**上 海 大 学**

**2019-2020学年冬季学期**

**《数据结构（2）》实验报告**

实 验 组 号： 01

上 课 老 师： 沈 俊

小 组 成 绩：

小组成员成绩表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 学号 | 姓名 | 贡献因子 | 成绩 |
| 1 | 17120193 | 张赛 | 20 |  |
| 2 | 18120189 | 林艺珺 | 20 |  |
| 3 | 18120206 | 徐奕婷 | 20 |  |
| 4 | 18120212 | 叶菁 | 20 |  |
| 5 | 18120221 | 陈思文 | 20 |  |

注：小组所有成员的贡献因子之和为100.

计算机工程与科学学院

2020年03月20日

实验四 排序

一．实验内容

（一）设计性实验：堆排序算法的改进

假设定义最大堆为满足如下性质的完全三叉树：

（1）空树为堆；

（2）每个结点最多有3棵子树，且结点的值不小于所有子树根的值，且所有子树均为最大堆。

按上述述要求完成三叉堆的存储结构设计，再实现三叉堆的向上和向下调整算法，最后完成利用三叉堆进行排序的算法。要求利用大数据量进行测试，同原有堆排序算法在时间上进行对比分析。

（二）综合性实验：运动会团体排名

[问题描述] 学校为了丰富学生生活，提高学生身体素质，每年秋天都会举办全校运动会。运动会设n个比赛项目，每个项目取前6名，分别得8、6、4、3、2、1分，破校记录的每个运动员会双倍计分。全校有m个学院分别组织多名学生参加运动会。学校以学院为单位统计得分，进行团体排名。

团体排名规则如下：

（1）每个学院的比赛得分为该学院所有学生得分的总和。按学院的比赛得分从高到低排。

（2）学院得分一样的按所获冠军人数递减排序；仍然一样，按所获亚军人数递减排序；仍然一样，再按所获季军人数递减排序。

（3）如果学院得分、冠军人数、亚军人数、季军人数都一样，则按参赛人数递减排序，仍然一样则名次并列。

要求根据n个项目的比赛结果，计算各学院的得分等情况，并完成团体排名。

[输入数据]

第1行有2个整数n和m，分别比赛项目数（项目编号为：1到n）和参赛学院数（学院编号为：1到m）。

接下来有m行数据（从第2行到第m+1行），分别表示1-m个学院参赛学生的编号，每行的第1个数k表示参赛学生数目，然后是k个学生的运动员编号。

最后，还有n行数据，分别表示n个项目的比赛结果。每行7个数，第1个数c表示破校记录人数，然后依次是前6名运动员的编号。

[输出数据]

按排名先后顺序输出n个学院的排名表，一共m行，每行2个数：名次和学院编号。

二．主要算法设计

（一）设计性实验：堆排序算法的改进

1. 问题分析与算法设计

对于三叉最大堆来说，堆顶元素是堆中关键字最大的数据元素。利用三叉堆大堆及其运算可以很容易实现堆排序的算法。三叉堆选择顺序存储，使用数组TriheapArr来存储数据元素，堆中数据元素数目CurrentSize，堆中数据元素最大数目为MaxSize。

（1） 初始堆构造

根据参数maxsize分配储存空间，并把数组中元素复制到heapArr数组，然后采用自下而上，自右向左的方法把序列逐步调整形成最大堆。

（2） 向下调整算法

当结点i的子树都为最大堆时，比较结点i的3个孩子结点，沿孩子结点关键字最大的分支进行调整，j指示i的三个孩子中关键字值最大的孩子结点，将结点i与结点j的关键字进行比较，若结点j的关键字大于结点i的，则交换结点i和结点j。再重复进行下一层的比较交换，直到不用交换时。

代码2.1 三叉堆的向下调整算法

|  |
| --- |
| 1. **template** <**class** ElemType> 2. **void** TriHeap<ElemType>::FilterDown(**int** Start, **int** End) 3. { 4. **int** i = Start, j; 5. ElemType temp = TriheapArr[i]; 6. j = 3 \* i + 1; 7. **while**(j <= End) { 8. **if**(j < End && TriheapArr[j] < TriheapArr[j+1]) { 9. j++; 10. **if**(j < End && TriheapArr[j] < TriheapArr[j+1]) j++; 11. } 12. **if**(j < End-1 && TriheapArr[j] >= TriheapArr[j+1] 13. && TriheapArr[j] < TriheapArr[j+2]) j+=2; 14. **if**(temp >= TriheapArr[j]) **break**; 15. **else** { 16. TriheapArr[i] = TriheapArr[j]; 17. i = j; 18. j = 3\*j+1; 19. } 20. } 21. TriheapArr[i] = temp; 22. } |

