

18340146_宋渝杰_实验 6

一、图的存储、遍历、最短路径算法

算法分析：

存储：

存储方面要求同时使用邻接矩阵和邻接表进行存储，要求可以从键盘直接输入图。在程序中事先建立一个 `vector<string>` 存储图的所有节点，`vector<vector<int>>` 存储邻接矩阵，`vector<node*>` 存储邻接表，之后可以通过【输入边的两个顶点和边长】来实现图的存储，具体算法如下：

- 1、输入总的操作步数（添加边的次数）
- 2、循环输入两个顶点的名字（string）和边长（int），输入后系统先判断两个顶点是否存在于图中，存在则继续操作，不存在则先添加该点于图中（通过扩展存储节点的 `vector`、邻接矩阵和邻接表），再继续操作。之后邻接矩阵对应位置存储边长，邻接表的对应链表添加对应节点（若已存在则不会再添加新节点）。
- 3、操作结束之后，输出“输入成功！”，退出存储模式。

遍历：

DFS：深度优先遍历一般采用压栈出栈方式实现，先建立一个标识数组，用来标识节点是否已遍历过。建立一个 `stack`，把其中一个节点压栈并输出，之后根据邻接矩阵判断节点连接状态，找到一个没有遍历过的新节点，将其压栈并输出，然后对新节点进

行相同操作。当寻找不到没有遍历过的新节点，则将 stack 顶部出栈，对新的顶部进行相同操作，直到所有节点被遍历过（即 stack 节点全部出栈）。

BFS：广度优先遍历又叫做层次遍历，实现原理和树的层次遍历类似，建立一个标识数组，用来标识节点是否已遍历过。建立一个 vector，将第一个节点 push_back，然后遍历 vector，遍历过程中将当前遍历的节点的所有未遍历过的新节点 push_back 入 vector，遍历过程中输出即可。

最短路径算法：

此处要求使用迪杰斯特拉算法，根据邻接矩阵进行图的分析。系统提示输入两个节点，然后输出最短路径。具体算法如下：

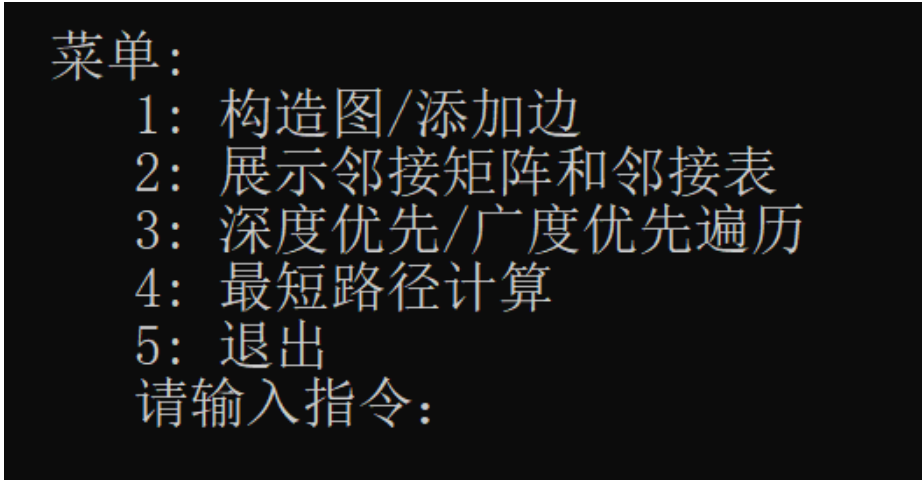
- 1、选择第一个节点，分析它连接的节点，选择最短的一条边的节点，并确定这是这两点之间最短的路径，此时已选择过两个节点；

- 2、选择已经选择过的节点们，分析它们连接的节点，选择最短的一条边的那个节点，判断它和第一个节点的直线距离和通过中间节点的路径的距离，选择短的那条并将其确定为这两点的最短路径；

- 3、重复步骤二，直到所有节点都被选择过，此时已生成第一个节点与其他所有节点的最短路径，最后输出之前输入的两个节点之间的最短路径即可。

流程：

程序开始时先输出菜单，提示用户输入相应数字进行功能的选择



```
菜单：
1： 构造图/添加边
2： 展示邻接矩阵和邻接表
3： 深度优先/广度优先遍历
4： 最短路径计算
5： 退出
请输入指令：
```

