使用前务必将解压后的Graph文件夹放到D盘路径下，即D:\Graph

1. 复杂度分析（设无向图的顶点数为n，边数为e）
2. 存储结构的建立
3. 时间复杂度

建立邻接表的存储结构时，首先需要为每个顶点的信息域赋值，时间复杂度为O(n)；其次，要将各个边表结点插入到与之相关联顶点的邻接表中。为了优化时间，这里采用从表头插入的方式，从而避免了每次插入都需要寻找链表表尾的操作，因此时间复杂度为O(e)。综上，建立邻接表的时间复杂度为O(n+e)。

建立邻接矩阵的存储结构时，首先建立一个大小为n­2的二维数组并将每个元素的值初始化为0（即表示“不邻接”），这一操作的时间复杂度为O(n­2)；其次。要将与每条边关联的两个顶点在矩阵中对应的项置为1（即表示“邻接”），这一步的时间复杂度为O(e)且O(e)的上界为O(n­2)。综上，建立邻接矩阵的时间复杂度为O(n­2)。

1. 空间复杂度

建立邻接表的存储结构时，需要建立一个顶点表保存顶点的信息和指向其第一个边表结点的指针，空间复杂度为O(n­)；每个顶点的边表中存储了与之关联的所有边的信息，整体的空间复杂度为O(e)。综上，邻接表的空间复杂度为O(n­+e)。

建立邻接矩阵的存储结构时，需要建立一个大小为n的数组存储所有顶点的信息，空间复杂度为O(n­)；此外，需要建立一个大小为n­2的二维数组存储n个顶点的邻接矩阵，空间复杂度为O(n­2­)。综上，邻接矩阵的空间复杂度为O(n­2­)。

1. 搜索算法
2. 时间复杂度

对图进行搜索遍历时，需要的时间分为两部分：访问每个顶点所花费的时间和寻找每个顶点的邻接点所花费的时间。

使用邻接表时，不论递归还是非递归，每个顶点都需要被访问一次，其时间复杂度为O(n­)；在寻找每个顶点的邻接点时，边表中的每条边都被访问了2次（因为是无向图），其时间复杂度为O(e)。综上，基于邻接表的DFS、BFS的时间复杂度为O(n­+e)。

使用邻接矩阵时，不论递归还是非递归，每个顶点都需要被访问一次，其时间复杂度为O(n­)；在寻找第i个顶点的邻接点时，需要扫描邻接矩阵第i行的所有值，时间复杂度为O(n­)。综上，基于邻接矩阵的DFS、BFS的时间复杂度为O(n­2)。

