《数据结构与算法》实验报告

<table>
<tr><td>实验名称</td><td colspan="5">校园导游咨询</td></tr>
<tr><td>姓名</td><td>陈思睿</td><td>学号</td><td>21030021007</td><td>日期</td><td>6.10</td></tr>
<tr><td>实验内容</td><td colspan="5">设计一个校园导游程序，为来访客人提供各种信息查询服务。测试数据根据实际情况指定。提示：一般情况下，校园的道路是双向通行的，可设校园平面图是一个无向图。顶点和边均含有相关信息。
1、设计西海岸校园平面图，所含 Point of Interest(PoI)不少于 10 个。以图中顶点表示校内各 PoI，存放 PoI 名称、代号、简介等信息；以边表示路径，存放路径长度等相关信息。（4 分）
2、为来访客人提供图中任意 PoI 相关信息的查询。（2 分）
3、为来访客人提供图中任意 PoI 的纹路查询，即查询任意两个 PoI 之间的一条最短的简单路径。（4 分）</td></tr>
<tr><td>实验目的</td><td colspan="5">掌握图的存储方法和最短路经算法。</td></tr>
</table>

实验步骤

1、设计西海岸校园平面图

①定义基本数据结构：景点 Viewpoint 和图 Graph。其中 Viewpoint 包含景点名称、序号、简介。图包含整个校园的所有景点、景点之间存在的路径以及权值，通过邻接矩阵来存储。

```
typedef struct {
    int num;       // 景点序号
    char *name;    // 景点名称
    char *blurb;   // 景点简介
} Viewpoint;
```

```
typedef struct {
    int numEdges, numVertices; // 顶点数和边数
    int Edge[14][14];          // 邻接矩阵
    Viewpoint point[14];       // 景点信息数组
} Graph;
```

②单个景点的创建

简介、名称都通过字符串存储。

```
// 创建景点
Viewpoint create_point(int num1, const char *name, const char *blurb) {
    Viewpoint v;
    v.num = num1;
    v.name = strdup(name);
    v.blurb = strdup(blurb);
    return v;
}
```

③西海岸校园景点信息的建立

```
// 创建图
void create(Graph* G) {
    G->numVertices = 14;
    G->numEdges = 18;
    G->point[0] = create_point(0, "北门", "很少有人走的地方");
    G->point[1] = create_point(1, "听海苑", "信息学院宿舍，独立卫浴");
    G->point[2] = create_point(2, "国旗处", "五星红旗迎风飘扬");
    G->point[3] = create_point(3, "东操场", "打篮球、跑步不错的地方");
    G->point[4] = create_point(4, "综合体", "图书馆、学习的去处，山字形大楼");
    G->point[5] = create_point(5, "信院南", "本专业的楼");
    G->point[6] = create_point(6, "信院北", "电子信息专业的楼");
    G->point[7] = create_point(7, "听海餐厅", "已经吃腻了的食堂");
    G->point[8] = create_point(8, "东门", "出门的地方、大部分走的地方、连接望海")
    G->point[9] = create_point(9, "正门", "没开放的门、还在修啊");
    G->point[10] = create_point(10, "西操场", "上体育课的地方，能再修大一点吗？")
    G->point[11] = create_point(11, "望海餐厅", "西区同学的吃饭处");
    G->point[12] = create_point(12, "工程楼", "工程学院的专业楼");
    G->point[13] = create_point(13, "望海苑", "工程学院的学生住所");
```

通过调用 creat_point()函数来构建每个景点的详细信息。（包含序号，景点名称和信息）

④初始化邻接矩阵：除了对角线其他都是-1。

```
// 初始化邻接矩阵
for (int i = 0; i < 14; i++) {
    for (int j = 0; j < 14; j++) {
        if (i == j) {
            G->Edge[i][j] = 0;
        } else {
            G->Edge[i][j] = -1;
        }
    }
}
```

邻接矩阵的创建：

```
G->Edge[0][1] = 8;
G->Edge[1][2] = 15;
G->Edge[1][7] = 7;
G->Edge[2][4] = 6;
G->Edge[3][4] = 3;
G->Edge[3][6] = 1;
G->Edge[3][7] = 3;
G->Edge[4][5] = 8;
G->Edge[5][6] = 10;
G->Edge[5][9] = 1;
G->Edge[6][8] = 12;
G->Edge[7][8] = 7;
G->Edge[8][10] = 8;
G->Edge[10][11] = 4;
G->Edge[10][12] = 18;
G->Edge[11][12] = 7;
G->Edge[11][13] = 3;
G->Edge[12][13] = 15;
```

