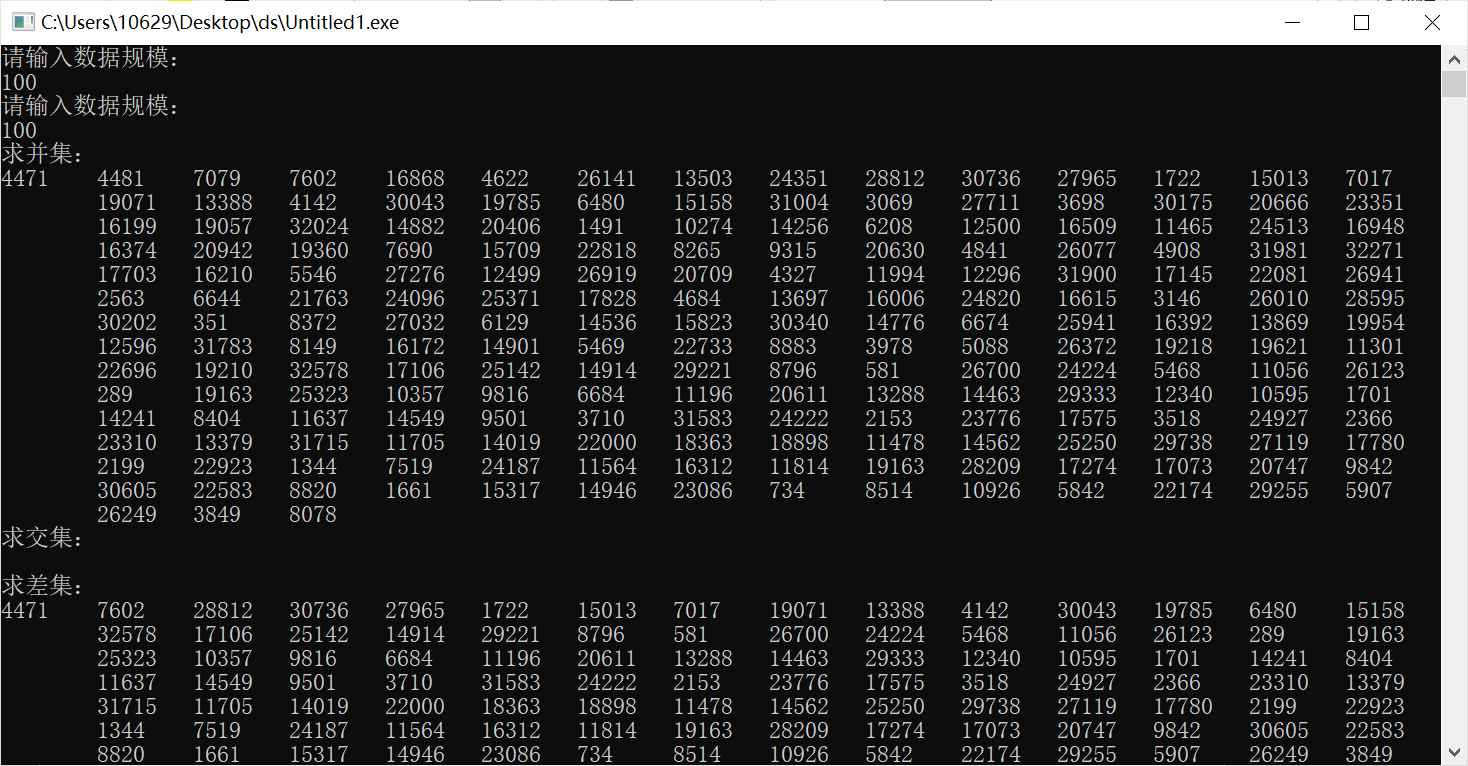
**四、实验研究**

1. 实验目的

