# 课程编号：B080109010

数据结构课程设计

总结报告



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **姓名** | **王斌** | **学号** | | **20144814** |
| **班级** | **1409** | **指导教师** | | **李雪亮** |
| **实验名称** | **数据结构课程设计** | | | |
| **开设学期** | **2016-2017第一学期** | | | |
| **开设时间** | **第10周——第12周** | | | |
| **报告日期** | **2016-11-24** | | | |
| **评定成绩** |  | | **评定人** |  |
| **评定日期** | **2016-11-28** |

**东北大学软件学院**

第一章需求分析

写出问题的定义、分析和研究意义。

1. 问题的定义

实现景区信息管理系统，该系统包含的主要功能有：创建与输出导游路线图、计算与打印导游路线图、寻找两个景点之间的最短路径、根据关键词查询定位具体景点并打印景点相关信息、按照热度，岔路口等特性对各个景点进行排序。

另外还有一个子系统——停车场车辆进出信息记录系统，负责记录和处理停车和候车等信息、计算各车辆停车时长和应付的费用。

1. 问题的分析
2. 用邻接链表存储景点分布图的信息，带权无向图的邻接链表，通过邻接列表完成创建景区景点分布图和输出景区景点分布图。
3. 使用深度遍历的方法对景点进行遍历，并输出景点景区旅游信息管理系统中制订旅游景点导游线路策略。
4. 采用迪杰斯特拉算法，使用一个优先级队列来更新每个节点间的距离。来达到求取景点之间的最短路径。
5. 根据用户输入的关键字进行景点的查找，关键字可以在景点名称也可以在景点介绍中。排序算法可以采用冒泡、快速、插入排序等。
6. 以栈模拟停车场的相关信息，以队列模拟车场外的便道，通过从终端读取的数据来进行停车场的更新。
7. 研究意义、

本次实验涉及到堆栈、队列、邻接表、临街矩阵、树、图以及迪杰斯克拉算法等，通过此次实验，了解并掌握数据结构和算法设计的方法，具备初步的独立分析和设计能力。学会了思考选择哪种数据结构和比较所采用的算法的复杂度和优良性。

第二章系统设计

2.1总体设计

给出设计思想，基本的数据结构，抽象数据类型的定义，主要的操作的定义（或类的定义），主要功能实现方法。

1.数据结构

①景点信息的数据结构

typedef struct Information{

char introduction[200];

int popularity;

int roadnum;

}Information;

②邻接链表数据结构

点信息：

typedef struct ArcNode{

//VertexType name;

int adjvex;

struct ArcNode \*next;

int weight;

}ArcNode;

边信息：

typedef struct Node{

VertexType name;

ArcNode \*first;

Information info;

}Node,AdjList[MAX\_VERTEX\_NUM];

③邻接表

typedef struct ALGraph{

int vexnum;//顶点数vertex

int arcnum;//边数arc

AdjList vertices;

}ALGraph;

④邻接矩阵

typedef struct

{

int arcs[MAX\_VERTEX\_NUM][MAX\_VERTEX\_NUM]; //邻接矩阵

int vexnum, arcnum; //无向网当前顶点数和弧数

VertexType vexs[MAX\_VERTEX\_NUM];//描述顶点的数组

Information info[MAX\_VERTEX\_NUM];//介绍顶点信息的数组

} MGraph;

⑤停车信息

typedef struct zanlind{

int number; //汽车车号

int ar\_time; //汽车到达时间

}zanInode;

⑥停车场的栈结构

typedef struct

{

zanInode car[N];

int top; //栈指针\*/

} SqStack; //定义顺序栈类型，用于描述停车场\*/

⑦停车场的队列结构

typedef struct

{

int CarNo[MM]; /\*车牌号\*/

int front,rear; /\*队首和队尾指针\*/

} SqQueue; /\*定义循环队类型，用于描述候车场\*/

1. 抽象数据类型和主要功能的实现

①void createGraph(ALGraph &G)//创建邻接链表

ArcNode \*p=NULL,\*node=NULL;//顶点指针

②void addInformation (ALGraph &G)//添加顶点信息

③void DFS(ALGraph G, int i)//图的深度遍历函数

ArcNode \*p;//顶点指针

④void DFSTraverse(ALGraph G）//图的深度遍历的输出

⑤void dijkstra(MGraph G,int v0,int vd) //采用Dijkstra算法求从顶点v0到其余各顶点的最短路径

int dist[MAX\_VERTEX\_NUM],path[MAX\_VERTEX\_NUM];//距离与路径

int s[MAX\_VERTEX\_NUM];//已求出最短路径的顶点集合,每求得一条最短路径 , 就将其加入到集合S中

⑥void DisPath(int dist[],int path[],int s[],int n,int v0,int vd,MGraph G)//由dist计算最短路径的方法

