程序图和DD路径

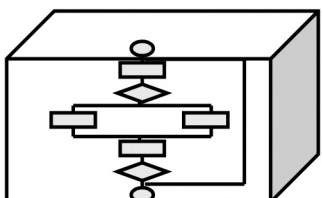
目录

■白盒测试概述

■程序图
■DD路径
■DD路径图
■拓扑路径
■可行路径

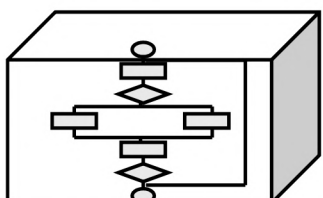
白盒测试概述

- 白盒测试又称为结构性测试、透明盒测试、逻辑驱动测试或基于代码的测试；
- 将测试对象（被测软件）看做内部逻辑完全可见的盒子，测试人员通过分析程序的逻辑结构来设计测试用例。



白盒测试概述

- 在不同点、不同分支检查程序的状态，从而确定程序的实际状态是否与预期状态一致
- 主要用于单元测试
- 路径测试和数据流测试



目录

- 白盒测试概述
- 程序图

- DD路径
- DD路径图
- 拓扑路径
- 可行路径

程序图

- 定义：程序图P = (V, E)，V是节点的集合、E是有向边的集合。其中，节点表示的是程序中的语句或语句片段，边表示程序语句或语句片段之间的控制流。
- 如果程序图P中存在两个节点i和j，且存在一条从i到j的边；这说明程序语句或语句片段j可以在i之后立即被执行。

```
if condition
  then command1;
  else command2;
endif
```

- 语句和语句片段
- 选择语句片段作为最小研究单位
- 选择语句片段的编号作为节点

示例：三角形程序

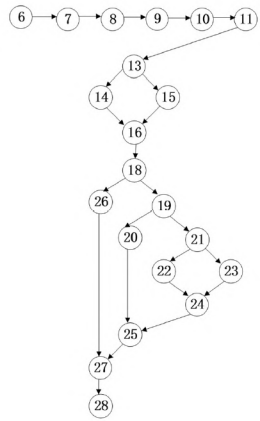
```
1.  // 三角形程序，接受三个输入作为三条边，输出三角形的类型
2.  Program triangle
3.  Int a, b, c; // 定义三个整型变量
4.  Bool isTriangle; // 记录是否为三角形
5.  // 输入参数
6.  Input(a) ; // 输入边a
7.  Output("输入的边a是:", a); // 输出边a
8.  Input(b) ; // 输入边b
9.  Output("输入的边b是:", b); // 输出边b
10. Input(c) ; // 输入边c
11. Output("输入的边c是:", c); // 输出边c
12. // 判断是否为三角形
13. If ((a+b>c) and (b+c>a) and (c+a>b))
14.     Then isTriangle = true;
15.     Else isTriangle = false;
16. EndIf
```

```
17. // 判断三角形类型并输出
18. If(isTriangle)
19.     Then If((a==b) and (b==c)) // 三条边都相等
20.             Then output("等边三角形");
21.             Else If((a≠b) and (b≠c) and (c≠a))// 三条边都不等
22.                     Then output("普通三角形");
23.                     Else output("等腰三角形");
24.                 EndIf
25.         EndIf
26.     Else output("构不成三角形");
27. EndIf
28. End triangle //程序结束
```

程序图：三角形程序

- 程序图只包含实际执行的语句

```
1.    // 三角形程序，接受三个输入作为三条边，输出三角形的类型
2.    Program triangle
3.    Int a, b, c; // 定义三个整型变量
4.    Bool isTriangle; // 记录是否为三角形
5.    // 输入参数
6.    Input(a) ; // 输入边a
7.    Output("输入的边a是：", a); // 输出边a
8.    Input(b) ; // 输入边b
9.    Output("输入的边b是：", b); // 输出边b
10.   Input(c) ; // 输入边c
11.   Output("输入的边c是：", c); // 输出边c
12.   // 判断是否为三角形
13.   If ((a+b>c) and (b+c>a) and (c+a>b))
14.     Then isTriangle = true;
15.     Else isTriangle = false;
16.   EndIf
17.   // 判断三角形类型并输出
18.   If(isTriangle)
19.     Then If((a==b) and (b==c)) // 三条边都相等
20.           Then output("等边三角形");
21.           Else If((a=b) and (b=c) and (c=a))// 三条边都不等
22.                 Then output("普通三角形");
23.                 Else output("等腰三角形");
24.                EndIf
25.           EndIf
26.     Else output("构不成三角形");
27.   EndIf
28.   End triangle //程序结束
```