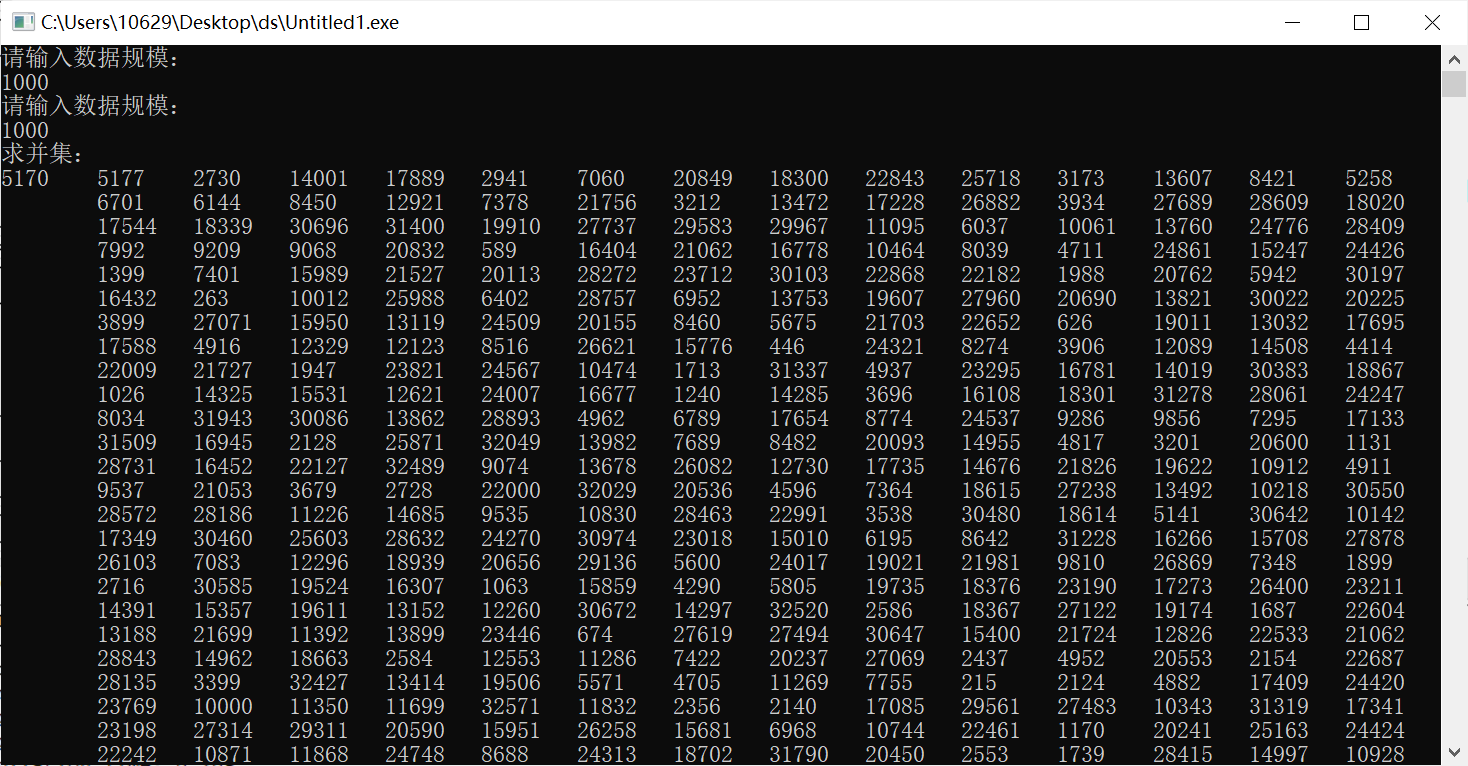
数据规模对算法时间效率和空间效率的影响；

2. 实验数据

