**Project 1** **图的相关算法实验**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 学号 | 2021111011004 | 姓名 | 王齐 | 上交时间 | 2021-11-06 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 题号 | 程序逻辑(40) | 算法新颖性(20) | 代码规范 (20) | 实验报告(20) | 总分 |
| 得分 |  |  |  |  |  |

**Project 6-1: 图的深度优先搜索**

**1. 问题描述**

写程序实现图的深度优先搜索算法（DFS），对输入的有向图要求输出DFS遍历结果。

**2. 算法描述**

图的存储结构（邻接矩阵）：

#define MaxVNum 100 //顶点数最大值

#define MaxDistance 65535 //两个顶点之间最大距离

typedef struct {

int Vex[MaxVNum]; //存储顶点信息

int Edge[MaxVNum][MaxVNum]; //存储边信息

int n, e; //顶点数，边数

} MGraph;

深度优先搜索（DFS）：

算法 DFS(MG, v, visited)

/\*图的深度优先递归搜索算法。MG代表使用邻接矩阵存储的图，v代表搜索的起始顶点，visited表示标志顶点是否被访问的辅助数组\*/

DFS1. [初始化]

PRANT(v).

visited[v] 1

DFS2. [深度优先搜索图]

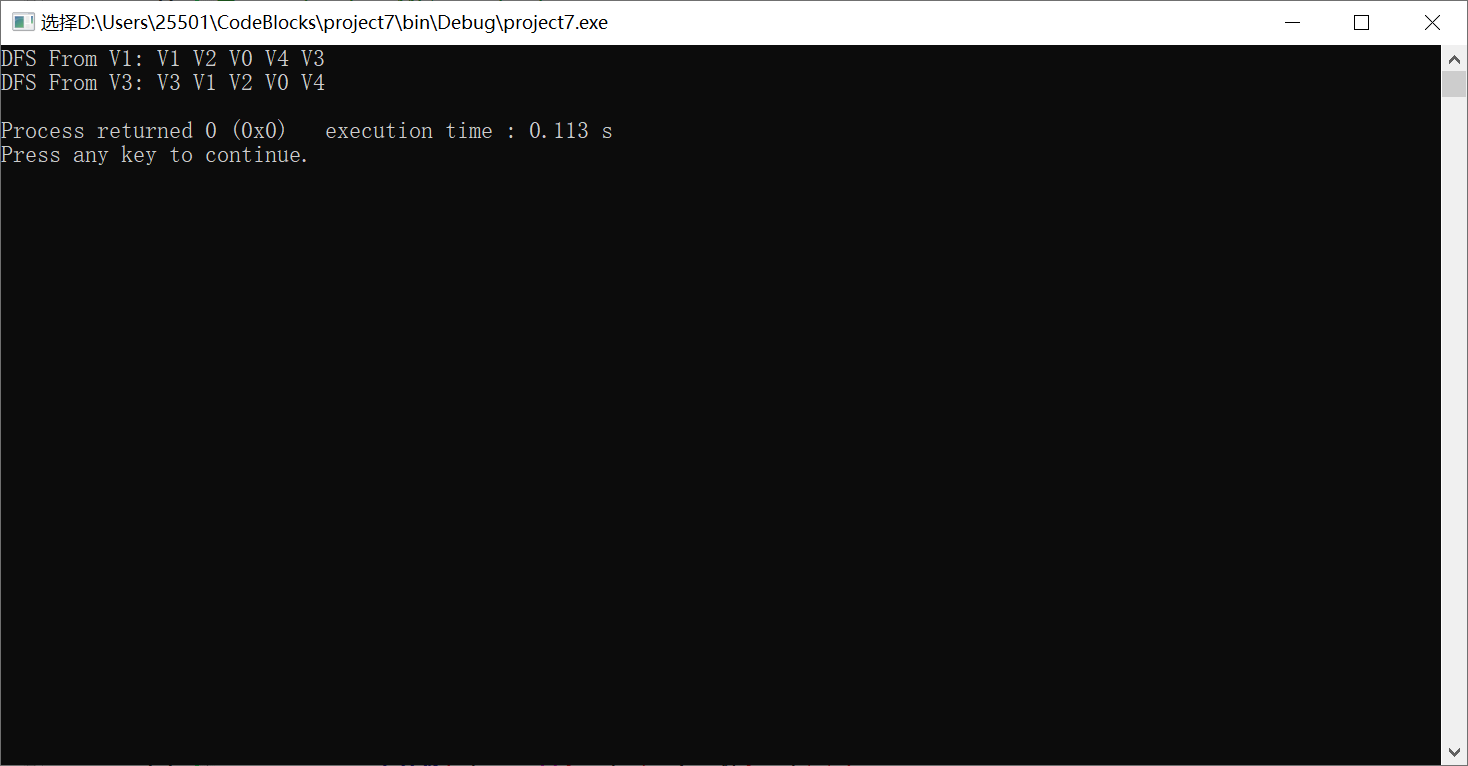
FOR k=1 to n DO

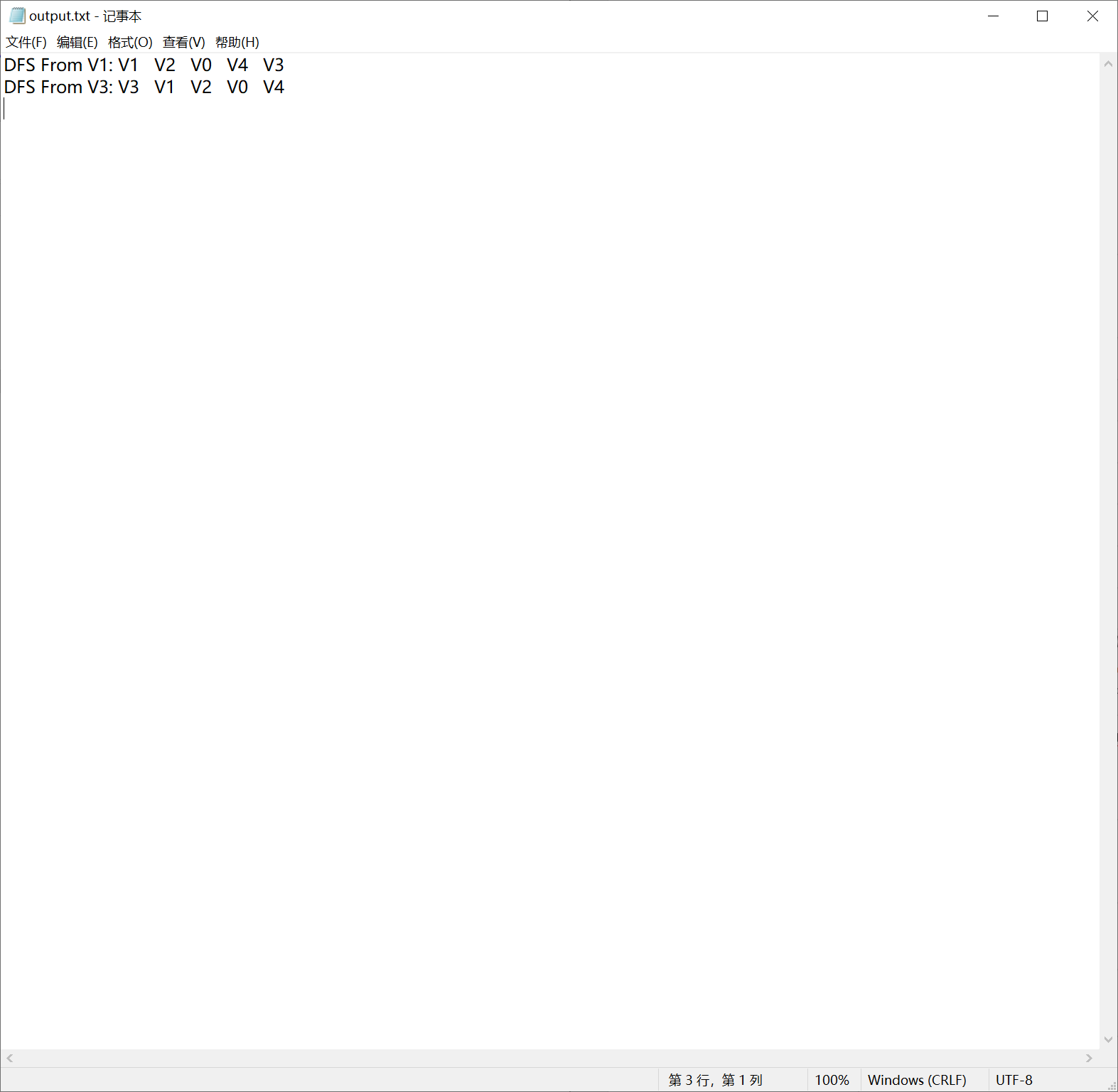
IF visited[k] = 0 THEN

DFS(MG, k, visited).

**3. 测试结果**

控制台输出结果：

output.txt输出结果：



**4. 分析与评论**

在进行图的深度优先搜索时，采用的是邻接矩阵的数据结构，通过递归函数来实现。

**Project 6-2: 图的广度优先搜索**

**1. 问题描述**

写程序实现图的广度优先搜索算法（BFS），对输入的有向图要求输出BFS遍历结果。

**2. 算法描述**

图的存储结构（邻接矩阵）：

#define MaxVNum 100 //顶点数最大值

#define MaxDistance 65535 //两个顶点之间最大距离

typedef struct {

int Vex[MaxVNum]; //存储顶点信息

int Edge[MaxVNum][MaxVNum]; //存储边信息

int n, e; //顶点数，边数

} MGraph;

广度优先搜索（BFS）：

算法 BFS(MG, v, visited)

/\*图的广度优先递归搜索算法。MG代表使用邻接矩阵存储的图，v代表搜索的起始顶点，visited表示标志顶点是否被访问的辅助数组\*/

DFS1. [初始化]

CREATE(Q). //创建队列Q

FOR i=1 TO n DO visited[i] 0

PRANT(v).

visited[v] 1

Q v //入队

DFS2. [广度优先搜索图]

WHILE NOT(ISEMPTY(Q)) D)