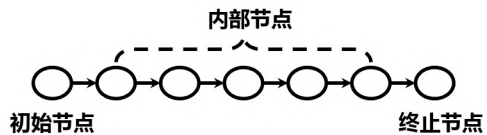
目录

DD路径

- DD路径（Decision to Decision Paths）是决策到决策的路径，是指一个语句序列；
- “决策”语句是指一个节点的入度或出度大于等于2；
- 在DD路径中，其语句序列内部没有分支；
- DD-路径可通过有程序图中的节点组成的路径来定义。

DD路径

- 图论中，这些路径称为链
  - 链是一条起始和终止节点不同的路径
  - 每个内部节点都满足入度=1和出度=1

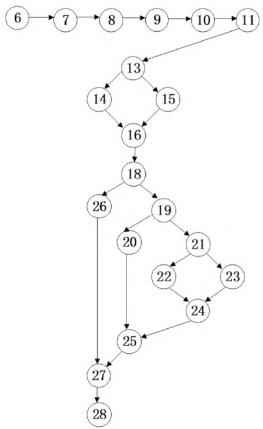


- 退化链：长度为0的链，即只有一个节点和0条边

DD路径

- 定义：DD路径是程序图中的一条链，分为如下5种情况：
  - 由一个入度为0的节点组成，对应于源节点；
  - 由一个出度为0的节点组成，对应于汇节点；
  - 由一个入度≥2或出度≥2的节点组成，对应于判定语句或其结束语句；
  - 由一个入度为1且出度为1的节点组成，对应于短分支；
  - 由长度≥1的最大链组成，对应于串行语句序列。

DD路径：三角形程序



节点	DD路径名称	定义情况
6	第一	1
7~11	A	5
13	B	3
14	C	4
15	D	4
16	E	3
18	F	3
19	G	3
20	H	4
21	I	3
22	J	4
23	K	4
24	L	3
25	M	3
26	N	4
27	O	3
28	最后	2

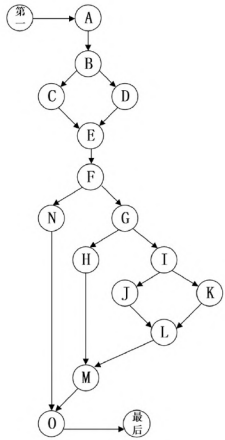
目录

- ■白盒测试概述
- ■程序图
- ■DD路径
- ■DD路径图

- ■拓扑路径
- ■可行路径

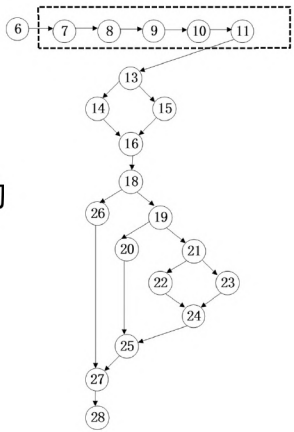
DD路径图

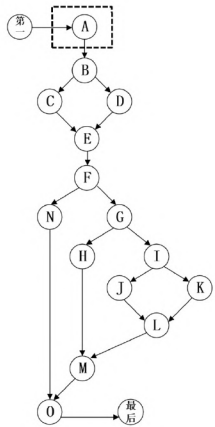
- 定义：DD路径图DP = {V, G}是一个有向图，V是节点的集合、E是有向边的集合。其中，节点表示的是程序中的DD路径，边表示连续DD路径之间的控制流。