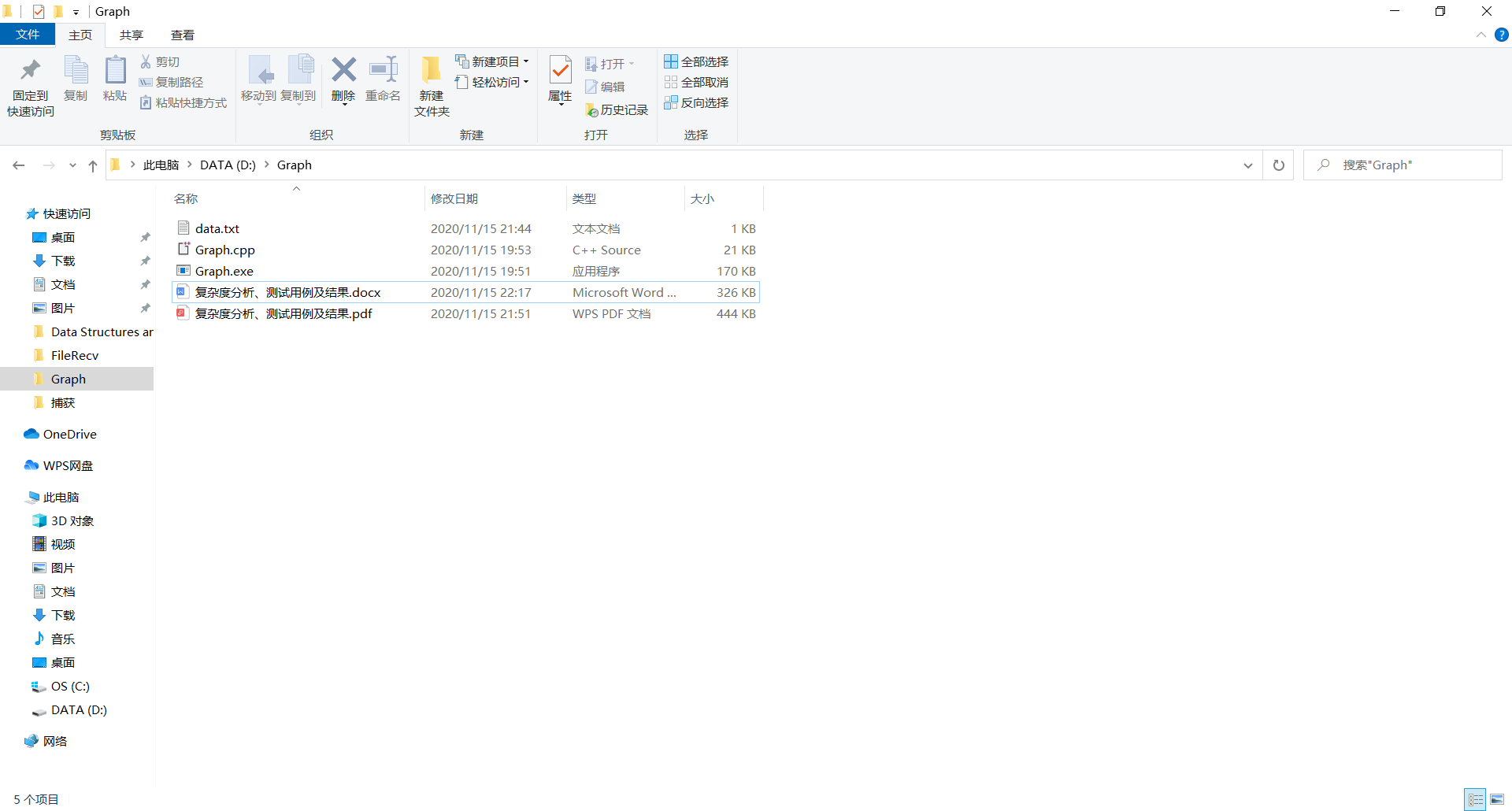
1. 空间复杂度

对图进行BFS或非递归的DFS时，需要为每个顶点声明一个标记是否访问过的标志变量，总的空间复杂度为O(n­)；此外，在进行遍历时，需要使用队列或栈记录访问过的结点，而队列或栈中的元素数量不会超过顶点数n，因此空间复杂度为O(n­)。综上，BFS和非递归的DFS空间复杂度均为O(n­)。

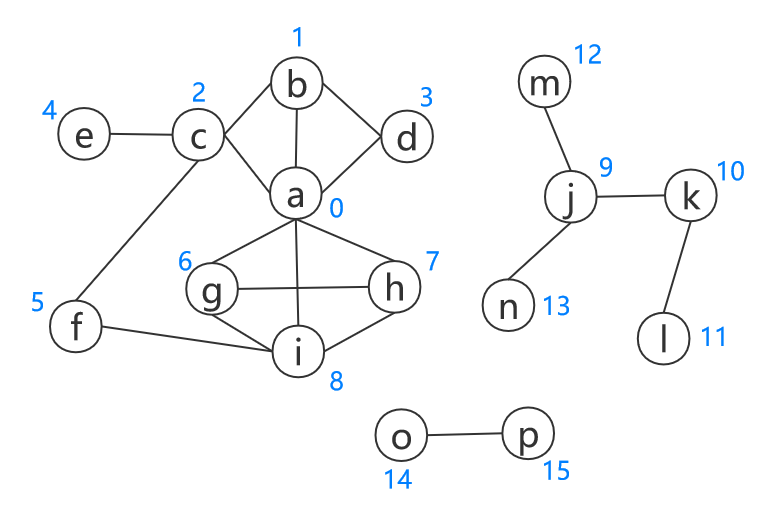
对图进行递归的DFS时，需要为每个顶点声明一个标记是否访问过的标志变量，总的空间复杂度为O(n­)；此外，递归过程需要一个工作栈，其空间复杂度为O(n­)。综上，递归的DFS空间复杂度为O(n­)。

