基本路径法

覆盖测试与路径测试

白盒测试的两种常用技术：

- 覆盖测试：在测试过程中，以覆盖某些程序元素为测试目标的测试。
- 路径测试：从流程图上讲，程序的一次执行对应于从入口到出口的一条路径，针对路径的测试即为路径测试。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。

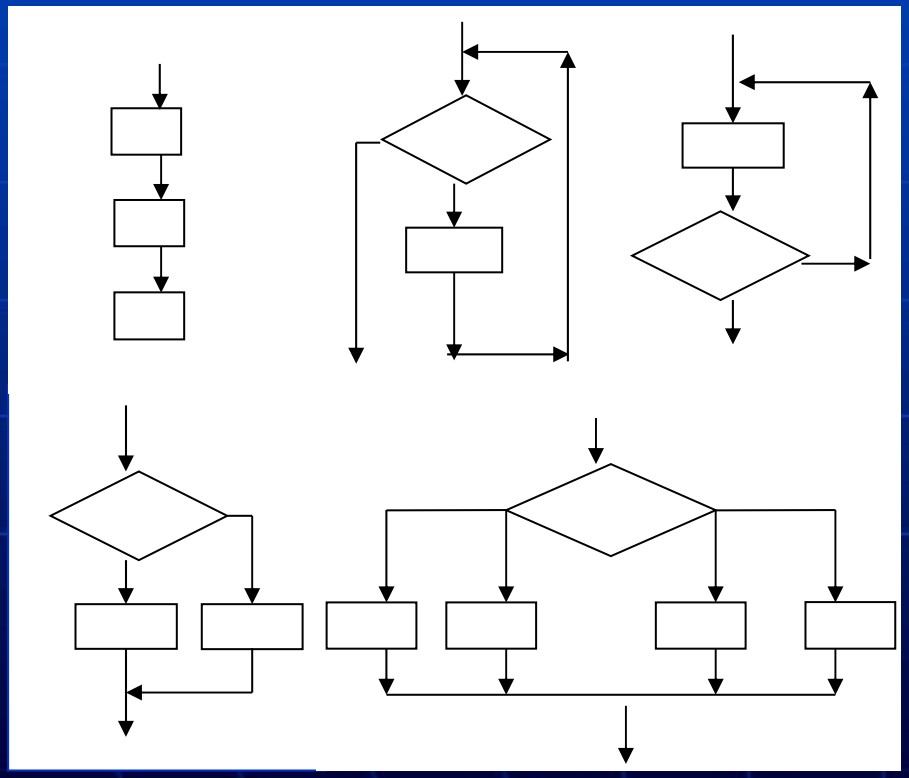
基本路径测试

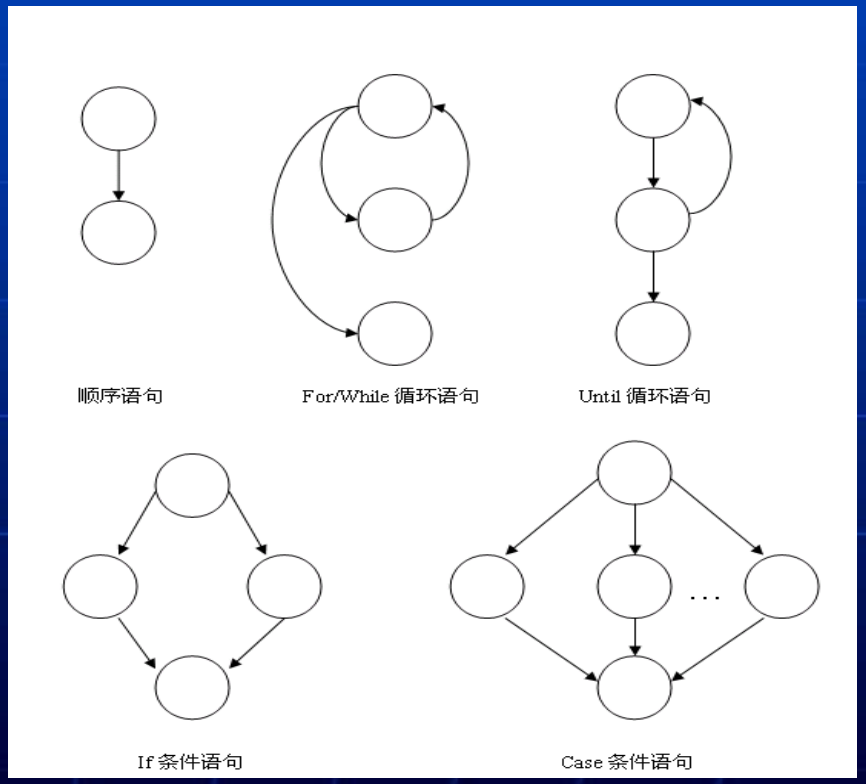
- 完成路径测试的理想情况是做到路径覆盖，但对于复杂性大的程序要做到所有路径覆盖（测试所有可执行路径）是不可能的。
- 在不能做到所有路径覆盖的前提下，如果某一程序的每一个独立路径都被测试过，那么可以认为程序中的每个语句都已经检验过了，即达到了语句覆盖。这种测试方法就是通常所说的基本路径测试方法。

