

使用前务必将解压后的 Graph 文件夹放到 D 盘路径下，即 D:\Graph

## 一、复杂度分析（设无向图的顶点数为 $n$ ，边数为 $e$ ）

### （一）存储结构的建立

#### 1. 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  $n^2$  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  $O(n^2)$ ；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  $O(n^2)$ 。综上，建立邻接矩阵的时间复杂度为  $O(n^2)$ 。

#### 2. 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  $n^2$  的二维数组存储  $n$  个顶点的邻接矩阵，空间复杂度为  $O(n^2)$ 。综上，邻接矩阵的空间复杂度为  $O(n^2)$ 。

### （二）搜索算法

#### 1. 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  $O(n^2)$ 。

#### 2. 空间复杂度

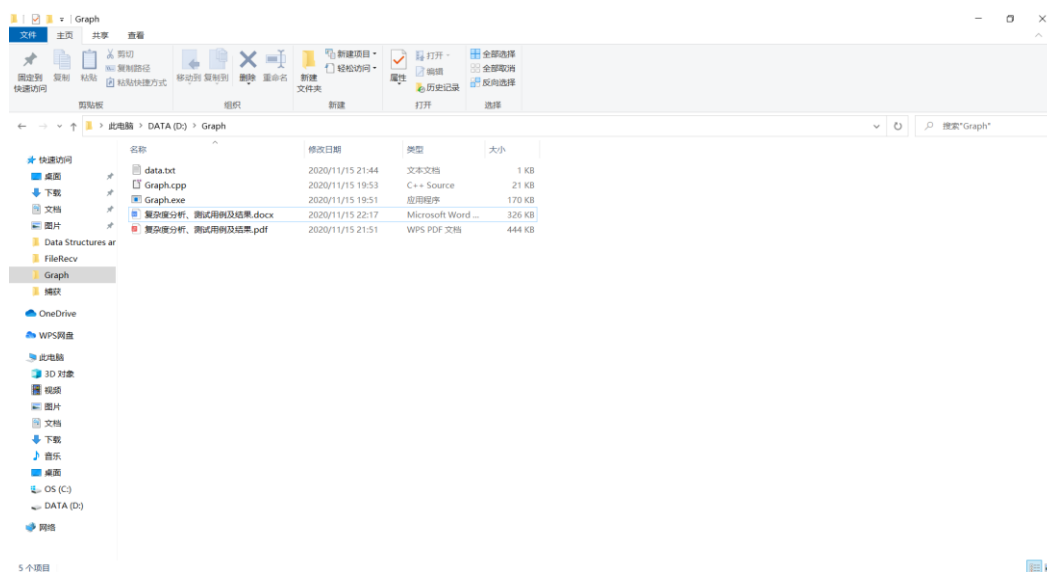
对图进行 BFS 或非递归的 DFS 时，需要为每个顶点声明一个标记是否访问过的标志变量，总的空间复杂度为  $O(n)$ ；此外，在进行遍历时，需要使用队列或栈记录访问过的结点，而队列或栈中的元素数量不会超过顶点数  $n$ ，因此空间复杂度为  $O(n)$ 。综上，BFS 和非递归的 DFS 空间复杂度均为  $O(n)$ 。

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，

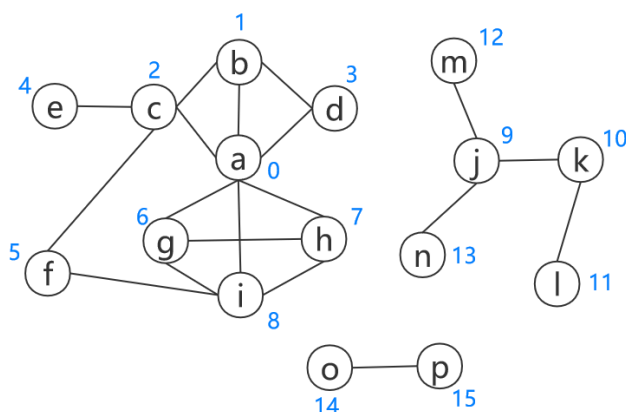
其空间复杂度为  $O(n)$ 。综上，递归的 DFS 空间复杂度为  $O(n)$ 。