程序图与DD路径图

- DD路径图是一种压缩格式的程序图





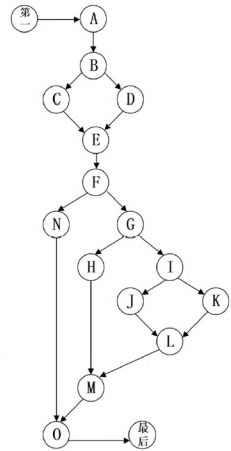
目录

- 白盒测试概述
- 程序图
- DD路径
- DD路径图
- 拓扑路径

- 可行路径

拓扑路径

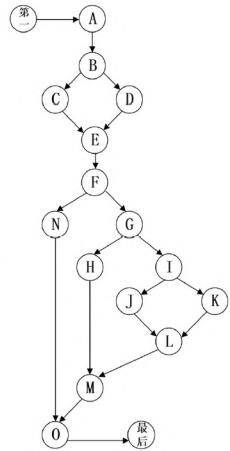
- 从图中可以得到8条拓扑上可行的路径
  - 1.第一 - A - B - C - E - F - N - O - 最后
  - 2.第一- A - B - D - E - F - N - O - 最后
  - 3.第一 - A - B - C - E - F - G - H - M - O -最后
  - 4.第一 - A - B - D - E - F - G – H - M - O -最后
  - 5.第一- A - B - C - E - F - G - I - J - L - M - O -最后
  - 6.第一- A - B - D - E - F - G - I - J - L - M - O -最后
  - 7.第一- A - B - C - E - F - G - I - K - L - M - O -最后
  - 8.第一- A - B - D - E - F - G - I - K - L - M - O -最后



目录

可行路径

- 如果分析程序源码实现，发现只有4条可行路径
  - 2.第一- A - B - D - E - F - N - O - 最后
  - 3.第一- A - B - C - E - F - G - H - M - O -最后
  - 5.第一- A - B - C - E - F - G - I - J - L - M - O -最后
  - 7.第一- A - B - C - E - F - G - I - K - L - M - O -最后



总结

- 程序图
- DD路径
- DD路径图
- 拓扑路径
- 可行路径

测试覆盖指标（一）

目录

- 覆盖率
  - 3种覆盖指标
  - 示例
- 逻辑覆盖指标

覆盖率

- 定义：覆盖率是用于度量测试完整性的一种手段
- 覆盖率 = 被执行到的测试项数量/总项数×100%
- 覆盖率对于软件测试有非常重要的作用
- 可以确定测试工作是否充分

- 节点覆盖
- 边覆盖
- 路径覆盖

目录

基于程序图的覆盖指标

- 定义：节点覆盖G_{node}是指采用给定的测试用例集作用于被测软件时，程序图中的所有节点都被遍历到，则称这组测试用例集满足节点覆盖。

基于程序图的覆盖指标

- 定义：边覆盖G_{edge}是指采用给定的测试用例集作用于被测软件时，程序图中的所有边都被遍历到，则称这组测试用例集满足边覆盖。

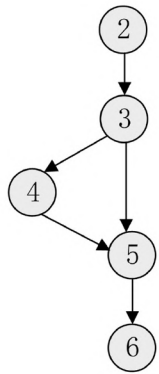
基于程序图的覆盖指标

- 定义：路径覆盖G_{path}是指采用给定的测试用例集作用于被测软件时，程序图中所有从源节点到汇节点的路径都被遍历到，则称这组测试用例集满足路径覆盖。

基于程序图的覆盖指标

- 节点覆盖最容易实现；
- 边覆盖在节点覆盖的基础上，更进了一步；
- 对于if-then这样的语句，如果采用节点覆盖，则会错过隐含的else子句；而采用边覆盖，则会考虑到这种情况。

```
int a;
scanf("%d",&a);
if(a==0)
        a=a+1;
a=a+2;
printf("%d",a)
```



基于程序图的覆盖指标

- 路径覆盖则在边覆盖的基础上又进了一步，考虑了从源节点到汇节点的所有路径；
- 如果一个程序中存在循环，则很难满足路径覆盖指标。

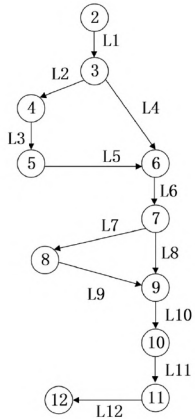
目录

示例：源码

```
1. Function fun(Int x, Int y, Int z):Int // 输出为整型类型
2.   Int m = n = 0;
3.   If((x > 0) and (y < 10))
4.      Then m = x + z;
5.           n = m * 3;
6.   EndIf
7.   If((x == 2) or (z > 6))
8.      Then n = x - y;
9.   EndIf
10.  n = m + n;
11.  Return n;   // 返回计算结果
12. End fun // 函数结束
```