1. 测试用例

使用前务必将解压后的Graph文件夹放到D盘路径下，即D:\Graph



**测试用例1：**

****

顶点数：16

边数：19

每个顶点的信息（只能为一个字符）：a、b、c、d、e、f、g、h、i、j、k、l、m、n、o、p

每条边连接的两个顶点的序号（序号从0开始）：

**[边1]**0 1

**[边2]**0 2

**[边3]**0 3

**[边4]**0 6

**[边5]**0 7

**[边6]**0 8

**[边7]**1 2

**[边8]**1 3

**[边9]**2 4

**[边10]**2 5

**[边11]**5 8

**[边12]**6 7

**[边13]**6 8

**[边14]**7 8

**[边15]**9 10

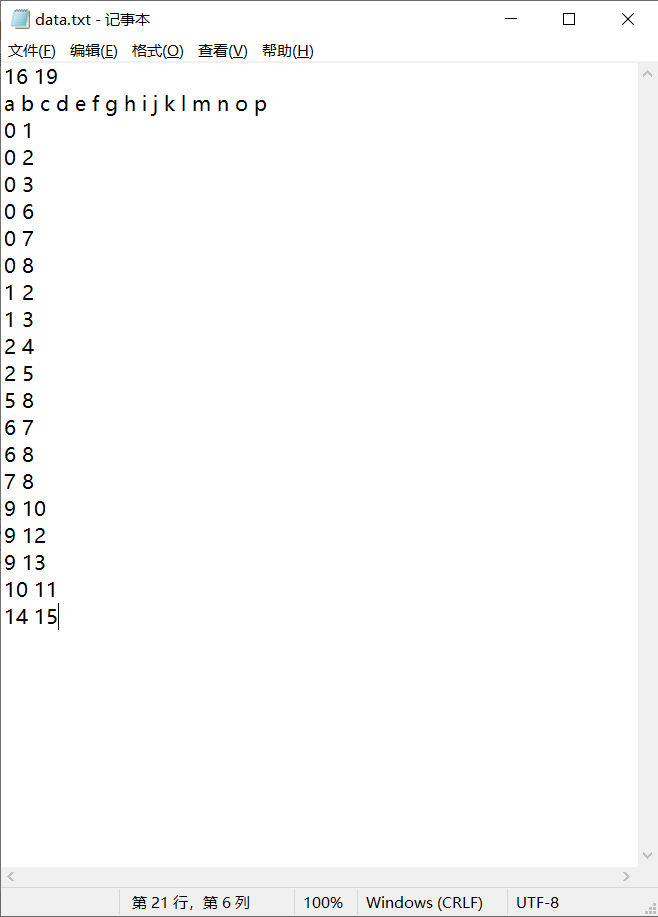
**[边16]**9 12

**[边17]**9 13

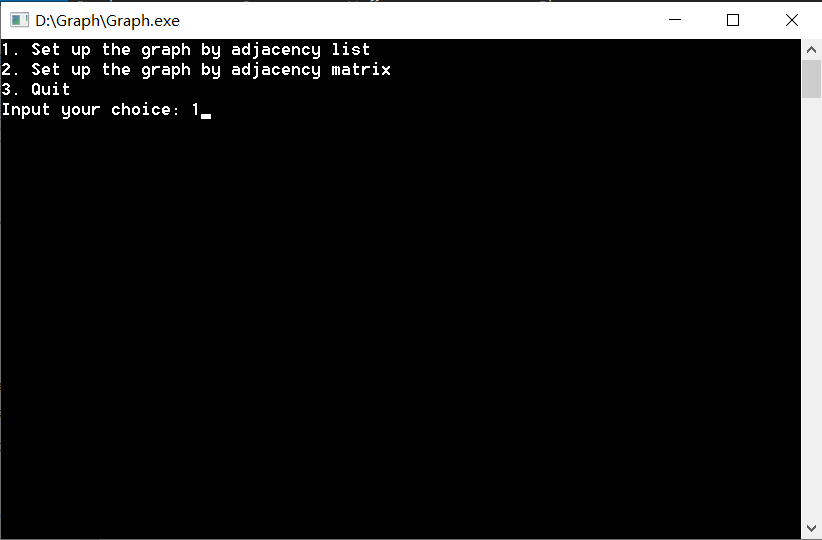
**[边18]**10 11

**[边19]**14 15

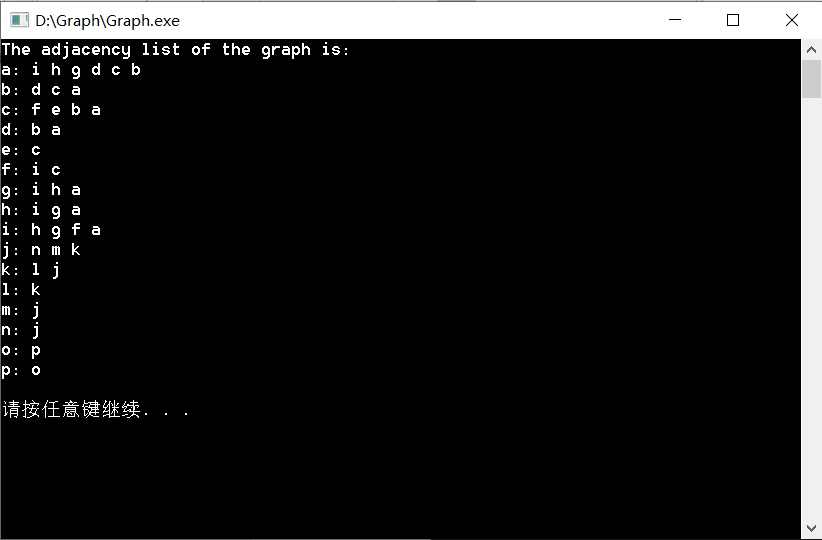
1. 将图中的信息输入到D:\Graph\data.txt中，第一行为顶点数和边数，第二行为每个顶点依次保存的信息（只能为一个字符），下面每行的两个数各表示一条边关联的两个顶点的序号（序号从0开始）。