( v Q. //出队

FOR k = 1 TO n DO

( IF visited[k] = 0 THEN

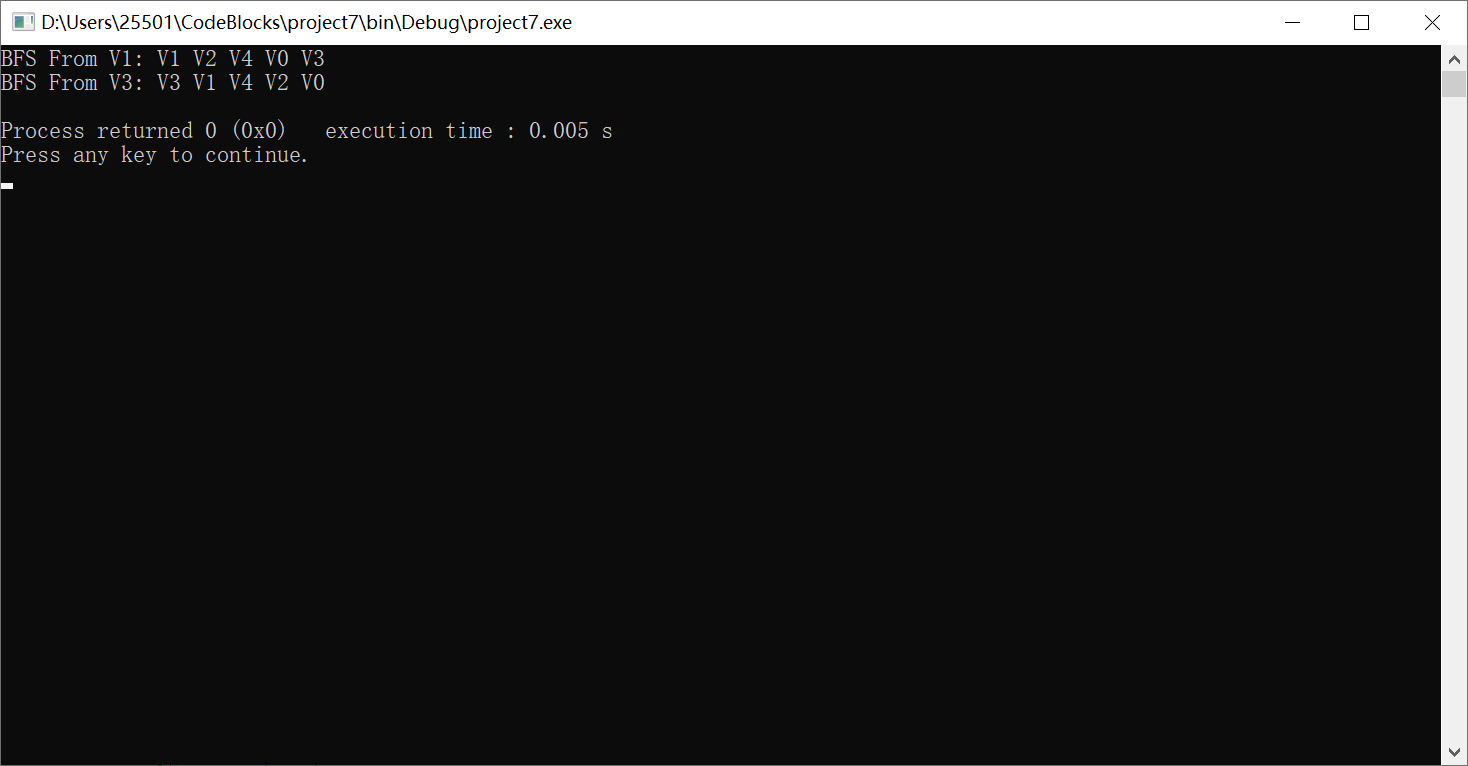
(PRANT(k).

visited[k] 1.

