**数据结构课程设计**

**班 级： 1616202**

**学 号：161620222**

**姓 名： 刘鸿搏**

**指导老师： 秦小麟**

**日 期： 2018.1.8**

**目录**

**购物网站信息管理………………………………3**

**迷宫问题…………………………………………5**

**二叉树的应用……………………………………7**

**Huffman编码与应用……………………………10**

**无线传感器网络 ………………………………12**

**排序算法比较 …………………………………15**

1. **购物网站信息管理**
   1. **题目描述**

1、购物网站信息管理（必做）（链表）

[问题描述]

设计一个程序，对商铺信息管理，商铺信息包括：商铺编号，商铺名，信誉度，（商品名称1，价格1，销量1），（商品名称2，价格2，销量2），（商品名称3，价格3，销量3）…。

假设商品名称包括(毛巾，牙刷，肥皂，牙膏，洗发水，沐浴露等)，每个商铺具有其中事先确定若干商品及价格，由文件输入，销量初始为0。

[基本要求]

（1）建立一个单向链表存储所有商铺信息（至少30个），以编号为序，编号从1开始递增，从文件中读取数据，并能将数据存储在文件。商铺信息结点的数据结构自行设计。

（2）可以增、删商铺。增加商铺，编号自动加一，插入链表尾部；删除商铺以编号为准，并修改后续结点的编号，保持编号连续性。

（3）可以增、删选定商铺中的商品，修改商品价格。

（4）查询某一种商品名称，建立一个双向循环链表，结点信息是包含该商品的所有商铺编号，商铺名，信誉度，商品名称，价格，销量，按销量高至低排序，并可逐一显示。

（5）购买某一商铺的商品，修改单向链表中商品的信息的销量。

（6）任何的商铺信息变化，实现文件存储。

* 1. **采用的数据结构**

采用线性表数据结构，用链表存储实现之。

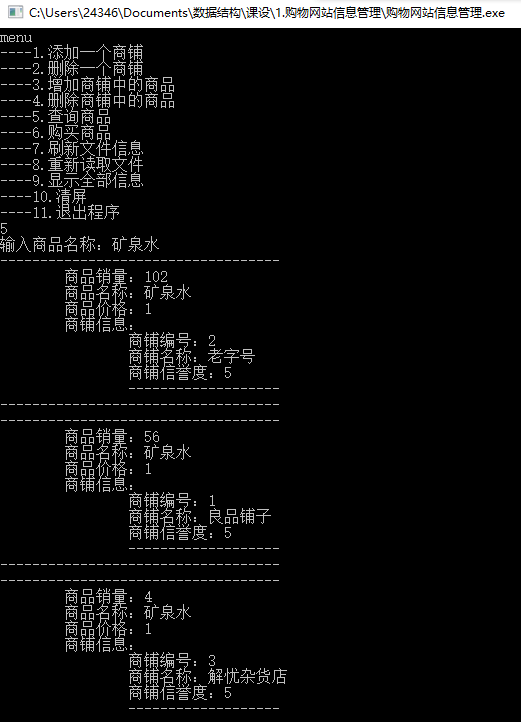
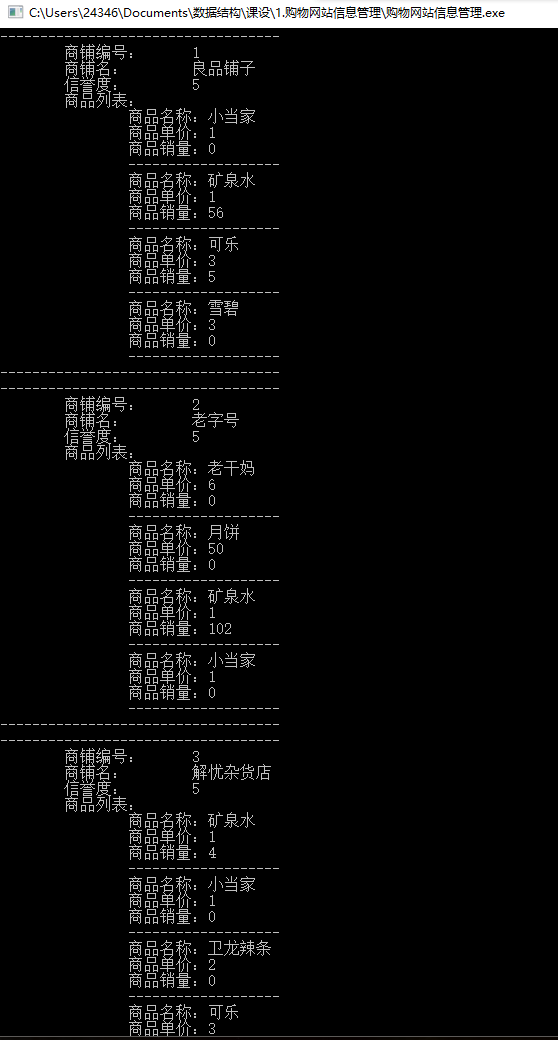
并将修改后的链表信息存入文件，商铺信息和商品信息分别存入shop.dat和goods.dat文件中。