（3） 向上调整算法

从插入结点j开始，比较结点j的关键字和其双亲结点i的关键字大小，如果结点j的关键字小于其双亲结点i的关键字，则交换两个结点值。向重复进行上一层的比较交换，直到不用交换。

代码2.2 三叉堆的向上调整算法

|  |
| --- |
| 1. **template**<**class** ElemType> 2. **void** TriHeap<ElemType>::FilterUp(**int** End) 3. { 4. **int** j = End,i; 5. ElemType temp = TriheapArr[j]; 6. i = (j - 1) / 3; 7. **while**(j > 0) { 8. **if** (TriheapArr[i] >= temp) **break**; 9. **else** { 10. TriheapArr[j] = TriheapArr[i]; 11. j = i; 12. i = (j - 1) / 3; 13. } 14. TriheapArr[j] = temp; 15. } 16. } |

（4） 三叉堆排序算法

对未排序的数据元素建立三叉堆初始最大堆，堆顶元素是堆中最大的数据元素，交换堆顶元素和堆尾元素，输出堆尾元素，对剩余元素调用向下调整算法重新形成最大堆，重复交换和调用直到所有数据元素输出。

代码2.3 利用三叉堆进行排序的算法

|  |
| --- |
| 1. **template**<class ElemType> 2. **void** TriHeapSort(ElemType \*elem, **int** n) 3. { 4. **int** i, j; 5. TriHeap<ElemType> tri(elem, n, n); 6. **for** (i = n - 1; i > 0; --i) 7. { 8. tri.TriSwap(0, i); 9. j = (i-1)/3; 10. **while**(j >= 0) 11. { 12. tri.FilterDown(0, i-1); 13. j--; 14. } 15. **if** (n <= 50) { 16. cout << "第" << n - i << "趟堆排序结果："; 17. tri.Traverse(Write<ElemType>); 18. cout << endl; 19. } 20. } 21. cout << "排序后序列为："; 22. tri.Traverse(Write<ElemType>); 23. cout << endl; 24. } |

2. 函数功能设计

|  |  |
| --- | --- |
| TriHeap类成员函数 | |
| TriHeap(int maxSize); | 构造空三叉堆 |
| TriHeap(ElemType a[],int maxsize,int n); | 根据数据构造三叉堆 |
| ~TriHeap(){delete []TriheapArr;} | 析构函数 |
| void Traverse(void (\*Visit)(const ElemType &)) const; | 遍历三叉堆 |
| void FilterDown(int Start, int End); | 三叉堆的向下调整算法 |
| void FilterUp(int End); | 三叉堆的向上调整算法 |
| void TriSwap(int m, int n) { Swap(TriheapArr[m], TriheapArr[n]); } | 交换三叉堆 |
| 排序算法 | |
| void TriHeapSort(ElemType \*elem, int n)； | 利用三叉堆进行排序的算法 |

（二）综合性实验：运动会团体排名

1. 问题分析与算法设计

要求输出n个学院的排名表，本题选择了较简单的冒泡进行名次排序。

为了利于比较，设计了结构体College，内含学院得分、冠军人数、亚军人数、季军人数、参赛人数等需要比较的信息，此外还有运动员编号、学院编号等信息。在结构体Sport中存入前六名运动员的编号，根据编号找到运动员所属的学院，把运动员在本项目中的得分、名次信息统计到College中去。

输出排名表时需注意，判断正在输出的学院与下一个待输出的学院是否并列，是则一并输出。

cmp函数

用于比较学院a和学院b的名次。若a在前，返回1；若a与b并列，返回0；否则返回-1。

2. 函数功能设计

|  |  |
| --- | --- |
| SportsMeeting.cpp文件结构 | |
| int cmp(College a, College b)； | 对两学院进行比较，结果返回值 |
| 输入数据读取模块 |  |
| 得分翻倍模块 |  |
| 学院运动员各名次数量统计模块 |  |
| 根据cmp的返回值进行冒泡排序模块 |  |

