本次作业建立的图均为有向图

1 邻接矩阵与邻接表建立的时间空间复杂度

1.1 邻接矩阵

```
//创建邻接矩阵
void CreatMGragh(MTGraph* G){
    cout<<"请输入顶点数："<<endl;
    cin>>G->n;
    cout<<"请输入边数："<<endl;
    cin>>G->e;
    cout<<"请输入顶点编号："<<endl;
    for(int i = 0;i<G->n;i++)
        cin>>G->vertex[i];
    for (int i = 0; i < G->n; ++i)
        for(int j = 0; j < G->n; ++j)
            G->edge[i][j] = 0;//初始化
    cout<<"请输入边和权值："<<endl;
    for(int k = 0;k<G->e;k++){
        int i,j,w;
        cin>>i>>j>>w;
        G->edge[i][j] = w;//w为权值
    }
}
```

1.1.1 时间复杂度

分析代码可知：$T = O(n^2 + n + e)$
又因为有向图，$e \leqslant n(n-1)$
所以，$T = O(n^2)$

1.1.2 空间复杂度

顶点表占用n，边表占用n^2
所以$S = O(n + n^2) = O(n^2)$

1.2 邻接表

```
void CreateGraph(AdjGraph *G){
    cout<<"请输入顶点数："<<endl;
```

```
    cin>>G->n;
    cout<<"请输入边数："<<endl;
    cin>>G->e;
    cout<<"请输入顶点编号："<<endl;
    for (int i = 0; i < G->n; ++i){
        cin>>G->vexlist[i].vertex;
        G->vexlist[i].firstedge = NULL;
    }
    int a,b,c;
    EdgeNode* temp;
    cout<<"请输入边和权值："<<endl;
    for (int i = 0; i < G->e; ++i){
        cin>>a>>b>>c;
        temp = G->vexlist[a].firstedge;//尾插
        G->vexlist[a].firstedge = new EdgeNode;
        G->vexlist[a].firstedge->adjvex = b;
        G->vexlist[a].firstedge->cost = c;
        G->vexlist[a].firstedge->next = temp;
    }
}
```