1、控制流图

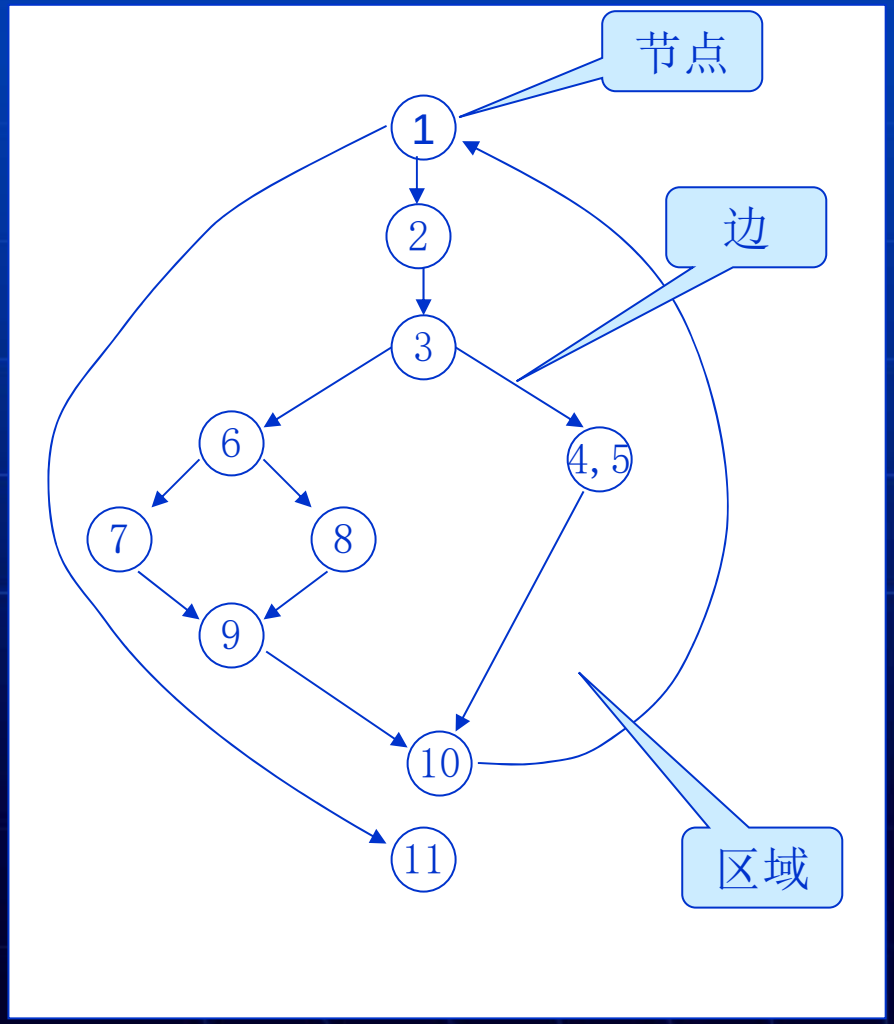
- 控制流图（可简称流图）是对程序流程图进行简化后得到的，它可以更加突出的表示程序控制流的结构。
- 控制流图中包括两种图形符号：节点和控制流线。
  - 节点由带标号的圆圈表示，可代表一个或多个语句、一个处理框序列和一个条件判定框。
  - 控制流线由带箭头的弧或线表示，可称为边。它代表程序中的控制流。

