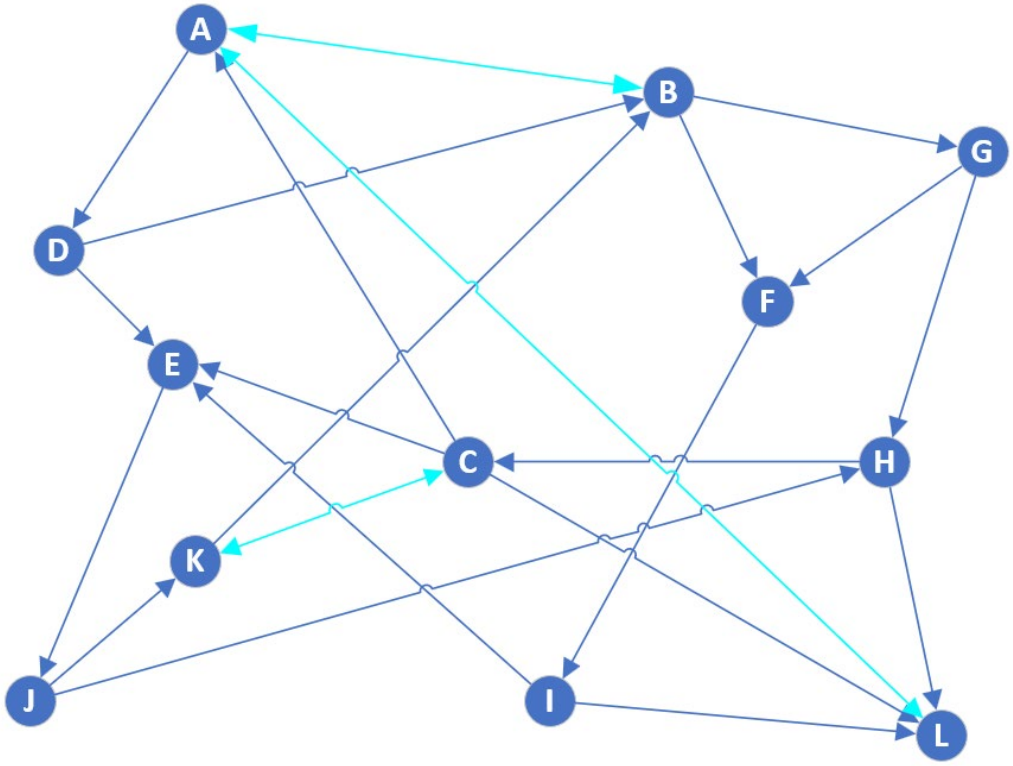
张宇杰 2022113573
以下程序均以该图作为输入：蓝色为单向边，浅蓝色为双向边
图形用 Visio Professional 绘制



输入序列如下：
可见于文件./my-graph/graph_1.txt

```
+v A
+v B
+v C
+v D
+v E
+v F
+v G
+v H
+v I
+v J
+v K
+v L

+e A D 1
+e C A 1
+e A L 1
+e L A 1
+e A B 1
+e B A 1
```

```
20.
21.  +e D B 1
22.  +e K B 1
23.  +e B F 1
24.  +e B G 1
25.
26.  +e C E 1
27.  +e C K 1
28.  +e K C 1
29.  +e H C 1
30.  +e C L 1
31.
32.  +e D E 1
33.
34.  +e E J 1
35.  +e I E 1
36.
37.  +e F I 1
38.  +e G F 1
39.
40.  +e G H 1
41.
42.  +e H L 1
43.  +e J H 1
44.
45.  +e J K 1
46.
47.  +e I L 1
```

一、分别实现无向图（或有向图）的邻接矩阵和邻接表存储结构的建立算法，分析和比较其建立算法的时间复杂度以及存储结构的空间占用情况
五、以适当的方式输入图的顶点和边，并显示相应的结果。要求顶点不少于 10 个，边不少于 13 条（图的规模越大越好）