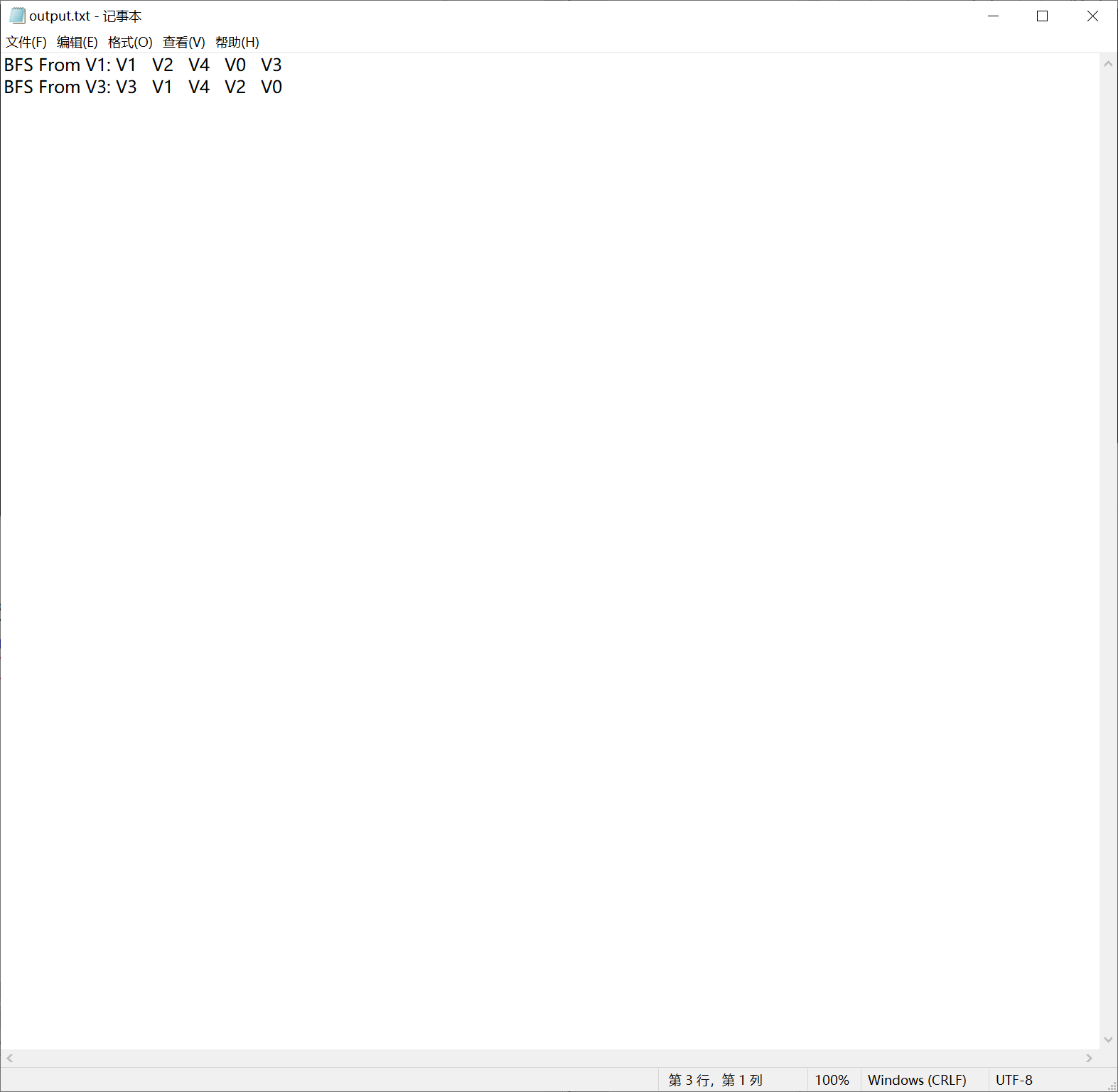
Q k .))) //入队

**3. 测试结果**

控制台输出结果：



output.txt输出结果：



**4. 分析与评论**

在进行图的深度优先搜索时，采用的是邻接矩阵的数据结构，通过队列来实现。

**Project 6-3: 图的最短路径实验**

**1. 问题描述**

写程序求出给定有向网的单源点到所有目标点的最短路径，并输出到这些最短路径。

**2. 算法描述**

图的存储结构（邻接矩阵）：

#define MaxVNum 100 //顶点数最大值

#define MaxDistance 65535 //两个顶点之间最大距离

typedef struct {

int Vex[MaxVNum]; //存储顶点信息

int Edge[MaxVNum][MaxVNum]; //存储边信息

int n, e; //顶点数，边数

} MGraph;

最短路径（Dijkstra）：

算法 Dijkstra(MG, n, v, dist, path)

/\*计算v到其他各顶点的最短距离。MG代表使用邻接矩阵存储的图，v代表起始顶点，n代表顶点个数\*/

D1. [初始化]

FOR i=1 TO n DO

( path[i] -1.

dist[i] max.

s[i] 0.) //数组s[i]记录i是否被访问过

dist[v] 0.

s[v] 1.

u v. //u为即将访问的顶点

D2. [求从初始顶点v到其他各顶点的最短距离]

FOR j=1 to n-1 DO

( FOR j = 1 TO n DO

( k = Vex[v].

IF (s[k] 1 AND dist[u] + Edge[k][j] < dist[k]) THEN

(dist[k]dist[u] + Edge[k][j].

path[k] *u.))*

ldist max.

