白盒测试技术
——基本路径法

黄轶文

目 录



课堂测试



- 说出逻辑覆盖测试包含的六种基本类型，并说出它们的基本测试要求。

➢例：

1、语句覆盖：使程序中的***语句至少测试一次。

基本路径测试



本次课将会接触到的新知识：

1.控制流图

2.环形复杂度

3.独立路径

基本路径测试

- 路径测试就是从一个程序的入口开始，执行所经历的各个语句的完整过程。从广义的角度讲，任何有关路径分析的测试都可以被称为路径测试。

- 完成路径测试的理想情况是做到路径覆盖，但对于复杂性大的程序要做到所有路径覆盖是不可能的。

基本路径测试

- 在不能做到所有路径覆盖的前提下，如果某一程序的每一个独立路径都被测试过，那么可以认为程序中的每个语句都已经检验过了，即达到了语句覆盖。这种测试方法就是通常所说的基本路径测试法。

基本路径测试

- 基本路径测试方法是在控制流图的基础上，通过分析控制结构的环形复杂度，导出执行路径的基本集，再从该基本集设计测试用例。基本路径测试方法包括4个步骤：

基本路径测试

（1）画出程序的控制流图。

（2）计算程序的环形复杂度，导出程序基本路径集中的独立路径条数，这是确定程序中每个可执行语句至少执行一次所必须的测试用例数目的上界。

基本路径测试

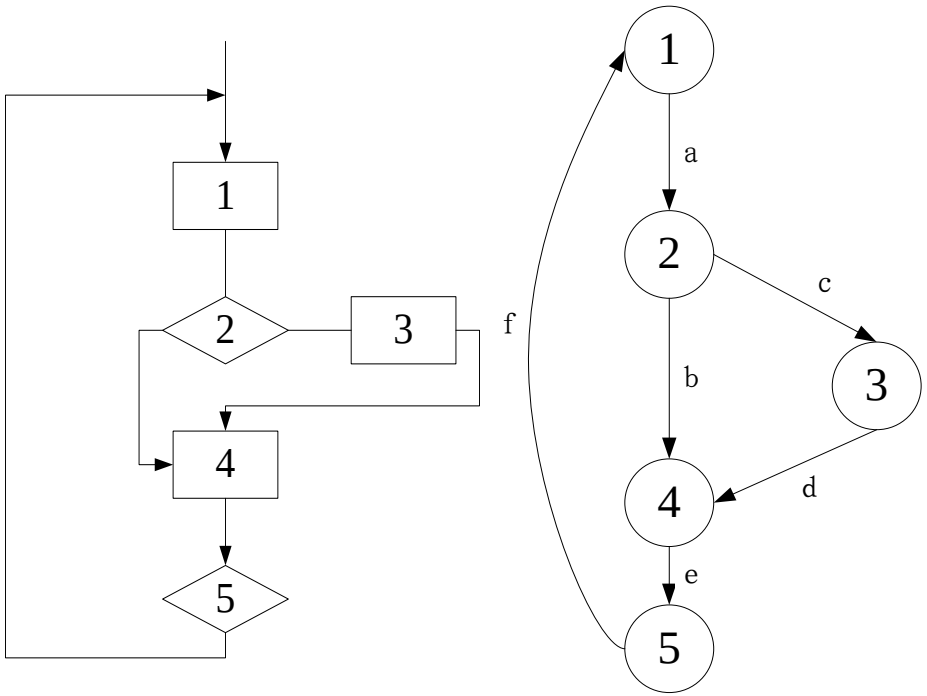
（3）导出基本路径集，确定程序的独立路径。

（4）根据(3)中的独立路径，设计测试用例的输入数据和预期输出。

控制流图

- 程序流程图又称框图，是我们最熟悉，也是最容易理解的一种程序控制结构的图形表示了。在这种图上的框里面常常标明了处理要求或者条件，但是，这些标注在做路径分析时是不重要的。为了**更加突出控制流的结构，需要对程序流程图做一些简化——称作“控制流图”**。