有向图，邻接矩阵

运行程序:

```
1.  #include <iostream>
2.  #include <fstream>
3.
4.  #include ".\src\MyGraph"
5.  using namespace MyGraph;
6.  using namespace std;
7.
```

```cpp
int main()
{
    system("chcp 65001");
    ifstream command("./graph_1.txt");

    AdjacencyMatrixGraph G1;
    G1.commandSequence(command);
    G1.showMatrix();

    cout << endl;
}
```

运行结果：

```
Active code page: 65001
       A       B       C       D       E       F       G       H       I       J       K       L
A      inf     1       inf     1       inf     inf     inf     inf     inf     inf     inf     1
B      1       inf     inf     inf     inf     1       1       inf     inf     inf     inf     inf
C      1       inf     inf     inf     1       inf     inf     inf     inf     inf     1       1
D      inf     1       inf     inf     1       inf     inf     inf     inf     inf     inf     inf
E      inf     inf     inf     inf     inf     inf     inf     inf     inf     1       inf     inf
F      inf     inf     inf     inf     inf     inf     inf     inf     1       inf     inf     inf
G      inf     inf     inf     inf     inf     1       inf     1       inf     inf     inf     inf
H      inf     inf     1       inf     inf     inf     inf     inf     inf     inf     inf     1
I      inf     inf     inf     inf     1       inf     inf     inf     inf     inf     inf     1
J      inf     inf     inf     inf     inf     inf     inf     1       inf     inf     1       inf
K      inf     1       1       inf     inf     inf     inf     inf     inf     inf     inf     inf
L      1       inf     inf     inf     inf     inf     inf     inf     inf     inf     inf     inf

(base) PS D:\File\大二秋\DSA\作业4\my-graph>
```

分析：

时间复杂度：每加一次边，对邻接矩阵进行一次修改，修改是 O(1)的，故为 O(E)

空间占用：显然为 $O(V^2)$

有向图，邻接表

运行程序：

```cpp
#include <iostream>
#include <fstream>

#include ".\src\MyGraph"
using namespace MyGraph;
using namespace std;

int main()
{
    system("chcp 65001");
    ifstream command("./graph_1.txt");

    AdjacencyListGraph G1;
    G1.commandSequence(command);
    G1.showList();

```

```
        cout << endl;
    }
```

运行结果：

```
(base) PS D:\File\大二秋\DSA\作业4\my-graph> .\test.exe
Active code page: 65001
A -> [D|1] -> [L|1] -> [B|1] -> end
B -> [A|1] -> [F|1] -> [G|1] -> end
C -> [A|1] -> [E|1] -> [K|1] -> [L|1] -> end
D -> [B|1] -> [E|1] -> end
E -> [J|1] -> end
F -> [I|1] -> end
G -> [F|1] -> [H|1] -> end
H -> [C|1] -> [L|1] -> end
I -> [E|1] -> [L|1] -> end
J -> [H|1] -> [K|1] -> end
K -> [B|1] -> [C|1] -> end
L -> [A|1] -> end

(base) PS D:\File\大二秋\DSA\作业4\my-graph>
```

分析：

时间复杂度：每加一次边，向对应邻接表插入一条边，而 vector 尾插的均摊时间复杂度为 O(1)，故整体时间复杂度为 O(E)

空间占用：显然为 O(V+E)

二、实现无向图（或有向图）的邻接矩阵和邻接表两种存储结构的相互转换算法

邻接矩阵转邻接表

运行程序：

```
#include <iostream>
#include <fstream>

#include ".\src\MyGraph"
using namespace MyGraph;
using namespace std;

int main()
{
    system("chcp 65001");
    ifstream command("./graph_1.txt");

    AdjacencyMatrixGraph G1;
    G1.commandSequence(command);
    G1.showMatrix();
```

```
        cout << endl;

        AdjacencyListGraph G2 = matrixToList(G1);
        G2.showList();
        cout << endl;
    }
```