* 1. **算法设计思想**

程序读取文件时会同时打开两个文件，将商铺与商品信息匹配后添加到链表尾。

由于通过write函数将包含其他结构体的结构体写入文件后，无法用read函数正常读取，所以将商铺和商品节点分存于两个文件。因此带来了读取文件信息时的商品与商品匹配问题，考虑到商品节点是按商铺次序顺序存到文件的，再在商品结构体中添加商铺编号信息就可以解决读取信息时的匹配问题。

查询同一商品在不同商铺的销量功能，从网站信息链表中提取出此商品销量链表，并在建立链表是时按销量递减排序。

* 1. **源程序**
  2. **测试数据和结果**
  3. **算法的时间复杂度和算法的改进方法**
     + - 1. **主要函数时间复杂度**

1. void init(shop \*head){} //从文件中读取信息

函数中包含：while(!infile1.eof()){while(!infile2.eof()){}}

**时间复杂度：O（n1+n2） n1为商铺数，n2为 商品数**

1. void refreshfile(shop \*head){} //向文件存储信息

函数中包含：for(;p!=NULL;p=p->next){for(;pg!=NULL;pg=pg->next){}}

**时间复杂度为：O（n1+n2） n1为商铺数，n2为 商品数**

1. void addshop(shop \*head){}

函数中包含：while(1){if(searchshopname(head,pn->shopname)){}}

**时间复杂度为：O（n1+n2） n1为商铺数，n2为 商品数**

1. void deleteshop(shop \*head){}

函数中包含：for(ps=ps->next;ps!=NULL;ps=ps->next,++i){}

**时间复杂度为：O（n） n为商铺数**

1. void purchase(shop \*head){}

函数中包含：for(;ps!=NULL;ps=ps->next){for(goods \*pg=ps->goodhead;pg->next!=NULL;pg=pg->next){}}

**时间复杂度为：O（n1+n2） n1为商铺数，n2为 商品数**

* + - * 1. **算法改进方法**

**所有算法时间复杂度都已最小化**

1. **迷宫问题**
   1. **题目描述**

2、迷宫问题（必做）（栈与队列），(选做)（深度搜索，广度搜索）

[问题描述]

利用栈操作实现迷宫问题求解。

[基本要求]

（1）从文件中读取数据，生成模拟迷宫地图，不少于10行10列。

（2）给出任意入口和出口，显示输出迷宫路线。

* 1. **采用的数据结构**

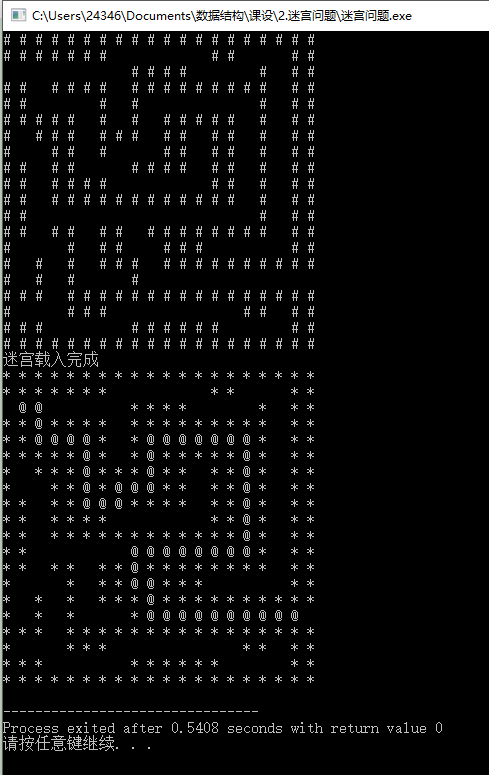
采用栈数据结构，用数组存储实现之。

* 1. **算法设计思想**

程序启动后从文件中读取迷宫信息，将其存入二维数组。

找到起点和终点，对起点调用path()函数，该函数思想是：从当前点开始，若在上下左右方向找不到路则将当前位置标记为已死胡同并返回；若找到了路，则将当前位置标记为已经过，并递归调用path()函数到下一位置。若到达终点，则将flag置1，并递归出栈，出栈过程中不对路线标记做改动。最后输出已经过节点即为所求路线。