FOR i=1 TO n DO

IF(dist[i]<ldist AND s[i]=0) THEN

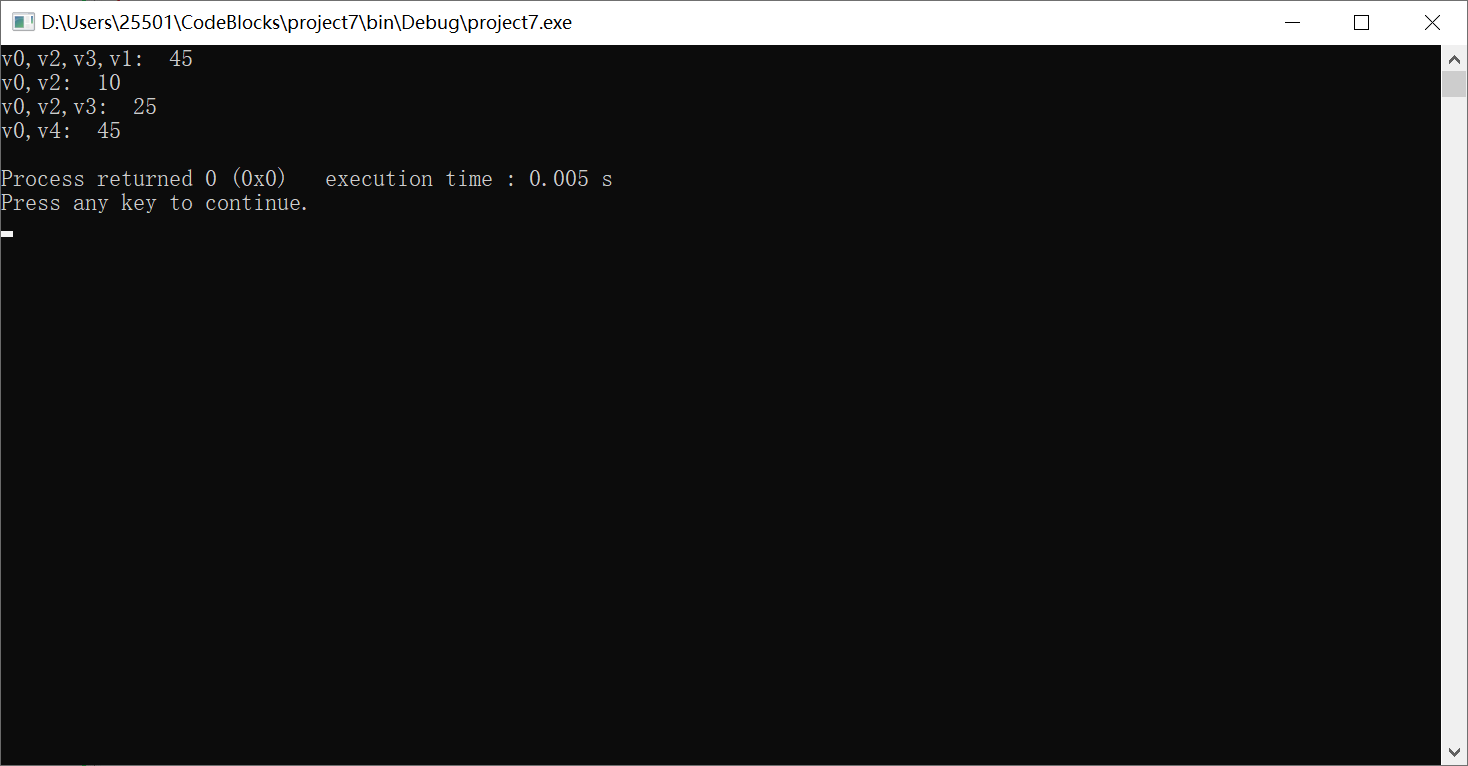
(ldist dist[i].

u i.)