1. 程序的控制流图



- 在控制流图中只有两种图形符号，它们是：

1. 节点：以标有编号的圆圈表示。
2. 边——控制流线或弧：以箭头表示。

1. 程序的控制流图

✓节点

1、标有编号的圆圈
2、程序流程图中矩形框所表示的处理
3、菱形表示的两个甚至多个出口判断
4、多条流线相交的汇合点

可代表一个或多个语句、一个处理框序列和一个条件判定框（假设不包含复合条件）

在选择或多分支结构中，分支的汇聚处应有一个汇聚节点。

1. 程序的控制流图

✓**边——控制流线或弧**

1、箭头

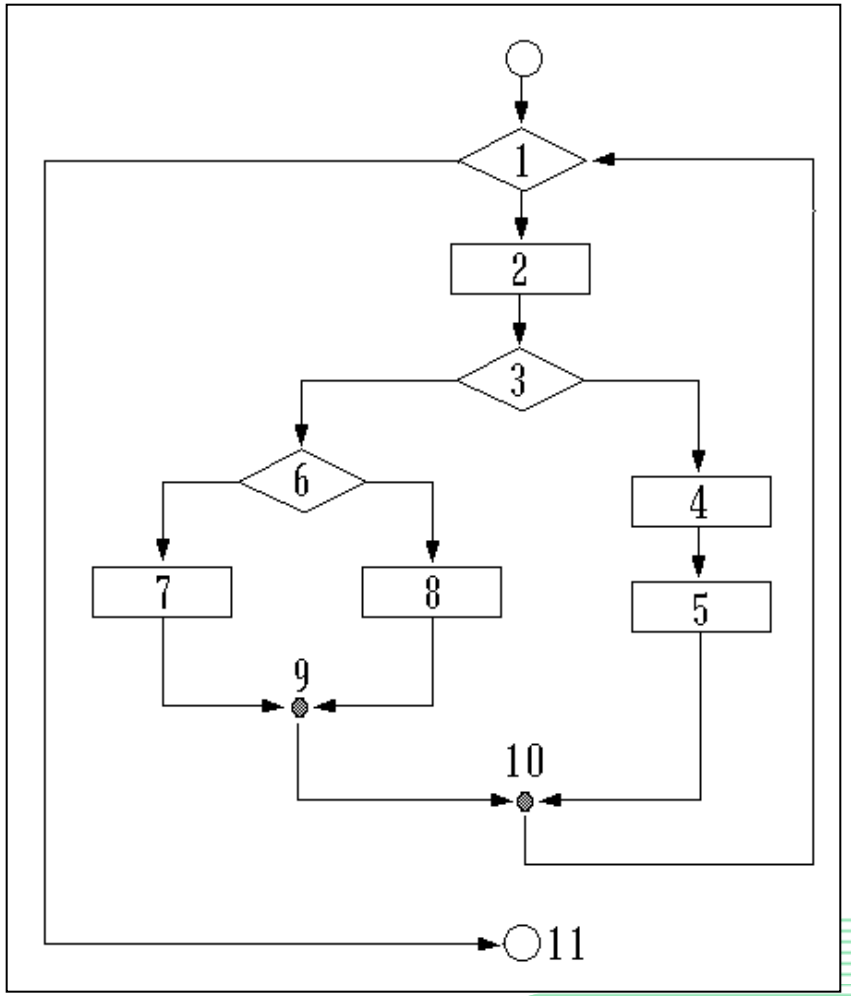
2、与程序流程图中的流线一致，表明了控制的顺序

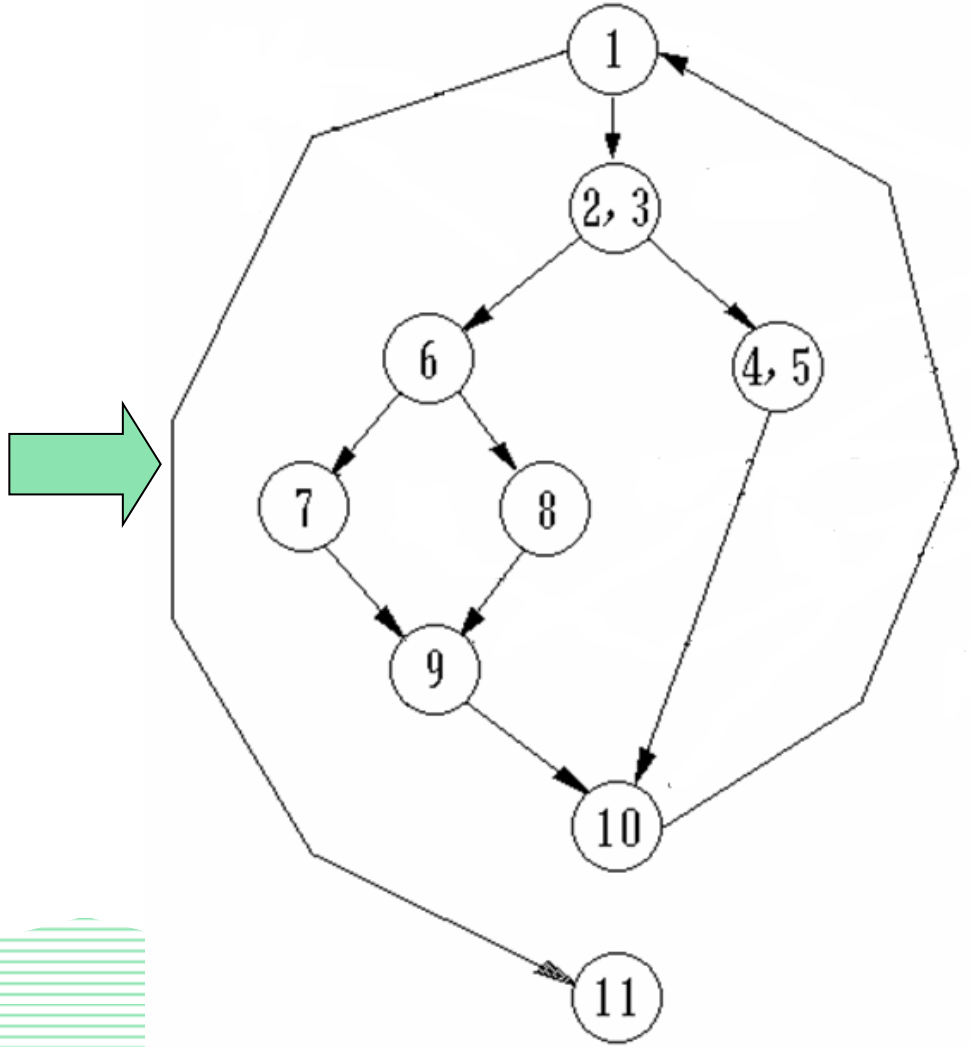
3、控制流线通常标有名字

1. 程序的控制流图

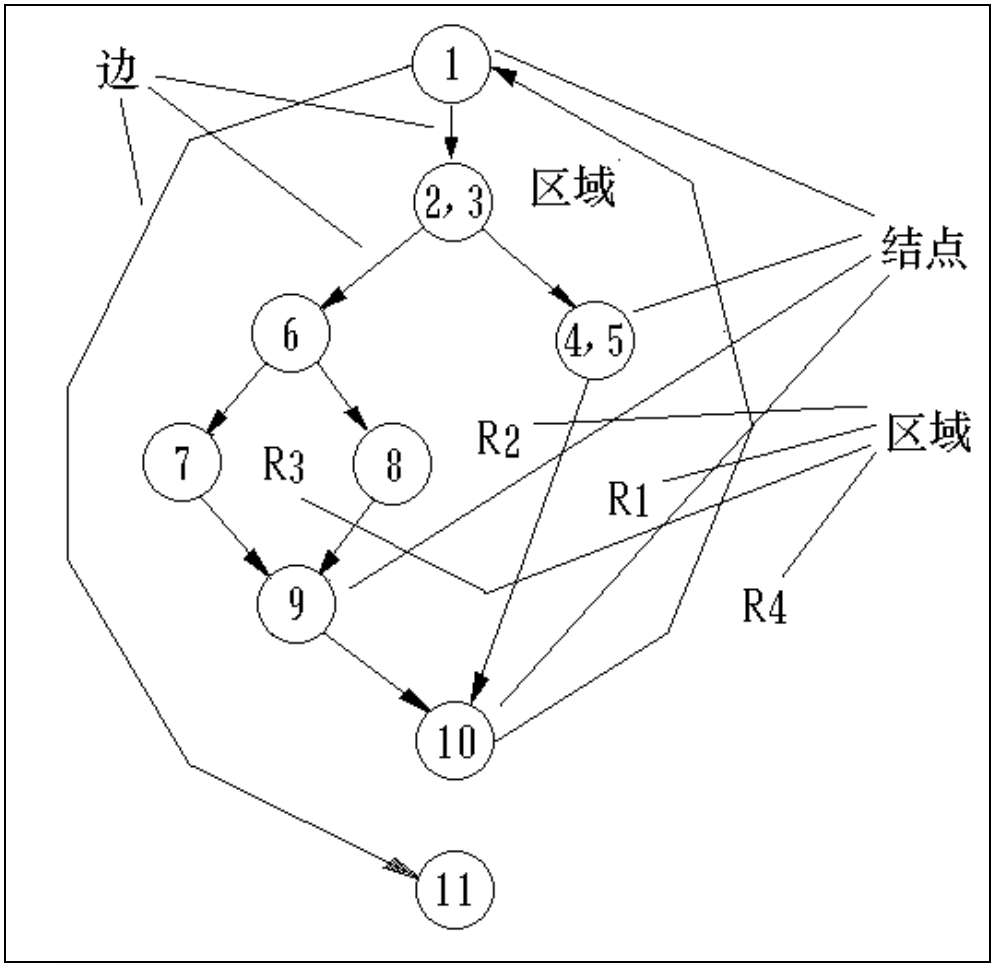
- **边和节点围成的范围叫做区域**，当对区域计数时，图形外的区域也应记为一个区域。