常见结构的流程图与控制流图



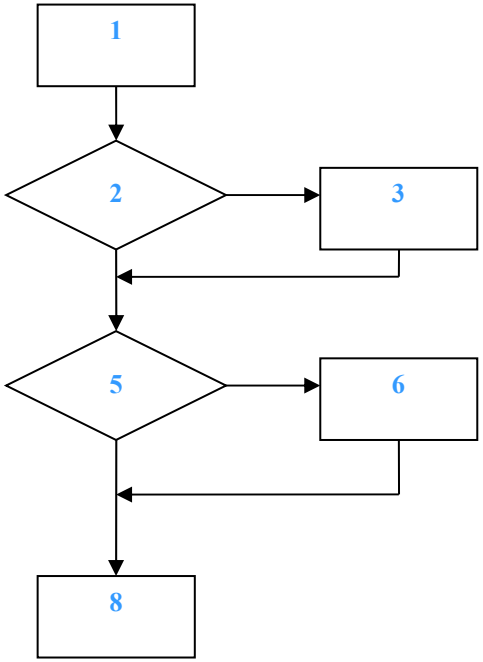


控制流图实例

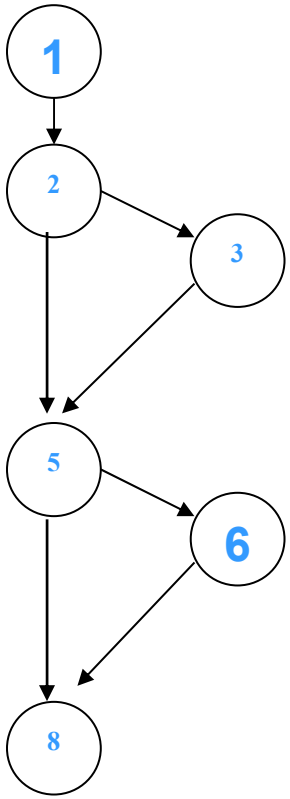


其中，包含条件的节点被称为**判定节点**（也叫谓词节点），由判定节点发出的边必须终止于某一个节点，由边和节点所限定的范围被称为**区域**。

如何根据流程图得到控制流图？

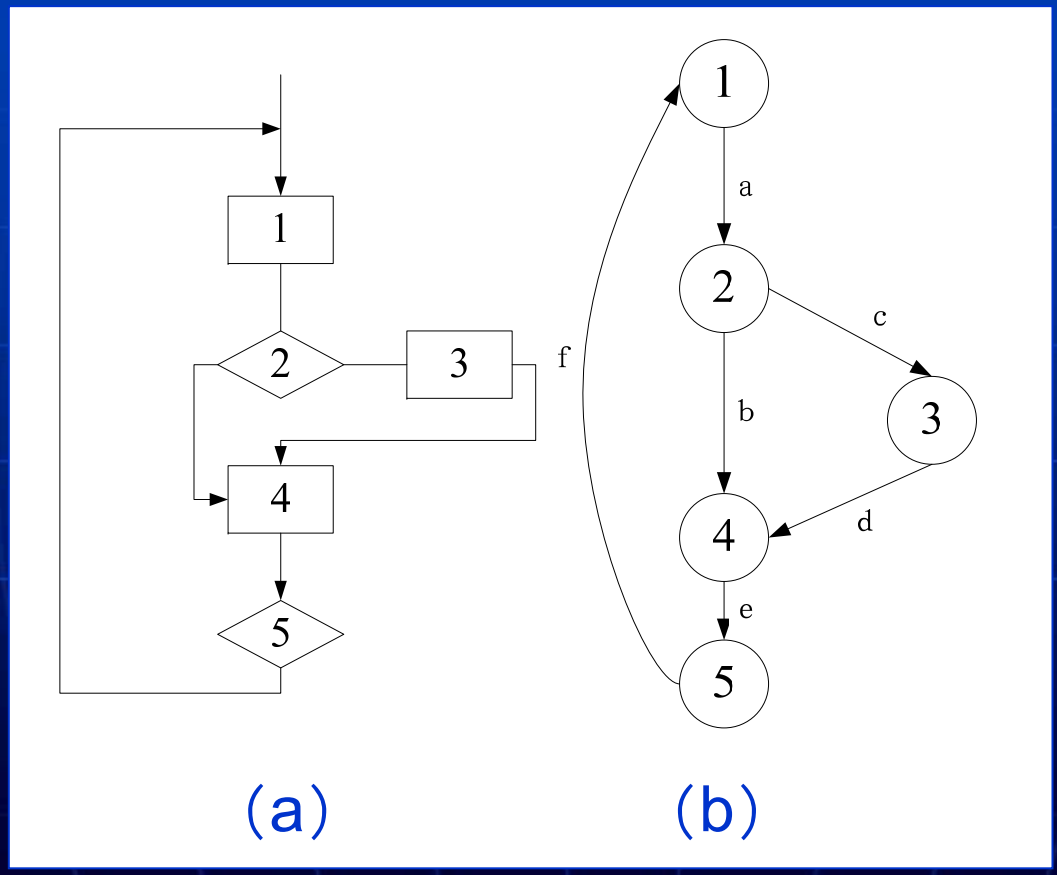


（a）程序流程图



（b）控制流图

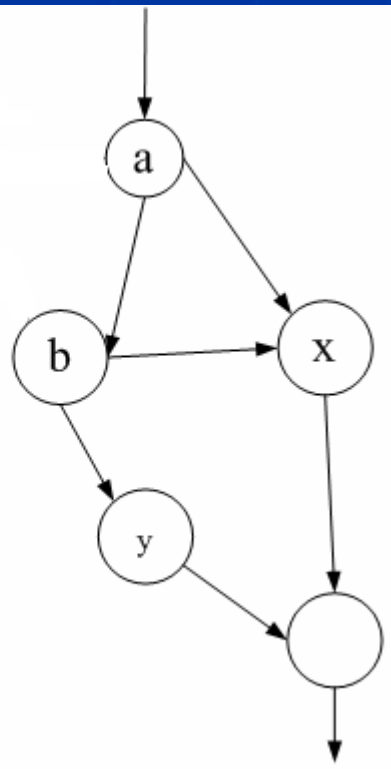
如何根据流程图得到控制流图？（续）



关于复合条件的处理

如何画复合条件的控制流图？

```
……
If a OR b
  Then call x
   Else call y
EndIf
……
```