⑦void ppth(int path[],int i,int v0,MGraph G)//输出各条最短路径

⑧void searchWord(ALGraph G)//搜索关键字

Node searchnode[MAX\_VERTEX\_NUM];

⑨void hotSort(ALGraph G)//热度排序

void roadSort(ALGraph G)//岔路口排序

⑩顺序栈的基本运算算法和循环队列的基本运算算法

void InitStack(SqStack \*&s)//初始化栈

int StackEmpty(SqStack \*s)//判断栈是否为空

int StackFull(SqStack \*s)//判断栈满

int Push(SqStack \*&s,int e1,int e2)//进栈

int Pop(SqStack \*&s,int &e1,int &e2)//出栈

void DispStack(SqStack \*s)//输出栈中的元素

void InitQueue(SqQueue \*&q)

int QueueEmpty(SqQueue \*q)

int QueueFull(SqQueue \*q)

int enQueue(SqQueue \*&q,int e)

int deQueue(SqQueue \*&q,int &e)

void DispQueue(SqQueue \*q)

⑾void parkinglotSystem()//停车系统

2.2程序设计

①创建邻接列表

通过对邻接表中的每一个节点进行遍历，将每一个节点包含的相邻节点的链表信息放入矩阵中

for(i=0; i<G.arcnum; i++)

{

printf("\n输入第%d边的两个顶点以及该边的权值：",i+1);

scanf("%s %s %d",&name1,&name2,&weight);

int m=0,n=0;

m=locateVex(G,name1);

n=locateVex(G,name2);

p=(ArcNode\*)malloc(sizeof(ArcNode));//产生一个新的链表结点

p->weight=weight;

p->adjvex=n;

//strcpy(p->adjvex,name2);

p->next=G.vertices[m].first;

G.vertices[m].first=p;

②深度遍历的函数

通过递归的方法，每次都从当前节点向相邻节点深入，访问完相邻节点接下去访问下一个相邻节点，直到访问完所有相邻节点然后回退。

void DFS(ALGraph G, int i) {

ArcNode \*p;

visited[i]=1;//true=1访问过

printf("%s ",G.vertices[i].name);

p=G.vertices[i].first;

while(p){

if(visited[p->adjvex]==0){

DFS(G,p->adjvex);//引用不需使用&G

}

p=p->next;

}

}

③迪杰斯克拉算法

给定好一个起始节点，用一个数组去记录其余每个节点与这个节点的最短距离。

void dijkstra(MGraph G,int v0,int vd) //采用Dijkstra算法求从顶点v0到其余各顶点的最短路径

{

int dist[MAX\_VERTEX\_NUM],path[MAX\_VERTEX\_NUM];//距离与路径

int s[MAX\_VERTEX\_NUM];//已求出最短路径的顶点集合,每求得一条最短路径 , 就将其加入到集合S中

int mindis,i,j,u,n=G.vexnum;

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

dist[i]=G.arcs[v0][i];//各顶点到v0的距离设为直接到达距离

s[i]=0;

if(G.arcs[v0][i]<INF)//如果可以直接相连

path[i]=v0;

else

path[i]=-1;

}

s[v0]=1;//先把v0设置为已求出最短距离的顶点

path[v0]=0;//v0到自身的路径为0

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

mindis=INF;//最短距离先设置为最大

u=-1;

for(j=0; j<n; j++)//找出距离当前顶点直接距离最近的顶点

if(s[j]==0&&dist[j]<mindis){

u=j;

mindis=dist[j];

}

s[u]=1;//以u作为当前顶点开始寻找

for(j=0; j<n; j++)

if(s[j]==0)

if(G.arcs[u][j]<INF&&dist[u]+G.arcs[u][j]<dist[j]){

dist[j]=dist[u]+G.arcs[u][j];

path[j]=u;