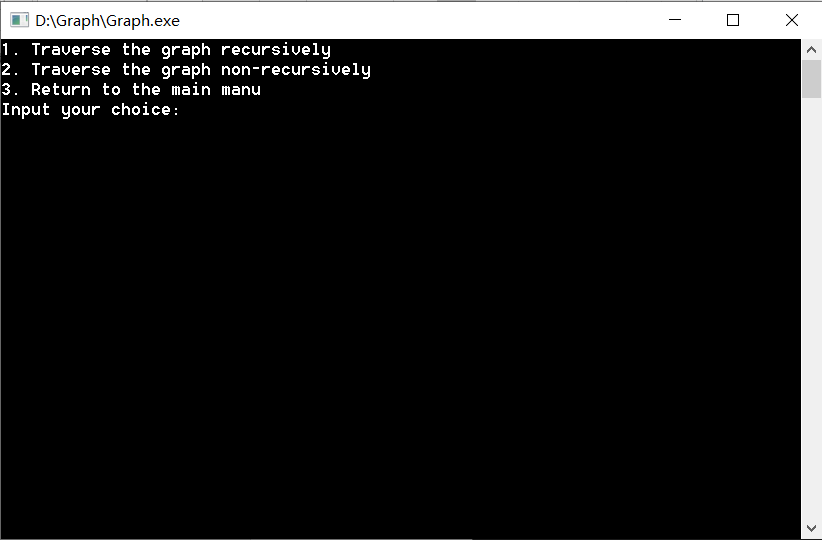
1. 运行D:\Graph\Graph.txt
2. 选择建立邻接表功能

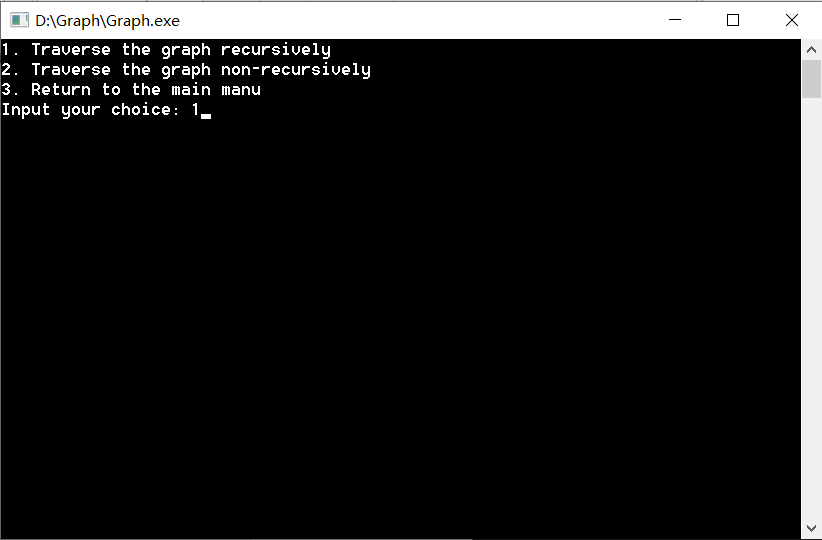


1. 得到图的邻接表

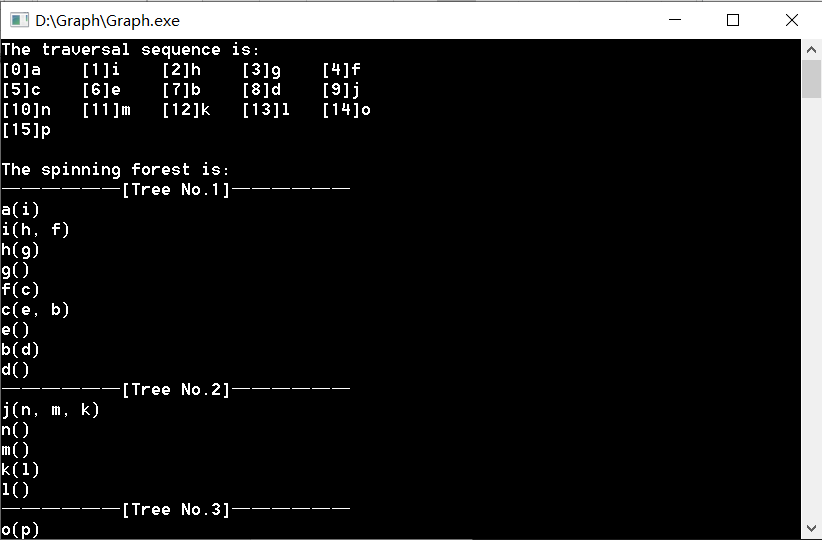