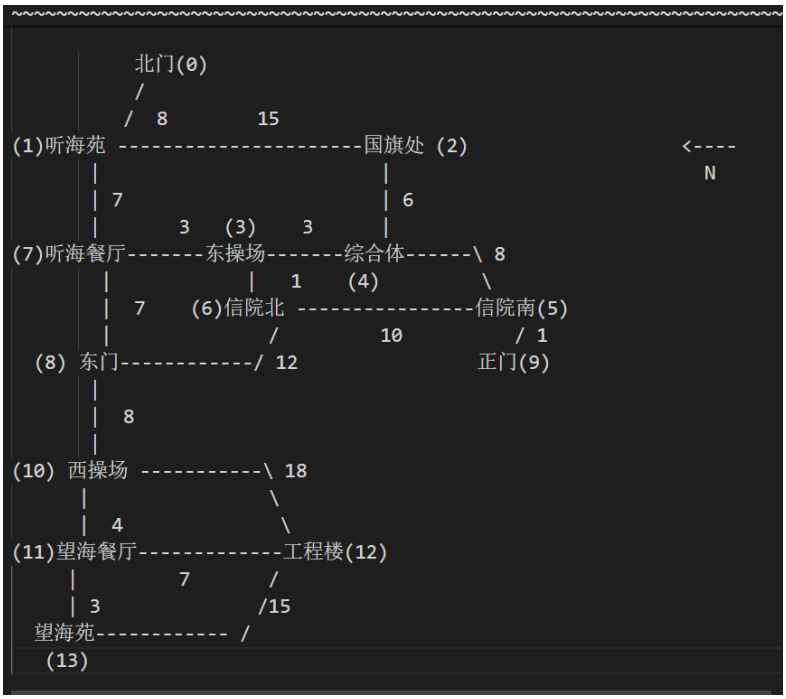
利用邻接矩阵的对称性（无向带权图）

```
for (int i = 1; i < 14; i++) {
    for (int j = i + 1; j < 14; j++) {
        G->Edge[j][i] = G->Edge[i][j];
    }
}
```

2、为来访客人提供图中任意 PoI 相关信息的查询。

①西海岸平面图

在 map.txt 文件中构建一个西海岸地图，包含各个景点以及不同景点之间的带权路径。（共计 14 个景点和 18 条路径）用户可以通过选项查询地图。



②读取地图并打印的函数 print_map()

```c
// 打印地图
void print_map(const char *filename) {
    FILE *file = fopen(filename, "r");
    if (file == NULL) {
        perror("Failed to open file");
        exit(EXIT_FAILURE);
    }
    char map[MAX_ROWS][MAX_COLS];
    int row = 0;
    while (fgets(map[row], MAX_COLS, file)) {
        row++;
    }
    fclose(file);
    for (int i = 0; i < row; i++) {
        printf("%s", map[i]);
    }
}
```

③任意 PoI 相关信息的查询。

```c
// 查询景点信息
void viewpoint_information(Graph* g) {
    printf("请选择你想要查询的景点(选择景点前的选项)\n");
    printf(" 0,北门      1,听海苑      2,国旗处     3,东操场   4,综合体\n");
    printf(" 5,信院南    6,信院北      7,听海餐厅   8,东门     9,正门\n");
    printf(" 10,西操场   11,望海餐厅   12,工程楼    13,望海苑\n");
    int choice;
    scanf("%d", &choice);
    printf("%s\n", g->point[choice].name);//输出名称
    printf("%s\n", g->point[choice].blurb);//展示相关信息
}
```

3、提供图中任意 PoI 的纹路查询

最短路径采用迪杰斯特拉算法，通过提示用户输出两个景点的序号，根据邻接矩阵来求最短路径

初始化

```
// Dijkstra算法求最短路径
void dijkstra(Graph* G) {
    int n = G->numVertices;
    int start, end;
    printf("请选择你的起点和终点(选择景点前的选项)\n");
    printf(" 0,北门      1,听海苑      2,国旗处      3,东操场   4,综合体\n");
    printf(" 5,信院南    6,信院北      7,听海餐厅    8,东门     9,正门\n");
    printf(" 10,西操场   11,望海餐厅   12,工程楼     13,望海苑\n");
    scanf("%d %d", &start, &end);
    int distance[MAX], previous[MAX], visited[MAX];
    int i, count, next_node, min_distance;

    // 初始化
    for (i = 0; i < n; i++) {
        distance[i] = INF;
        previous[i] = -1;
        visited[i] = 0;
    }
    distance[start] = 0;
```