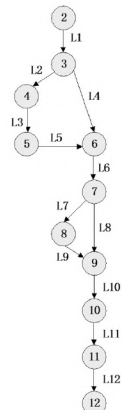
示例：程序图

```
1. Function fun(Int x, Int y, Int z):Int // 输出为整型类型
2.   Int m = n = 0;
3.   If((x > 0) and (y < 10))
4.     Then m = x + z;
5.          n = m * 3;
6.   EndIf
7.   If((x == 2) or (z > 6))
8.     Then n = x - y;
9.   EndIf
10.  n = m + n;
11.  Return n;   // 返回计算结果
12. End fun // 函数结束
```



节点覆盖测试用例

- 由于fun函数相对简单，所有的语句在出现在同一条路径上
- 只需要设计1条测试用例即可满足要求
- 白盒测试技术未指定如何设计测试用例，仅仅给出了测试用例应该满足的度量标准
- 可设计出完全不同的测试用例集合

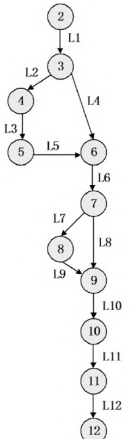


编号	输入			覆盖的节点
	x	y	z	
1	3	6	9	2，3，4，5，6，7，8，9，10，11，12

边覆盖测试用例

- 设计2条测试用例即可满足边覆盖要求

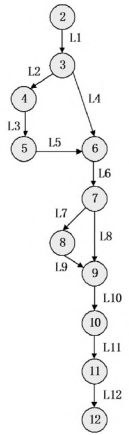
编号	输入			覆盖的边
	x	y	z	
1	2	5	8	L1，L2，L3，L5，L6，L7，L9，L10，L11，L12
2	3	12	5	L1，L4，L6，L8，L10，L11，L12



路径覆盖测试用例

- 4条路径
- P1 = {2-3-4-5-6-7-8-9-10-11-12}
- P2 = {2-3-4-5-6-7-9-10-11-12}
- P3 = {2-3-6-7-8-9-10-11-12}
- P4 = {2-3-6-7-9-10-11-12}

编号	输入			覆盖的路径
	x	y	z	
1	2	5	8	P1
2	3	5	3	P2
3	3	12	8	P3
4	3	12	5	P4



目录

- 覆盖率
  - 3种覆盖指标
  - 示例
- 逻辑覆盖指标

逻辑覆盖指标

- 语句覆盖
- 判定覆盖
- 条件覆盖
- 判定-条件覆盖
- 条件组合覆盖
- 路径覆盖

逻辑覆盖指标

- 语句覆盖是指设计若干个测试用例，当将它们作用于被测程序后，程序中的每一条可执行语句至少被执行一次。
- 由于程序图来源于程序代码，语句覆盖与节点覆盖相同。

逻辑覆盖指标

- 判定覆盖也称为分支覆盖，是指设计一组测试用例，当它们作用于被测软件时，程序中每个判定的取真分支和取假分支至少各执行一次。
- 判定覆盖等同于边覆盖。

逻辑覆盖指标

- 条件覆盖是指设计足够多的测试用例，当它们作用于被测软件后，程序中每个判定内的每个条件的各种可能取值至少被执行一次。
- 条件覆盖关注的是更为细致的条件取值情况，而不仅仅是整个判定的取值。

逻辑覆盖指标

- 判定-条件覆盖是判定覆盖和条件覆盖的结合，要求设计一组测试用例，针对被测程序运行完这些测试用例后，不仅程序中每个判定的各种取值至少被执行一次，而且每个判定中每个条件的各种取值也至少被执行一次。

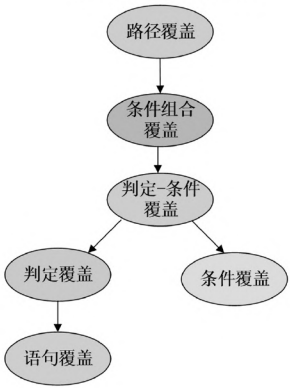
逻辑覆盖指标

- 条件组合覆盖是指设计足够多的测试用例，运行被测软件后，程序中每个判定的所有条件的可能取值组合都至少被执行一次。
- 如果一个判定中包含n个简单条件，为达到条件组合覆盖，需要执行2^n种情况。

逻辑覆盖指标

- 路径覆盖是指设计一组测试用例，当它们作用于被测软件时，程序中的所有路径都至少被覆盖一次。

逻辑覆盖指标



总结

- 覆盖率定义
- 6种测试覆盖指标