}

}

printf("\n输出最短路径和最短路径长度为:\n");

DisPath(dist,path,s,n,v0,vd,G);

}

④停车算法

if (!StackFull(St)) /\*停车场不满\*/

{

Push(St,no,time);

printf("恭喜您停车成功！>>您的车在停车场位置为:%d\n",St->top+1);

}

else /\*停车场满\*/

{

if (!QueueFull(Qu)) /\*候车场不满\*/

{

enQueue(Qu,no);

printf("抱歉停车场已满，已停在候车场！>>您的车在候车场位置为:%d\n",Qu->rear);

}

else

printf("抱歉停车场与候车场均满，无法停车或候车！\n");

}

第三章系统实现与调试

对实践任务书中要求的内容分别描述实现及调试过程。重点介绍每部分内容实现过程以及实现过程中遇到的难点和解决方案

3.1 导游线路图的创建

1.实现过程：

创立邻接表。根据实验要求，创建一个存储各个节点信息的结构体，然后每一个节点中会包含一个存储相邻节点的信息的链表。

采用深度遍历的办法，遍历到每个节点的具体信息，通过链表将结构体进行连接。

1. 难点

通过遍历每个节点，进行邻接链表的建立。

1. 解决措施

函数中用迭代的方式进行遍历，并用链表进行连接。

3.2输出导游路线图及其图中的回路部分

3.3输出两个景点之间最短路径和最短距离

1.实现过程：

迪杰斯克拉算法计算最短路径。首先得到一个游客输入的起点，然后根据迪杰斯特拉算法，计算出从该顶点到其他所有顶点的最短路径。通过计算出起点与其他相邻节点的当前最短距离，对节点进行整体的遍历。最后找到最短的路径。

1. 难点

最短路径寻找和输出

1. 解决措施

通过迪杰斯克拉算法遍历每一个节点，并找出最短的路径，通过dist函数记录并输出最短路径上面的节点和最短路径。

3.4输出道路修建规划图

3.5查找及排序

查找

1.实验过程：

通过游客输入的关键字，遍历各个景点的名字及简介，然后使用c语言中的字符串函数进行匹配，找到包含关键字的景点，将包含这些关键字的所有景点的景点名称以及景点简介全部放在一个字符串，然后打印输出。

1. 难点

字符串和节点信息的比较

1. 解决措施

通过自带的字符串函数进行匹配对比，找到合适的信息字符串进行输出。

排序：

1. 实现过程

冒泡排序（欢迎度）：通过比较相邻的元素，根据比较的结果就进行交换，对每一对相邻的节点做相同的操作，不断重复上面的步骤。

快速排序（岔路数）：通过对数据的大小进行比较，将要排序的数据分割成独立的两部分，然后再按此方法对这两部分数据分别进行快速排序，然后通过递归不断替换中间元素直到排序完成。

1. 难点

如何进行排序

1. 解决方法

冒泡排序，快速排序

3.6输出车辆的进出信息

1. 实现过程：

首先按照要求，创建一个数组栈，一个链表队列，然后实现进场出场的逻辑，当车辆要进场时，判断停车场这个栈是否是满的，如果是满的，便放入队列中，如果队列中没有车，再判断停车场是否满了，如果没满就进行压栈操作，即进入停车场，如果满了就进入队列；如果车辆要出场时，我先让这辆车前面的车按顺序压入临时栈中，然后这辆车出场后，先对其进行收费，再将临时栈中的车出栈，回到停车场栈，最后再判断便道上是否有车，如果有车则进行压栈，进入停车场。

1. 难点、

栈和队列的实现

1. 解决方式

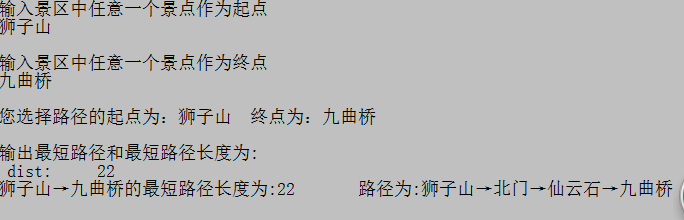
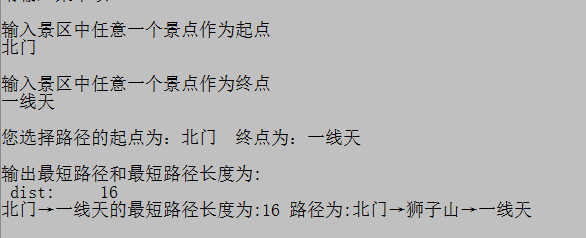
进行栈和队列的判断，是否为空和为满。压栈和出栈，进行栈顶和栈底的++ 和--操作。