根据初始起点、不断更新相应的最短路径，同时更新相邻结点的距离

```
for (count = 0; count < n - 1; count++) {
    min_distance = INF;

    // 找到距离最小的未访问节点
    for (i = 0; i < n; i++)
        if (!visited[i] && distance[i] < min_distance) {
            min_distance = distance[i];
            next_node = i;
        }

    // 标记为已访问
    visited[next_node] = 1;

    // 更新相邻节点的距离
    for (i = 0; i < n; i++)
        if (!visited[i] && G->Edge[next_node][i] != -1 && distance[next_node] != INF
            && distance[next_node] + G->Edge[next_node][i] < distance[i]) {
            distance[i] = distance[next_node] + G->Edge[next_node][i];
            previous[i] = next_node;
        }
}
```

根据相应的前驱结点构造相应的两个结点之间的最短路径 path 并输出。（如果存在最短的路径）

```c
// 输出路径
int path[MAX];
int path_index = 0;
int current = end;
while (current != -1) {
    path[path_index++] = current;
    current = previous[current];
}

if (distance[end] == INF) {
    printf("没有路径从 %d 到 %d\n", start, end);
    return;
}

printf("最短路径从 %s 到 %s: ", G->point[start].name, G->point[end].name);
for (i = path_index - 1; i >= 0; i--) {
    printf("%s", G->point[path[i]].name);
    if (i > 0) printf(" -> ");
}
printf("\n");
printf("总距离: %d(单位:10m)\n", distance[end]);
```

4、主函数（菜单的构建）

提示用户进行选择，该程序有三个功能、分别是查看平面地图、查询任意景点详细信息以及查询任意两个景点之间的最短路径

```c
const char *filename = "D:\\code\\c\\formal_test6\\map.txt";
Graph *g = (Graph *)malloc(sizeof(Graph));
create(g);
printf("*****************************************************\n");
printf("欢迎来到中国海洋大学景点查询功能\n");
printf("请选择功能\n");
printf("1.查看校园平面图\n");
printf("2.查询任意景点信息\n");
printf("3.查询任意景点之间路径\n");
printf("其他任意键退出\n");
printf("*****************************************************\n");
```

采用 switch 语句实现循环选择、通过用户输入不同的选择调用不同的函数，同时用户可以持续输出，直到选择退出为止，而不是只能执行一次程序就退出。

```
int choice;
scanf("%d", &choice);
while (choice == 1 || choice == 2 || choice == 3) {
    switch (choice) {
        case 1:
            print_map(filename);
            break;
        case 2:
            viewpoint_information(g);
            break;
        case 3:
            dijkstra(g);
            break;
        default:
            break;
    }
    printf("请选择功能\n");
    printf("1.查看校园平面图\n");
    printf("2.查询任意景点信息\n");
    printf("3.查询任意景点之间路径\n");
    printf("其他任意键退出\n");
    printf("*************************************************\n");
    scanf("%d", &choice);
}
```

5、实验结果

1、菜单功能的完整

展示完整的选项

```
*************************************************
欢迎来到中国海洋大学景点查询功能
请选择功能
1.查看校园平面图
2.查询任意景点信息
3.查询任意景点之间路径
其他任意键退出
*************************************************
```

2、查看校园平面图

```
*****************************************************
欢迎来到中国海洋大学景点查询功能
请选择功能
1.查看校园平面图
2.查询任意景点信息
3.查询任意景点之间路径
其他任意键退出
*****************************************************
1
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~ sea~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


  (1)听海苑 ----------------------国旗处 (2)                   <---
  -
        |                         |                           N


        | 7                       | 6
        |       3    (3)    3     |
  (7)听海餐厅-------东操场-------综合体------\ 8
         |            |   1     (4)        \
         |  7    (6)信院北  ----------------信院南(5)
         |              /        10          / 1
    (8) 东门------------/ 12               正门(9)


        |
        |  8
        |
  (10) 西操场 -----------\ 18
       |                 \
       |  4               \
  (11)望海餐厅-------------工程楼(12)
       |        7        /
       | 3              /15
    望海苑------------ /
     (13)请选择功能
  1.查看校园平面图
  2.查询任意景点信息
  3.查询任意景点之间路径
  其他任意键退出
  *****************************************************
```

选项执行正确
地图打印完整

2、任意景点的查询
输入选项 2、菜单先提示你进行选择、通过选择不同景点前的序号，输出该景点的介绍

```
2
请选择你想要查询的景点(选择景点前的选项)
 0,北门      1,听海苑     2,国旗处     3,东操场   4,综合体
 5,信院南    6,信院北     7,听海餐厅   8,东门     9,正门
 10,西操场   11,望海餐厅  12,工程楼    13,望海苑
4
综合体
图书馆、学习的去处，山字形大楼
```