运行结果:

```
(base) PS D:\File\大二秋\DSA\作业4\my-graph> .\test.exe
Active code page: 65001
        A       B       C       D       E       F       G       H       I       J       K       L
A       inf     1       inf     1       inf     inf     inf     inf     inf     inf     inf     1
B       1       inf     inf     inf     inf     1       1       inf     inf     inf     inf     inf
C       1       inf     inf     inf     1       inf     inf     inf     inf     inf     1       1
D       inf     1       inf     inf     1       inf     inf     inf     inf     inf     inf     inf
E       inf     inf     inf     inf     inf     inf     inf     inf     inf     1       inf     inf
F       inf     inf     inf     inf     inf     inf     inf     inf     1       inf     inf     inf
G       inf     inf     inf     inf     inf     1       inf     1       inf     inf     inf     inf
H       inf     inf     1       inf     inf     inf     inf     inf     inf     inf     inf     1
I       inf     inf     inf     inf     1       inf     inf     inf     inf     inf     inf     1
J       inf     inf     inf     inf     inf     inf     inf     1       inf     inf     1       inf
K       inf     1       1       inf     inf     inf     inf     inf     inf     inf     inf     inf
L       1       inf     inf     inf     inf     inf     inf     inf     inf     inf     inf     inf

A -> [B|1] -> [D|1] -> [L|1] -> end
B -> [A|1] -> [F|1] -> [G|1] -> end
C -> [A|1] -> [E|1] -> [K|1] -> [L|1] -> end
D -> [B|1] -> [E|1] -> end
E -> [J|1] -> end
F -> [I|1] -> end
G -> [F|1] -> [H|1] -> end
H -> [C|1] -> [L|1] -> end
I -> [E|1] -> [L|1] -> end
J -> [H|1] -> [K|1] -> end
K -> [B|1] -> [C|1] -> end
L -> [A|1] -> end

(base) PS D:\File\大二秋\DSA\作业4\my-graph>
```

结果正确

分析：时间复杂度为 $O(V^2)$

邻接表转邻接矩阵

运行程序:

```
    #include <iostream>
    #include <fstream>

    #include ".\src\MyGraph"
    using namespace MyGraph;
    using namespace std;

    int main()
    {
        system("chcp 65001");
        ifstream command("./graph_1.txt");

        AdjacencyListGraph G1;
        G1.commandSequence(command);
```

```
        G1.showList();
        cout << endl;

        AdjacencyMatrixGraph G2 = listToMatrix(G1);
        G2.showMatrix();
        cout << endl;
    }
```

运行结果：

```
(base) PS D:\File\大二秋\DSA\作业4\my-graph> .\test.exe
Active code page: 65001
A -> [D|1] -> [L|1] -> [B|1] -> end
B -> [A|1] -> [F|1] -> [G|1] -> end
C -> [A|1] -> [E|1] -> [K|1] -> [L|1] -> end
D -> [B|1] -> [E|1] -> end
E -> [J|1] -> end
F -> [I|1] -> end
G -> [F|1] -> [H|1] -> end
H -> [C|1] -> [L|1] -> end
I -> [E|1] -> [L|1] -> end
J -> [H|1] -> [K|1] -> end
K -> [B|1] -> [C|1] -> end
L -> [A|1] -> end

        A       B       C       D       E       F       G       H       I       J       K       L
A       inf     1       inf     1       inf     inf     inf     inf     inf     inf     inf     1
B       1       inf     inf     inf     inf     1       1       inf     inf     inf     inf     inf
C       1       inf     inf     inf     1       inf     inf     inf     inf     inf     1       1
D       inf     1       inf     inf     1       inf     inf     inf     inf     inf     inf     inf
E       inf     inf     inf     inf     inf     inf     inf     inf     inf     1       inf     inf
F       inf     inf     inf     inf     inf     inf     inf     inf     1       inf     inf     inf
G       inf     inf     inf     inf     inf     1       inf     1       inf     inf     inf     inf
H       inf     inf     1       inf     inf     inf     inf     inf     inf     inf     inf     1
I       inf     inf     inf     inf     1       inf     inf     inf     inf     inf     inf     1
J       inf     inf     inf     inf     inf     inf     inf     1       inf     inf     1       inf
K       inf     1       1       inf     inf     inf     inf     inf     inf     inf     inf     inf
L       1       inf     inf     inf     inf     inf     inf     inf     inf     inf     inf     inf

(base) PS D:\File\大二秋\DSA\作业4\my-graph>
```

结果正确

分析：时间复杂度为 O(V+E)