1. 选择DFS功能，然后选择递归遍历功能

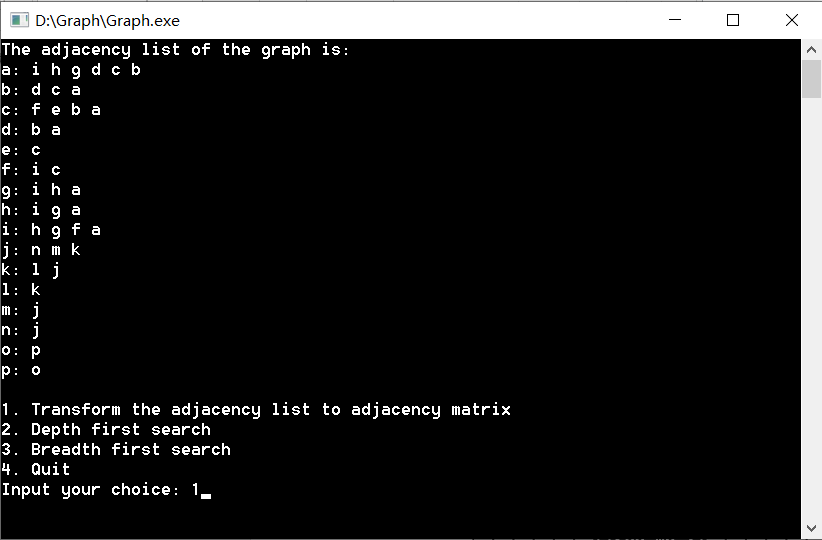




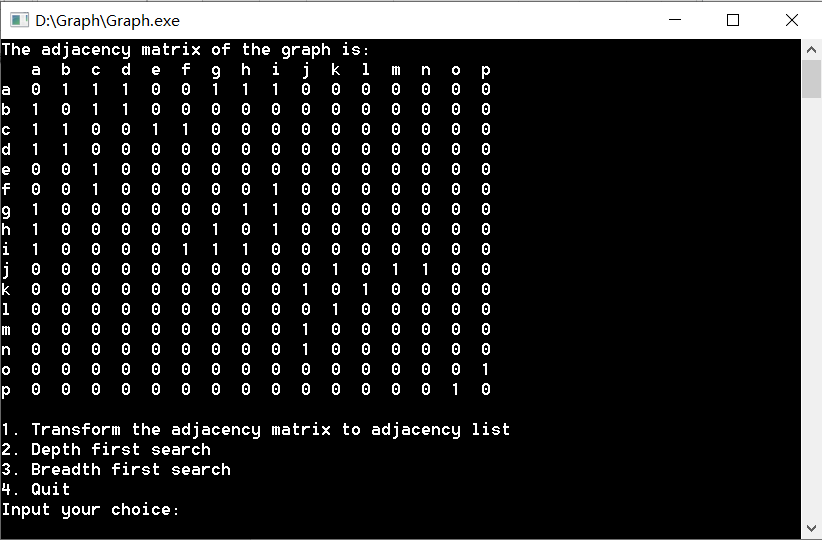
1. 得到遍历序列、编号和DFS生成森林（括号前为树结点的信息，括号中为该树结点各子结点的信息）



