**郑 州 轻 工 业 学 院**

**实 验 报 告**

**课程名称：** 数据结构

**实验名称：** 图及其应用

**院 （系）：** 计算机与通信工程学院

**姓 名：** 刘亚炜

**学 号：** 541707090128

**专业班级：** 物联网工程17-1

**指导教师：** 孟颍辉

2019 **年** 6 **月** 5 **日**

**实验报告成绩评定表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 评定项目 | 内 容 | 满 分 | 评 分 | 总 分 |
| 实验态度 | 实验认真，态度端正，遵守纪律，出勤情况。 | 20 |  |  |
| 实验过程 | 按实验要求完成各种功能或操作，代码书写规范，注释清晰，设计严谨，运行结果正确。 | 40 |  |
| 报告撰写 | 报告字迹整洁、内容丰富、条理清楚；图、表、文字表达准确规范，上交及时。 | 40 |  |
| 总成绩 | 采用五级分制：优、良、中、及格、不及格 |  | | |
| **评语：**  **指导老师签字： 年 月 日** | | | | |

实验报告正文

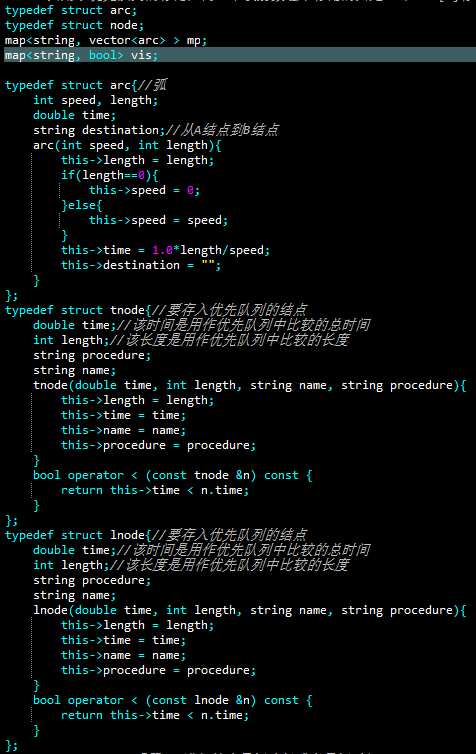
1. **实验目的**

掌握图的存储、构建、搜索等操作和应用，能用最短路径及其搜索等算法编制较综合性的程序，求解最优路线问题，进行程序设计、数据结构和算法设计等方面的综合训练。

1. **实验内容及要求**
2. 任务描述

设计一个城市交通咨询模拟系统，利用该系统实现至少两种最优决策：最短路程到达、最省时到达等线路规划。

1. 主要数据类型与变量



1. 算法或程序模块





1. **测试**
   1. 方案

两种结构体数据类型，一种是顶点，另一种是弧

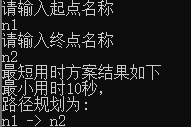
采用map的映射模式来创建数组的第一维

结点连通状态采用vector进行弧的存储

使用优先队列对其进行bfs规划，其中node只是优先队列中的结点

node只用于优先队列的存储，而二维映射数组中存储的只是arc，vec[0]存储的是arc类型，但是它代表的是一个结点

* 1. 结果



1. **总结与讨论**

改进版bfs,使用优先队列作为载体，寻找最短路径时就先找到最短的length元素先出队，寻找最短用时时则先让最短用时的元素出队，另外使用了动态数组，节省空间，bfs节省时间