1.2.1 时间复杂度

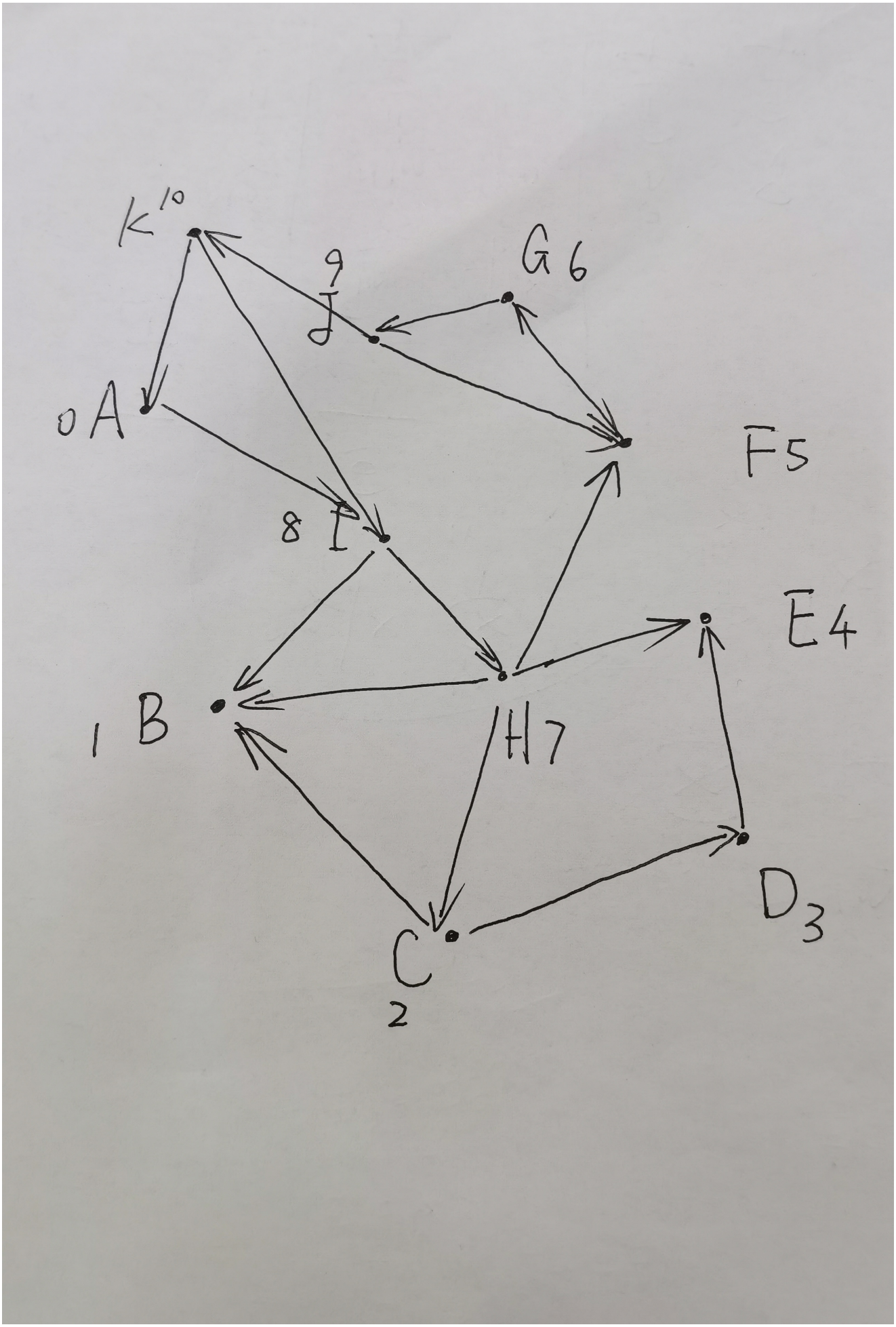
分析代码可知：$T = O(n + e)$

1.2.2 空间复杂度

由于是链式存储，$S = O(n + e)$

2 测试数据与结果数据

2.1 测试数据：



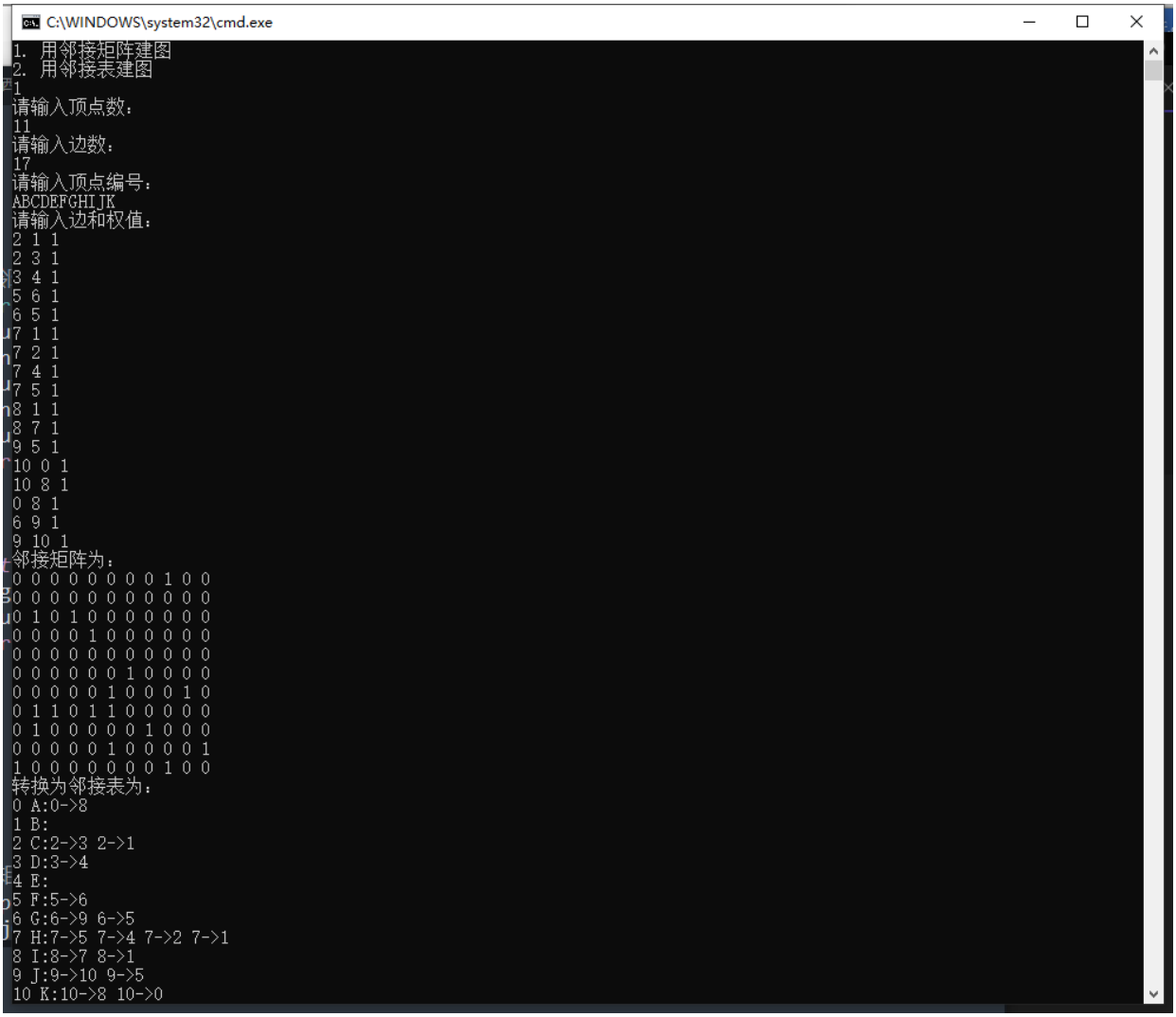
建立如图所示的图