第四章系统测试

测试方法：输出两点间的最短距离

测试用例（应该给出几组具有不同特征的数据进行测试）：北门到狮子山，狮子山到九曲桥。

测试结果：



出现的问题及解决办法：

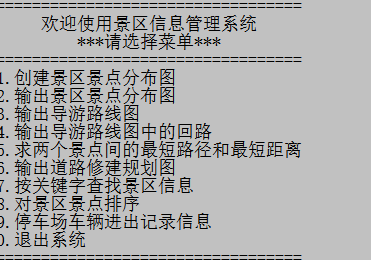
问题：路径遍历错误，节点的连接比较乱

解决方法：重新修改遍历的代码，用一个变量去记录遍历之后的节点和距离。通过一个新的函数，去比较遍历之后的节点和距离。反复操作，变量记录好已经遍历过的节点。

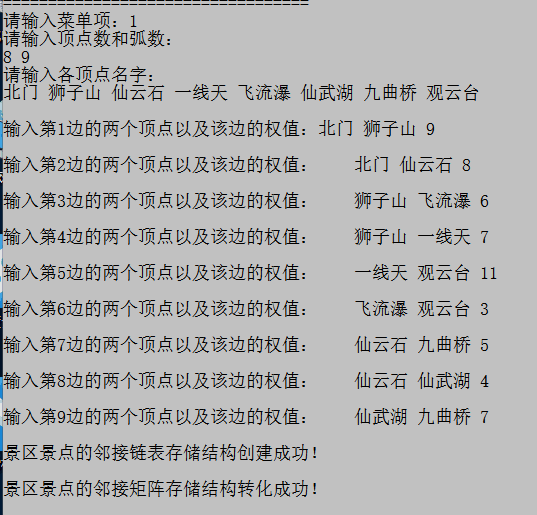
第五章结论

程序的最终实现结果。效果图截图

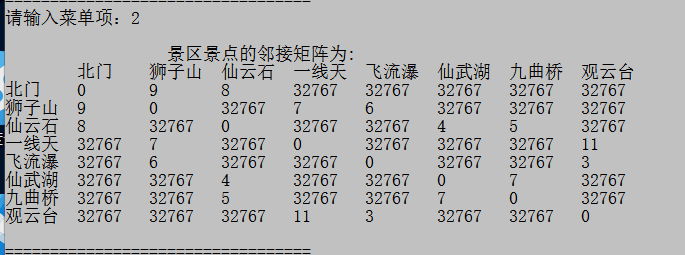
1. 首页菜单栏



1. 创建景区景点分布图



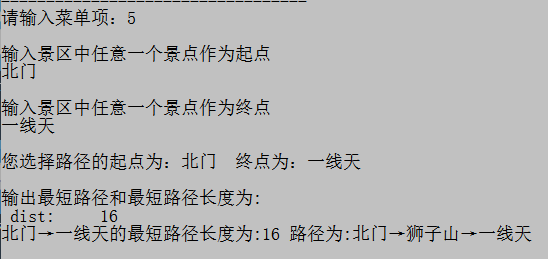
1. 输出景区景点分布图



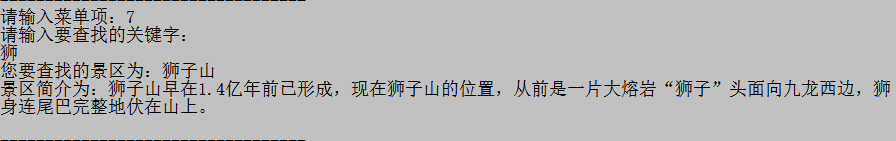
1. 输出导游路线图

QQ图片20161128233247

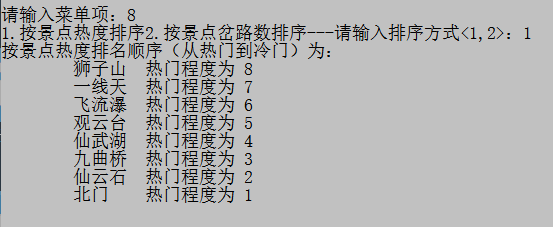
1. 求两个景点之间的最短距离和最短路径

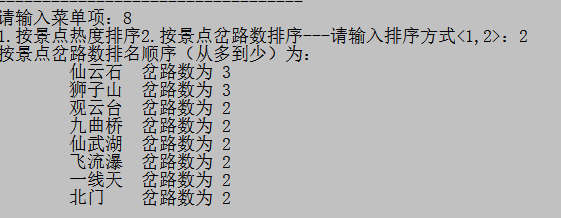
、

1. 按关键字查找景区信息

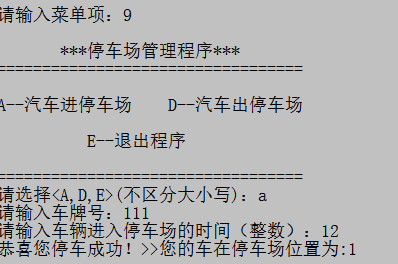
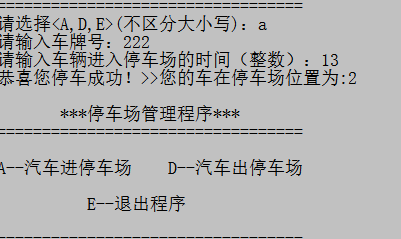


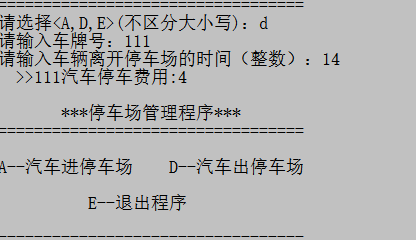
1. 对景区景点进行排序





1. 停车场停车进出信息记录





概括地写出系统实现的功能，主要创新点（可选功能或者自主加入的功能），遇到的困难，系统遗留的问题及可能解决的途径。

主要创新点：

1. 将迪杰斯克拉算法进行拆分，将通过遍历找出来的节点进行存储，将最短路径也进行存储，最后从存储的数组中进行取值并进行运算。  