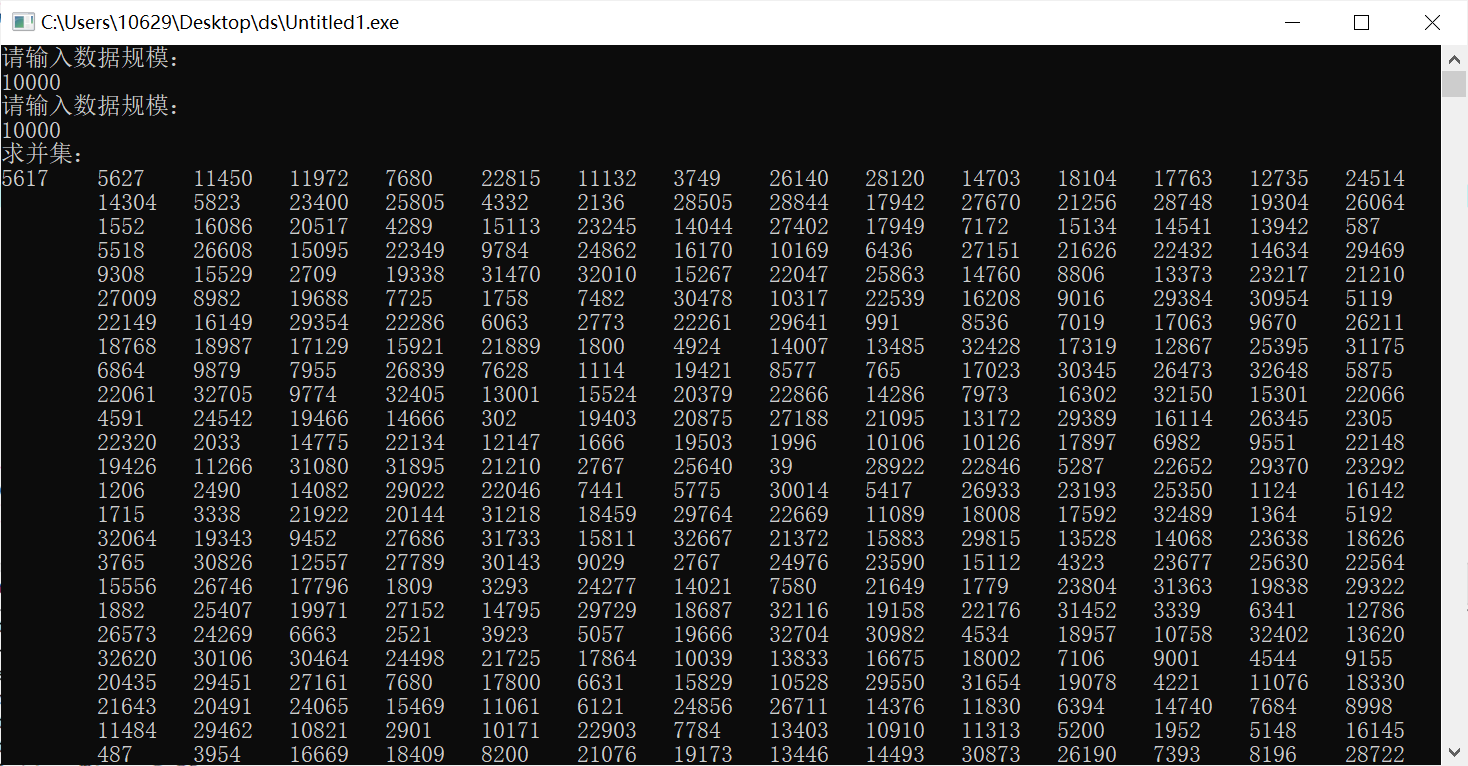
1. 数据规模：100；



1. 数据规模：1000；