三．主要数据组织

（一）设计性实验：堆排序算法的改进

代码3.1 三叉堆类

|  |
| --- |
| 1. **template**<**class** ElemType> 2. **class** TriHeap 3. { 4. private: 5. ElemType \*TriheapArr; 6. **int** CurrentSize; 7. **int** MaxSize; 8. public : 9. TriHeap(**int** maxSize); 10. TriHeap(ElemType a[],**int** maxsize,**int** n); 11. ~TriHeap(){**delete** **[]**TriheapArr;} 12. void Traverse(**void** (\*Visit)(const ElemType &)) const; 13. **void** FilterDown(**int** Start, **int** End); 14. **void** FilterUp(**int** End); 15. **void** TriSwap(**int** m, **int** n) {Swap(TriheapArr[m], TriheapArr[n]); } 16. }; |

（二）综合性实验：运动会团体排名

根据题目要求，为学院及运动项目各设计一个结构体。学院结构体中包括int类型的学院得分、运动员编号数组、冠军人数、亚军人数、季军人数及学院编号；运动项目结构体中包括int类型的破纪录人数、前六名编号及六名运动员的得分。数据结构设计如下：

代码3.2

|  |
| --- |
| 1. **const** int MAX = 100; 2. **struct** College 3. { 4. **int** score; //学院得分 5. **int** athlete[MAX]; //运动员编号 6. **int** No\_1; //冠军人数 7. **int** No\_2; //亚军人数 8. **int** No\_3; //季军人数 9. **int** number; //参赛人数 10. **int** college; //学院编号 11. }; 12. **struct** Sport 13. { 14. **int** num; //破纪录人数 15. **int** person[6]; //前六名编号 16. **int** person\_score[6]; //六名运动员的得分 17. }; |

四．测试分析

（一）设计性实验：堆排序算法的改进

样例1 普通型

|  |
| --- |
| 1. **int** k[] = {55, 63, 21, 72, 63, 38, 16, 46, 92, 87, 100, 12}; 2. **int** kk = 12; 3. TriHeapSort(k, kk); // 三叉堆排序 |

样例1 测试结果

A screenshot of a cell phone

Description automatically generated

图4-1-1 样例1测试

样例2 满三叉树的最大堆 正序

|  |
| --- |
| 1. **int** l[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}; 2. **int** ll = 13; 3. TriHeapSort(l, ll); // 三叉堆排序 |

样例2 测试结果

A screenshot of a cell phone

Description automatically generated

图4-1-2 样例2测试

样例3 满三叉树的最大堆 逆序

|  |
| --- |
| 1. **int** m[] = {13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1}; 2. **int** mm = 13; 3. TriHeapSort(m, mm); // 三叉堆排序 |

样例3 测试结果

A screenshot of a cell phone

Description automatically generated

图4-1-3 样例3测试

样例4 100个随机数三叉堆排序（与原有堆排序算法对比）

|  |
| --- |
| 1. **int** nn = 100; 2. **int** n[nn]; 3. for(**int** j=0;j<nn;++j) 4. n[j]=rand()%100; 5. TriHeapSort(n,nn); // 三叉堆排序 6. HeapSort(n,nn); // 堆排序 |

样例4 测试结果

*由于过程量庞大，此处省却排序中间结果，可运行程序直接查看。*

A screenshot of a cell phone

Description automatically generated

图4-1-4样例4测试

样例5 1000个随机数三叉堆排序（与原有堆排序算法对比）

|  |
| --- |
| 1. **int** rr = 1000; 2. **int** r[rr]; 3. for (**int** j = 0; j < rr; ++j) 4. r[j] = rand() % 100; 5. TriHeapSort(n,nn); // 三叉堆排序 6. HeapSort(n,nn); // 堆排序 |

样例5 测试结果

*由于过程量庞大，此处省却排序中间结果，可运行程序直接查看。*

A screenshot of a cell phone

Description automatically generated

图4-1-5(1) 样例5三叉堆测试

A picture containing building, window, large, white

Description automatically generated

图4-1-5(2) 样例5堆测试

（二）综合性实验：运动会团体排名

[输入数据]

第1行有2个整数n和m，分别比赛项目数（项目编号为：1到n）和参赛学院数（学院编号为：1到m）。

接下来有m行数据（从第2行到第m+1行），分别表示1-m个学院参赛学生的编号，每行的第1个数k表示参赛学生数目，然后是k个学生的运动员编号。

最后，还有n行数据，分别表示n个项目的比赛结果。每行7个数，第1个数c表示破校记录人数，然后依次是前6名运动员的编号。

[输出数据]