    2.将各种函数和算法已经封装成单独的文件，可以通过include导入文件。进行文件之中的相互调用。

主要的不足：

1. 创立的文件太多，在互相调用和导入的过程中，产生了混乱，条例和逻辑比较混乱。
2. 迪杰斯克拉算法书写的比较繁琐，有很多的冗余。
3. 没有去深入探索回路功能的具体实现，导致在我的程序之中，没有去实现这部分的功能。

参考文献

例：

[1] 数据结构、算法与应用：C++语言描述 [Data Structures,Algorithms,and Applications in C++][M].[机械工业出版社](http://book.jd.com/publish/%E6%9C%BA%E6%A2%B0%E5%B7%A5%E4%B8%9A%E5%87%BA%E7%89%88%E7%A4%BE_1.html" \t "_blank" \o "机械工业出版社)出版时间：2000-01-01.

[2] 数据结构(C语言版) [M].北京: 中国铁道出版社, 2011-08-01.

**附录：**

**《数据结构课程设计》实验成绩评定表**

****

|  |  |  |  |
| --- | --- | --- | --- |
| **评价内容** | **具 体 要 求** | **分值** | **得分** |
|
| **平时表现** | **课程设计过程中，无缺勤、迟到、早退现象，学习态度积极。课设过程未做与课设无关的事情。** | **10** |  |
| **报告质量** | **实验报告格式规范，体例符合要求；报告内容充实、正确，实验目的归纳合理到位，思考题回答准确。** | **40** |  |
| **创新性** | **在实践内容实现上，有自己独立的想法，并在功能实现上有一定的创新性。** | **10** |  |
| **实验内容** | **能够按实验要求合理设计并开发出程序，认真记录实验数据，原理及实验结果分析准确，归纳总结充分。程序代码书写规范，简洁。代码块区别显著，有重要位置的注释内容。** | **40** |  |
| **总 分** | | |  |