1. 程序的控制流图

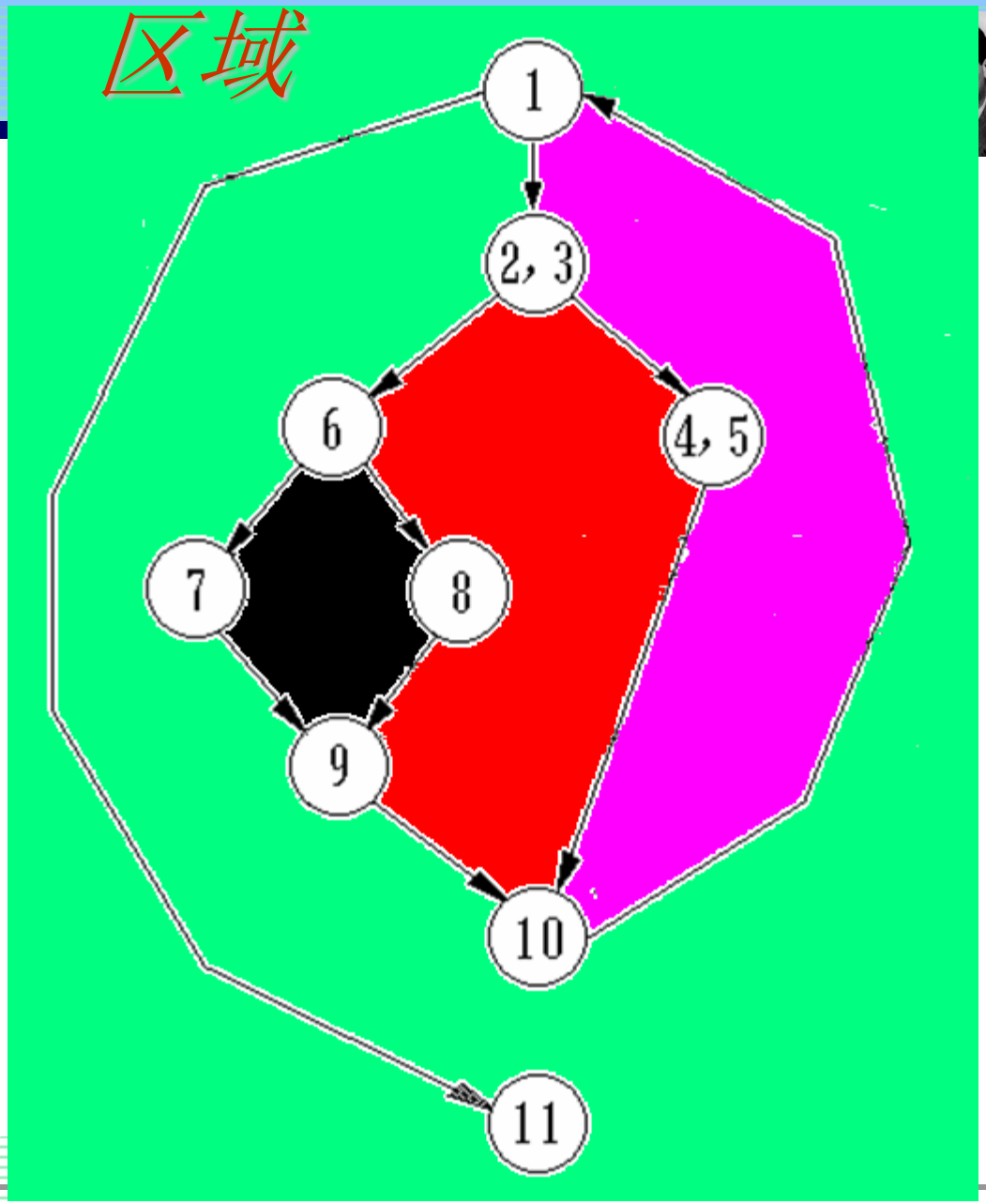


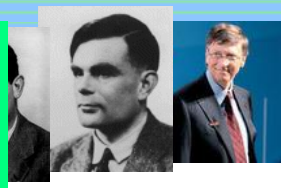


1. 程序的控制流图



区域





1. 程序的控制流图

常见结构的程序流程图

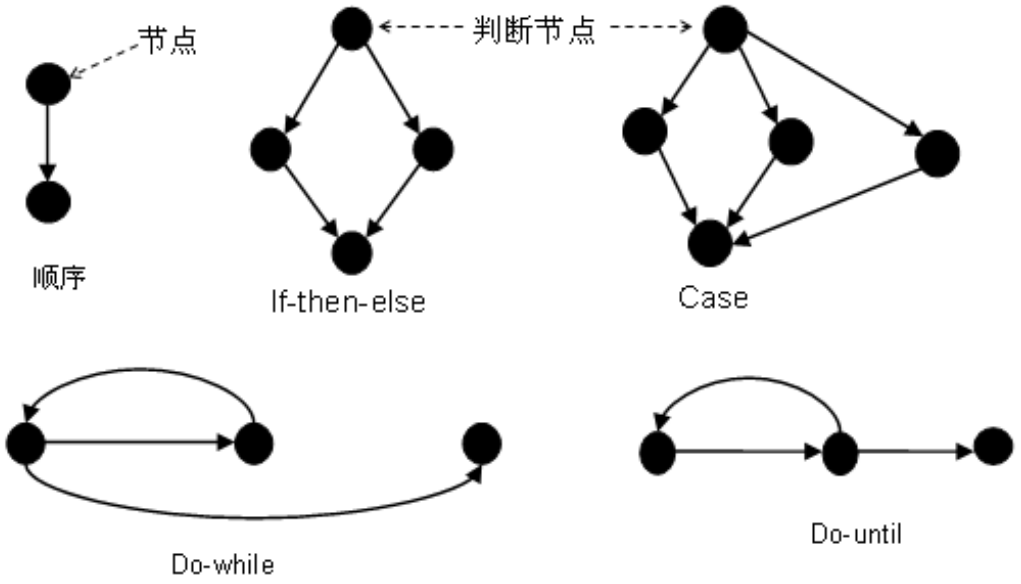


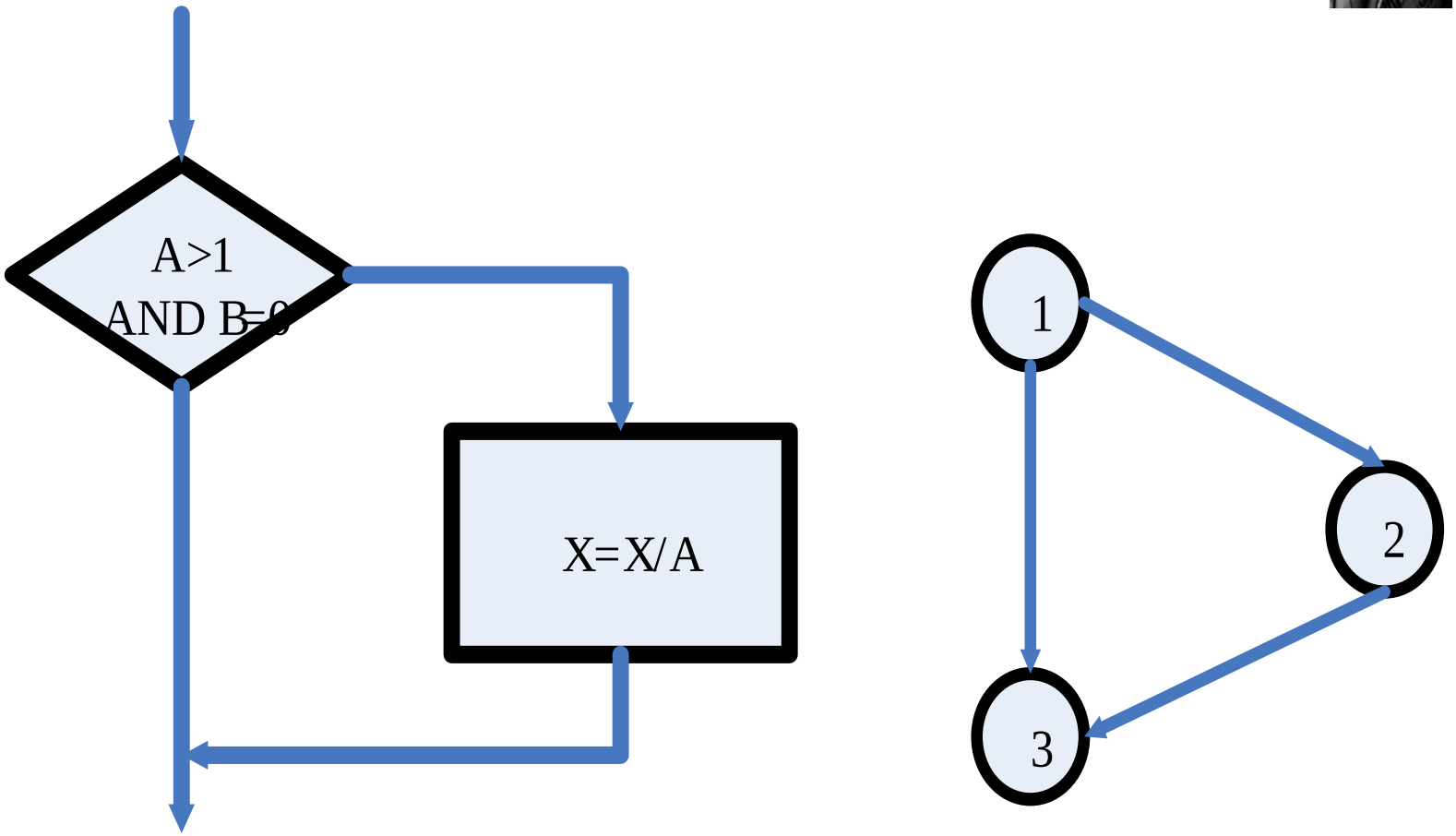
图 5-7 程序流程的基本图元

其中，包含条件的节点被称为**判定节点**（也叫谓词节点），由判定节点发出的边必须终止于某一个节点，由边和节点所限定的范围被称为**区域**。

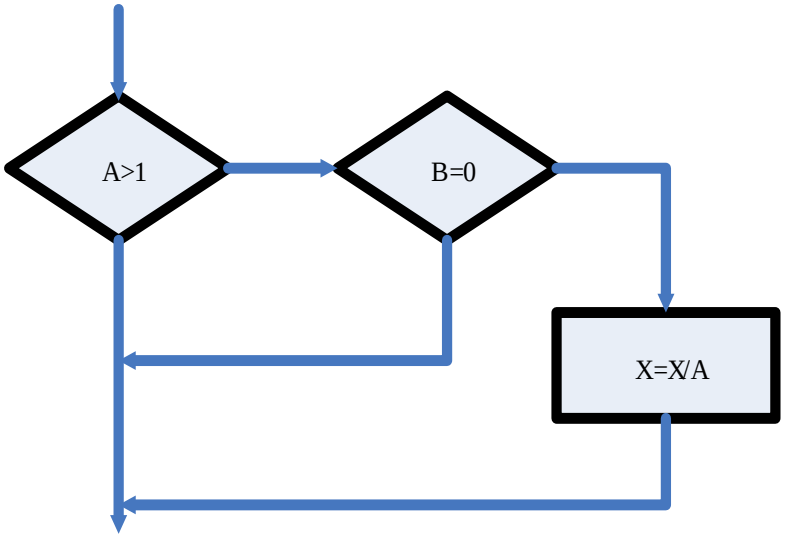
1. 程序的控制流图

- **复合条件的控制流图**

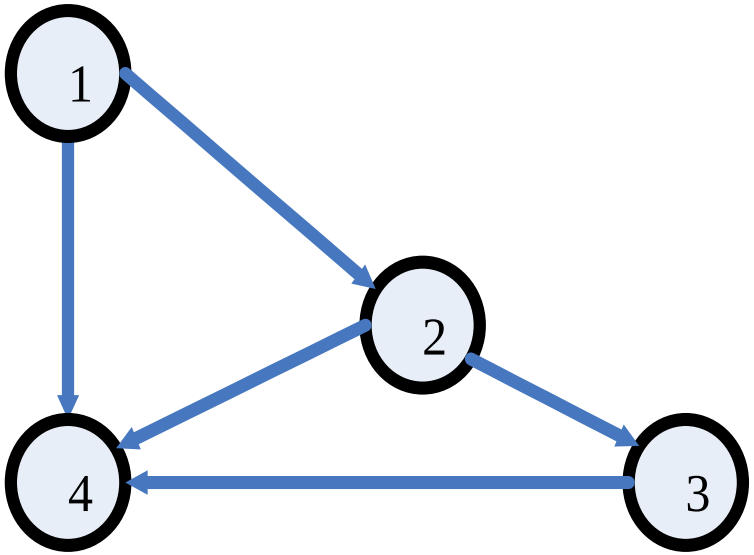
  如果判断中的条件表达式是由一个或多个逻辑运算符 (OR，AND, ...) 连接的复合条件表达式，则需改为 一系列只有单个条件的嵌套的判断。