3、任意领个景点之间的路径查询
例如：
①北门到望海苑：北门 -> 听海苑 -> 听海餐厅 -> 东门 -> 西操场 -> 望海餐厅 -> 望海苑

同时输出总距离。

```
3
请选择你的起点和终点(选择景点前的选项)
 0,北门      1,听海苑      2,国旗处     3,东操场   4,综合体
 5,信院南    6,信院北      7,听海餐厅   8,东门     9,正门
 10,西操场   11,望海餐厅   12,工程楼    13,望海苑
0 13
最短路径从 北门 到 望海苑: 北门 -> 听海苑 -> 听海餐厅 -> 东门 ->
西操场 -> 望海餐厅 -> 望海苑
总距离: 37(单位:10m)
```

②听海苑 到 信院南: 听海苑 -> 听海餐厅 -> 东操场 -> 信院北 -> 信院南

```
3
请选择你的起点和终点(选择景点前的选项)
 0,北门      1,听海苑      2,国旗处     3,东操场   4,综合体
 5,信院南    6,信院北      7,听海餐厅   8,东门     9,正门
 10,西操场   11,望海餐厅   12,工程楼    13,望海苑
1 5
最短路径从 听海苑 到 信院南: 听海苑 -> 听海餐厅 -> 东操场 -> 信院
北 -> 信院南
总距离: 21(单位:10m)
```

4、退出功能

除了 1、2、3 选项其他任意键都会退出程序

```
请选择功能
1.查看校园平面图
2.查询任意景点信息
3.查询任意景点之间路径
其他任意键退出
**************************************************
4
PS D:\code>
```

6、源代码：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <limits.h>

#define MAX_ROWS 100
#define MAX_COLS 200
#define INF INT_MAX
#define MAX 200

typedef struct {
int num;      // 景点序号
```

```
char *name;   // 景点名称
char *blurb;  // 景点简介
} Viewpoint;

typedef struct {
int numEdges, numVertices; // 顶点数和边数
int Edge[14][14];          // 邻接矩阵
Viewpoint point[14];       // 景点信息数组
} Graph;

// 创建景点
Viewpoint create_point(int num1, const char *name, const char *blurb) {
Viewpoint v;
v.num = num1;
v.name = strdup(name);
v.blurb = strdup(blurb);
return v;
}

// 创建图
void create(Graph* G) {
G->numVertices = 14;
G->numEdges = 18;
G->point[0] = create_point(0, "北门", "很少有人走的地方");
G->point[1] = create_point(1, "听海苑", "信息学院宿舍，独立卫浴");
G->point[2] = create_point(2, "国旗处", "五星红旗迎风飘扬");
G->point[3] = create_point(3, "东操场", "打篮球、跑步不错的地方");
G->point[4] = create_point(4, "综合体", "图书馆、学习的去处，山字形大楼");
G->point[5] = create_point(5, "信院南", "本专业的楼");
G->point[6] = create_point(6, "信院北", "电子信息专业的楼");
G->point[7] = create_point(7, "听海餐厅", "已经吃腻了的食堂");
G->point[8] = create_point(8, "东门", "出门的地方、大部分走的地方、连接望海");
G->point[9] = create_point(9, "正门", "没开放的门、还在修啊");
G->point[10] = create_point(10, "西操场", "上体育课的地方，能再修大一点吗？");
G->point[11] = create_point(11, "望海餐厅", "西区同学的吃饭处");
G->point[12] = create_point(12, "工程楼", "工程学院的专业楼");
G->point[13] = create_point(13, "望海苑", "工程学院的学生住所");

// 初始化邻接矩阵
for (int i = 0; i < 14; i++) {
for (int j = 0; j < 14; j++) {
if (i == j) {
G->Edge[i][j] = 0;
} else {
```

```
G->Edge[i][j] = -1;
}
}
}

G->Edge[0][1] = 8;
G->Edge[1][2] = 15;
G->Edge[1][7] = 7;
G->Edge[2][4] = 6;
G->Edge[3][4] = 3;
G->Edge[3][6] = 1;
G->Edge[3][7] = 3;
G->Edge[4][5] = 8;
G->Edge[5][6] = 10;
G->Edge[5][9] = 1;
G->Edge[6][8] = 12;
G->Edge[7][8] = 7;
G->Edge[8][10] = 8;
G->Edge[10][11] = 4;
G->Edge[10][12] = 18;
G->Edge[11][12] = 7;
G->Edge[11][13] = 3;
G->Edge[12][13] = 15;

for (int i = 1; i < 14; i++) {
for (int j = i + 1; j < 14; j++) {
G->Edge[j][i] = G->Edge[i][j];
}
}
}