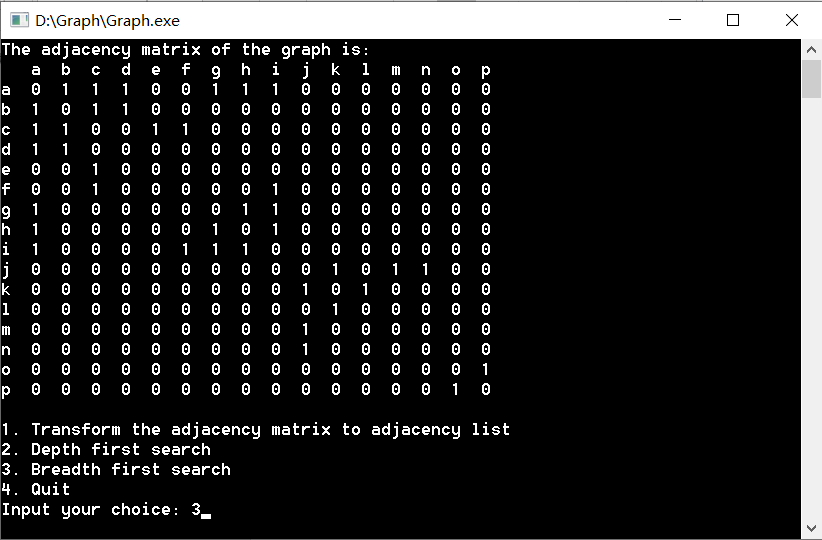
1. 选择邻接表转邻接矩阵功能



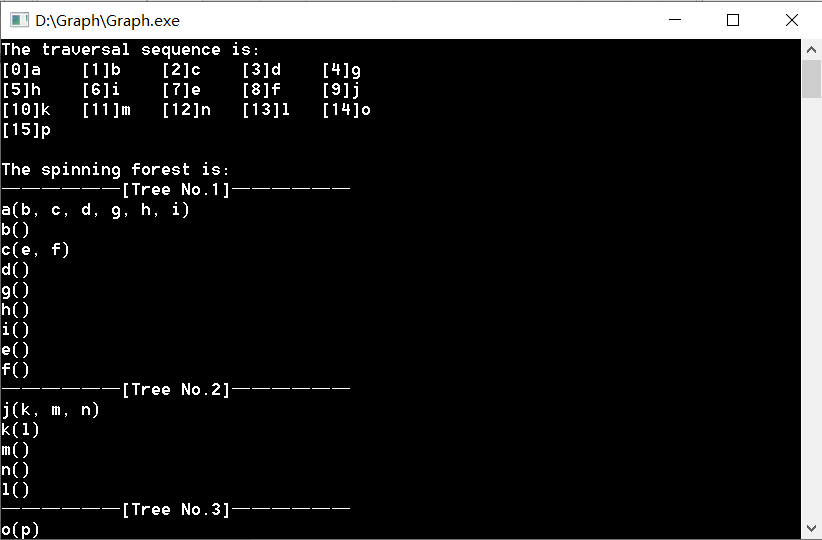
1. 得到图的邻接矩阵



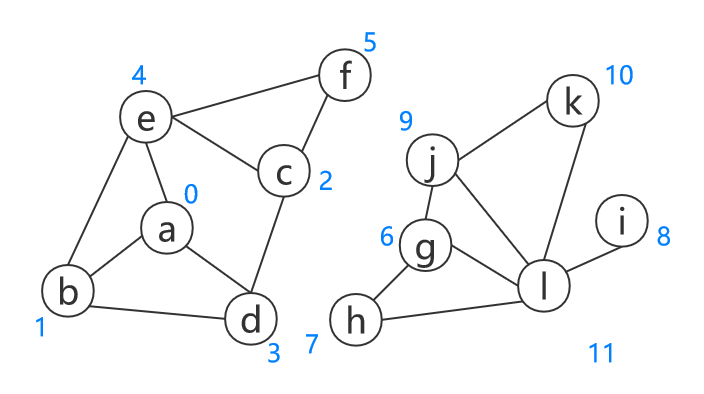
1. 选择BFS功能



1. 得到遍历序列、编号和BFS生成森林（括号前为树结点的信息，括号中为该树结点各子结点的信息）



**测试用例2：**



顶点数：12

边数：17

每个顶点的信息（只能为一个字符）：a、b、c、d、e、f、g、h、i、j、k、l

每条边连接的两个顶点的序号（序号从0开始）：

**[边1]**0 1

**[边2]**0 3

**[边3]**0 4

**[边4]**1 3

**[边5]**1 4

**[边6]**2 