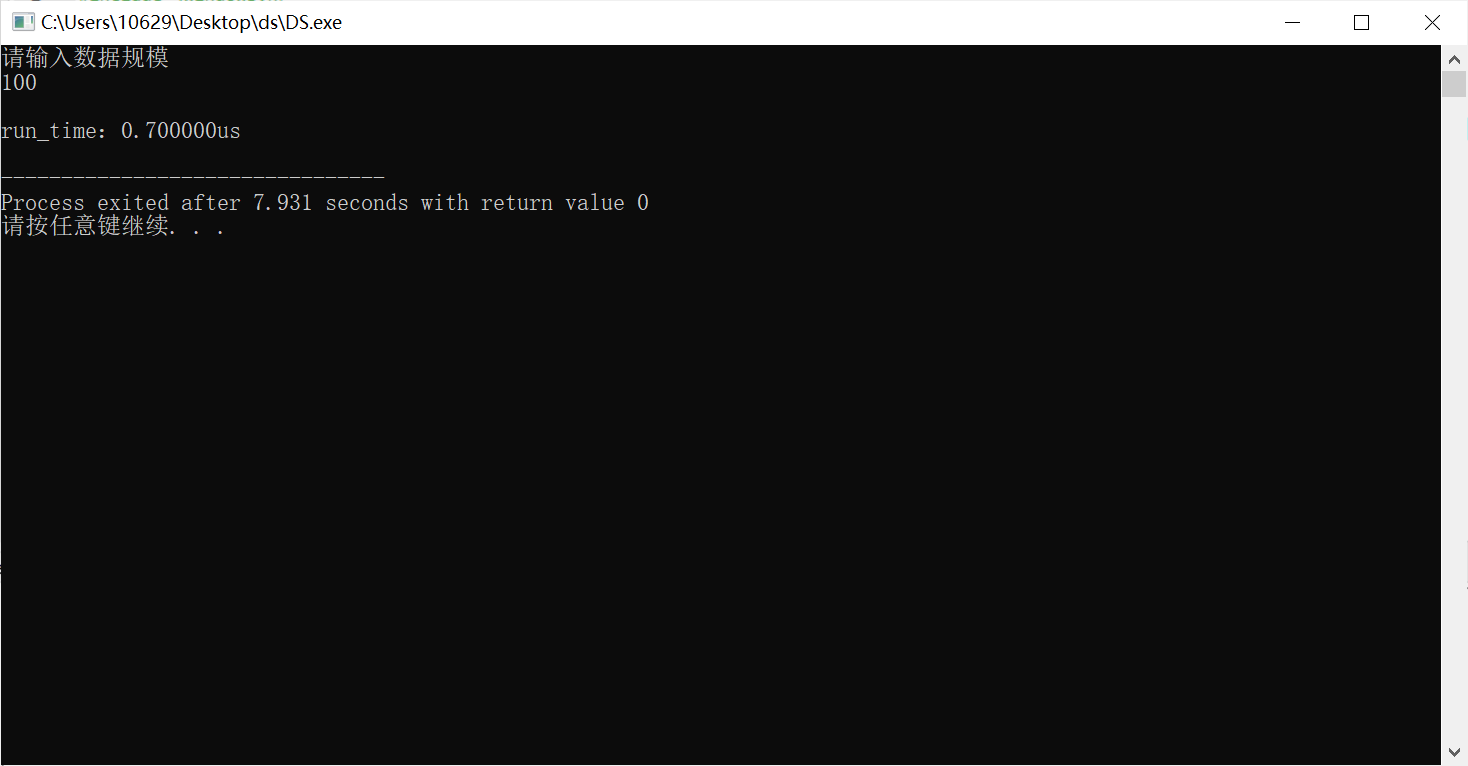