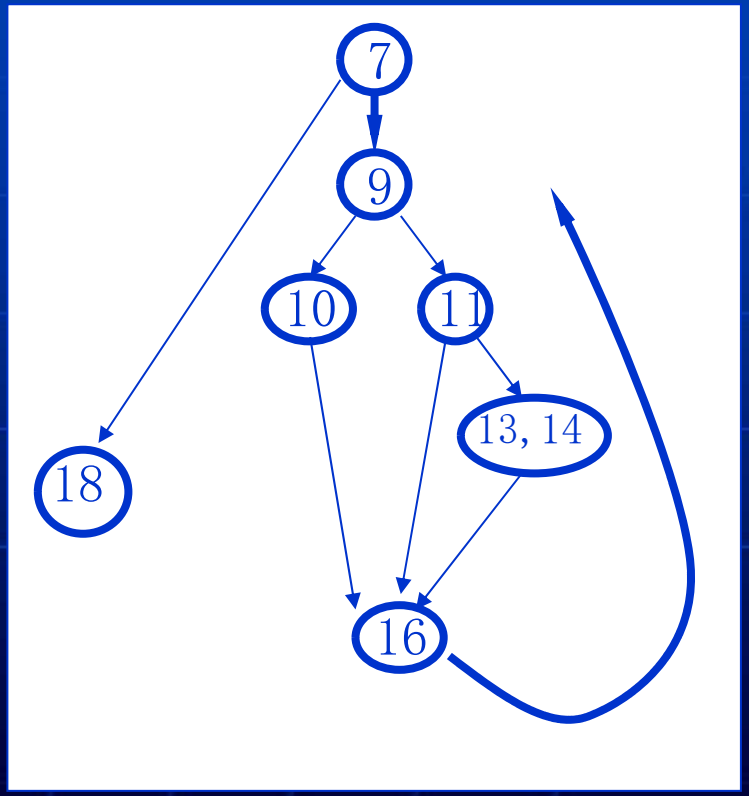


如何由源程序直接得到控制流图？

```
1  main ()
2  {
3   int num1=0, num2=0, score=100;
4   int i;
5   char str;
6   scanf (“%d, %c\n”, &i, &str);
7   while (i<5)
8   {
9      if (str=’T’)
10                num1++;
11     else if (str=’F’)
12     {
13       score=score-10;
14       num2 ++;
15      }
16     i++;
17  }
18  printf (“num1=%d, num2=%d, score=%d\n”, num1, num2, score);
19 }
```