## 二、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

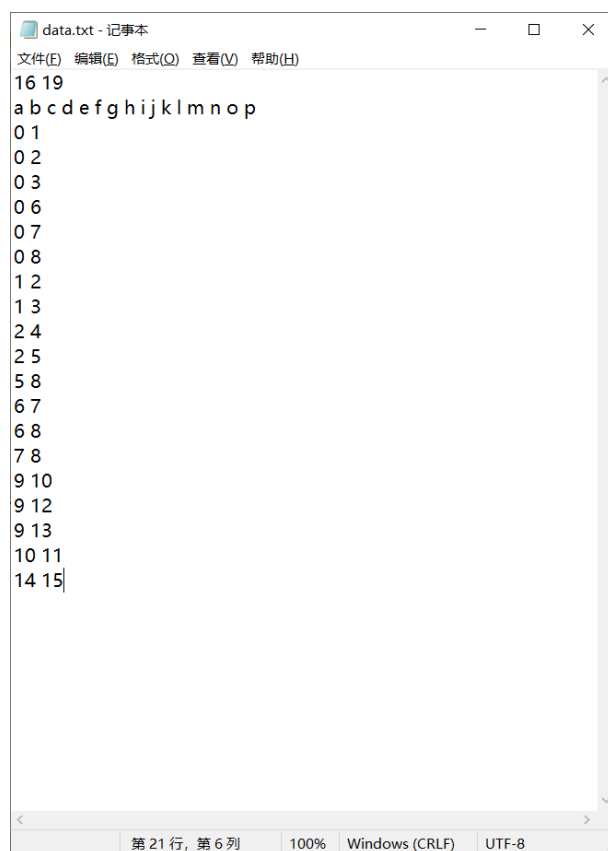
[边 7] 1 2

[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）。



```
data.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
16 19
a b c d e f g h i j k l m n o p
0 1
0 2
0 3
0 6
0 7
0 8
1 2
1 3
2 4
2 5
5 8
6 7
6 8
7 8
9 10
9 12
9 13
10 11
14 15|
第 21 行, 第 6 列 100% Windows (CRLF) UTF-8
```

2. 运行 D:\Graph\Graph.txt
3. 选择建立邻接表功能

```
D:\Graph\Graph.exe
1. Set up the graph by adjacency list
2. Set up the graph by adjacency matrix
3. Quit
Input your choice: 1_
```

#### 4. 得到图的邻接表

```
D:\Graph\Graph.exe
The adjacency list of the graph is:
a: i h g d c b
b: d c a
c: f e b a
d: b a
e: c
f: i c
g: i h a
h: i g a
i: h g f a
j: n m k
k: l j
l: k
m: j
n: j
o: p
p: o
请按任意键继续. . .
```

#### 5. 选择 DFS 功能，然后选择递归遍历功能

```
D:\Graph\Graph.exe
1. Traverse the graph recursively
2. Traverse the graph non-recursively
3. Return to the main menu
Input your choice:
```

```
D:\Graph\Graph.exe
1. Traverse the graph recursively
2. Traverse the graph non-recursively
3. Return to the main menu
Input your choice: 1_
```

6. 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）

```
D:\Graph\Graph.exe
The traversal sequence is:
[0]a [1]i [2]h [3]g [4]f
[5]c [6]e [7]b [8]d [9]j
[10]n [11]m [12]k [13]l [14]o
[15]p

The spinning forest is:
-----[Tree No.1]-----
a(i)
i(h, f)
h(g)
g()
f(c)
c(e, b)
e()
b(d)
d()
-----[Tree No.2]-----
j(n, m, k)
n()
m()
k(l)
l()
-----[Tree No.3]-----
o(p)
```

7. 选择邻接表转邻接矩阵功能

```
D:\Graph\Graph.exe
The adjacency list of the graph is:
a: i h g d c b
b: d c a
c: f e b a
d: b a
e: c
f: i c
g: i h a
h: i g a
i: h g f a
j: n m k
k: l j
l: k
m: j
n: j
o: p
p: o

1. Transform the adjacency list to adjacency matrix
2. Depth first search
3. Breadth first search
4. Quit
Input your choice: 1_
```

8. 得到图的邻接矩阵

```
D:\Graph\Graph.exe
The adjacency matrix of the graph is:
a b c d e f g h i j k l m n o p
a 0 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0
b 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
c 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0
d 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
e 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
f 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0
g 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0
h 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0
i 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0
j 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0
k 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0
l 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
m 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
n 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
o 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
p 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0

1. Transform the adjacency matrix to adjacency list
2. Depth first search
3. Breadth first search
4. Quit
Input your choice:
```