1： 进行构造图/添加边操作： 程序初始时不存在图，第一次选择该功能时显示“构造图”，之后再次选择该操作时，将会是在构造好的图中进行“添加边”操作。

2： 展示邻接矩阵和邻接表： 在图的构造过程中已经构造好了邻接矩阵和邻接表，此处直接输出即可。

3： DFS/BFS 输出： 根据上述算法对图进行深度/广度输出。

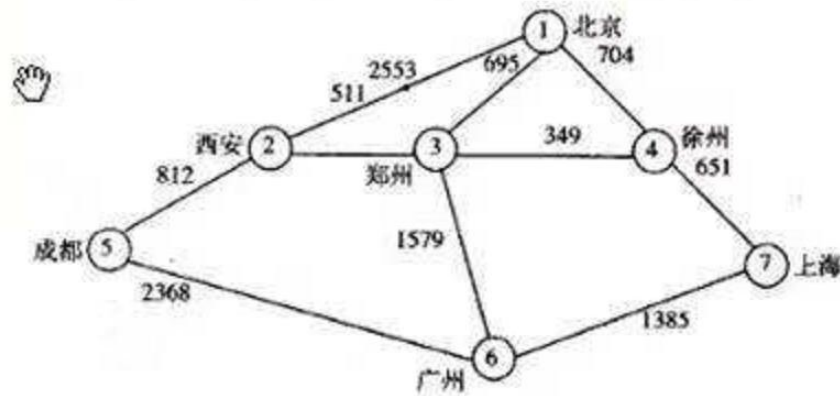
4： 最短路径： 系统提示输入两个顶点，系统根据上述迪杰斯特拉算法输出最短路径。

5： 退出程序。

程序测试：

测试样例 1：

添加图：



```
菜单：
1: 构造图/添加边
2: 展示邻接矩阵和邻接表
3: 深度优先/广度优先遍历
4: 最短路径计算
5: 退出
请输入指令： 1
请输入图的边的条数：
10
请输入两个顶点的名称以及边长：
北京 西安 2553
请输入两个顶点的名称以及边长：
北京 郑州 695
请输入两个顶点的名称以及边长：
北京 徐州 704
请输入两个顶点的名称以及边长：
西安 郑州 511
请输入两个顶点的名称以及边长：
郑州 徐州 349
请输入两个顶点的名称以及边长：
徐州 上海 651
请输入两个顶点的名称以及边长：
上海 广州 1385
请输入两个顶点的名称以及边长：
郑州 广州 1579
请输入两个顶点的名称以及边长：
西安 成都 812
请输入两个顶点的名称以及边长：
成都 广州 2368
输入成功！
```

输出邻接矩阵和邻接表：

邻接矩阵：							
	北京	西安	郑州	徐州	上海	广州	成都
北京	0	2553	695	704	-1	-1	-1
西安	2553	0	511	-1	-1	-1	812
郑州	695	511	0	349	-1	1579	-1
徐州	704	-1	349	0	651	-1	-1
上海	-1	-1	-1	651	0	1385	-1
广州	-1	-1	1579	-1	1385	0	2368
成都	-1	812	-1	-1	-1	2368	0

邻接表：							
北京	-->	西安	-->	郑州	-->	徐州	
西安	-->	北京	-->	郑州	-->	成都	
郑州	-->	北京	-->	西安	-->	徐州	--> 广州
徐州	-->	北京	-->	郑州	-->	上海	
上海	-->	徐州	-->	广州			
广州	-->	上海	-->	郑州	-->	成都	
成都	-->	西安	-->	广州			

深度/广度优先遍历：

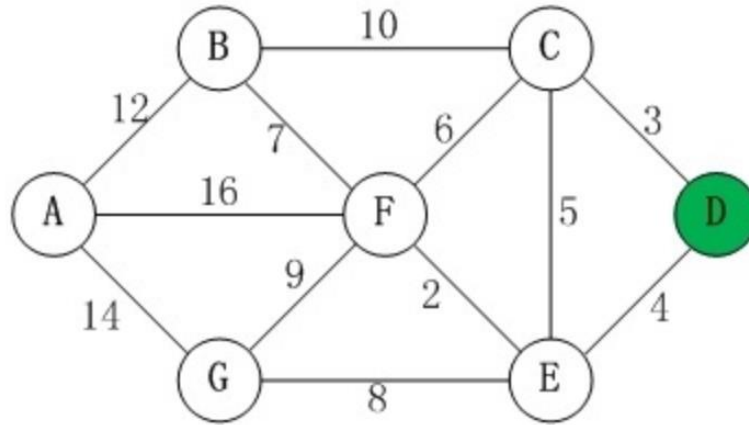
DFS：							
北京	西安	郑州	徐州	上海	广州	成都	
BFS：							
北京	西安	郑州	徐州	成都	广州	上海	