3

**[边7]**2 4

**[边8]**2 5

**[边9]**4 5

**[边10]**6 7

**[边11]**6 9

**[边12]**6 11

**[边13]**7 11

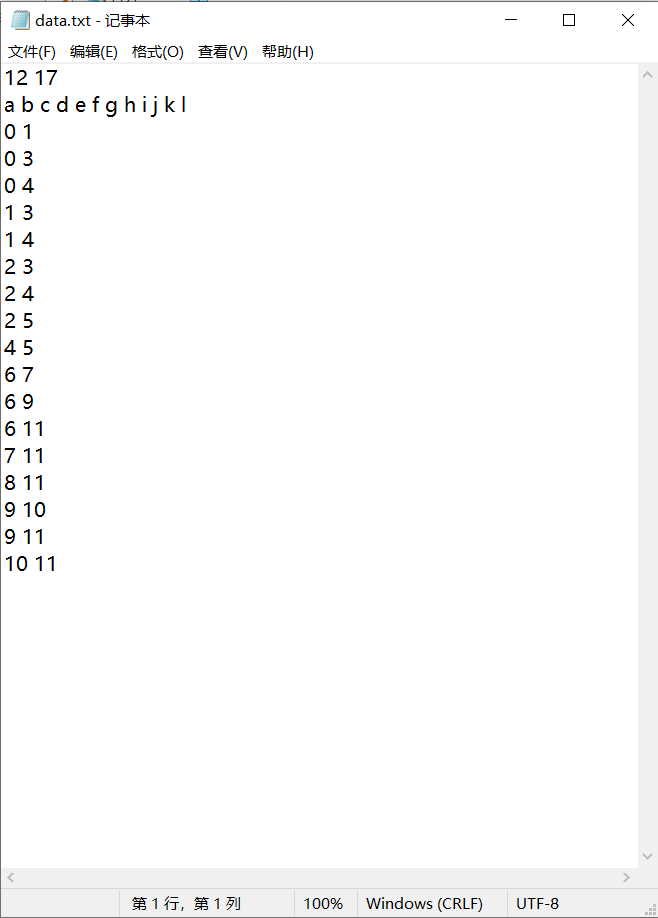
**[边14]**8 11

**[边15]**9 10

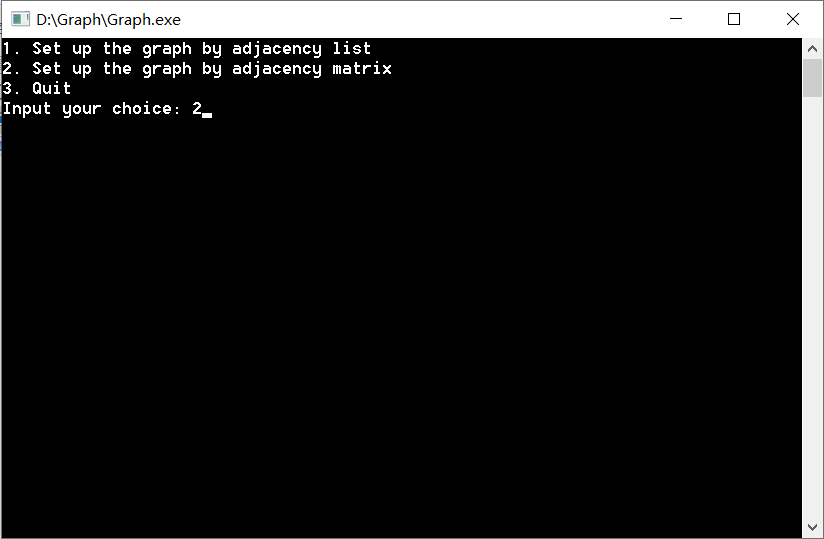
**[边16]**9 11

**[边17]**10 11

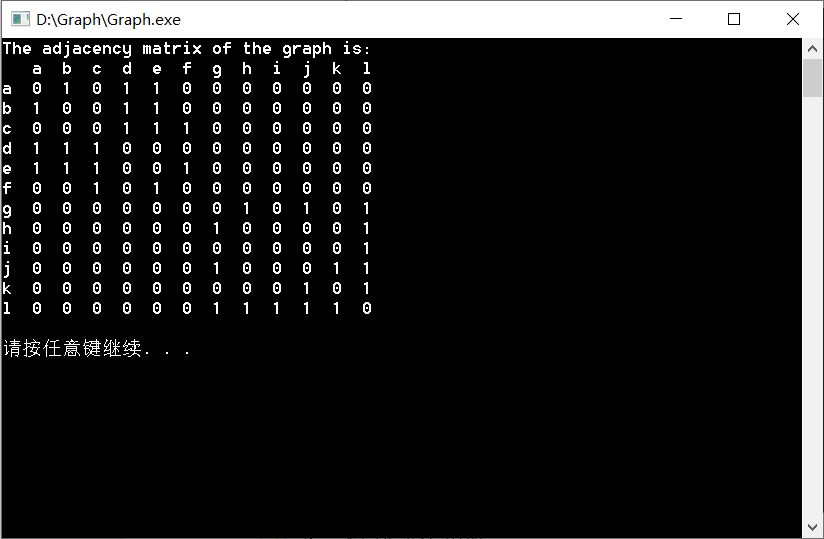
1. 将图中的信息输入到D:\Graph\data.txt中，第一行为顶点数和边数，第二行为每个顶点依次保存的信息（只能为一个字符），下面每行的两个数各表示一条边关联的两个顶点的序号（序号从0开始）。