// 打印地图
void print_map(const char *filename) {
FILE *file = fopen(filename, "r");
if (file == NULL) {
perror("Failed to open file");
exit(EXIT_FAILURE);
}
char map[MAX_ROWS][MAX_COLS];
int row = 0;
while (fgets(map[row], MAX_COLS, file)) {
row++;
}
fclose(file);
```

```
for (int i = 0; i < row; i++) {
printf("%s", map[i]);
}
}

// 查询景点信息
void viewpoint_information(Graph* g) {
printf("请选择你想要查询的景点(选择景点前的选项)\n");
printf(" 0,北门     1,听海苑     2,国旗处     3,东操场   4,综合体\n");
printf(" 5,信院南    6,信院北     7,听海餐厅   8,东门     9,正门\n");
printf(" 10,西操场   11,望海餐厅   12,工程楼    13,望海苑\n");
int choice;
scanf("%d", &choice);
printf("%s\n", g->point[choice].name);//输出名称
printf("%s\n", g->point[choice].blurb);//展示相关信息
}

// Dijkstra 算法求最短路径
void dijkstra(Graph* G) {
int n = G->numVertices;
int start, end;
printf("请选择你的起点和终点(选择景点前的选项)\n");
printf(" 0,北门     1,听海苑     2,国旗处     3,东操场   4,综合体\n");
printf(" 5,信院南    6,信院北     7,听海餐厅   8,东门     9,正门\n");
printf(" 10,西操场   11,望海餐厅   12,工程楼    13,望海苑\n");
scanf("%d %d", &start, &end);
int distance[MAX], previous[MAX], visited[MAX];
int i, count, next_node, min_distance;

// 初始化
for (i = 0; i < n; i++) {
distance[i] = INF;
previous[i] = -1;
visited[i] = 0;
}
distance[start] = 0;

for (count = 0; count < n - 1; count++) {
min_distance = INF;

// 找到距离最小的未访问节点
for (i = 0; i < n; i++)
if (!visited[i] && distance[i] < min_distance) {
min_distance = distance[i];
```

```
next_node = i;
}

// 标记为已访问
visited[next_node] = 1;

// 更新相邻节点的距离
for (i = 0; i < n; i++)
if (!visited[i] && G->Edge[next_node][i] != -1 && distance[next_node] != INF
&& distance[next_node] + G->Edge[next_node][i] < distance[i]) {
distance[i] = distance[next_node] + G->Edge[next_node][i];
previous[i] = next_node;
}
}

// 输出路径
int path[MAX];
int path_index = 0;
int current = end;
while (current != -1) {
path[path_index++] = current;
current = previous[current];
}

if (distance[end] == INF) {
printf("没有路径从 %d 到 %d\n", start, end);
return;
}

printf("最短路径从 %s 到 %s: ", G->point[start].name, G->point[end].name);
for (i = path_index - 1; i >= 0; i--) {
printf("%s", G->point[path[i]].name);
if (i > 0) printf(" -> ");
}
printf("\n");
printf("总距离: %d(单位:10m)\n", distance[end]);
}

int main() {
const char *filename = "D:\\code\\c\\formal_test6\\map.txt";
Graph *g = (Graph *)malloc(sizeof(Graph));
create(g);
printf("************************************************\n");
printf("欢迎来到中国海洋大学景点查询功能\n");
```

```
printf("请选择功能\n");
printf("1.查看校园平面图\n");
printf("2.查询任意景点信息\n");
printf("3.查询任意景点之间路径\n");
printf("其他任意键退出\n");
printf("*************************************************\n");
int choice;
scanf("%d", &choice);
while (choice == 1 || choice == 2 || choice == 3) {
switch (choice) {
case 1:
print_map(filename);
break;
case 2:
viewpoint_information(g);
break;
case 3:
dijkstra(g);
break;
default:
break;
}
printf("请选择功能\n");
printf("1.查看校园平面图\n");
printf("2.查询任意景点信息\n");
printf("3.查询任意景点之间路径\n");
printf("其他任意键退出\n");
printf("*************************************************\n");
scanf("%d", &choice);
}
return 0;}
```

实验总结

本次实验利用西海岸平面图实现了校园路径查询功能、通过本次实验、熟悉了迪杰斯特拉算法求最短路径的方法、熟悉了基本的图的存储、构造以及相关的算法。通过本次实验、我也加深了图算法的相应知识、另外本次实验结合相关的背景，也较有意义。