三、在上述两种存储结构上，分别实现无向图（或有向图）的深度优先搜索(递归和非递归)和广度优先搜索算法。并以适当的方式存储和展示相应的搜索结果，包括：深度优先或广度优先生成森林（或生成树）、深度优先或广度优先序列和深度优先或广度优先编号。并分析搜索算法的时间复杂度和空间复杂度。

P.S. 在本次程序设计中，邻接表存储的图 AdjacencyListGraph 与邻接矩阵存储的图 AdjacencyMatrixGraph 均共用了父类 Graph 的各种搜索方法
故以下的程序以及运行结果中，仅展示对邻接表存储的图 AdjacencyListGraph 的搜索结果（因为两种存储结构的代码一致，结果也一致）
仅在分析复杂度时讨论二者的不同

深度优先搜索（递归）

运行程序：

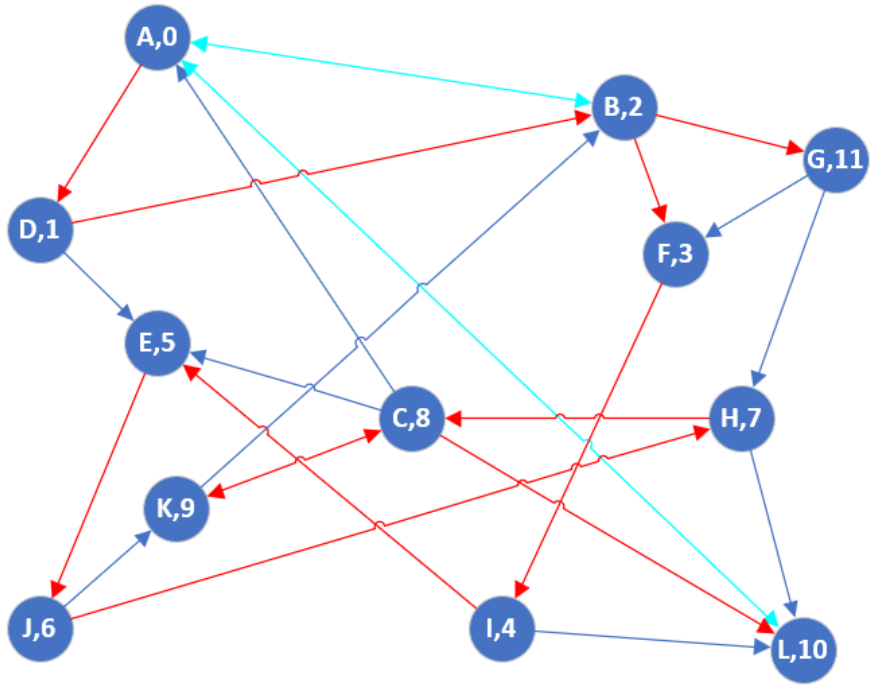
```
#include <iostream>
#include <fstream>
```

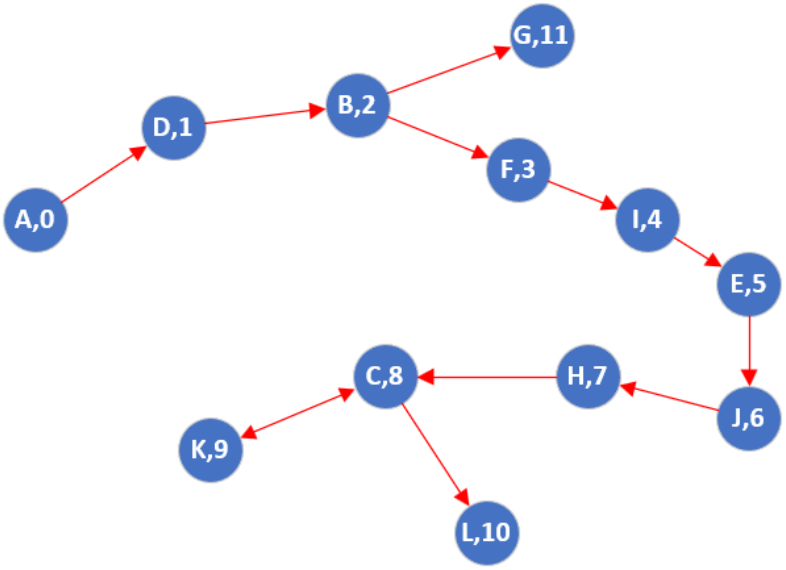
```
3.
4.      #include ".\src\MyGraph"
5.      using namespace MyGraph;
6.      using namespace std;
7.
8.      int main()
9.      {
10.         system("chcp 65001");
11.         ifstream command("./graph_1.txt");
12.
13.         AdjacencyListGraph G1;
14.         G1.commandSequence(command);
15.         G1.showList();
16.         cout << endl;
17.
18.         G1.dfs();
19.         cout << endl;
20.     }
```