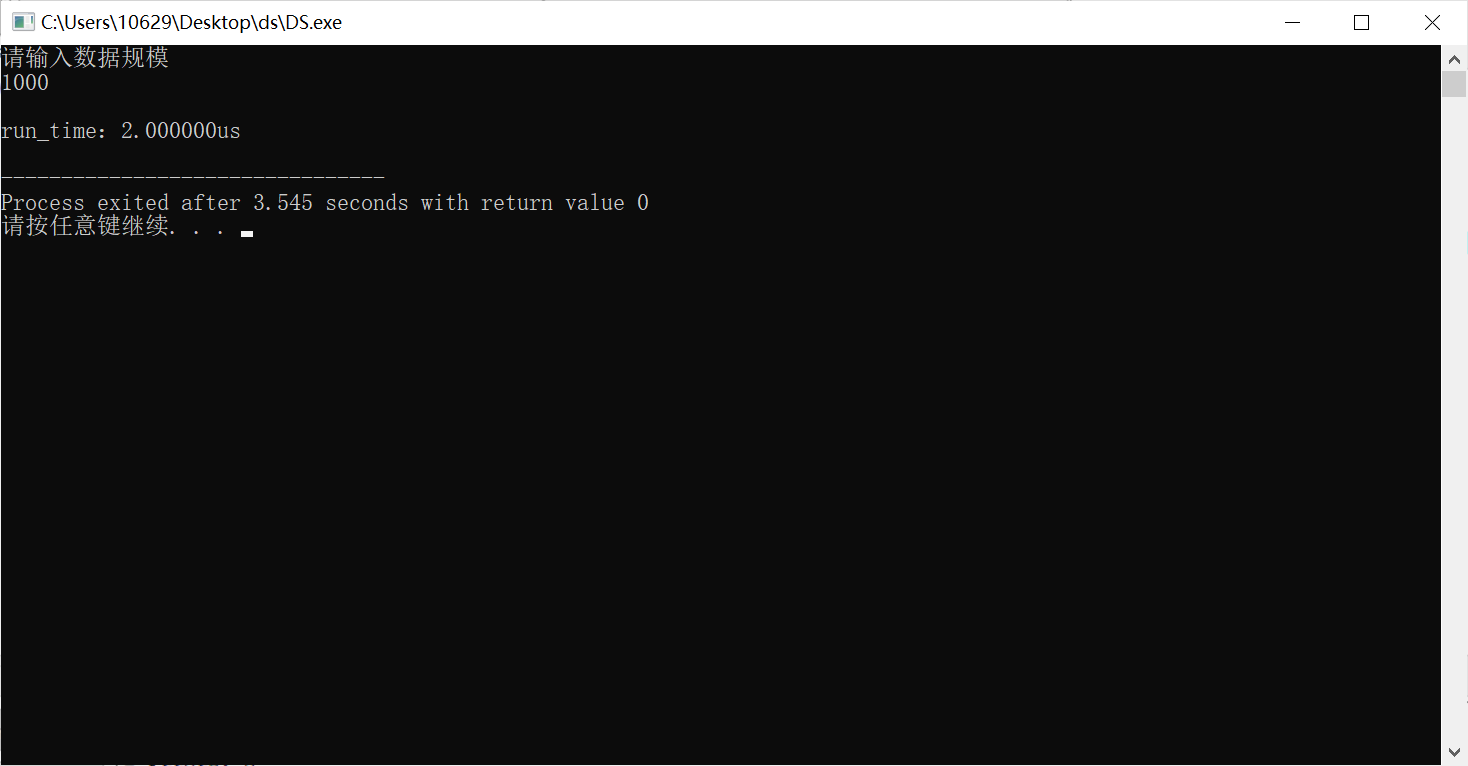
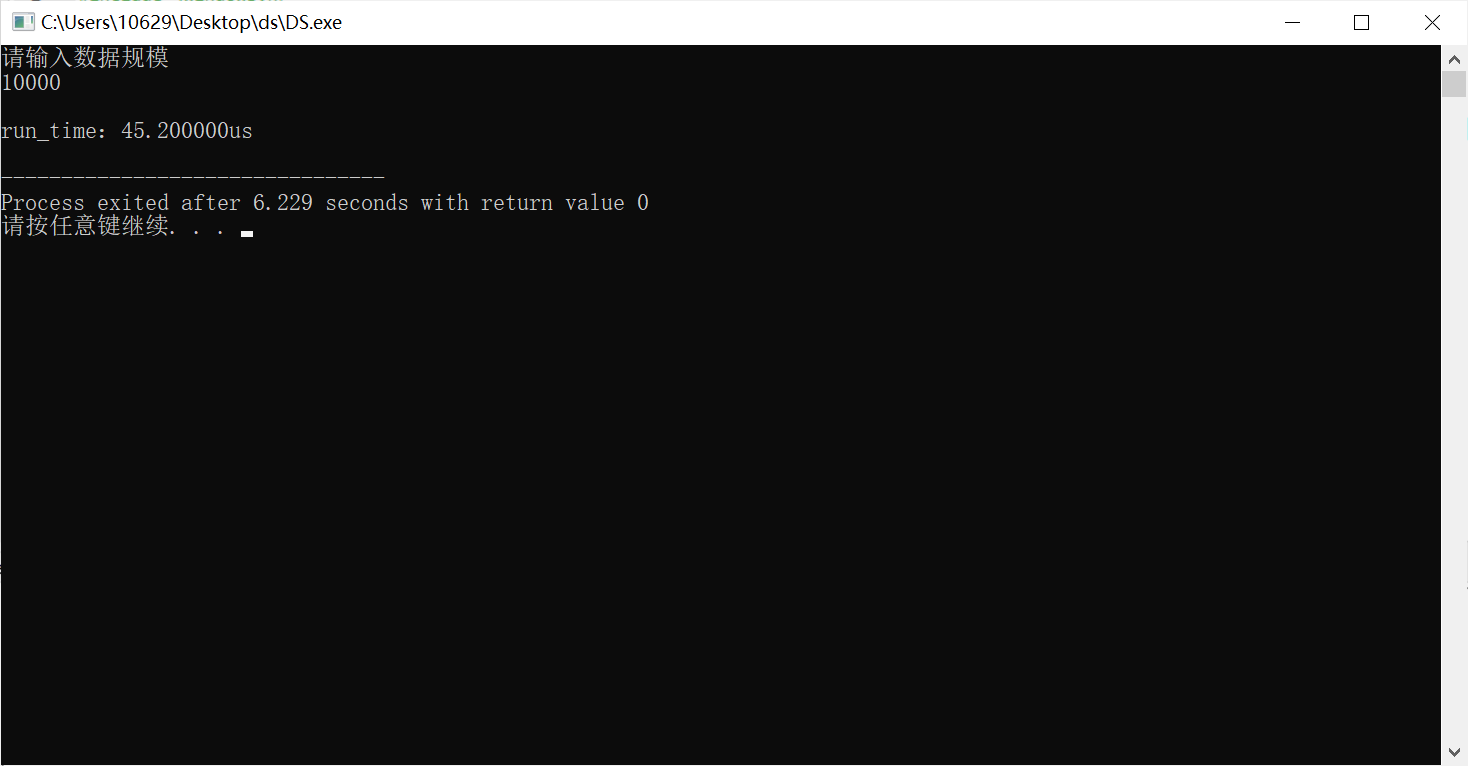