按排名先后顺序输出n个学院的排名表，一共m行，每行2个数：名次和学院编号。

样例1 题目给定样例

[输入样例]

2 3

4 101 301 402 203

5 102 201 302 303 401

3 203 103 403

2 101 301 102 201 203 103

2 303 401 402 303 103 403

[输出样例]

1 2

2 1

3 3

样例1 测试结果

A screenshot of a cell phone

Description automatically generated

图4-2-1 样例1测试结果

样例2 两个学院完全并列第一

[输入样例]

2 3

4 101 201 301 401

4 102 202 302 402

4 103 203 303 403

2 101 202 301 402 103 303

2 102 201 302 401 203 403

[输出样例]

1 1

1 2

3 3

样例2 测试结果

A screenshot of a cell phone

Description automatically generated

图4-2-2 样例2测试结果

样例3 三个学院完全并列第一

[输入样例]

3 4

6 101 201 301 401 501 601

6 102 202 302 402 502 602

6 103 203 303 403 503 603

1 104

2 101 201 301 401 501 601

2 102 202 302 402 502 602

2 103 203 303 403 503 603

[输出样例]

1 1

1 2

1 3

4 4

样例3 测试结果

A screenshot of a cell phone

Description automatically generated

图4-2-3 样例3测试结果

样例4 三个学院完全并列第二

[输入样例]

3 5

6 101 201 301 401 501 601

6 102 202 302 402 502 602

6 103 203 303 403 503 603

6 104 204 304 404 504 604

1 105

2 104 404 301 401 501 601

2 204 504 302 402 502 602

2 304 604 303 403 503 603

[输出样例]

1 4

2 1

2 2

2 3

5 5

样例4 测试结果

A screenshot of a cell phone

Description automatically generated

图4-2-4 样例4测试结果

样例5 两个学院得分相同但冠军数不同

1号学院有1名破纪录的冠军，2号学院有2名未破纪录的冠军，另有相同数量未破纪录的亚、季军。

[输入样例]

3 4

6 101 201 301 401 501 601

6 102 202 302 402 502 602

6 103 203 303 403 503 603

1 104

1 101 301 401 501 601 103

0 102 302 402 502 602 203

0 202 303 403 503 603 104

[输出样例]

1 2

2 1

3 3

4 4

样例5 测试结果

A screenshot of a cell phone

Description automatically generated

图4-2-5 样例5测试结果

样例6 两个学院得分相同但亚军数不同

1号学院有1名破纪录的亚军，2号学院有2名未破纪录的亚军，另有相同数量破纪录的冠军、未破纪录的季军。

[输入样例]

3 4

6 101 201 301 401 501 601

6 102 202 302 402 502 602

6 103 203 303 403 503 603

1 104

2 101 201 401 501 601 203

1 102 202 402 502 602 303

0 103 302 104 403 503 603

[输出样例]

1 2

2 1

3 3

4 4

样例6 测试结果

A screenshot of a cell phone

Description automatically generated

图4-2-6 样例6测试结果

样例7 两个学院得分相同但季军数不同

1号学院有1名破纪录的季军，2号学院有2名未破纪录的季军，另有相同数量破纪录的冠军、亚军。

[输入样例]

3 4

6 101 201 301 401 501 601

6 102 202 302 402 502 602

6 103 203 303 403 503 603

1 104

3 101 201 301 501 601 104

2 102 202 302 502 602 303

0 103 203 402 503 403 603

[输出样例]

1 2

2 1

3 3

4 4

样例7 测试结果

A screenshot of a cell phone

Description automatically generated

图4-2-7 样例7测试结果

样例8 两个学院得分相同但参赛人数不同

1号学院、2号学院有相同数量破纪录的冠军、亚军、季军。

[输入样例]

3 3

7 101 201 301 401 501 601 701

6 102 202 302 402 502 602

6 103 203 303 403 503 603

3 101 201 301 401 501 601

3 102 202 302 402 502 602

0 103 203 303 403 503 603

[输出样例]