输出结果:

```
(base) PS D:\File\大二秋\DSA\作业4\my-graph> .\test.exe
Active code page: 65001
A -> [D|1] -> [L|1] -> [B|1] -> end
B -> [A|1] -> [F|1] -> [G|1] -> end
C -> [A|1] -> [E|1] -> [K|1] -> [L|1] -> end
D -> [B|1] -> [E|1] -> end
E -> [J|1] -> end
F -> [I|1] -> end
G -> [F|1] -> [H|1] -> end
H -> [C|1] -> [L|1] -> end
I -> [E|1] -> [L|1] -> end
J -> [H|1] -> [K|1] -> end
K -> [B|1] -> [C|1] -> end
L -> [A|1] -> end

==================================
深度优先序列: A D B F I E J H C K L G
深度优先编号 及 深度优先生成森林信息:
标签    A       B       C       D       E       F       G       H       I       J       K       L
编号    0       2       8       1       5       3       11      7       4       6       9       10
父节点  null    D       H       A       I       B       B       J       F       E       C       C
==================================
(base) PS D:\File\大二秋\DSA\作业4\my-graph>
```

转换为图形:





分析：

时间复杂度：分为两个部分：访问每个节点花费的时间，以及在每个节点找邻居花费的时间

对邻接矩阵：

1：V 个节点需要 O(V)

2：对邻接矩阵，对于节点 i，需要扫描第 i 行的每一个元素才能找到所有的邻居，需要 O(V)

故总复杂度 O(V^2)

对邻接表：

1：n 个节点需要 O(V)

2：对邻接表，总共的找邻居时间复杂度就是遍历边表的时间复杂度。对于有向图：O(V)

故总复杂度 O(V+E)

空间复杂度:
无论是邻接矩阵还是邻接表，递归工作栈的最大深度为 V，故空间复杂度为 O(V)

深度优先搜索（非递归）

运行程序:

```cpp
#include <iostream>
#include <fstream>

#include ".\src\MyGraph"
using namespace MyGraph;
using namespace std;

int main()
{
    system("chcp 65001");
    ifstream command("./graph_1.txt");

    AdjacencyListGraph G1;
    G1.commandSequence(command);
    G1.showList();
    cout << endl;

    G1.dfs_no_rec();
    cout << endl;
}
```

运行结果:

```
父节点  null    D       H       A       I       B       B       J       F       E       C       C
==================================
(base) PS D:\File\大二秋\DSA\作业4\my-graph> .\test.exe
Active code page: 65001
A -> [D|1] -> [L|1] -> [B|1] -> end
B -> [A|1] -> [F|1] -> [G|1] -> end
C -> [A|1] -> [E|1] -> [K|1] -> [L|1] -> end
D -> [B|1] -> [E|1] -> end
E -> [J|1] -> end
F -> [I|1] -> end
G -> [F|1] -> [H|1] -> end
H -> [C|1] -> [L|1] -> end
I -> [E|1] -> [L|1] -> end
J -> [H|1] -> [K|1] -> end
K -> [B|1] -> [C|1] -> end
L -> [A|1] -> end

==================================
深度优先序列(非递归): A D B F I E J H C K L G
深度优先编号 及 深度优先生成森林信息:
标签:   A       B       C       D       E       F       G       H       I       J       K       L
编号:   0       2       8       1       5       3       11      7       4       6       9       10
父节点: null    D       H       A       I       B       B       J       F       E       C       C
==================================
(base) PS D:\File\大二秋\DSA\作业4\my-graph>
```