3）数据规模：10000；



1. 实验方案及实验结果

|  |  |
| --- | --- |
| 数据规模 | 运算时间（us） |
| 100 | 0.2 |
| 1000 | 3 |
| 10000 | 28.6 |

4. 实验结果评价分析

根据相关的理论知识对所得到的实验结果进行解释和分析，如分析XX因素对算法世界效率和空间效率的影响，阐明实验结论。

数据规模对算法时间和空间效率的影响；

数据规模越大，算法时间和空间效率越低；数据规模越小，算法时间和空间效率越高；

经过本次实验，发现实验结论与预期结论相符，实验基本成功；

**五、总结与优化**

分析现有方案的不足，进一步分析可能的优化方案，并给出实现思路

不足：元素的插入，删除和查找算法时间和空间效率有待提高；

思路：利用二叉排序树和二叉平衡树来解决规模较大的数据集合也许更优；

**六、心得和体会**

时间转眼即逝，一转眼一学期的数据结构课就已经快要结束了，我对第一节课的时候老师向我们介绍的场景还历历在目，老师兴致勃勃的介绍着数据结构课的作用，重要性。老师每节课都充满活力让我们每节课都不好意思打盹。在学习这门课程以前，我觉得编程只要会简单的写代码就好了。然而实际上数据结构才是计算机界的大佬。着计算机应用领域的延伸与扩展，数据结构类的问题占据了现在计算机技术领域的大部分，数据元素之间的关系已经不是普通方程式能够表达的了，已经变得非常复杂，所以数据结构就成为了不可缺少的角色。孙老师用在线视频和线下讲课结合来让我们学习数据结构，教学方式新颖也让我们有了学习的乐趣，在学期初，我觉得数据结构还是比较简单的，可能是由于之前c 语言学习的还不错，但是对指针掌握的不够熟练，老师给很多设计题，感觉是不小的挑战，导致在数据结构中接触到与指针有关的问题，面对线性表，堆栈，队列，二叉树等问题的时候，都会稍显吃力。但是在不断学习数据结构的过程中不断强化对指针理解，现在我对指针的相关基本知识并且能够熟练运用了。这一学期的学习下来我觉得学好数据结构有以下几点经验：

1. 初步了解算法思想、原理 整明白抽象结构，想要弄清楚一个算法的实现，首先要大致知道这个算法的原理，这是最简单的一步，也是最基础的一步。

2. 查找各种算法资料 发现许多不同的算法，其实书上列举的只是数据结构的一小部分，我们通过查阅资料可以发现很多其他不同的排序算法，而且对于一个算法，也有各种不同的实现方法。有了这些知识储备和实践，我们可以提出问题与同学交流或者解答同学的问题。

3.我们还是要注重看懂弄懂会做书上的代码，毕竟那是基础，最重要的是了解他们的目的，在此基础上深入的了解算法的实现过程，而不是在死记硬背，那样会事倍功半。

4. 坚持上机操作，实践出真知 和所有计算机类的学习一样，数据结构是非常需要动手的一门课程，看书看一天也不如把书上的代码打到电脑上实践一遍，发现错误弄懂，那收获必定匪浅的

5. 勤于练习，寻找感觉 算法是解决问题的方法，掌握树上的只是以后，要去找一些具有综合性的题目来做一做，这些问题里面包含了各种知识点，例如同时蕴含了排序，堆栈的相关知识，只有在解决这类问题的时候，才能知道如何去灵活准确的运用各种算法，才能检验我们是否扎实了课本上的知识。

这学期的数据结构课感觉自己学的还不是很尽力，没有尽力去对待他，但是学到了这种新的学习方式，都是大学生了，应该有自律能力，该学的必须要学，体验了这种全新的上课方式，掌握了学习的主动权，而不是像木头一样坐在桌子上听老师讲，总的来说这学期还是学到了很多东西，希望以后有问题还能向孙老师请教，在这里再次感谢孙老的半年的指导。