# 

# 《数据结构》

# 课程设计报告

**设计题目：**分布式视频管理系统

**姓 名\_\_\_\_\_\_\_\_\_\_\_戴天宇\_\_\_\_\_\_\_\_**

**学 号\_\_\_\_\_\_\_\_\_\_\_10211470\_\_\_\_\_\_\_\_**

**班 级\_\_\_\_\_\_\_\_\_\_\_2010211312\_\_\_\_\_**

**同组组员\_\_\_\_\_戴天宇 陈天阳 李昕阳**

**实验报告日期： 2012.5.31**

1. 需求分析（详细说明要实现的功能和功能间的关系）

* 有一台管理PC机和多台普通PC机，每台普通PC机上有存放一些视频文件；

管理PC机和其他PC机通过网络互联在一起，组成网状结构（加权图）；

管理PC机通过遍历整附图，来查询这些PC机，获取每台PC机上的视频文件的信息，包括：文件名，视频类型，视频格式，视频大小，所在的PC机名称等；

* 管理系统将所有视频资源按照视频类型以树状结构组织起来，叶子节点必须是视频文件；

视频资源会动态的增加或者减少，需要能够进行调整；有的PC机上会动态增加一些视频文件，或者减少一些视频文件，管理系统要能够相应的调整树的结构；

* 在界面上可以查看不同类型的视频文件，通过在这棵类型树上选定某种视频类型，在列表中将该类型的所有视频资源显示出来；
* 显示结果时，可以按照视频资源的名称、资源的格式、资源的大小进行排序；（选择三种排序算法，分析排序的性能）；
* 每台PC机可以提出检索要求，通过文件名，检索与目标字符串相同的视频文件；（选定两种检索方法，比较性能，按照检索算法的要求可以重新组织视频文件信息）

在界面上输入PC机名称，检索的文件名；（PC机名称为提出查询的PC机）

* 在检索出的视频文件中，计算每个文件所处PC机与提出检索要求PC机的最短距离，并将每条最短路径保存下来；然后按照此距离进行排序，将排序结果显示出来。

主要负责分别采用归并排序，冒泡排序和选择排序；统计排序所用时间，对三种不同排序方法的性能进行了比较。

1. 算法思想（对于实现较为复杂的功能，要写出所使用的算法和算法的思路）

排序部分使用了冒泡排序，归并排序，选择排序。

//两路归并排序//

使用两路归并排序对视频文件名进行了排序

归并（Merge）排序法是将两个（或两个以上）有序表合并成一个新的有序表，即把待排序序列分为若干个子序列，每个子序列是有序的。然后再把有序子序列合并为整体有序序列。申请空间，使其大小为两个已经排序序列之和，该空间用来存放合并后的序列,设定两个指针，最初位置分别为两个已经排序序列的起始位置，比较两个指针所指向的元素，选择相对小的元素放入到合并空间，并移动指针到下一位置，重复步骤直到某一指针达到序列尾，将另一序列剩下的所有元素直接复制到合并序列尾。

时间复杂度O(nlogn) 空间复杂度O(n)

/\*private void mergesort(int[] arry, string[] nameset, int start, int end)

{

//若data为null，或只剩下1 or 0个元素，返回，不排序

if (null == arry || arry.Length <= 1 || null == nameset || nameset.Length <= 1)

{

return ;

}

int middle = (start + end) >> 1;

int[] left = new int[middle], right = new int[end - middle], result = new int[end];

int i = 0, j = 0;

foreach (int x in data)//开始排序

{

if (i < middle)//填充左数组

{

left[i] = x;

i++;

}

else//填充右数组

{

right[j] = x;

j++;

}

}

if (start < end)

{

// 归并的基本思想

// 排左边

mergesort(arry, nameset, start, middle);

// 排右边

mergesort(arry, nameset, middle + 1, end);

// 合并

merge(arry, nameset, start, middle, end);

}

}

\*/

//冒泡排序

使用冒泡排序对视频文件大小进行了排序

依次比较相邻的两个数，将小数放在前面，大数放在后面。即在第一趟：首先比较第1个和第2个数，将小数放前，大数放后。然后比较第2个数和第3个数，将小数放前，大数放后，如此继续，直至比较最后两个数，将小数放前，大数放后。至此第一趟结束，将最大的数放到了最后。在第二趟：仍从第一对数开始比较（因为可能由于第2个数和第3个数的交换，使得第1个数不再小于第2个数），将小数放前，大数放后，一直比较到倒数第二个数（倒数第一的位置上已经是最大的），第二趟结束，在倒数第二的位置上得到一个新的最大数（其实在整个数列中是第二大的数）。如此下去，重复以上过程，直至最终完成排序。

冒泡排序总的时间复杂度为O(n\*n)。

void BubbleSort(int[] arry,string[] nameset)

{

for (int j = 1; j < nameset.Length; j++)

{

for (int i = 0; i < nameset.Length - 1; i++)