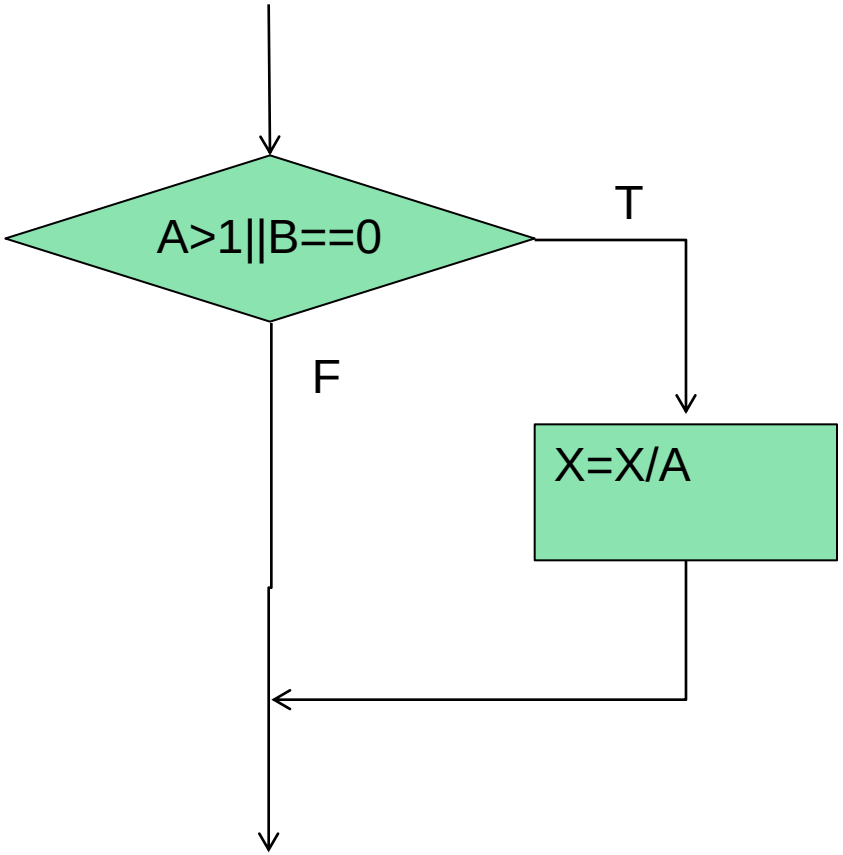
（a） 流程图

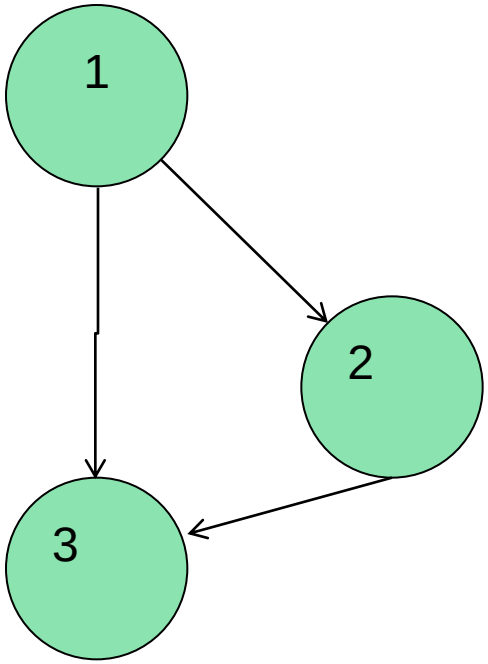


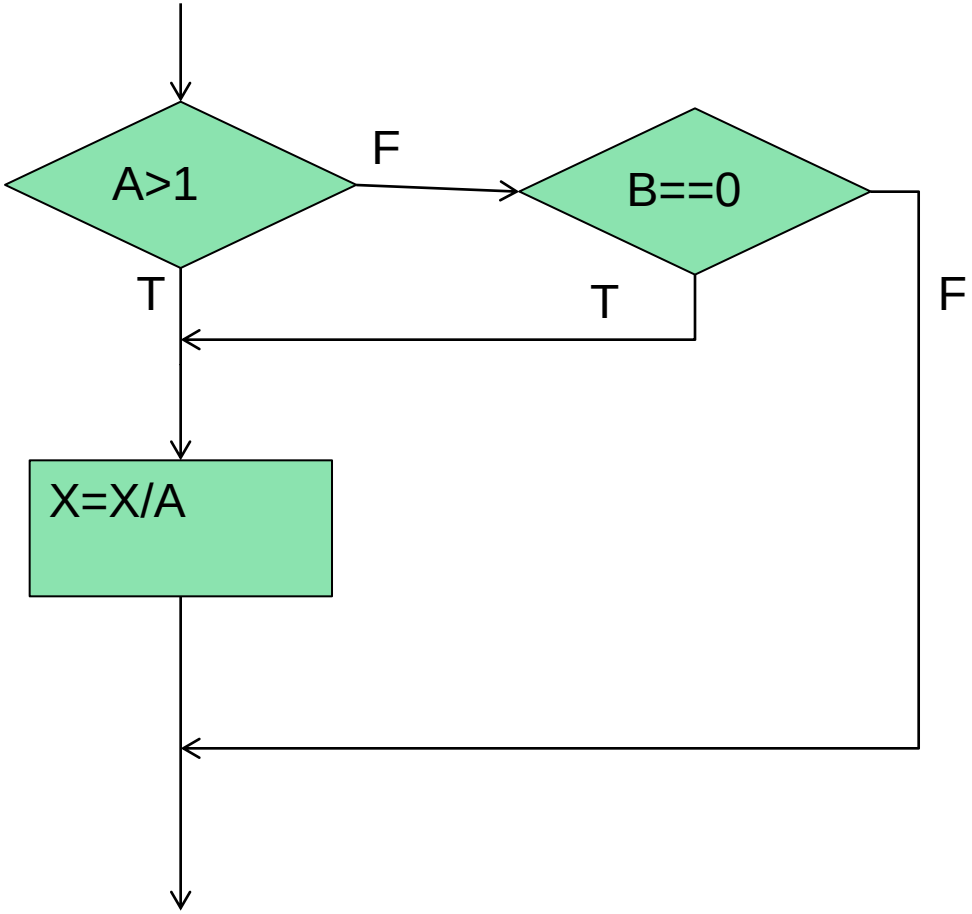
（c） 详细流程图

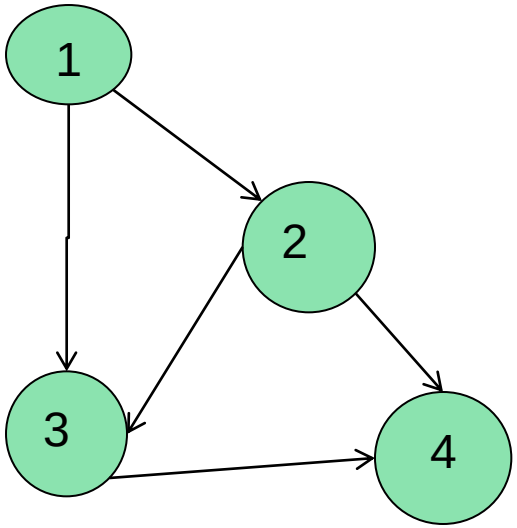


（d） 流程图c 对应的流图


A>1||B==0
T
F
X=X/A


1
2
3

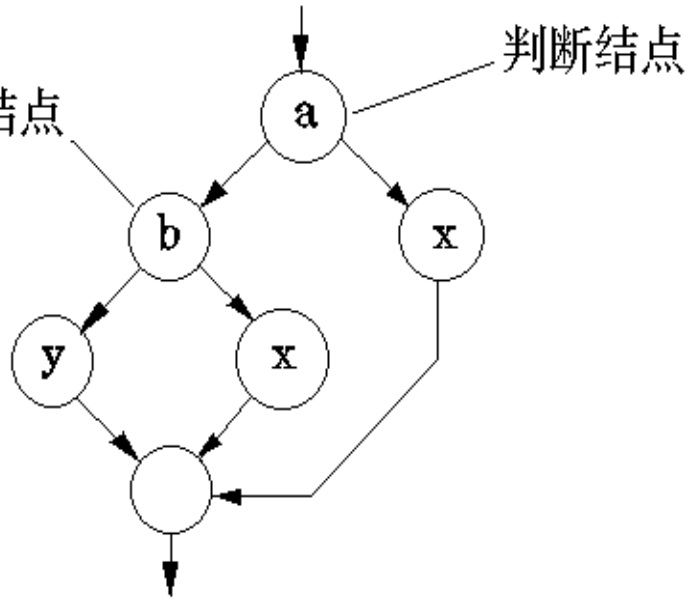

A>1
F
B==0
T
T
F
X=X/A


1
2
3
4

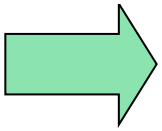
1.

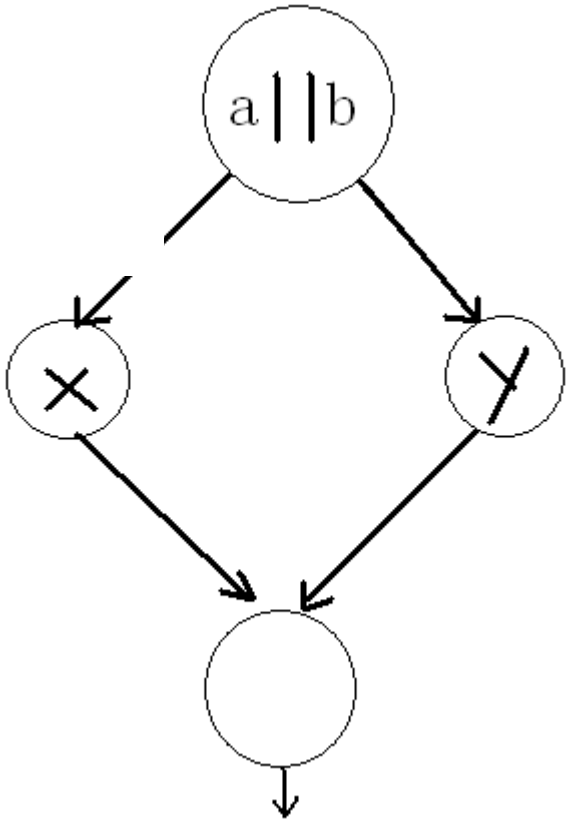
```
        .
        .
        .
if   a OR b
   then  procedure  x
   else  procedure  y;
        .
        .
        .
```

(a)