有11个顶点，17条边
由于权重与本次作业各算法无关，输入均为：

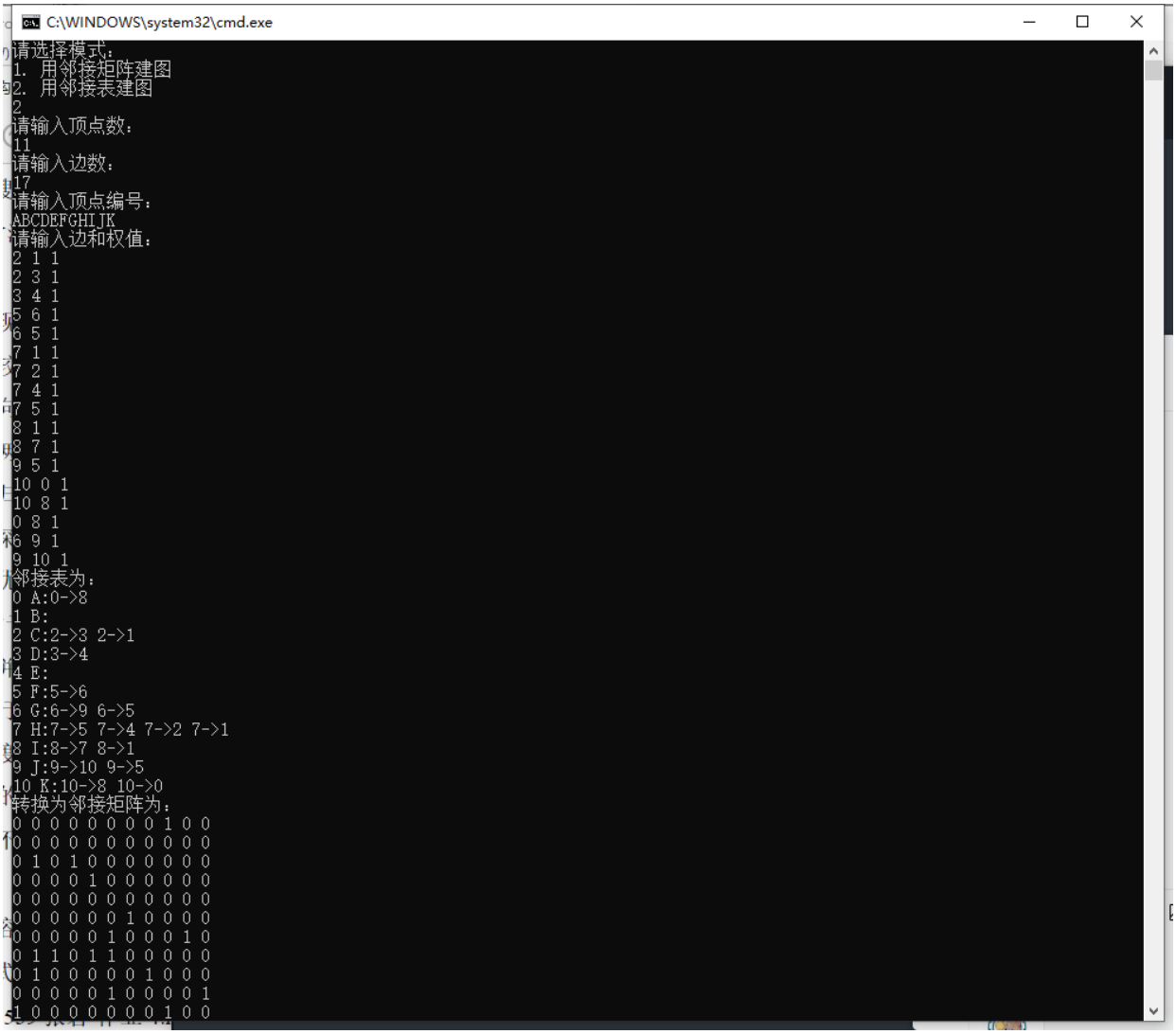
```
2 1 1
2 3 1
3 4 1
5 6 1
6 5 1
7 1 1
7 2 1
7 4 1
7 5 1
8 1 1
8 7 1
9 5 1
10 0 1
10 8 1
0 8 1
6 9 1
9 10 1
```