* 1. **源程序**
  2. **测试数据和结果**



* 1. **算法的时间复杂度和算法的改进方法**
     + - 1. **主要函数时间复杂度**

void path(int pos[2]){}

函数包含四个并列的递归调用。

**时间复杂度为O（n） n为迷宫总面积**

* + - * 1. **算法改进方法**

**可以改用非递归方法提升算法效率。**

1. **二叉树的应用**
   1. **题目描述**

3、二叉树的应用 (必做)（二叉树）

[问题描述]

编程实现二叉树的建立，先序、中序、后序（递归和非递归方法）、层序遍历。求二叉树的高度、宽度，结点数。判断是否为二叉排序树。

[基本要求]

（1） 从文件中读入建树信息，树的节点数目不小于20个，树的高度不小于4。

（2） 采用二叉链表结构。

（3） 至少2组输入数据，分别是二叉排序树和不是二叉排序树。

* 1. **采用的数据结构**

采用二叉树数据结构，用链表存储实现之。

* 1. **算法设计思想**

先序遍历递归算法：

void POT(BiTNode \*p){

if(p){

cout<<p->data<<"\t";

POT(p->lchild);

POT(p->rchild);

};

}

先序遍历非递归算法：

void PreOrderTraverseNR(){

stack<BiTNode \*> node;

BiTNode \*n=bt;

while(n!=NULL||!node.empty()){

while(n!=NULL){

cout<<n->data<<"\t";

node.push(n);

n=n->lchild;

}

if(!node.empty()){

n=node.top();

node.pop();

n=n->rchild;

}

}

cout<<endl;

}

判断是否为二叉排序树只需检查中序遍历结果是否为有序数列即可。

* 1. **源程序**
  2. **测试数据和结果**



* 1. **算法的时间复杂度和算法的改进方法**
     + - 1. **主要函数时间复杂度**

void POT(BiTNode \*p){};

函数包含递归调用遍历了所有节点

**时间复杂度为O（n） n为节点数**

void CBT(BiTNode\* &root){};

函数包含递归调用读取文件信息以及每次递归都有for循环置文件读取指针于待读取字符之前。

**时间复杂度为O（n\*n） n为节点数**

void BTD(BiTNode \*root){};

函数包含递归遍历了每个节点。

**时间复杂度为O（n） n为节点数**

void PreOrderTraverseNR(){};

函数包含while(n!=NULL||!node.empty()){while(n!=NULL){}}非递归前序遍历

**时间复杂度为O（n） n为节点数**

* + - * 1. **算法改进方法**

CBT（）建树函数若改为非递归调用，就不必加for循环多次重复读取文件，时间复杂度变为O（n）

1. **Huffman编码与应用**
   1. **题目描述**

4、Huffman编码与解码(必做)（Huffman编码、二叉树）

[问题描述]

对一篇不少于2000字符的英文文章，统计各字符出现的次数，实现Huffman编码，以及对编码结果的解码。

[基本要求]

（1） 输出每个字符出现的次数和编码。

（2） 在Huffman编码后，要将编码表和英文文章编码结果保存到文件中，编码结果必须是二进制形式，即0 1的信息用比特位表示，不能用字符’0’和’1’表示。

（3） 提供读编码文件生成原文件的功能。

* 1. **采用的数据结构**

采用二叉树数据结构，用链表和数组存储实现之。

* 1. **算法设计思想**

typedef struct HTNode{

int weight;

int parent,lchild,rchild;

}HTNode;