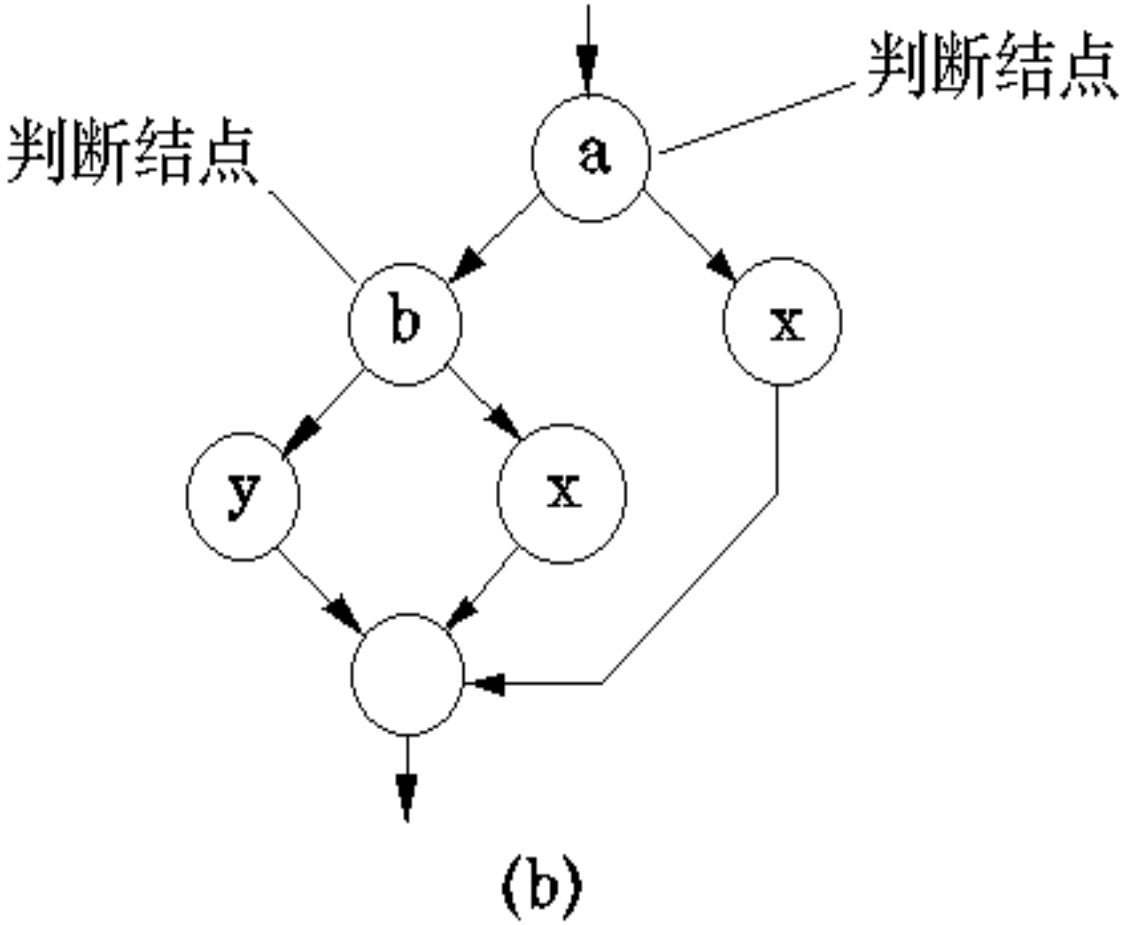


(b)





1. 程序的控制流图



(b)

2. 程序环路复杂性

- 环路复杂性即McCabe复杂性度量，在进行程序的基本路径测试时，从程序的环路复杂性可导出程序基本路径集合中的独立路径条数。

- 程序的环路复杂性给出了程序基本路径集中的独立路径条数，这是确保程序中每个可执行语句至少执行一次所必需的测试用例数目的上界。

2. 程序环路复杂性

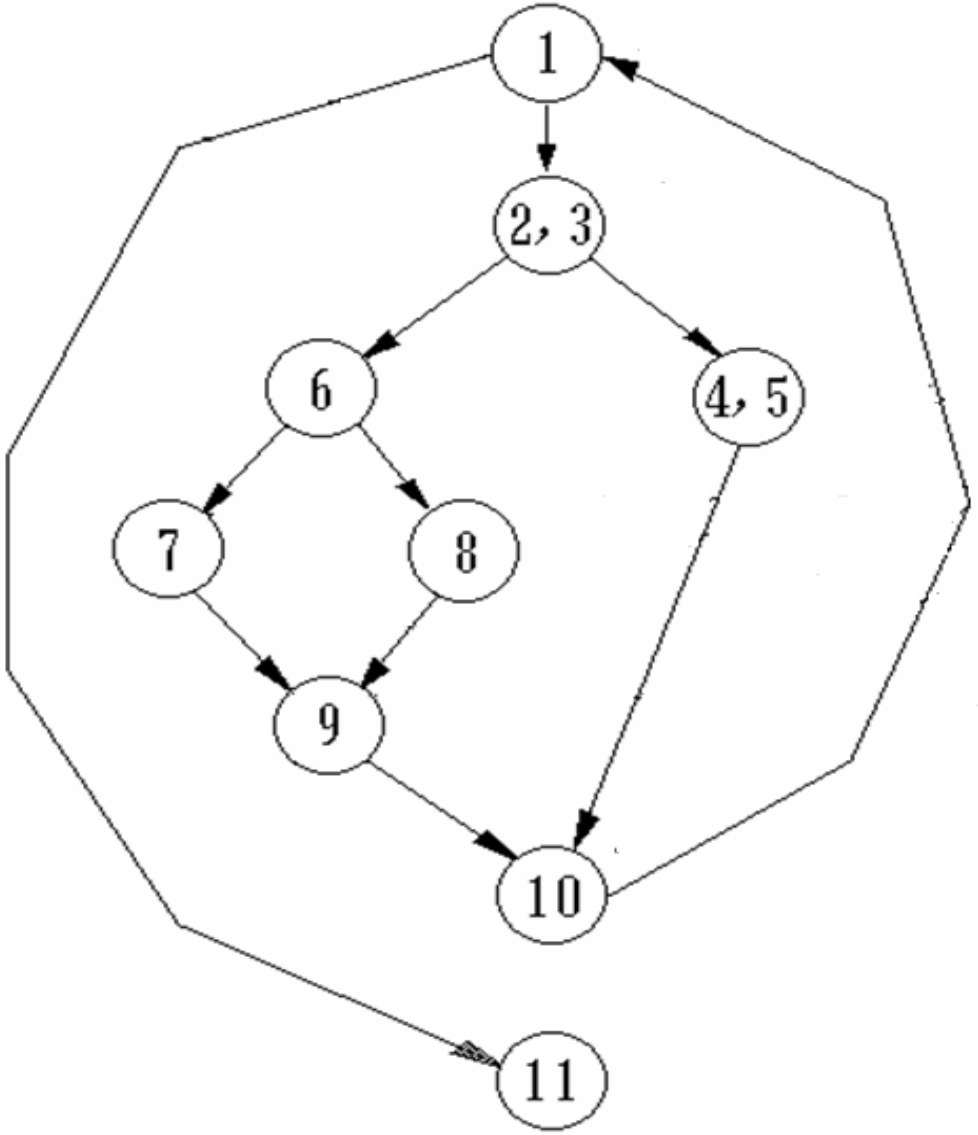
- 独立路径：指包括一组以前没有处理的语句或条件的一条路径。

- **从控制流图来看，一条独立路径是至少包含有一条在其它独立路径中从未有过的边的路径。**

- **一条路径这里指一条“程序通路”。**

2. 程序环路复杂性

思考:根据独立路径的定义,在图示的控制流图中，有哪些独立的路径?



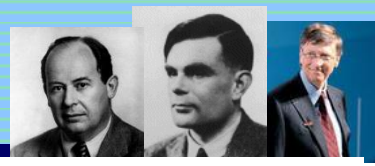
2. 程序环路复杂性

- **path1：1 - 11**
  path2：1 - 2 - 3 - 4 - 5 - 10 - 1 - 11
  path3：1 - 2 - 3 - 6 - 8 - 9 - 10 - 1 - 11
  path4：1 - 2 - 3 - 6 - 7 - 9 - 10 - 1 - 11
- **路径 path1，path2，path3，path4组成了控制流图的一个基本路径集。**