**附：程序的源代码**

**#include <bits/stdc++.h>**

**using namespace std;**

**//两种结构体数据类型，一种是顶点，另一种是弧**

**//采用map的映射模式来创建数组的第一维**

**//结点连通状态采用vector进行弧的存储**

**//使用优先队列对其进行bfs规划，其中node只是优先队列中的结点**

**//node只用于优先队列的存储，而二维映射数组中存储的只是arc，vec[0]存储的是arc类型，但是它代表的是一个结点**

**typedef struct arc;**

**typedef struct node;**

**map<string, vector<arc> > mp;**

**map<string, bool> vis;**

**typedef struct arc{//弧**

**int speed, length;**

**double time;**

**string destination;//从A结点到B结点**

**arc(int speed, int length){**

**this->length = length;**

**if(length==0){**

**this->speed = 0;**

**}else{**

**this->speed = speed;**

**}**

**this->time = 1.0\*length/speed;**

**this->destination = "";**

**}**

**};**

**typedef struct tnode{//要存入优先队列的结点**

**double time;//该时间是用作优先队列中比较的总时间**

**int length;//该长度是用作优先队列中比较的长度**

**string procedure;**

**string name;**

**tnode(double time, int length, string name, string procedure){**

**this->length = length;**

**this->time = time;**

**this->name = name;**

**this->procedure = procedure;**

**}**

**bool operator < (const tnode &n) const {**

**return this->time > n.time;**

**}**

**};**

**typedef struct lnode{//要存入优先队列的结点**

**double time;//该时间是用作优先队列中比较的总时间**

**int length;//该长度是用作优先队列中比较的长度**

**string procedure;**

**string name;**

**lnode(double time, int length, string name, string procedure){**

**this->length = length;**

**this->time = time;**

**this->name = name;**

**this->procedure = procedure;**

**}**

**bool operator < (const lnode &n) const {**

**return this->length > n.length;**

**}**

**};**

**void bfs\_time(){//采用bfs进行搜索最短路径或者最短时间**

**cout<<"请输入起点名称\n";**

**string begin;**

**cin >> begin;**

**if(!mp.count(begin)){**

**cout<<"没有这个结点，即将退出\n";**

**system("pause");**

**system("cls");**

**return;**

**}**

**cout<<"请输入终点名称\n";**

**string end;**

**cin >> end;**

**if(!mp.count(end)){**

**cout<<"没有这个结点，即将退出\n";**

**system("pause");**

**system("cls");**

**return;**

**}**

**priority\_queue<tnode> que;**

**que.push(tnode(0, mp[begin][0].length, begin, begin));**

**while(!que.empty()){**

**tnode now = que.top();**

**que.pop();**

**if(now.name == end){**

**cout<<"最短用时方案结果如下\n";**

**cout<<"最小用时"<<now.time<<"秒，\n路径规划为:\n"<<now.procedure<<"\n";**

**vis.clear();**

**return;**

**}**

**vis[now.name] = true;**

**for(vector<arc>::iterator it = mp[now.name].begin(); it != mp[now.name].end(); it++) {**

**if(!vis[it->destination]){**

**que.push(tnode(it->time+now.time, it->length+now.length, it->destination, now.name+" -> "+it->destination));**

**}**

**}**

**}**

**cout<<"没有找到合适的路径\n";**

**return;**

**}**

**void bfs\_length(){**

**cout<<"请输入起点名称\n";**

**string begin;**

**cin >> begin;**

**if(!mp.count(begin)){**

**cout<<"没有这个结点，即将退出\n";**

**system("pause");**

**system("cls");**

**return;**

**}**

**cout<<"请输入终点名称\n";**

**string end;**

**cin >> end;**

**if(!mp.count(end)){**

**cout<<"没有这个结点，即将退出\n";**

**system("pause");**

**system("cls");**

**return;**

**}**

**priority\_queue<lnode> que;**

**que.push(lnode(0, mp[begin][0].length, begin, begin));**

**while(!que.empty()){**

**lnode now = que.top();**

**que.pop();**

**if(now.name == end){**

**cout<<"最短用时方案结果如下\n";**

**cout<<"最小用时"<<now.time<<"秒，路径规划为"<<now.procedure<<"\n";**

**vis.clear();**

**return;**

**}**

**vis[now.name] = true;**

**for(vector<arc>::iterator it = mp[now.name].begin(); it != mp[now.name].end(); it++) {**

**if(!vis[it->destination]){**

**que.push(lnode(it->time+now.time, it->length+now.length, it->destination, now.name+" -> "+it->destination));**

**}**

**}**

**}**

**cout<<"没有找到合适的路径\n";**

**return;**

**}**

**void menu(){**

**cout<<"实验三 图及其应用\n";**

**cout<<"1.