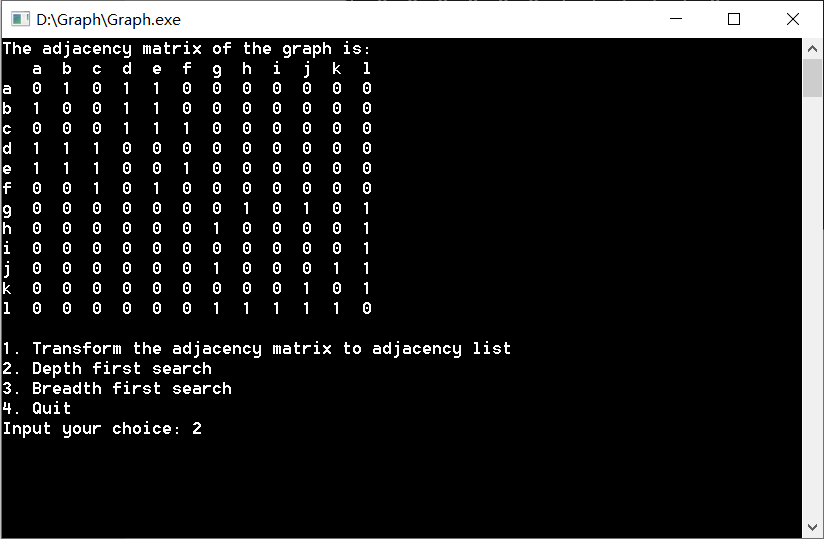
1. 运行D:\Graph\Graph.txt
2. 选择建立邻接表功能

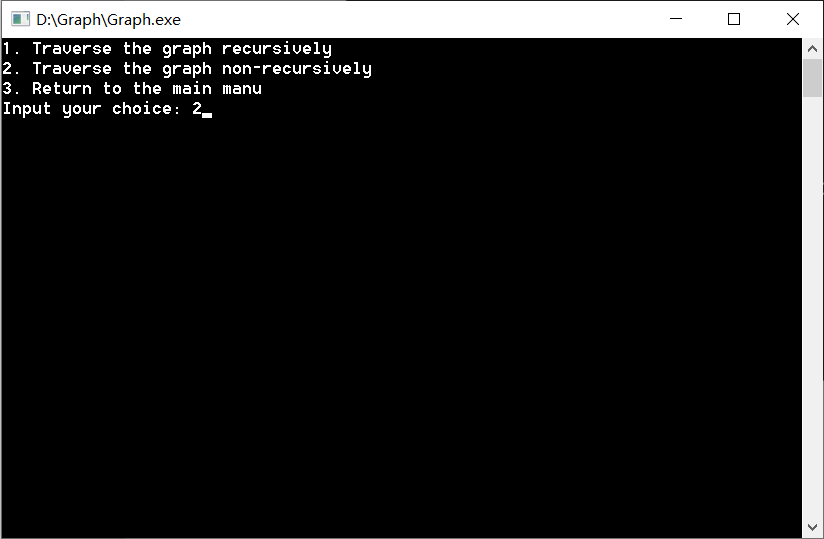


1. 得到邻接矩阵

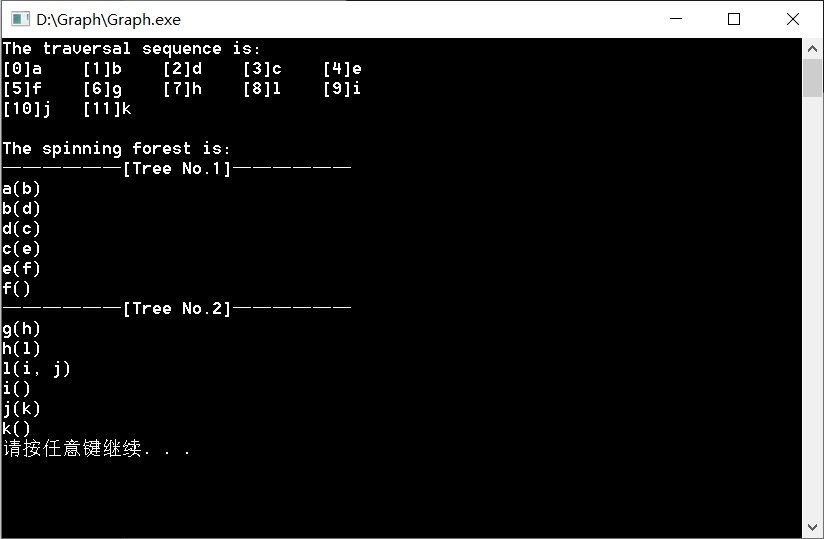


1. 选择DFS功能，然后选择非递归遍历功能

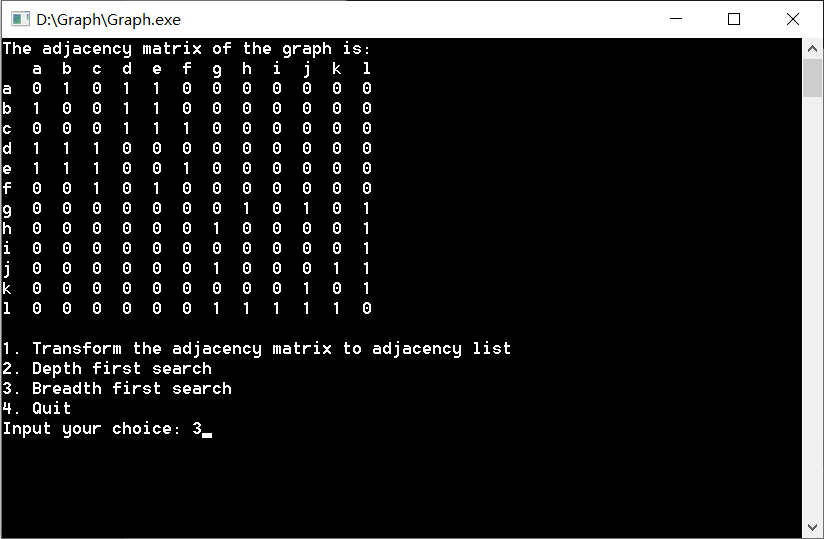




1. 得到遍历序列、编号和DFS生成森林（括号前为树结点的信息，括号中为该树结点各子结点的信息）



1. 选择BFS功能



1. 得到遍历序列、编号和BFS生成森林（括号前为树结点的信息，括号中为该树结点各子结点的信息）