2.2 相互转化算法

1. 邻接矩阵转邻接表：


C:\WINDOWS\system32\cmd.exe
1. 用邻接矩阵建图
2. 用邻接表建图
1
请输入顶点数：
11
请输入边数：
17
请输入顶点编号：
ABCDEFGHIJK
请输入边和权值：
2 1 1
2 3 1
3 4 1
5 6 1
6 5 1
7 1 1
7 2 1
7 4 1
7 5 1
8 1 1
8 7 1
9 5 1
10 0 1
10 8 1
0 8 1
6 9 1
9 10 1
邻接矩阵为：
0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0
0 1 0 1 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0 0 1 0
0 1 1 0 1 1 0 0 0 0 0
0 1 0 0 0 0 0 1 0 0 0
0 0 0 0 0 1 0 0 0 0 1
1 0 0 0 0 0 0 0 1 0 0
转换为邻接表为：
0 A:0->8
1 B:
2 C:2->3 2->1
3 D:3->4
4 E:
5 F:5->6
6 G:6->9 6->5
7 H:7->5 7->4 7->2 7->1
8 I:8->7 8->1
9 J:9->10 9->5
10 K:10->8 10->0

2. 邻接表转邻接矩阵：



2.3 DFS与BFS

2.3.1 邻接矩阵DFS与BFS结果数据

递归

```
邻接矩阵为：
0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0
0 1 0 1 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0 0 1 0
0 1 1 0 1 1 0 0 0 0 0
0 1 0 0 0 0 0 1 0 0 0
0 0 0 0 0 1 0 0 0 0 1
1 0 0 0 0 0 0 0 1 0 0
转换为邻接表为：
0 A:0->8
1 B:
2 C:2->3 2->1
3 D:3->4
4 E:
5 F:5->6
6 G:6->9 6->5
7 H:7->5 7->4 7->2 7->1
8 I:8->7 8->1
9 J:9->10 9->5
10 K:10->8 10->0
广度优先搜索序列为：
A I B H C E F D G J K
广度优先编号为：
1 3 5 8 6 7 9 4 2 10 11
深度优先搜索序列为：
A I B H C D E F G J K
深度优先编号为：
1 3 5 6 7 8 9 4 2 10 11
```