创建结点和弧\n";**

**cout<<"2.输入起点和终点帮你进行最短时间规划\n";**

**cout<<"3.输入起点和终点帮你进行最短路径规划\n";**

**cout<<"4.删除弧(修路等添加事件)\n";**

**cout<<"5.删除结点\n";**

**cout<<"0.退出本系统\n";**

**}**

**void creat\_node\_and\_arc(){**

**cout<<"请输入你想创建结点的数量\n";**

**int num = 0;**

**cin >> num;**

**for(int i = 0; i < num; i++) {**

**if(i==0){**

**cout<<"请输入你想要创建的结点名称\n";**

**}**

**string str;**

**cin >> str;**

**vector<arc> vec;**

**arc a = arc(0, 0);**

**vec.push\_back(a);**

**mp[str] = vec;**

**}**

**cout<<"请输入你想创建的弧的数量\n";**

**num = 0;**

**cin >> num;**

**for(int i = 0; i < num; i++) {**

**cout<<"请输入第"<<i+1<<"组数据\n";**

**cout<<"这条弧的起点是谁？（输入一个字符串）\n";**

**string str1;**

**cin >> str1;**

**if(!mp.count(str1)){**

**cout<<"没有这个结点，请重新输入\n";**

**system("pause");**

**system("cls");**

**i--;**

**continue;**

**}**

**cout<<"这条弧的终点是谁？（输入一个字符串）\n";**

**string str2;**

**cin >> str2;**

**if(!mp.count(str2)){**

**cout<<"没有这个结点，请重新输入\n";**

**system("pause");**

**system("cls");**

**i--;**

**continue;**

**}**

**cout<<"这条弧的长度和通行速度（请按照 长度-通行速度 方式输入）\n";**

**int length, speed;**

**cin >> length >> speed;**

**cout<<"这条弧是单向的还是双向的？（单向输入1，双向输入2）\n";**

**int type;**

**cin >> type;**

**arc a = arc(speed, length);**

**a.destination = str2;**

**mp[str1].push\_back(a);**

**if(type==2){**

**a.destination = str1;**

**mp[str2].push\_back(a);**

**}**

**}**

**cout<<"创建成功\n";**

**}**

**void delete\_node(){**

**cout<<"输入你想删除的结点名称：\n";**

**string ans;**

**cin >> ans;**

**if(mp.count(ans)){**

**mp[ans].clear();**

**mp.erase(ans);**

**cout<<"删除成功\n";**

**}else{**

**cout<<"没有这个结点\n";**

**}**

**}**

**void delete\_arc(){**

**cout<<"输入你想删除的弧\n";**

**cout<<"这个弧的起点是谁？\n";**

**string begin;**

**cin >> begin;**

**cout<<"这个弧的终点是谁？\n";**

**string end;**

**cin >> end;**

**cout<<"你想删除这个弧的单向还是双向？单向输入1，双向输入2\n";**

**int type;**

**cin >> type;**

**bool flag = 0;**

**for(vector<arc>::iterator it = mp[begin].begin(); it != mp[begin].end(); it++) {**

**if(it->destination == end){**

**mp[begin].erase(it);**

**flag = 1;**

**break;**

**}**

**}**

**if(type==2){**

**for(vector<arc>::iterator it = mp[end].begin(); it != mp[end].end(); it++) {**

**if(it->destination == begin){**

**mp[end].erase(it);**

**flag = 1;**

**break;**

**}**

**}**

**}**

**if(flag){**

**cout<<"操作成功\n";**

**}else{**

**cout<<"操作失败\n";**

**}**

**}**

**int main(){**

**while(1){**

**menu();**

**int choose;**

**cin >> choose;**

**switch(choose){**

**case 1:**

**creat\_node\_and\_arc();**

**break;**

**case 2:**

**bfs\_time();**

**break;**

**case 3:**

**bfs\_time();**

**break;**

**case 4:**

**delete\_arc();**

**break;**

**case 5:**

**delete\_node();**

**break;**

**case 0:**

**return 0;**

**default:**

**system("cls");**

**cout<<"输入有误请重新输入"<<endl;**

**system("pause");**

**system("cls");**

**}**

**}**

**}**