1 1

2 2

3 3

样例8 测试结果

A screenshot of a cell phone

Description automatically generated

图4-2-8 样例8测试结果

五．课程设计中遇到的问题和解决方法

（一）设计性实验：堆排序算法的改进

多个样例在不同的case语句中构造三叉堆，选择其中一个case时，其他case的构造三叉堆直接跳过了，编译会出问题。只要在每个case加“{}”符号，将每个构造三叉堆的作用域限制在case中，互不影响，就可解决问题。

（二）综合性实验：运动会团体排名

调试时发生了栈溢出。检查后发现是因为局部变量占用空间过大导致栈的上溢。

此时定义语句如下：const int MAX = 10000; College clg[MAX];Sport spt[MAX];

改为const int MAX = 100就可解决问题。

以后会注意在定义大结构、大数组时小心。

六．实验总结

17120193 张赛

本次设计性实验中，小组选择框对于三叉堆排序的实现。在实验进行前期，我对堆的调整算法进行了编写，但在没有理解题意的情况下，单独编写调整算法并没有实现类的构造，给负责后续功能的组员带来了不必要的麻烦。在后续的实验中，即时调整，也克服了构造方面的问题。

在后续的实验中，也设计了计时器，对三叉堆用于大量数据的排序性能进行测试，并且和二叉堆排序的性能作了对应的比较。相对来说，三叉堆在处理100000级以上量数组的排序时，有明显的时间优势。

18120189 林艺珺

本次上机实验中，我主要负责了综合性实验的测试部分以及整体的查错和报告整理。在测试过程中，我发现运动会团体排名一题，类似ACM的比赛题，因此对于输出的要求极高。仔细研读几遍题目后，发现组员理解的与题意有所偏差，因而对源程序进行了修改。在测试方面，根据题意的排序方式，设计了多组测试数据，以确保面面俱到，再次深刻体会到了软件测试的重要性。在大数据量测试三叉堆排序的过程中，发现大量的中间过程输出非常不利于测试查看，因此对于数据量加入了判断条件，以便测试结果的输出。

这次实验分工默契、进度可控，整体上在预计的时间内完成了任务。这应该是最后一次数据结构的小组实验了，本学期从一开始的流程生疏，到这次的分工明确、流程管理妥当，一是从中感受到了数据结构这门课程对于实践的高标准高要求，二是意识到计算机学习中实操、上手的重要性，三是懂得了团队配合的技巧与道理，明白了自己能力不足的时候，可靠、互相信任的团队是完成任务的重要存在。

18120206 徐奕婷

这次实验加深了我对数据结构排序算法的认识。

我主要负责综合性实验的设计，选用了结构体的数据结构，把需要比较的数据都存放在结构体中，这样利于排序比较，但同时也带来了不便。由于结构体设计的过大，在主函数中定义结构体数组时，如果数组太大就会造成栈溢出。

实验的设计在我看来是中规中矩的，由于水平受限，我还不能很好地用一些更高效的方式来编写，所以也没有什么突出的创新点。不过设计过程中对排序进行了复习巩固，我认为还是不错的，有些理论会在实际操作中印象更加深刻。

数据结构是一门很需要实操的课程，浅尝辄止是远不够的，哪怕是已经掌握的知识也要时时运用，才不会淡忘。

18120212 叶菁

这次实验让我对堆排序有了更深入的理解，利用三叉堆进行堆排序是原堆排序算法的改进，通过TriHeap类的创建，多次调用FilterDown()，创建了初始堆。堆中最大数据元素在堆的最顶端，重复输出顶端元素再调整成最大堆能依次获取序列中未输出的最大元素。通过大数据量与原有堆排序在时间上对比比较，发现改进后的堆排序算法慢于原堆排序算法，三叉堆的三个孩子间的比较次数比二叉堆多1次。在这次实验中，我们小组通过微信群分工、讨论，合作无间。

18120221 陈思文

三叉堆的存储结构类似二叉树堆的设计。在实现三叉堆的向上和向下调整算法时，产生了一些问题。我在设计向上调整算法时，模仿了二叉树堆的向上调整算法。然而一开始只考虑了一个结点的第一个孩子为最小或最大的状况。一开始自己测试了一个数据，觉得没什么问题。但小组成员后续的测试发现我的算法不够完善。通过测试出现的问题，我添加了有三个孩子的情况下，第一个孩子是中间值的情况。这次实验感谢小组成员的测试帮助，使算法更加完善。这次实验也告诉我以后设计算法时，要尽量考虑到所有情况。