9. 选择 BFS 功能

```
D:\Graph\Graph.exe
The adjacency matrix of the graph is:
a b c d e f g h i j k l m n o p
a 0 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0
b 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
c 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0
d 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
e 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
f 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0
g 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0
h 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0
i 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0
j 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0
k 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0
l 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
m 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
n 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
o 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
p 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0

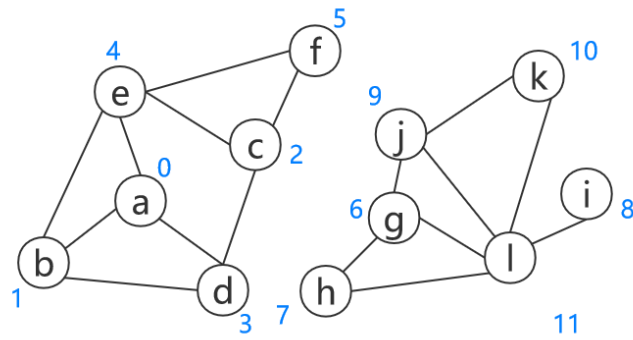
1. Transform the adjacency matrix to adjacency list
2. Depth first search
3. Breadth first search
4. Quit
Input your choice: 3_
```

10. 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）

```
D:\Graph\Graph.exe
The traversal sequence is:
[0]a [1]b [2]c [3]d [4]g
[5]h [6]i [7]e [8]f [9]j
[10]k [11]m [12]n [13]l [14]o
[15]p

The spinning forest is:
-----[Tree No.1]-----
a(b, c, d, g, h, i)
b()
c(e, f)
d()
g()
h()
i()
e()
f()
-----[Tree No.2]-----
j(k, m, n)
k(l)
m()
n()
l()
-----[Tree No.3]-----
o(p)
```

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

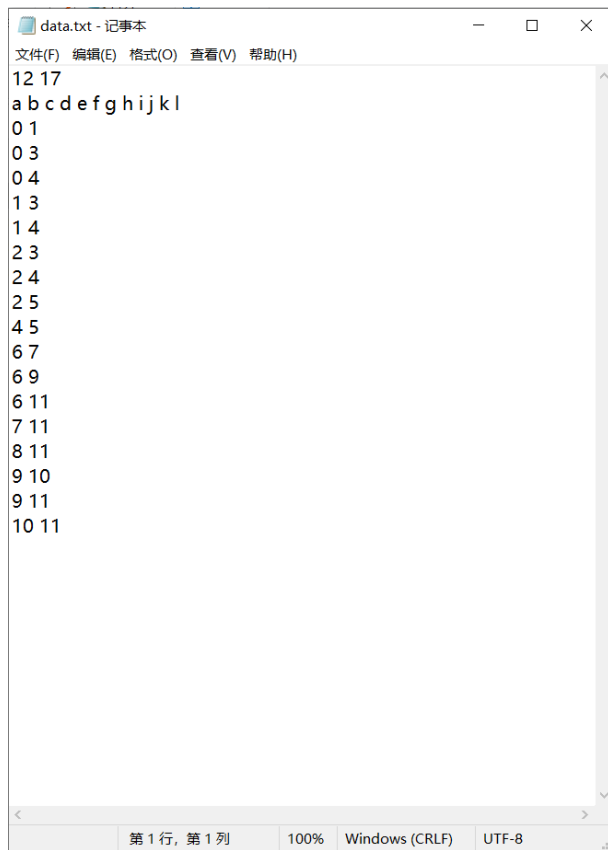
[边 14]8 11

[边 15]9 10

[边 16]9 11

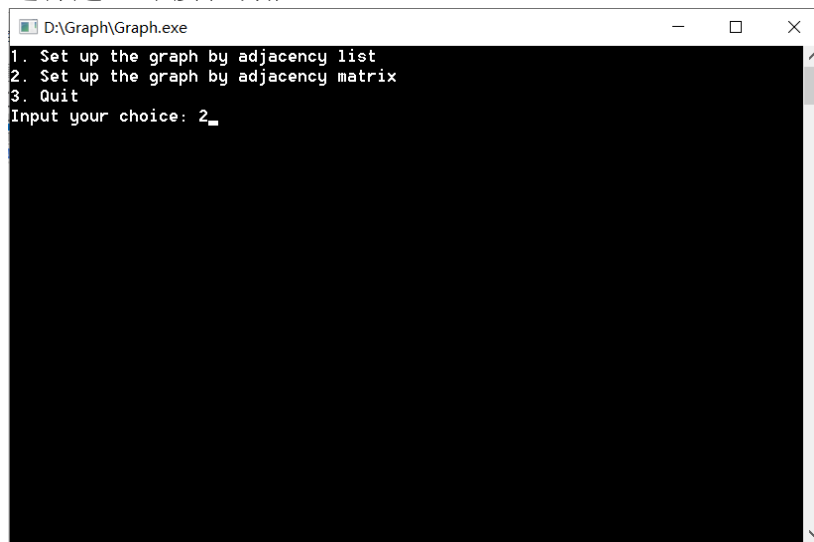
[边 17]10 11

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）。



```
data.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
12 17
a b c d e f g h i j k l
0 1
0 3
0 4
1 3
1 4
2 3
2 4
2 5
4 5
6 7
6 9
6 11
7 11
8 11
9 10
9 11
10 11
第 1 行, 第 1 列 100% Windows (CRLF) UTF-8
```

