组号: 8

图片包含 游戏机, 画

描述已自动生成

上海大学计算机工程与科学学院

**实 验 报 告**

（数据结构2）

学 期：2025-2025年春季

组 长： 高世衡

学 号： 23122725

指导教师： 朱能军

成绩评定： （教师填写）

二〇二五年3月31日

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **小组信息** | | | | |
| 登记序号 | 姓名 | 学号 | 贡献比 | 签名 |
| 1 | 刘天宇 | 23122724 | 33% |  |
| 2 | 高世衡 | 23122725 | 33% |  |
| 3 | 冯俊佳 | 23122721 | 33% |  |
|  |  |  |  |  |

|  |  |
| --- | --- |
| **实验概述** | |
| 实验0 | （熟悉上机环境、进度安排、评分制度；确定小组成员） |
| 实验一 | 有向网的邻接矩阵验证及拓展 |
| 实验二 | 无向网的邻接表验证及拓展 |
| 实验三 | 查找算法验证及设计 |
| 实验四 | (*实验题目*) |

实验三

一、**实验题目**

1. 掌握静态查找表、动态查找表等的定义和存储结构的实现。

2. 分析各种查找表的查找性能，讨论它们的适用情况。  
 3. 根据实际问题的需要，选择相应的查找表解决实际问题。

二、**实验内容**

（1）查找3个数组的最小共同元素。

有3个整数数组 a[]、b[]和c[], 各有aNum、bNum和cNum个元素（aNum, bNum, cNum <= n），而且三者都己经从小到大排列。设计并编写算法找出最小共同元素以及该元素在3个数组中出现的位置，若没有共同元素，则显示“NOT FOUND”，要求算法在最坏情况下的时间复杂度为O(n)。

【输入】

第一行三个数，表示三个数组元素个数，接下来三行是各有序数组。

【输出】

若存在，则输出四个数，分别表示共同元素以及在3个数组中出现的位置；否则输出“NOT FOUND”。

【输入样例】

2 4 5

3 4

4 7 8 9

1 2 3 4 5

【输出样例】

4 2 1 4

（2）求两个有序序列的中位数。

有两个长度为n的有序序列，如果将这两个序列合并成一个有序序列，则处于第n个位置的元素称为这两个序列的中位数。请设计2种求两个有序序列的中位数的算法，要求其中一种算法在最坏情况下的时间复杂度为O(logn)。

【输入】

第一行两个正整数m，n，表示两个序列的长度，接下来两行是各有m，n个元素的有序序列。

【输出】

一个数，就是所求中位数的值。

【输入样例】

8 8

1 3 5 7 9 11 13 15

12 14 16 18 20 22 24 26

【输出样例】

13

（3）二叉排序树的验证和拓展

对于在二叉排序树上删除结点的问题，教材中介绍了4种算法，并实现了其中第1种算法，现要求完成另1种：即移动当前节点左子树到其后继左子树的方式，并用多组测试数据对教材实现的和本题实现的这2种算法进行性能测试，分析比较它们的查找性能。

三、**解决方案**

1、算法设计

1.1 查找3个数组的最小共同元素

* 数据结构和算法思想：

哈希集合（unordered\_set）：

本实验使用了unordered\_set<int>类型的哈希集合，将三个数组分别存储进三个哈希集合中，然后遍历第一个数组中的元素，查找是否也同时存在于另两个集合中，从而判断是否为三个数组的共同元素。相比于map结构体方式记录位置，本实验采用的是值存在性判断方式，强调时间效率和简洁性，适合在只需要找出某个元素是否共同存在的场景下使用。

* 主要操作：

数据读入与集合构建：

1. 用户输入数组a、b和c的元素。
2. 使用unordered\_set<int>对a、b、c分别构建三个集合set\_a、set\_b、set\_c。查找最小共同元素：
3. 遍历数组a中的元素。
4. 对每一个num = a[i]，判断其是否同时存在于set\_b和set\_c。
5. 一旦发现满足条件的第一个元素，则认为其为“最小共同元素”（以 a 中出现顺序为准）。

位置信息定位与输出：

1. 使用find函数在数组中寻找共同元素的位置。
2. 将三个数组中该元素所在的位置（从1开始计数）打印输出。

* 改进方法：

对于哈希表和哈希集合的选取，本实验考虑到哈希表中存在结构体嵌套与键值对应的复杂性，在本实验中可通过哈希集合来避免，提升程序运行效率，因此本实验采用了哈希集合的数据结构。

若需求需输出多个共同元素或按值大小选取最小值，可在遍历中记录所有符合条件的元素后再选最小值进一步改进。

1.2 求两个有序序列的中位数

* 数据结构和算法思想：
* 主要操作：
* 改进方法：

1.3 二叉排序树的验证和拓展

* 数据结构和算法思想：
* 主要操作：
* 改进方法：

1. 源程序代码

2.1查找3个数组的最小共同元素

本算法用于查找三个整数数组中共同存在的最小元素（三者都己经从小到大排列），并输出其在每个数组中的位置（从1开始计数）。程序使用 unordered\_set 哈希集合来快速判断某个元素是否存在于数组中，从而提升查找效率。具体核心代码如下：