如何由源程序直接得到控制流图？（续）

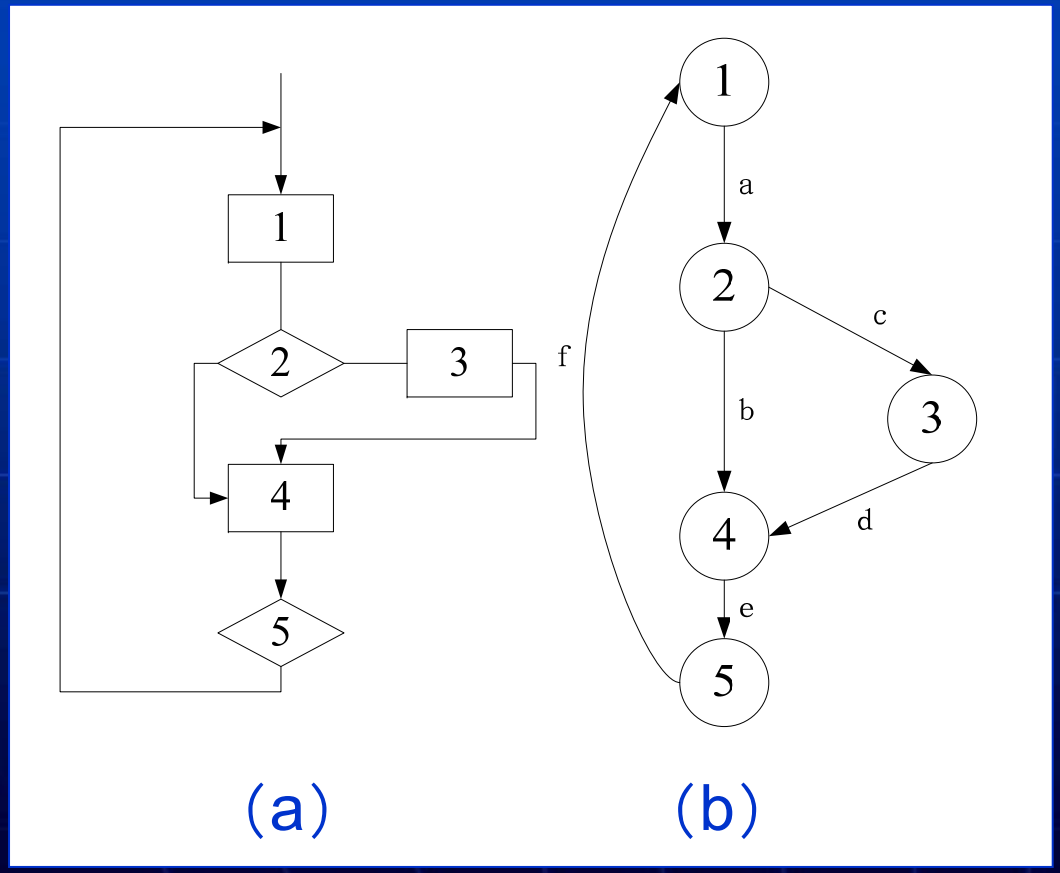
根据源代码可以导出程序的控制流图，如图所示。每个圆圈代表控制流图的节点，可以表示一个或多个语句。圆圈中的数字对应程序中某一行的编号。箭头代表边的方向，即控制流方向。



2、图矩阵

- 图矩阵是控制流图的矩阵表示形式。
- 图矩阵是一个方形矩阵，其维数等于控制流图的节点数。矩阵中的每列和每行都对应于标识的节点，矩阵元素对应于节点间的边。
- 通常，控制流图中的结点用数字标识，边则用字母标识。如果在控制流图中从第 i 个结点到第 j 个结点有一个标识为 x 的边相连接，则在对应图矩阵的第 i 行第 j 列有一个非空的元素 x 。

图矩阵实例



	1	2	3	4	5
1		a			
2			c	b	
3				d	
4					e
5	f				

3、环形复杂度及其计算方法

- 环形复杂度又称为圈复杂度，以图论为基础，为我们提供了非常有用的软件度量。可用如下三种方法之一来计算环形复杂度：
- 控制流图中区域的数量对应于环形复杂度。
- 给定控制流图G的环形复杂度—V(G)，定义为