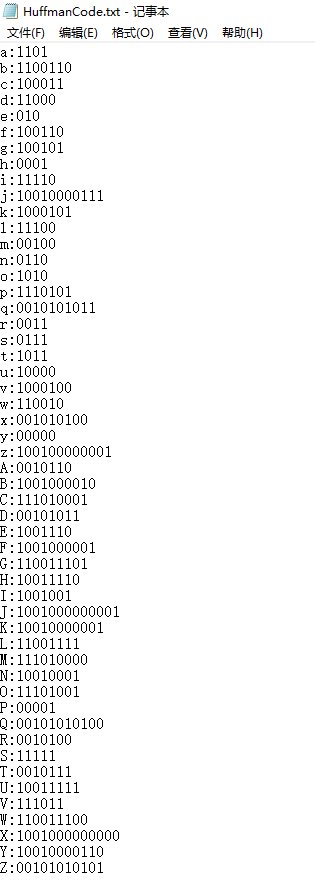
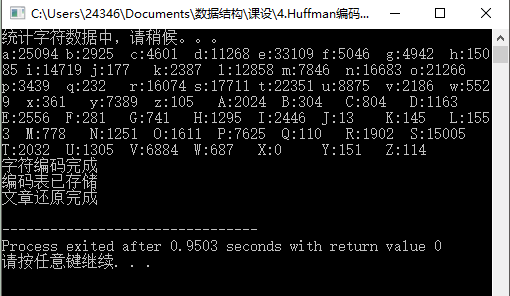
typedef char\*\* HuffmanCode; //动态存储哈夫曼编码表

程序自动读取文本文件信息，统计大小写英文字母出现次数，根据统计结果进行哈夫曼编码，并将编码结果保存在文本文件中，然后按照编码将文章编码为二进制文件，最后根据编码后的文件还原文章到文本文件。

建树原理：在节点中找到父节点为0的两个权值最小的节点，当做新节点的孩子，将新节点添加到所有节点中。

求哈夫曼编码原理：从叶节点到根逆向求编码，编码尾部加‘\0’以便strcpy

* 1. **源程序**
  2. **测试数据和结果**

还原结果：

* 1. **算法的时间复杂度和算法的改进方法**
     + - 1. **主要函数时间复杂度**

void HuffmanCoding(HTNode\* &HT,HuffmanCode &HC,int \*letters,int n){};

函数包含建树和求编码各嵌套两个for循环。

**时间复杂度为O（n\*n） n为节点数**

* + - * 1. **算法改进方法**

算法参照教材哈夫曼编码算法，无需改进。

1. **无线传感器网络**
   1. **题目描述**

5、无线传感器网络(必做) （图）

[问题描述]

在一个直角坐标（设为100\*100）平面中，随机分布n无线传感器结点，汇聚点为原点（0，0），要求将每个无线传感器结点的信息传输到汇聚点，传输可以通过多跳方式实现，设无线传感器结点最大无线通信距离为10，传输能耗与距离平方成正比，传输时间与距离成正比。

问题一：建立每个结点传输到汇聚点的最短时间通道，并找出无法实现传输的结点，将其排除。

问题二：建立整个网络传输到汇聚点的平均能耗最小网络结构。

[基本要求]

输入格式： 输入的第一行包含一个正整数n，表示无线传感器结点数量。结点使用1, 2, 3, ……n依次标号。 接下来n行，每行包含三个整数ni, xi, yi，其中xi, yi表示第ni个结点的坐标，要求从文本文件中输入。

输出格式：

问题一：输出每个结点到汇聚点的最短时间通道，包括最短时间和经过结点。

问题二：输出整个网络到汇聚点的平均能耗最小网络结构，包括平均最小能耗和连接方式。

* 1. **采用的数据结构**

采用图数据结构，用二维数组存储实现之。

* 1. **算法设计思想**

程序随机生成指定数目的传感器数据并存储到dat文件，然后读取文件存储到二维数组中。

寻找各传感器到原点最短时间路径：从原点开始递归调用findmin()函数，该函数依次执行以下操作：修改临时路径记录，比较临时时间记录与当前传感器时间记录值决定是否修改最小时间，比较临时功耗记录与当前传感器功耗记录决定是否修改最小功耗，将当前传感器标记为已访问，递归调用findmin（）函数，恢复临时时间、功耗、路径、访问记录。

* 1. **源程序**
  2. **测试数据和结果**



* 1. **算法的时间复杂度和算法的改进方法**
     + - 1. **主要函数时间复杂度**

void findmin(int n){};

函数包含循环递归调用

**时间复杂度为O（n^n） n为传感器数**

* + - * 1. **算法改进方法**

时间复杂度太大，若相邻传感器数目太多就会花费很长时间。将递归改为非递归会好一些。

1. **排序算法比较**
   1. **题目描述**

6、排序算法比较 （必做）（排序）

[问题描述]

利用随机函数产生10个样本的20000个随机整数（其中之一已经是正序，之一是逆序），利用直接插入排序、希尔排序，冒泡排序、快速排序、选择排序、堆排序，归并排序、基数排序8种排序方法进行排序（结果为由小到大的顺序），并统计每一种排序算法对不同样本所耗费的时间。