2. 运行 D:\Graph\Graph.txt
3. 选择建立邻接表功能



```
D:\Graph\Graph.exe
1. Set up the graph by adjacency list
2. Set up the graph by adjacency matrix
3. Quit
Input your choice: 2_
```

4. 得到邻接矩阵



```
D:\Graph\Graph.exe
The adjacency matrix of the graph is:
a b c d e f g h i j k l
a 0 1 0 1 1 0 0 0 0 0 0 0
b 1 0 0 1 1 0 0 0 0 0 0 0
c 0 0 0 1 1 1 0 0 0 0 0 0
d 1 1 1 0 0 0 0 0 0 0 0 0
e 1 1 1 0 0 1 0 0 0 0 0 0
f 0 0 1 0 1 0 0 0 0 0 0 0
g 0 0 0 0 0 0 0 1 0 1 0 1
h 0 0 0 0 0 0 1 0 0 0 0 1
i 0 0 0 0 0 0 0 0 0 0 0 1
j 0 0 0 0 0 0 1 0 0 0 1 1
k 0 0 0 0 0 0 0 0 0 1 0 1
l 0 0 0 0 0 0 1 1 1 1 1 0

请按任意键继续...
```

5. 选择 DFS 功能，然后选择非递归遍历功能

```
D:\Graph\Graph.exe
The adjacency matrix of the graph is:
a b c d e f g h i j k l
a 0 1 0 1 1 0 0 0 0 0 0 0
b 1 0 0 1 1 0 0 0 0 0 0 0
c 0 0 0 1 1 1 0 0 0 0 0 0
d 1 1 1 0 0 0 0 0 0 0 0 0
e 1 1 1 0 0 1 0 0 0 0 0 0
f 0 0 1 0 1 0 0 0 0 0 0 0
g 0 0 0 0 0 0 0 1 0 1 0 1
h 0 0 0 0 0 0 1 0 0 0 0 1
i 0 0 0 0 0 0 0 0 0 0 0 1
j 0 0 0 0 0 0 1 0 0 0 1 1
k 0 0 0 0 0 0 0 0 0 1 0 1
l 0 0 0 0 0 0 1 1 1 1 1 0

1. Transform the adjacency matrix to adjacency list
2. Depth first search
3. Breadth first search
4. Quit
Input your choice: 2
```

```
D:\Graph\Graph.exe
1. Traverse the graph recursively
2. Traverse the graph non-recursively
3. Return to the main menu
Input your choice: 2_
```

6. 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）

```
D:\Graph\Graph.exe
The traversal sequence is:
[0]a [1]b [2]d [3]c [4]e
[5]f [6]g [7]h [8]l [9]i
[10]j [11]k

The spinning forest is:
-----[Tree No.1]-----
a(b)
b(d)
d(c)
c(e)
e(f)
f()
-----[Tree No.2]-----
g(h)
h(l)
l(i, j)
i()
j(k)
k()
请按任意键继续. . .
```

## 7. 选择 BFS 功能

```
D:\Graph\Graph.exe
The adjacency matrix of the graph is:
a b c d e f g h i j k l
a 0 1 0 1 1 0 0 0 0 0 0 0
b 1 0 0 1 1 0 0 0 0 0 0 0
c 0 0 0 1 1 1 0 0 0 0 0 0
d 1 1 1 0 0 0 0 0 0 0 0 0
e 1 1 1 0 0 1 0 0 0 0 0 0
f 0 0 1 0 1 0 0 0 0 0 0 0
g 0 0 0 0 0 0 0 1 0 1 0 1
h 0 0 0 0 0 0 1 0 0 0 0 1
i 0 0 0 0 0 0 0 0 0 0 0 1
j 0 0 0 0 0 0 1 0 0 0 1 1
k 0 0 0 0 0 0 0 0 0 1 0 1
l 0 0 0 0 0 0 1 1 1 1 1 0

1. Transform the adjacency matrix to adjacency list
2. Depth first search
3. Breadth first search
4. Quit
Input your choice: 3_
```

## 8. 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）

```
D:\Graph\Graph.exe
The traversal sequence is:
[0]a [1]b [2]d [3]e [4]c
[5]f [6]g [7]h [8]j [9]l
[10]k [11]i

The spinning forest is:
-----[Tree No.1]-----
a(b, d, e)
b()
d(c)
e(f)
c()
f()
-----[Tree No.2]-----
g(h, j, l)
h()
j(k)
l(i)
k()
i()
请按任意键继续. . .
```