  V(G) = E-N+2

  其中，E是控制流图中边的数量，N是控制流图中的节点数量。
- 给定控制流图G的环形复杂度—V(G)，也可定义为

  V(G) = P+1

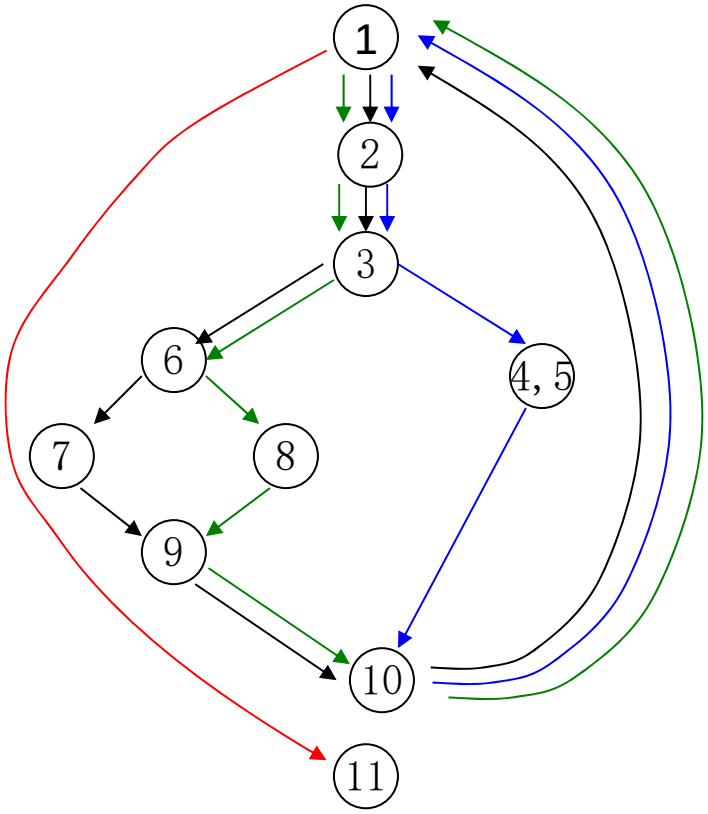
  其中，P是控制流图G中判定节点的数量。

4、独立路径

- 独立路径是指程序中至少引入了一个新的处理语句集合或一个新条件的程序通路。
- 采用流图的术语，即独立路径必须至少包含一条在本次定义路径之前不曾用过的边。

独立路径实例

独立路径：至少沿一条新的边移动的路径



路径1：1-11

路径2：1-2-3-4-10-1…