与递归实现的 dfs 一致
分析：复杂度同递归实现的 dfs

广度优先搜索

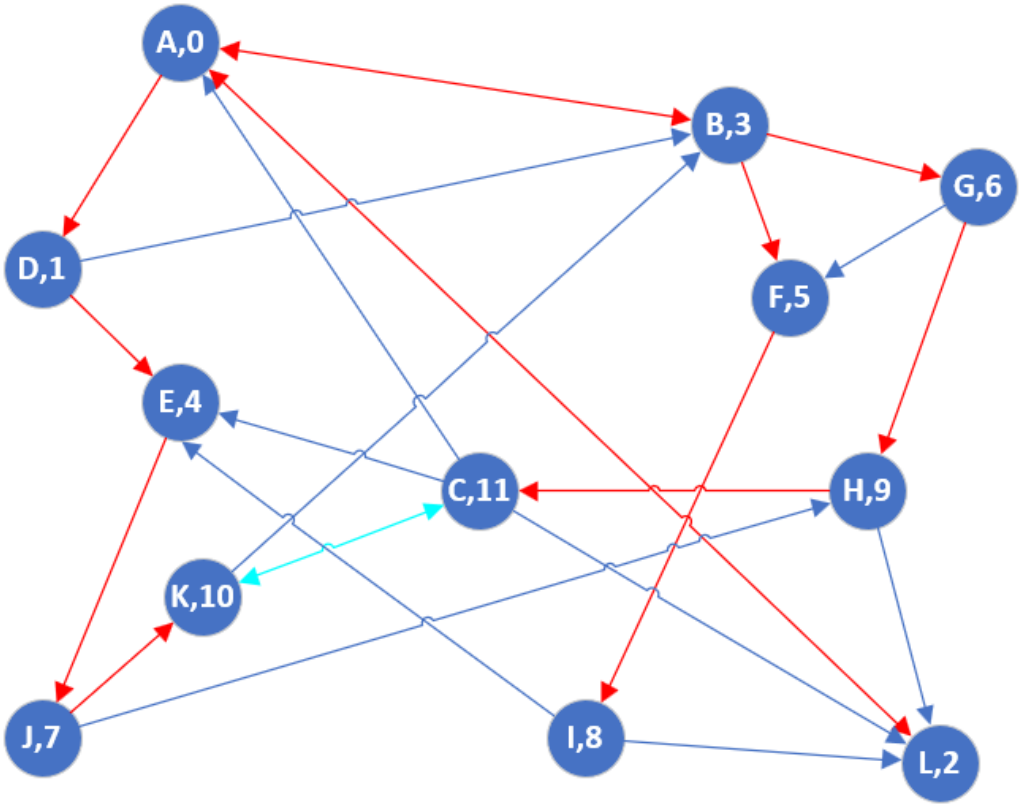
运行程序:

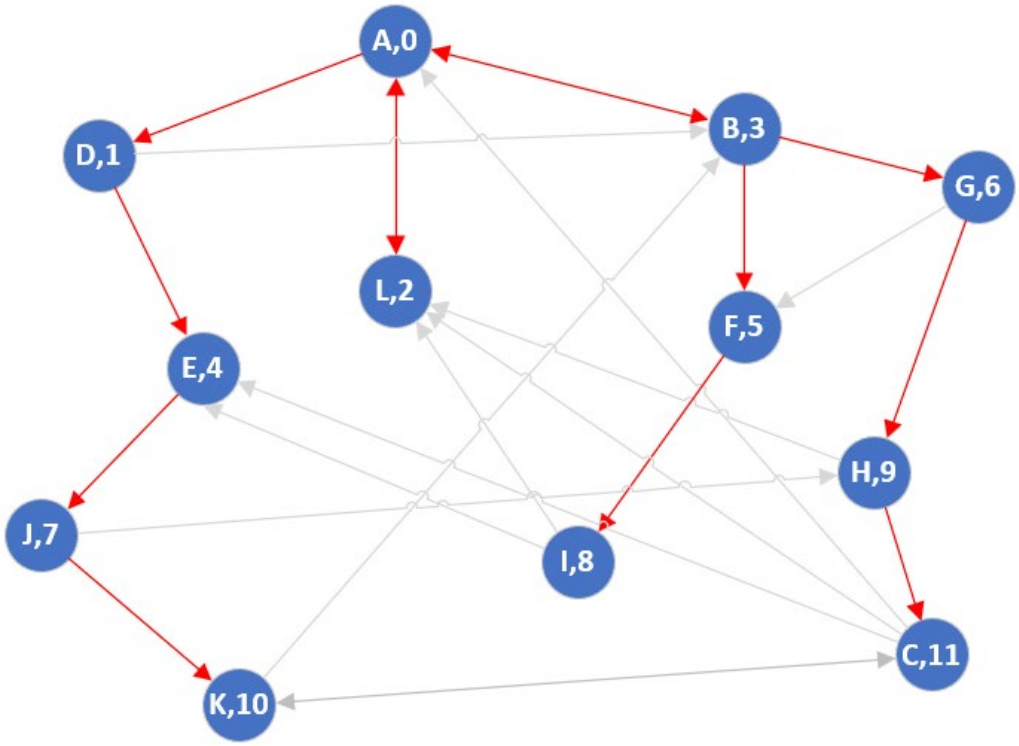
```cpp
#include <iostream>
#include <fstream>

#include ".\src\MyGraph"
using namespace MyGraph;
using namespace std;

int main()
{
    system("chcp 65001");
    ifstream command("./graph_1.txt");

    AdjacencyListGraph G1;
    G1.commandSequence(command);
    G1.showList();
    cout << endl;

    G1.bfs();
    cout << endl;
}
```

运行结果:

```
---------------------------------
(base) PS D:\File\大二秋\DSA\作业4\my-graph> .\test.exe
Active code page: 65001
A -> [D|1] -> [L|1] -> [B|1] -> end
B -> [A|1] -> [F|1] -> [G|1] -> end
C -> [A|1] -> [E|1] -> [K|1] -> [L|1] -> end
D -> [B|1] -> [E|1] -> end
E -> [J|1] -> end
F -> [I|1] -> end
G -> [F|1] -> [H|1] -> end
H -> [C|1] -> [L|1] -> end
I -> [E|1] -> [L|1] -> end
J -> [H|1] -> [K|1] -> end
K -> [B|1] -> [C|1] -> end
L -> [A|1] -> end

=================================
广度优先序列: A D L B E F G J I H K C
广度优先编号 及 深度优先生成森林信息:
标签:   A       B       C       D       E       F       G       H       I       J       K       L
编号:   0       3       11      1       4       5       6       9       8       7       10      2
父节点: null    A       H       A       D       B       B       G       F       E       J       A
=================================
(base) PS D:\File\大二秋\DSA\作业4\my-graph>
```

转换为图形:


A,0
B,3
G,6
D,1
F,5
E,4
C,11
H,9
K,10
J,7
I,8
L,2


A,0
D,1
B,3
G,6
L,2
F,5
E,4
H,9
J,7
I,8
C,11
K,10

分析:

时间复杂度：两部分：访问每个节点花费的时间，在每个节点找邻居花费的时间
对邻接矩阵：
1：V 个节点需要 O(V);
2：由于是邻接矩阵，对于节点 i，需要扫描第 i 行的每一个元素，需要 O(V);
总复杂度 O(V^2)
对邻接表：
1：n 个节点需要 O(V)
2：对邻接表，总共的找邻居时间复杂度就是遍历边表的时间复杂度。对于有向图：O(V)
故总复杂度 O(V+E)
空间复杂度:
无论是邻接矩阵还是邻接表，工作队列的最大长度为 V，故空间复杂度为 O(V)