2. 程序环路复杂性

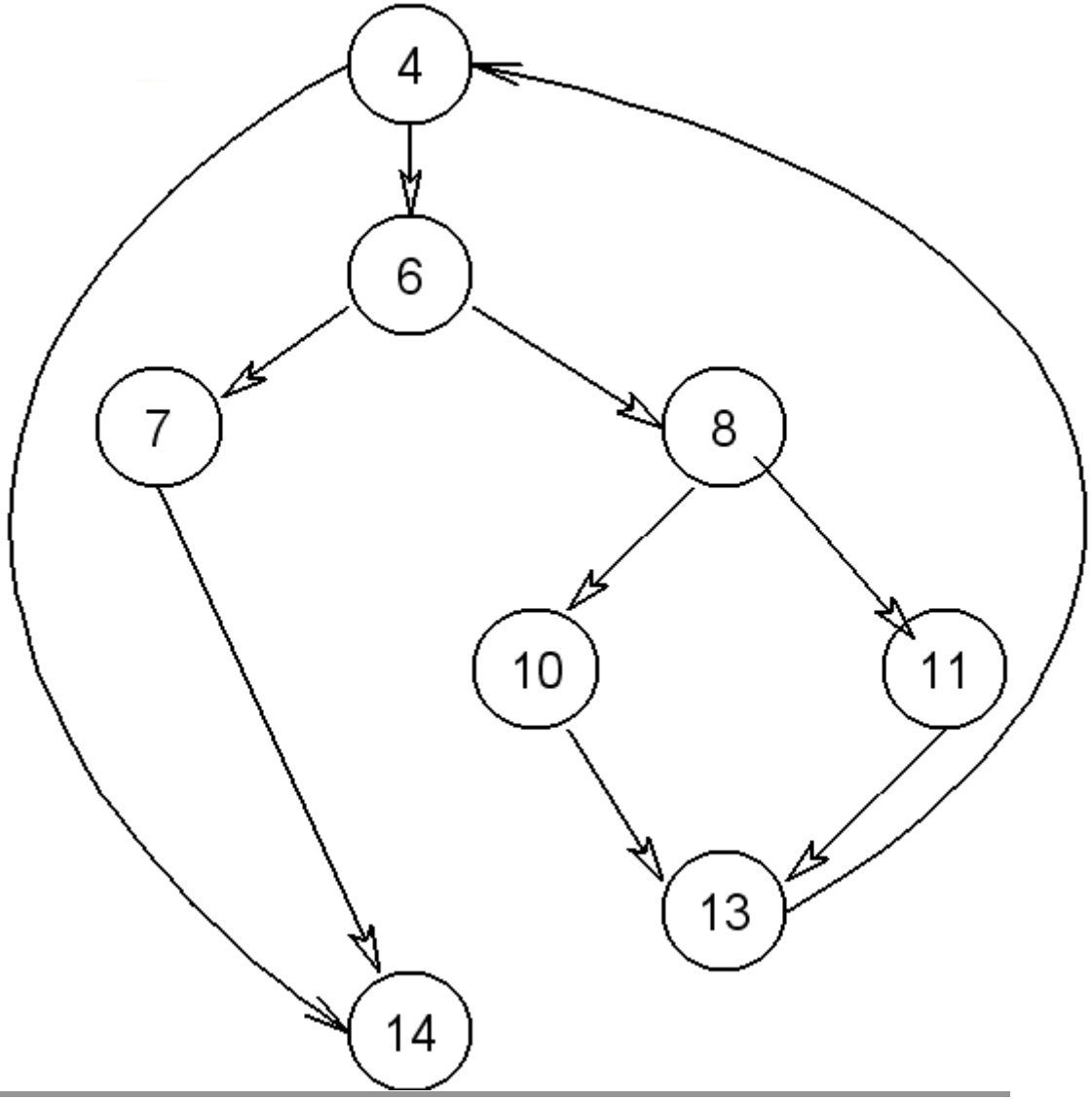
- **程序环路复杂性**计算方法（三种）：
- (1)流图中**区域的数量**对应于环形复杂度
- (2)给定流图G的环形复杂度V(G),定义为 **V(G)=E-N+2**, E是流图中边的数量，N是流图中节点的数量。
- (3) **V(G)=P+1**, P是流图G中的判定节点数。

2. 程序环路复杂性



如右图的控制流图，环形复杂度的计算方法如下：

- ✓ 流图中有四个区域；
- ✓ V(G)=10条边-8结点+2=4；
- ✓ V(G)=3个判定结点+1=4。



练习P76第3题

3. 导出测试用例

- 导出测试用例，确保基本路径集中的每一条路径的执行。

- 根据判断结点给出的条件，选择适当的数据以保证某一条路径可以被测试到 — 用逻辑覆盖方法。

3. 导出测试用例

- 每个测试用例执行之后，与预期结果进行比较。如果所有测试用例都执行完毕，则可以确信程序中所有的可执行语句至少被执行了一次。

- 必须注意，一些独立的路径，往往不是完全孤立的，有时它是程序正常的控制流的一部分，这时，这些路径的测试可以是另一条路径测试的一部分。

基本路径测试



- **基本路径测试法的步骤：**

(1)以详细或源代码作为基础，导出程序的控制流图。

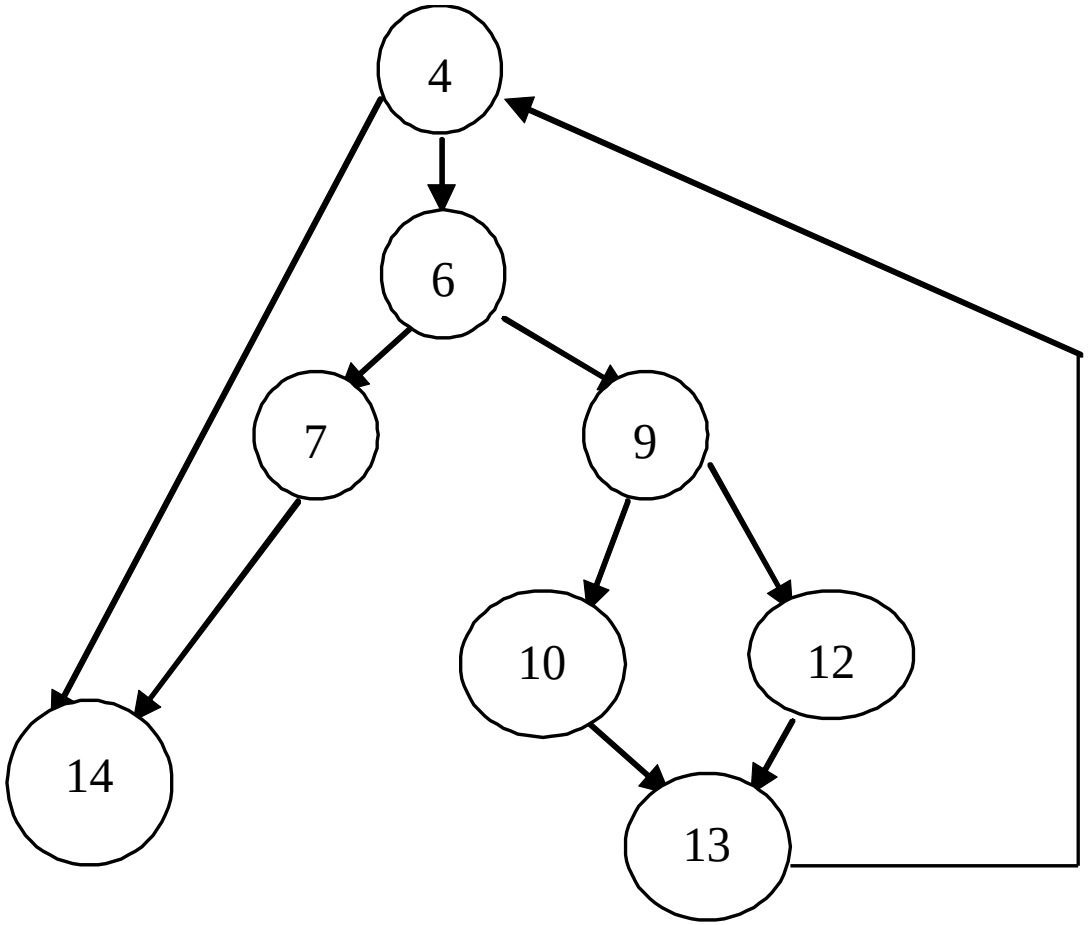
(2)计算得到控制流图G的环路复杂度V(G)

(3)确定基本路径集，生成测试用例，确保基本路径集中每条路径的执行。

- 例：如下所示的c语言函数：

```
void sort(int irecordnum, int itype)
1 {
2 int x=0;
3   int y=0;
4   while(irecordnum-->0)
5   {
6      if(itype= =0)
7         break;
8      else
9         if(itype==1)
10            x=x+10;
11        else
12           y=y+20;
13   }
14 }
```

程序段的流图



计算其环形复杂度

- 环形复杂度为:
- V(G)=E-N+2

  =10-8+2=4
- 或者

  V(G)=P+1

  =3+1=4

导出基本路径集

- 导出基本路径集，列出程序的独立路径，得：
- 路径1：4->14
- 路径2：4->6->7->14
- 路径3：4->6->9->10->13->4->14
- 路径4：4->6->9->12->13->4->14

设计测试用例

- 根据上一步得出的独立路径，涉及测试用例，如下：

	输入数据	预期输出
TC1	irecordnum=0 itype=0	x=0 y=0
TC2	irecordnum=1 itype=0	x=0 y=0
TC3	irecordnum=1 itype=1	x=10 y=0
TC4	irecordnum=1 itype=2	x=0 y=20

执行测试用例

因为被测试的模块是一个函数，所以要加载测试用例，必须编写一个驱动模块，即由一个主程序来调用该函数。为了采集结果，可以用以下两种方法。

1）修改被测试函数，使其返回验证的预期结果。

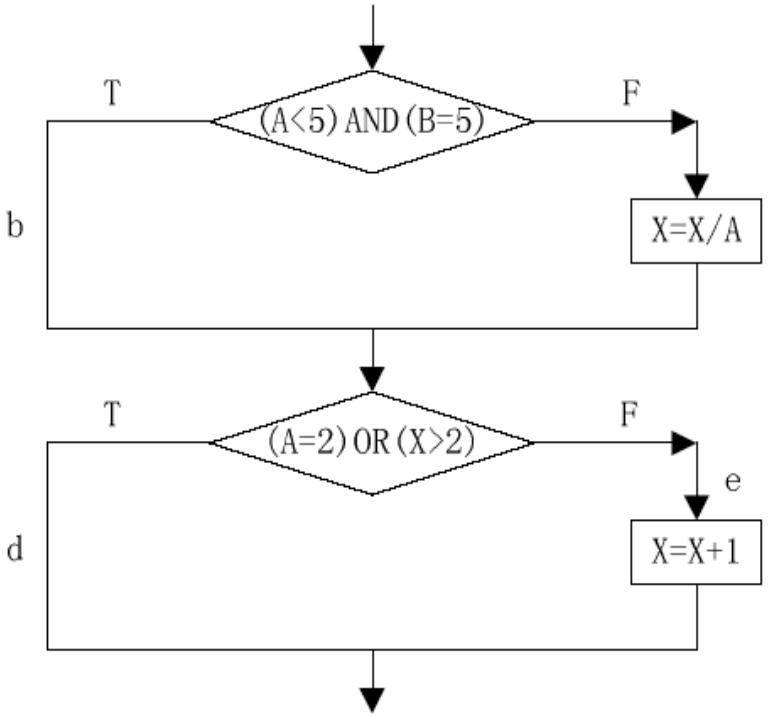
2）添加一个全局变量，在程序中把验证的值赋值给该全局变量。

因为第 1 种方式对程序影响比较大，因此采用第 2 种方式。

```
void  Sort ( int  iRecordNum, int iType )
1 {
2    int  x=0;
3    int  y=0;
4    while ( iRecordNum-- > 0 )
5    {
6      If ( iType==0 )
7            x=y+2; break;
8     else  If ( iType==1 )
9           x=y+10;
10     else
11          x=y+20;
12   }
13 ret=x;
14 }
```

```
void  Sort(int iRecordNum,int iType)
int ret;
int main()
{
      sort(0,0);
      if(0==ret)
         printf("pass");
      else
         printf("fail");
      return 0;
}
```