[基本要求]

（1） 原始数据存在文件中，用相同样本对不同算法进行测试；

（2） 屏幕显示每种排序算法对不同样本所花的时间；

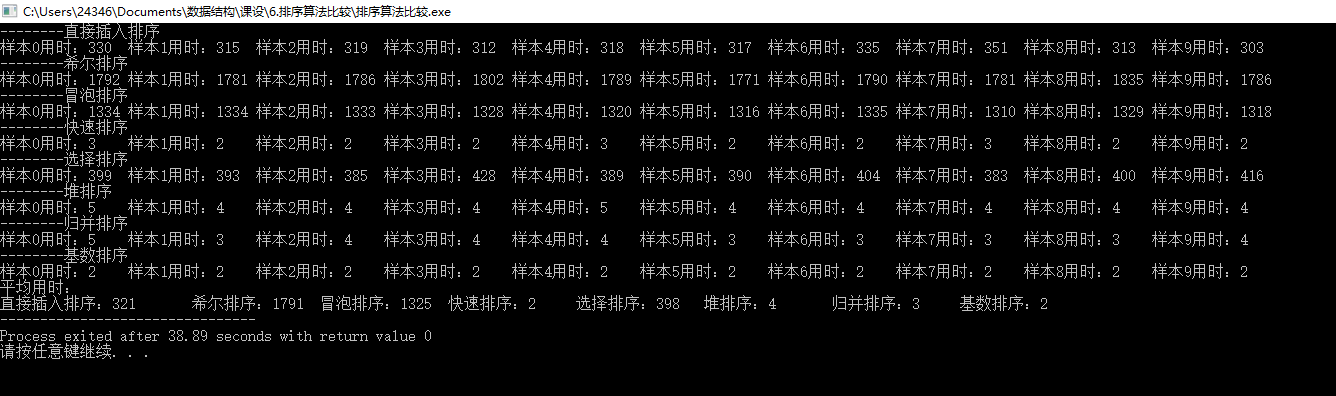
* 1. **采用的数据结构**

采用线性表数据结构，用数组存储实现之。

* 1. **算法设计思想**

程序随机生成10个dat文件，各自存储20000个随机数。然后使用八种排序算法对样本排序，记录每次排序时长（毫秒为单位），最后计算每种排序算法的平均用时。

* 1. **源程序**
  2. **测试数据和结果**



* 1. **算法的时间复杂度和算法的改进方法**
     + - 1. **主要函数时间复杂度**

void InsertionSort(int data[]){};

**时间复杂度为O（n\*n） n为数据个数**

void ShellSort(int\*data,unsigned int len){};

**时间复杂度为O（n^（3/2）） n为数据个数**

void BubbleSort(int data[]){};

**时间复杂度为O（n\*n） n为数据个数**

void QuickSort(int a[], int low, int high){};

**时间复杂度为O（nlogn） n为数据个数**

void SelectSort(int data[]){};

**时间复杂度为O（n\*n） n为数据个数**

void HeapSort(int array[],int length)};

**时间复杂度为O（nlogn） n为数据个数**

void MergeSort(int \*data, int start, int end, int \*result){};

**时间复杂度为O（nlogn） n为数据个数**

void RadixSort(int data[], int n){};

**时间复杂度为O（d（n+rd）） n为数据个数**

* + - * 1. **算法改进方法**

排序算法无需改进。

**结束语：**

这些天每天平均花2-4小时写课设，一天一道题。可以说每道题都满足基本题目要求，并且自己添加了很多错误处理方法，程序健壮性较好，尤其第一题，有了之前课设经验可以检验处理很多错误输入。

当然程序还是有很多改进之处的，例如由于时间关系，没有尝试用Qt做GUI，迷宫问题可以添加一个随机生成迷宫的算法等等。

学完数据结构，经过这次课设，感觉收获颇多，自己对C/C++有了更深的认识，编程能力更进一步，了解到了算法的重要性，为处理大数据打下基础，感觉到C/C++宝刀未老，激起了自己学习Qt的兴趣。

可能学习其他语言不需要掌握数据结构知识，但作为计算机专业学生，学习数据结构有利于我们深入理解数据处理方法、培养重视算法效率意识，例如可以用自定义数据结构进行无限大数之间的运算。数据结构在以后的编程开发中无疑会发挥重要作用。

代码行：

第一题 618行

第二题 174行

第三题 418行

第四题 394行

第五题 329行

第六题 536行

总代码行 2470行