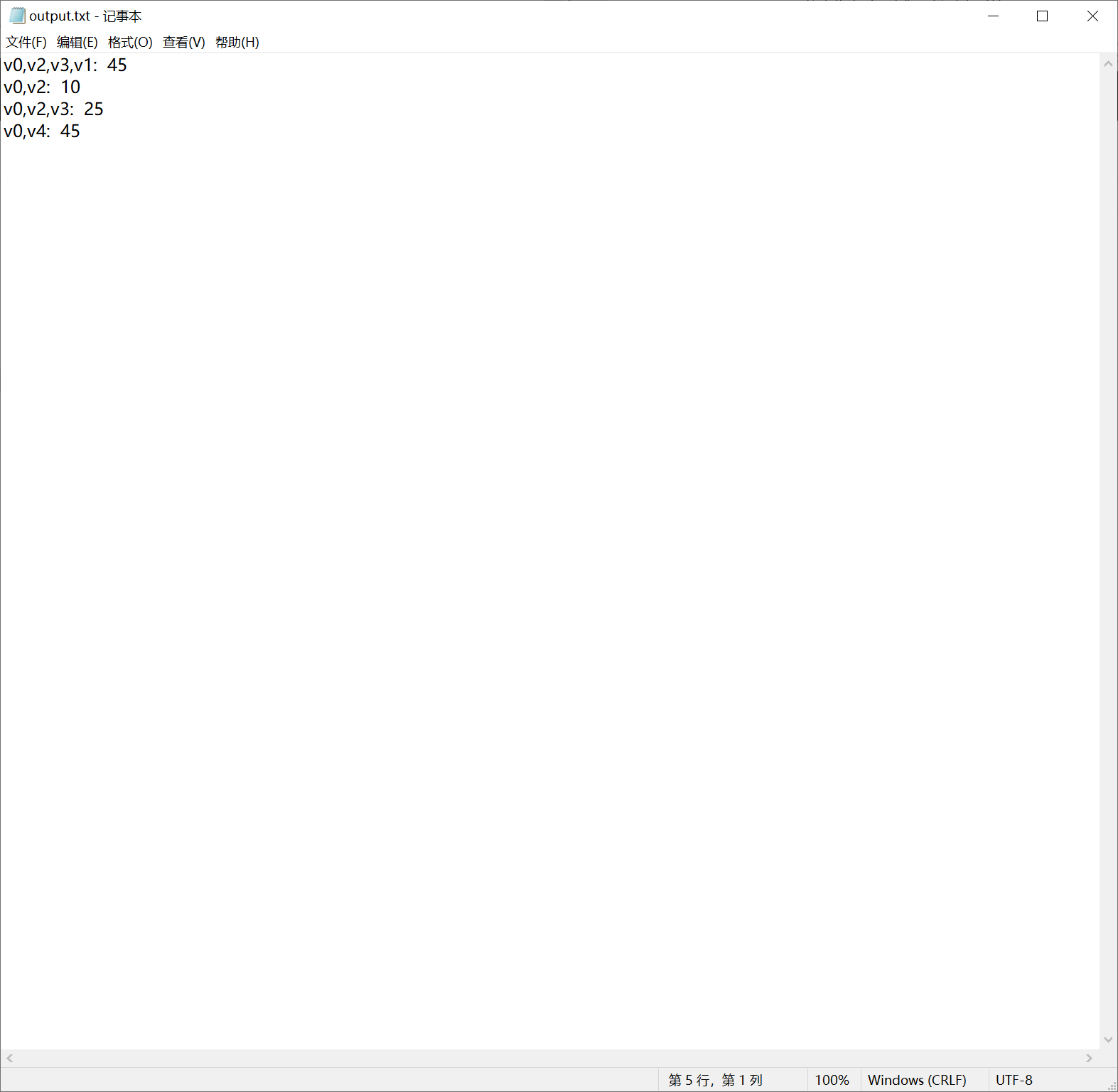
s[u] 1.)

**3. 测试结果**

控制台输出结果：



output.txt输出结果：



**4. 分析与评论**

在求顶点之间最短距离是，采用邻接矩阵的数据结构，通过Dijkstra算法实现。