实训04练习1

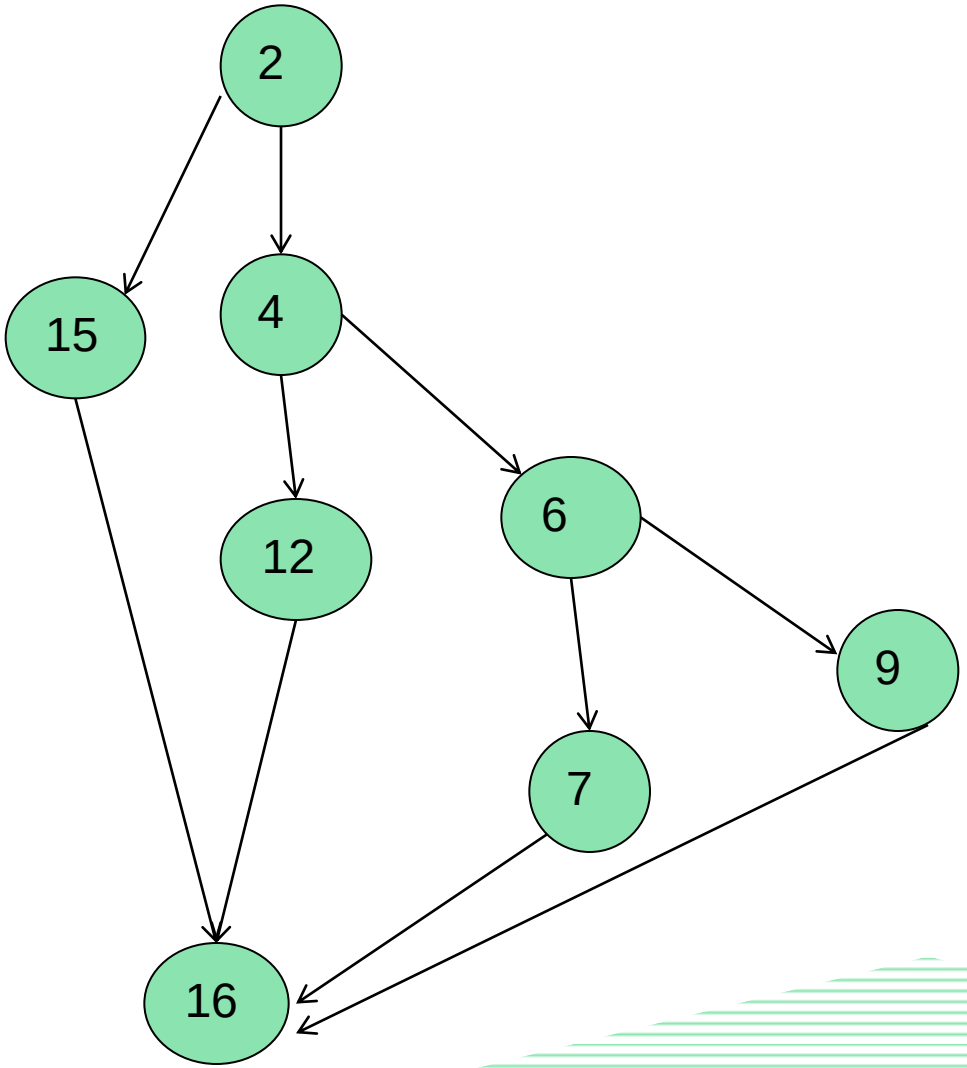


- 根据左图给出的程序流程图，完成以下要求：

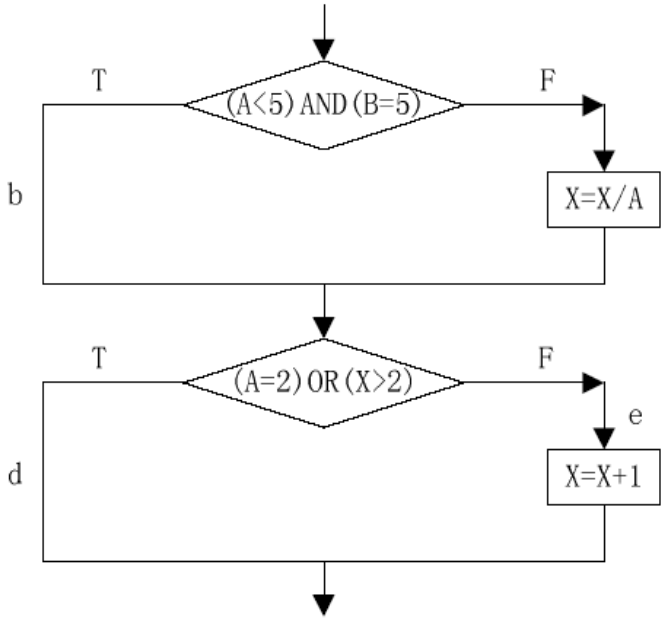
（1）画出相应的控制流图。

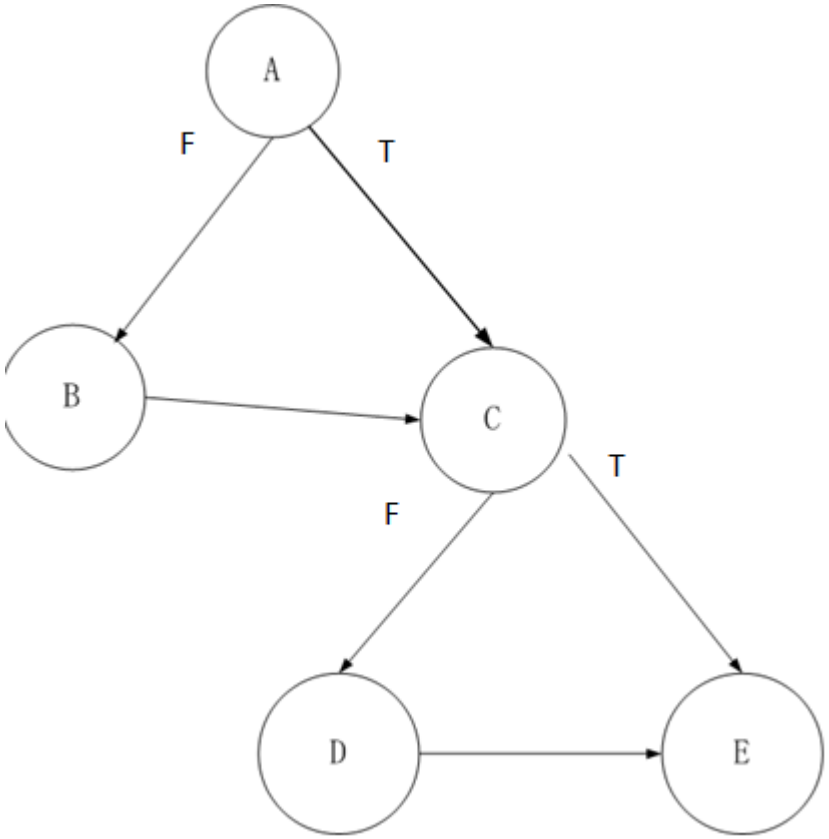
（2）计算环形复杂度。

（3）找出程序的独立路径集合。

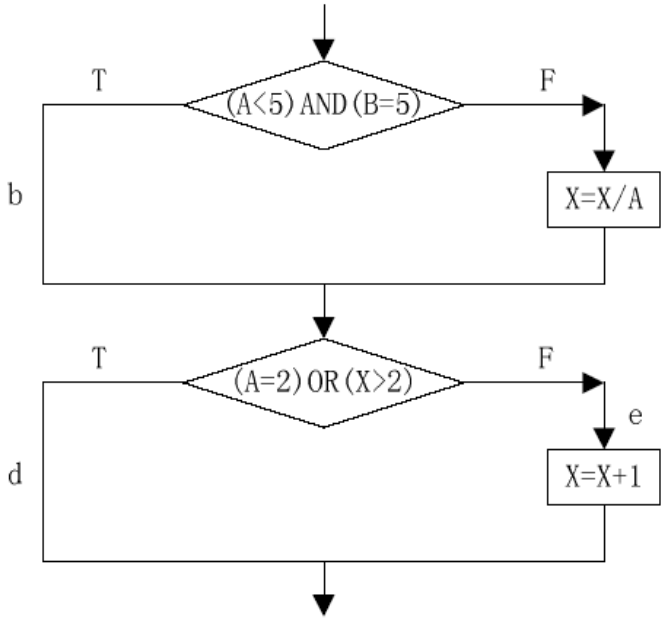

2
15
4
12
6
9
7
16

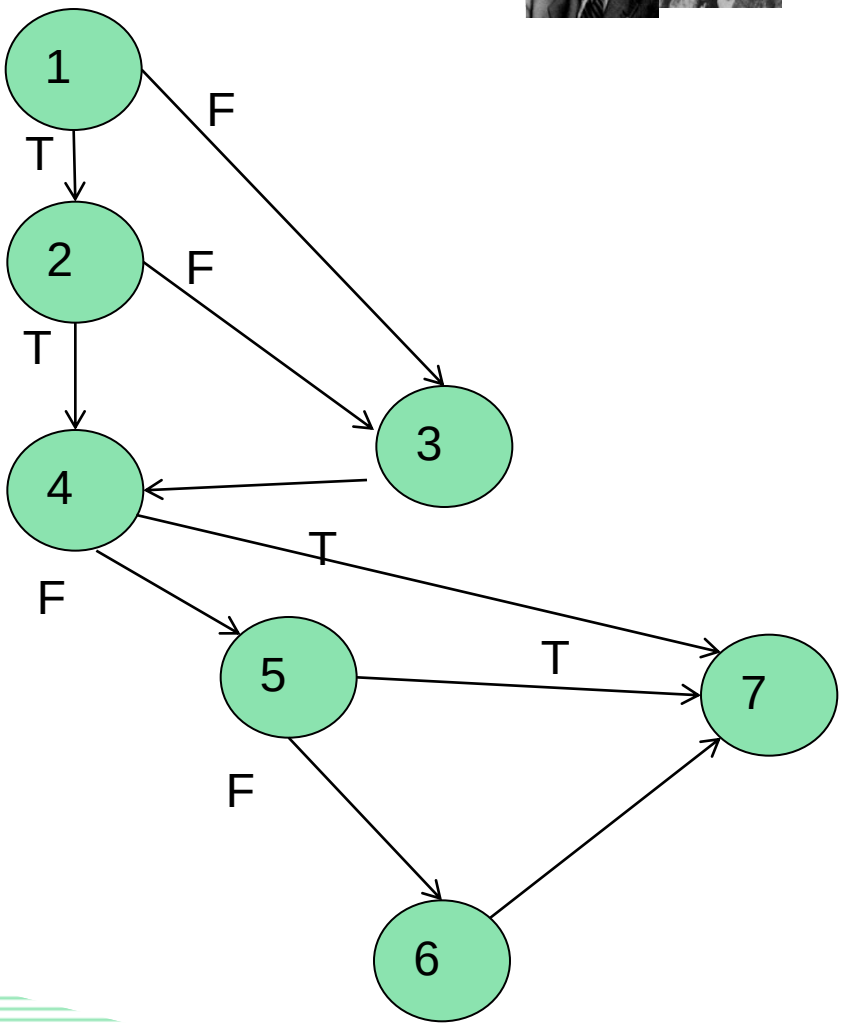
实训04练习1



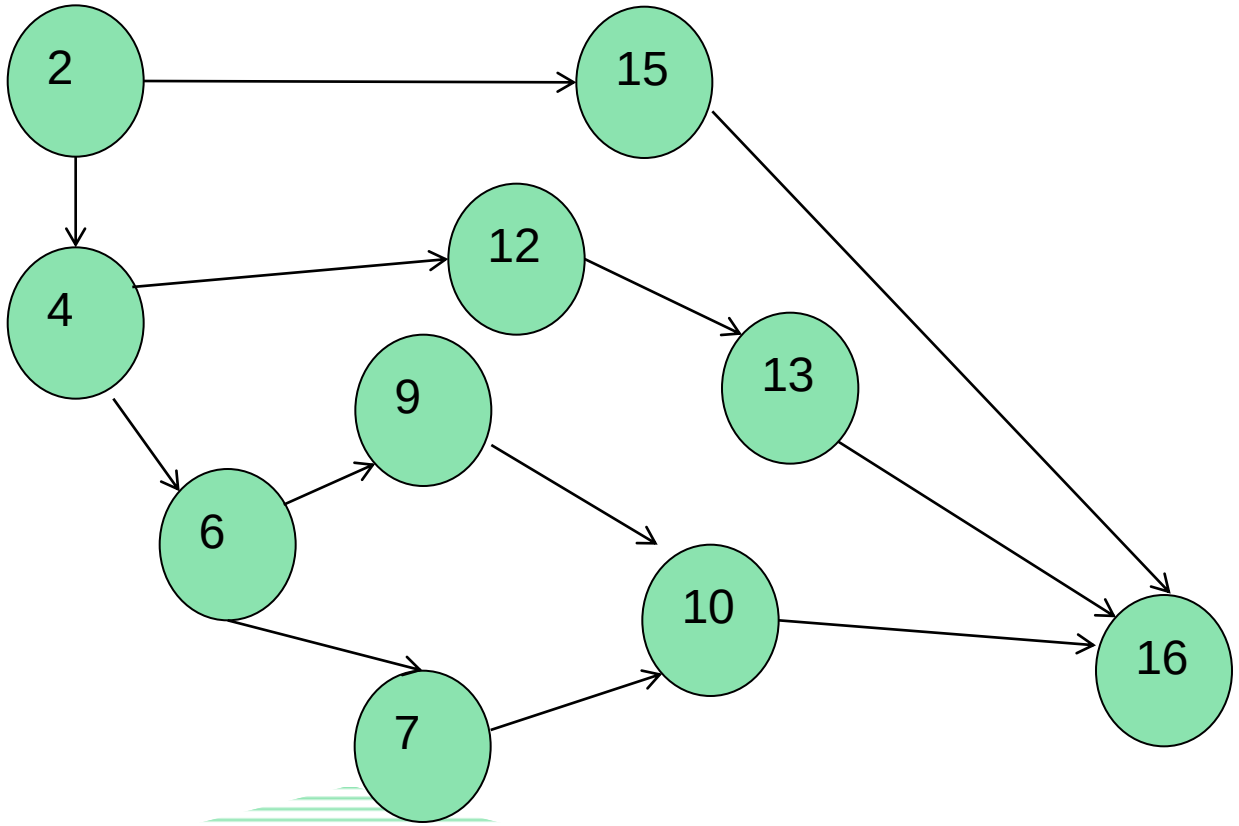


实训04练习1



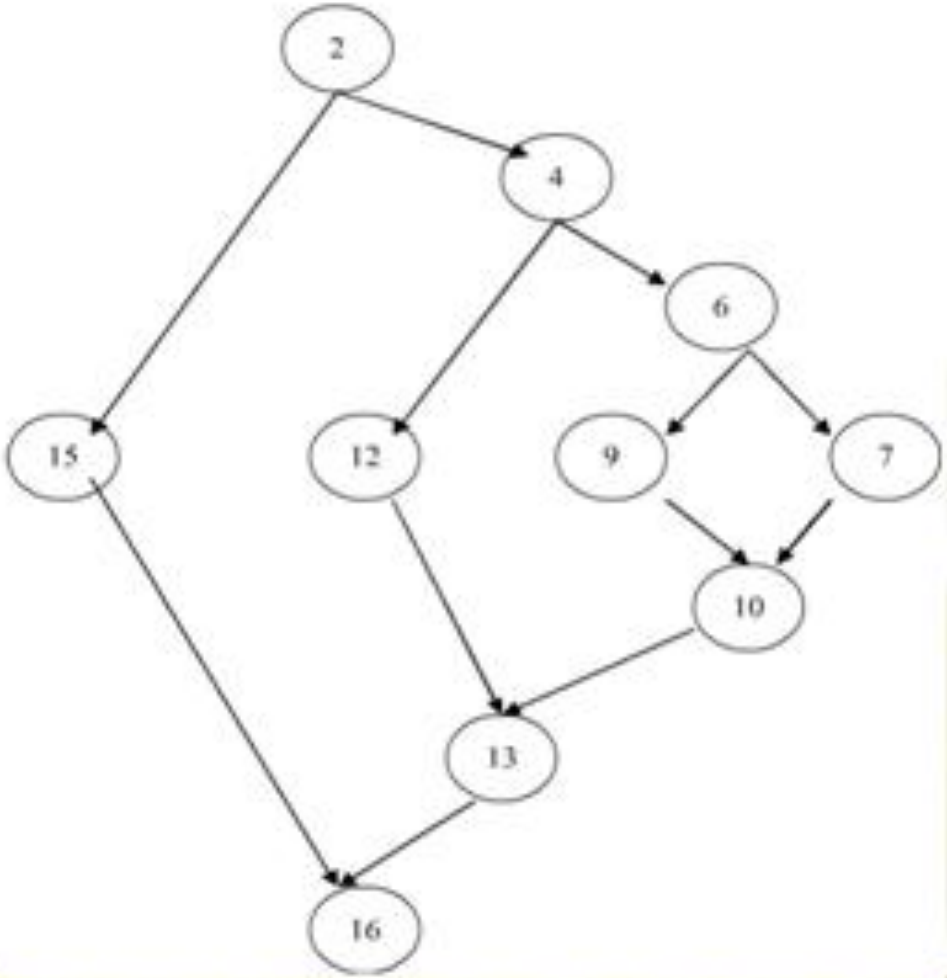


```
  Int IsLeap(int year)
1  {
2      if (year % 4==0)
3      {
4        if (year % 100==0)
5        {
6           if (year % 400==0)
7              leap=1;
8           else
9              leap=0;
10       }
11       else
12          leap=1;
13     }
14     else
15       leap=0;
16     return leap;
17 }
```



实训04练习2

1、控制流图：



2、V（G）=4

实训04练习2

用例号	路径	输入数据	预期结果
UC1	2-15-16	1001,1003	0
UC2	2-4-12-13-16	1004,1008,1012,1016	1
UC3	2-4-6-9-10-13-16	1100,1300,1400,1500	0
UC4	2-4-6-7-10-13-16	1200,2000	1