四、(2) 对于有向图，采用"邻接表"存储结构，设计和实现计算每个顶点入度、出度和度的算法，并分析其时间复杂度

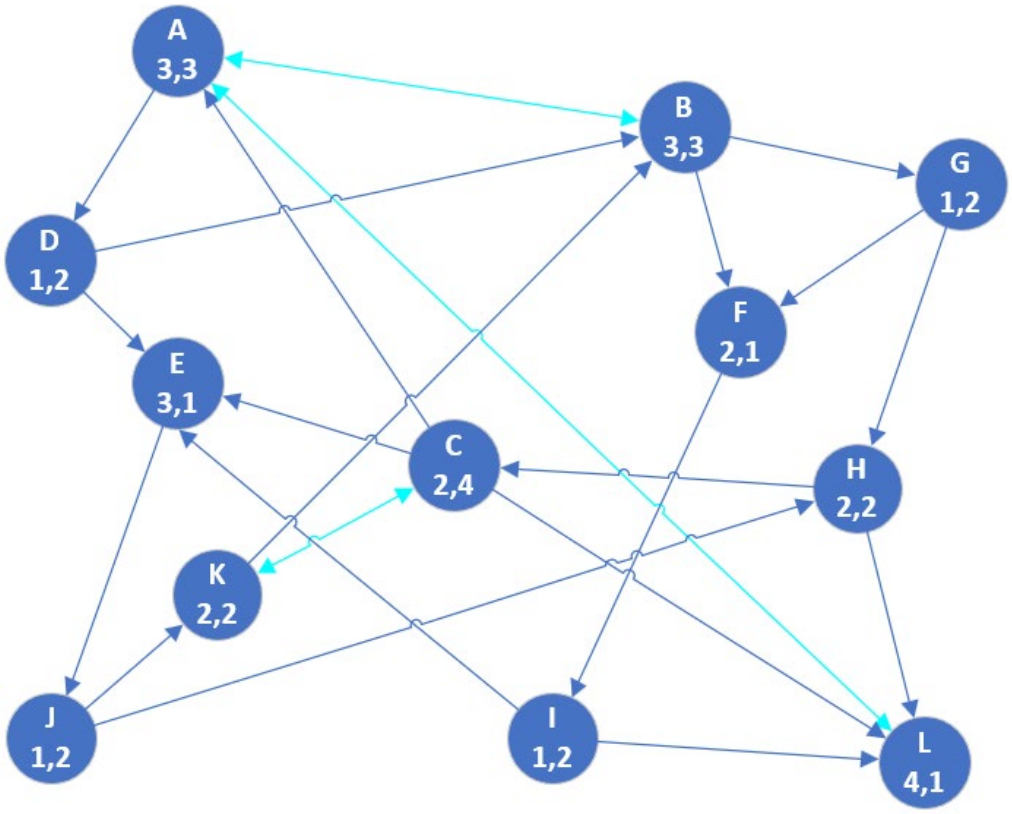
运行程序:

```
#include <iostream>
#include <fstream>

#include ".\src\MyGraph"
using namespace MyGraph;
using namespace std;

int main()
{
    system("chcp 65001");
    ifstream command("./graph_1.txt");

    AdjacencyListGraph G1;
    G1.commandSequence(command);
    G1.showList();
    cout << endl;

    G1.showDegree();
    cout << endl;
}
```

运行结果:

```
(base) PS D:\File\大二秋\DSA\作业4\my-graph> .\test.exe
Active code page: 65001
A -> [D|1] -> [L|1] -> [B|1] -> end
B -> [A|1] -> [F|1] -> [G|1] -> end
C -> [A|1] -> [E|1] -> [K|1] -> [L|1] -> end
D -> [B|1] -> [E|1] -> end
E -> [J|1] -> end
F -> [I|1] -> end
G -> [F|1] -> [H|1] -> end
H -> [C|1] -> [L|1] -> end
I -> [E|1] -> [L|1] -> end
J -> [H|1] -> [K|1] -> end
K -> [B|1] -> [C|1] -> end
L -> [A|1] -> end

标签    A       B       C       D       E       F       G       H       I       J       K       L
入度    3       3       2       1       3       2       1       2       1       1       2       4
出度    3       3       4       2       1       1       2       2       2       2       2       1
度      6       6       6       3       4       3       3       4       3       3       4       5
(base) PS D:\File\大二秋\DSA\作业4\my-graph>
```

绘制在图上:



分析:

时间复杂度:

要想知道一个节点的入度与出度，对于邻接表来说，必须遍历整个边表才能得知一个节点的入度与出度。时间复杂度为 O(E)

当然，求解所有节点的入度与出度的时间复杂度也是 O(E)，需要占用空间 O(V)