非递归

```
邻接矩阵为：
0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0
0 1 0 1 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0 0 1 0
0 1 1 0 1 1 0 0 0 0 0
0 1 0 0 0 0 0 1 0 0 0
0 0 0 0 0 1 0 0 0 0 1
1 0 0 0 0 0 0 0 1 0 0
转换为邻接表为：
0 A:0->8
1 B:
2 C:2->3 2->1
3 D:3->4
4 E:
5 F:5->6
6 G:6->9 6->5
7 H:7->5 7->4 7->2 7->1
8 I:8->7 8->1
9 J:9->10 9->5
10 K:10->8 10->0
广度优先搜索序列为：
A I B H C E F D G J K
广度优先编号为：
1 3 5 8 6 7 9 4 2 10 11
深度优先搜索序列为：
A I B H C D E F G J K
深度优先编号为：
1 3 5 6 7 8 9 4 2 10 11
请按任意键继续. . .
```

2.3.2 邻接表DFS与BFS结果数据

递归

```
邻接表为：
0 A:0->8
1 B:
2 C:2->3 2->1
3 D:3->4
4 E:
5 F:5->6
6 G:6->9 6->5
7 H:7->5 7->4 7->2 7->1
8 I:8->7 8->1
9 J:9->10 9->5
10 K:10->8 10->0
转换为邻接矩阵为：
0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0
0 1 0 1 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0 0 1 0
0 1 1 0 1 1 0 0 0 0 0
0 1 0 0 0 0 0 1 0 0 0
0 0 0 0 0 1 0 0 0 0 1
1 0 0 0 0 0 0 0 1 0 0
广度优先搜索序列为：
A I H B F E C G D J K
广度优先编号为：
1 4 7 9 6 5 8 3 2 10 11
深度优先搜索序列为：
A I H F G J K E C D B
深度优先编号为：
1 11 9 10 8 4 5 3 2 6 7
```

非递归

```
邻接表为：
0 A:0->8
1 B:
2 C:2->3 2->1
3 D:3->4
4 E:
5 F:5->6
6 G:6->9 6->5
7 H:7->5 7->4 7->2 7->1
8 I:8->7 8->1
9 J:9->10 9->5
10 K:10->8 10->0
转换为邻接矩阵为：
0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0
0 1 0 1 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0 0 1 0
0 1 1 0 1 1 0 0 0 0 0
0 1 0 0 0 0 0 1 0 0 0
0 0 0 0 0 1 0 0 0 0 1
1 0 0 0 0 0 0 0 1 0 0
广度优先搜索序列为：
A I H B F E C G D J K
广度优先编号为：
1 4 7 9 6 5 8 3 2 10 11
深度优先搜索序列为：
A I H F G J K E C D B
深度优先编号为：
1 11 9 10 8 4 5 3 2 6 7
各顶点入度为：
1 3 1 1 2 3 1 1 2 1 1
各顶点出度为：
1 0 2 1 0 1 2 4 2 2 2
各顶点度为：
2 3 3 2 2 4 3 5 4 3 3
请按任意键继续. . .
```

2.3.1 DFS与BFS时间复杂度

2.3.1.1 邻接矩阵

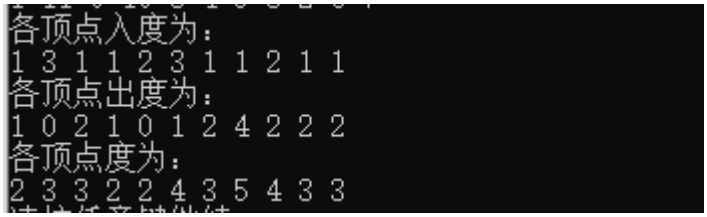
矩阵每个点都要遍历一遍，所以$T = O(n^2)$

2.3.1.2 邻接表

每个节点都要遍历一遍，所以$T = O(n + e)$

2.4 求邻接表入度出度和度算法

2.4.1 结果数据



2.4.2 时间复杂度分析

每个节点都要遍历一遍，所以$T = O(n + e)$