路径3：1-2-5-8-9-10-1…

路径4：1-5-6-7-9-10-1…

注：“…”表示后面剩下的路径是可以选择的，原因在于存在循环结构。

5、程序基本路径集及其确定方法

- 程序基本路径集是指由若干条独立路径组成的集合，其数量由环形复杂度确定。
- McCabe开发了一种算法，用于确定程序的基本路径集合，方法如下：

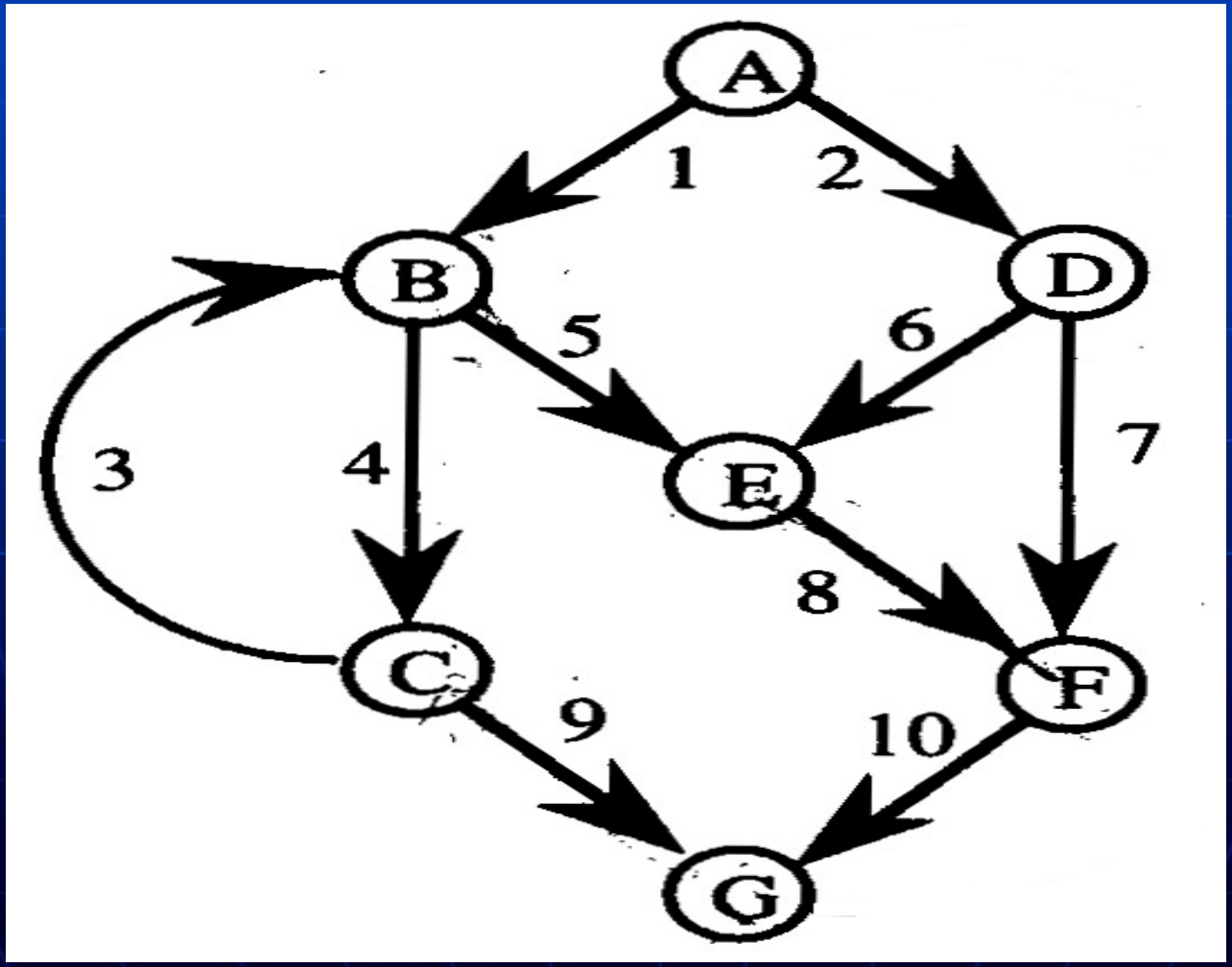
1、选择一个基线路径。

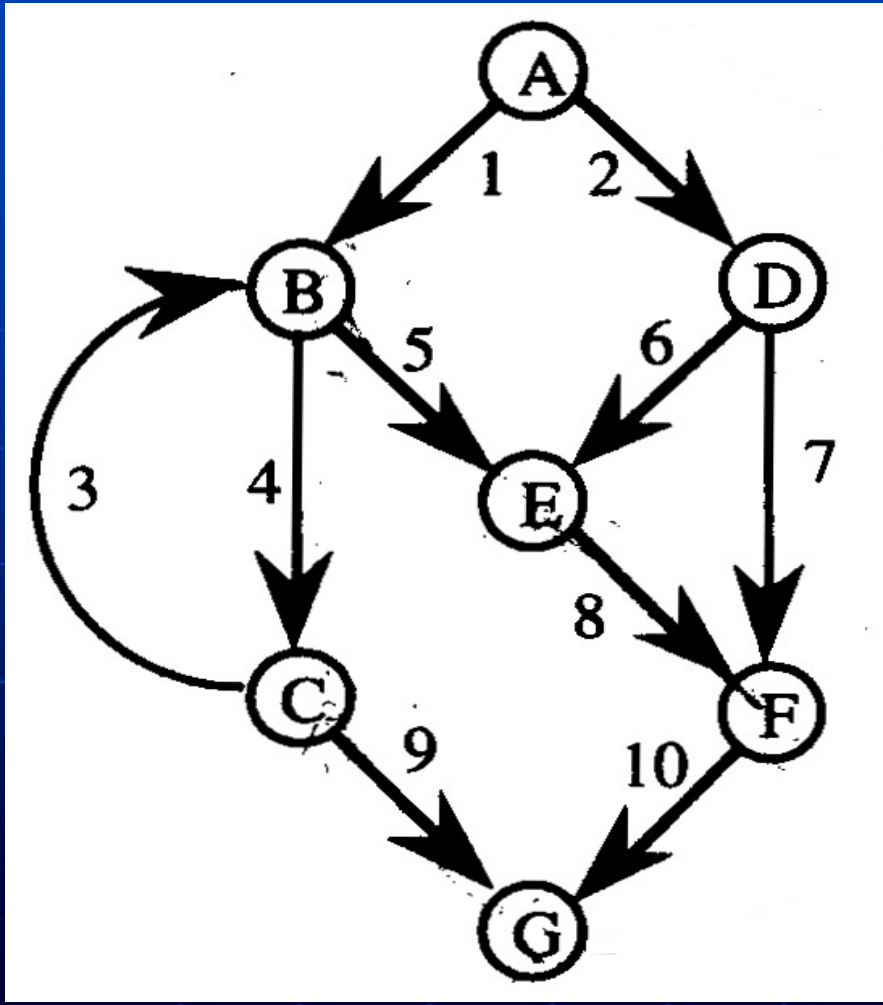
2、沿基线路径后退，碰到判定节点后翻转，将翻转后的路径作为基线路径，重复本步骤，直到所有的节点都被翻转。