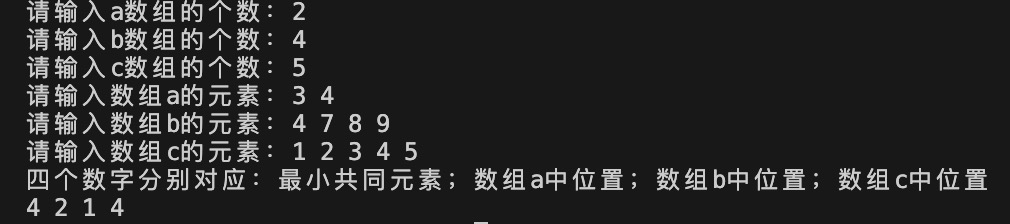
1. // 查找三个数组中共同的最小元素
2. **int** main()
3. {
4. **int** aNum, bNum, cNum;
5. cout << "请输入a数组的个数：";
6. cin >> aNum;
7. cout << "请输入b数组的个数：";
8. cin >> bNum;
9. cout << "请输入c数组的个数：";
10. cin >> cNum;
12. vector<**int**> a(aNum), b(bNum), c(cNum);
14. cout << "请输入数组a的元素：";
15. **for** (**int** i = 0; i < aNum; ++i)
16. cin >> a[i];
18. cout << "请输入数组b的元素：";
19. **for** (**int** i = 0; i < bNum; ++i)
20. cin >> b[i];
22. cout << "请输入数组c的元素：";
23. **for** (**int** i = 0; i < cNum; ++i)
24. cin >> c[i];
26. // 使用unordered\_set存储每个数组的元素
27. unordered\_set<**int**> set\_a(a.begin(), a.end());
28. unordered\_set<**int**> set\_b(b.begin(), b.end());
29. unordered\_set<**int**> set\_c(c.begin(), c.end());
31. **int** result = -1;
32. **for** (**int** i = 0; i < aNum; ++i) {
33. **int** num = a[i];
34. // 检查num是否在b和c中
35. **if** (set\_b.count(num) && set\_c.count(num)) {
36. result = num;
37. **break**;
38. }
39. }
40. **if** (result != -1) {
41. // 查找原数组中的位置（从1开始）
42. **int** pos\_a = find(a.begin(), a.end(), result) - a. begin() + 1;
43. **int** pos\_b = find(b.begin(), b.end(), result) - b. begin() + 1;
44. **int** pos\_c = find(c.begin(), c.end(), result) - c. begin() + 1;
45. cout << "四个数字分别对应：最小共同元素；数组a中位置； 数组b中位置；数组c中位置" << endl;
46. cout << result << " " << pos\_a << " " << pos\_b < < " " << pos\_c << endl;
47. }
48. **else** {
49. cout << "NOT FOUND" << endl;
50. }
51. **return** 0;
52. }

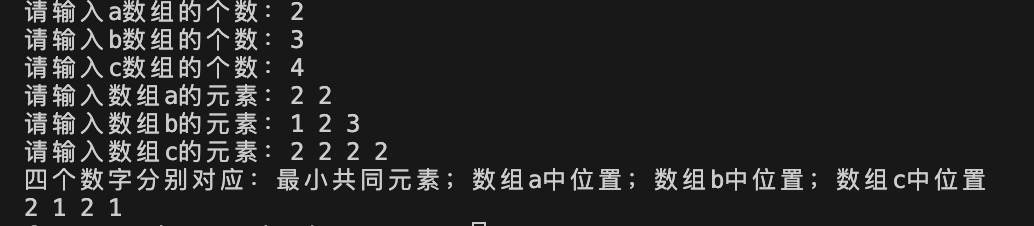
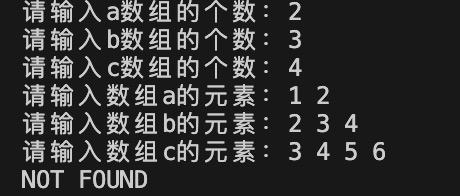
2.2 求两个有序序列的中位数

2.3 二叉排序树的验证和拓展

1. 实验结果

3.1 展示实验结果

* 查找3个数组的最小共同元素



* 求两个有序序列的中位数
* 二叉排序树的验证和拓展

3.2 结果分析

实验结果均与与其结果一致。

1. 算法分析

4.1 查找3个数组的最小共同元素

* 时间复杂度

哈希集合提供均摊O(1)的插入与查找操作，因此包括哈希构建、查找最小公共元素以及定位其在各数组中的位置的总耗时均为线性。考虑到a、b、c的大小均不超过n，因此整体时间复杂度为O(n)。

* 空间复杂度

本程序使用了三个unordered\_set<int>类型的哈希集合，分别存储数组a、b和c中的所有元素。由于每个集合中最多存储对应数组中不重复的元素个数，因此哈希集合的总空间复杂度为O(aNum + bNum + cNum)（aNum、bNum、cNum分别为数组a, b, c的元素个数）。

在输入数组元素时，数组空间程序使用了三个vector<int>类型的数组a、b和c，分别用于存储输入的三个数组数据，因此这一部分的空间复杂度为O(aNum + bNum + cNum)。

因此，由于程序使用了3个输入数组和3个哈希集合，空间均与元素数量成线性关系，故总的空间复杂度可以视为O(max(aNum, bNum, cNum) )，即三者中的最大值。

4.2 求两个有序序列的中位数

* 时间复杂度
* 空间复杂度

4.3 二叉排序树的验证和拓展

* 时间复杂度
* 空间复杂度

1. 总结与心得