最短路径：

```
请输入指令： 4
请输入两个顶点： 广州 北京
两顶点的最短距离为： 2274
```

测试样例 2:

添加图



```
请输入指令： 1
请输入图的边的条数：
12
请输入两个顶点的名称以及边长：
A B 12
请输入两个顶点的名称以及边长：
A F 16
请输入两个顶点的名称以及边长：
A G 14
请输入两个顶点的名称以及边长：
B F 7
请输入两个顶点的名称以及边长：
F G 9
请输入两个顶点的名称以及边长：
E G 8
请输入两个顶点的名称以及边长：
E F 2
请输入两个顶点的名称以及边长：
B C 10
请输入两个顶点的名称以及边长：
C F 6
请输入两个顶点的名称以及边长：
C E 5
请输入两个顶点的名称以及边长：
C D 3
请输入两个顶点的名称以及边长：
D E 4
输入成功！
```

输出邻接矩阵和邻接表

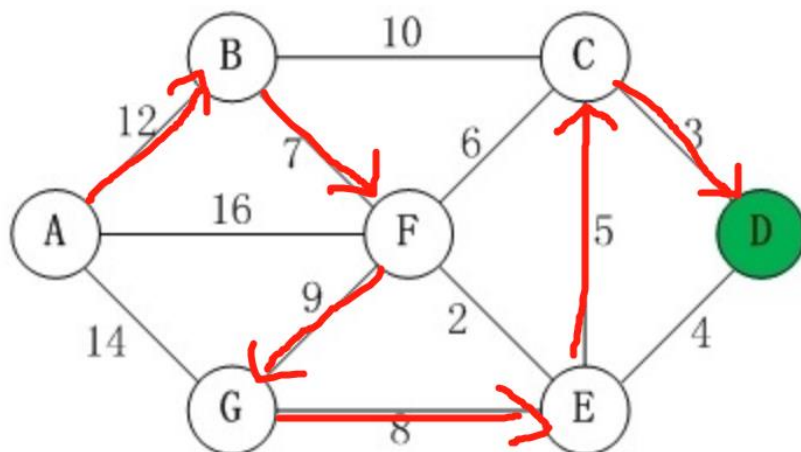
```
请输入指令： 2
邻接矩阵：
  A    B    F    G    E    C    D
A    0   12   16   14   -1   -1   -1
B   12    0    7   -1   -1   10   -1
F   16    7    0    9    2    6   -1
G   14   -1    9    0    8   -1   -1
E   -1   -1    2    8    0    5    4
C   -1   10    6   -1    5    0    3
D   -1   -1   -1   -1    4    3    0

邻接表：
A --> B --> F --> G
B --> A --> F --> C
F --> A --> B --> G --> E --> C
G --> A --> F --> E
E --> G --> F --> C --> D
C --> B --> F --> E --> D
D --> C --> E
```

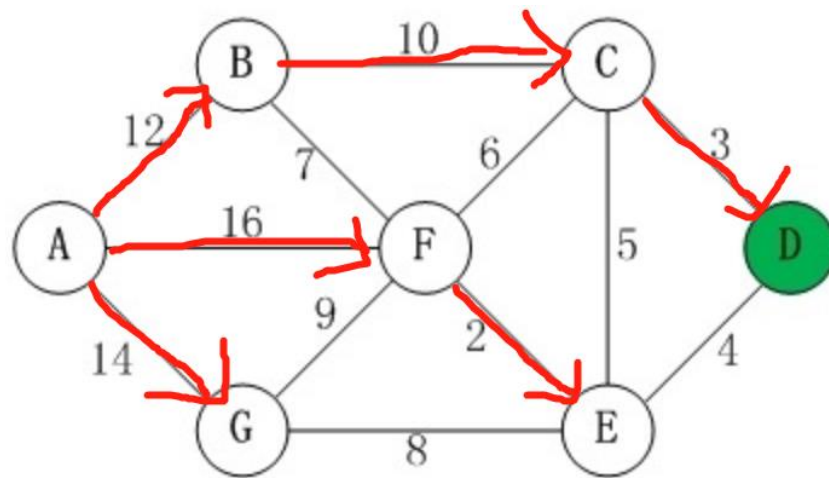
深度/广度优先遍历

```
请输入指令： 3
DFS:
  A    B    F    G    E    C    D
BFS:
  A    B    F    G    C    E    D
```

DFS:



BFS:



最短路径:

```
请输入指令: 4
请输入两个顶点: D A
两顶点的最短距离为: 22
```

路径为 D->E->F->A, 长度为 22。