注意：为遵循先易后难的原则，对于循环，一般先让路径跳过循环，然后考虑进入循环。

- 基本路径集通常并不唯一。

程序基本路径集确定案例





在A处翻转

在C处翻转

P1：A，B，C，G

P2：A，B，C，B，…

在B处翻转

P3：A，B，E，F，G

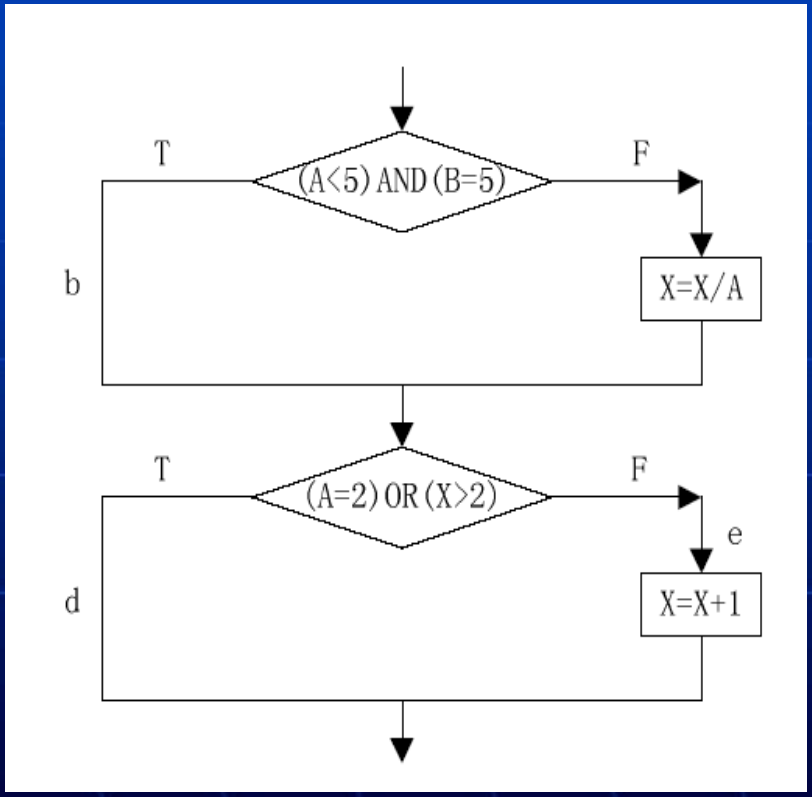
P4：A，D，E，F，G

在D处翻转

P5：A，D，F，G

注:“…”表示后面剩下的路径是可以选择的，原因在于存在循环结构。

课堂练习一



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）给出相应的图矩阵。

（3）计算环形复杂度。

（4）找出程序的基本路径集合。

基本路径测试法测试步骤

基本路径测试步骤：

1. 画出程序的控制流图
2. 计算流图G的环路复杂性V(G)
3. 确定只包含独立路径的基本路径集
4. 根据上面的独立路径，设计测试用例，使程序分别沿上面的独立路径执行。