{

if (String.Compare(nameset[i] , nameset[i + 1])>0)

{

//交换名字数组顺序

string tempname = nameset[i];

nameset[i] = nameset[i + 1];

nameset[i + 1] = tempname;

//交换序列号数组顺序

int temp = arry[i];

arry[i] = arry[i + 1];

arry[i + 1] = temp;

}

}

}

}

//选择排序

使用选择排序，对视频文件类型进行了排序

n个记录的文件的直接选择排序可经过n-1趟直接选择排序得到有序结果：

初始状态：无序区为R[1..n]，有序区为空。

第1趟排序，在无序区R[1..n]中选出关键字最小的记录R[k]，将它与无序区的第1个记录R[1]交换，使R[1..1]和R[2..n]分别变为记录个数增加1个的新有序区和记录个数减少1个的新无序区。

第i趟排序，第i趟排序开始时，当前有序区和无序区分别为R[1..i-1]和R(i..n)。该趟排序从当前无序区中选出关键字最小的记录 R[k]，将它与无序区的第1个记录R交换，使R[1..i]和R分别变为记录个数增加1个的新有序区和记录个数减少1个的新无序区。

这样，n个记录的文件的直接选择排序可经过n-1趟直接选择排序得到有序结果。

复杂度分析：选择排序是不稳定算法，最好的情况是最好情况是已经排好顺序，只要比较   
n\*(n-1)/2次即可，最坏情况是逆序排好的，那么还要移动 O(n)次,由于是低阶故而不考虑   
不难得出选择排序的时间复杂度是 O(n^2)

public void SelectSort(int[] arry,string[] nameset)

{

int min;

for (int i = 0; i < nameset.Length - 1; i++)

{

min = i;

for (int j = i + 1; j < nameset.Length; j++)

{

if (String.Compare(nameset[j], nameset[min])<0)

min = j;

}

string temp = nameset[min];

nameset[min] = nameset[i];

nameset[i] = temp;

int tem = arry[min];

arry[min] = arry[i];

arry[i] = tem;

}

}

1. 整体设计（整个设计包含哪些函数，函数间的相互关系，每个函数所实现的功能，主要函数实现的过程）

按照给定格式的加权图读取整个网络的拓扑形式；根据拓扑形式，使用迪杰特斯拉算法求出每台PC到个个PC的最短路径长度，保存为路程表，用于之后的最短路程排序；建立一棵树，将所有文件按照类别分类添加到树上，视频文件作为叶子节点；在按照视频类别进行文件检索和显示的时候，即按照广度遍历搜索方式进行检索；在视频的添加和删除过程中，对树的节点进行操作，添加删除叶子节点，即删除了对应的视频文件；在排序部分，根据排序关键字的不同，分别采用归并排序，冒泡排序和选择排序；统计排序所用时间，对三种不同排序方法的性能进行了比较。

个人完成部分是对查找到的文件按不同关键字进行排序并输出结果。

1. 整体时间复杂度分析（对于主要函数分析时间复杂度）

两路归并排序 时间复杂度O(nlogn) 空间复杂度O(n)

冒泡排序 冒泡排序总的时间复杂度为O(n\*n)。

选择排序 选择排序是不稳定算法，最好的情况是最好情况是已经排好顺序，只要比较   
n\*(n-1)/2次即可，最坏情况是逆序排好的，那么还要移动 O(n)次,由于是低阶故而不考虑，时间复杂度是 O(n^2)

1. 测试（给出测试实例和测试的结果）

运行环境：

VS2010

Windows 7

未排序

名称排序

格式排序

大小排序

1. 设计中的问题和心得（设计中遇到的问题和解决的方法）

设计中，对不同关键字进行排序的时候，选择了不同的排序方法，对于视频文件名关键字，选择了归并排序，按照字符转换后的大小排序；对于视频文件大小关键字，选择了冒泡排序，时间复杂度虽然有所增加，但是保证了稳定排序；对于视频类型的排序，使用了选择排序，一是为了比较各种常用排序的使用方法，二是考虑到了文件类型有限。

1. 设计的改进（设计中可以进一步优化和改进的地方）

设计中在排序算法的设计上，考虑了快速排序，冒泡排序，选择排序，归并排序，最后选择了后三者，旨在保证排序过程的稳定性，但是在文件量过多的情况下会使时间消耗增加。在设计过程中还可以考虑使用多关键字不稳定排序，但是能力有限，不稳定程度比较高，达不到预期要求，不过在这个部分还有很大的提升空间。

1. 设计思考（通过此次设计掌握了哪些内容，有什么收获）

通过本次程序设计，对多种排序方式有了更多的了解，已经可以熟练掌握并根据数据特点和具体要求选择最适合的算法。同时尝试了使用多关键字不稳定排序等综合性较强的算法进行设计，增强了自己的能力。在这次程序设计中，输入输出和排序方式选择方面都需要与界面控制相联系，在接口的设计上，函数嵌套、调用方面也有了更多的实践经验。