Structural Testing

Software Testing

Structural Testing

A complementary approach to functional testing is called structural / white box testing. It permits us to examine the internal structure of the program.

Path Testing

Path testing is the name given to a group of test techniques based on judiciously selecting a set of test paths through the program. If the set of paths is properly chosen, then it means that we have achieved some measure of test thoroughness.

This type of testing involves:

1. generating a set of paths that will cover every branch in the program.
2. finding a set of test cases that will execute every path in the set of program paths.

Software Testing

Flow Graph

The control flow of a program can be analysed using a graphical representation known as flow graph. The flow graph is a directed graph in which nodes are either entire statements or fragments of a statement, and edges represents flow of control.

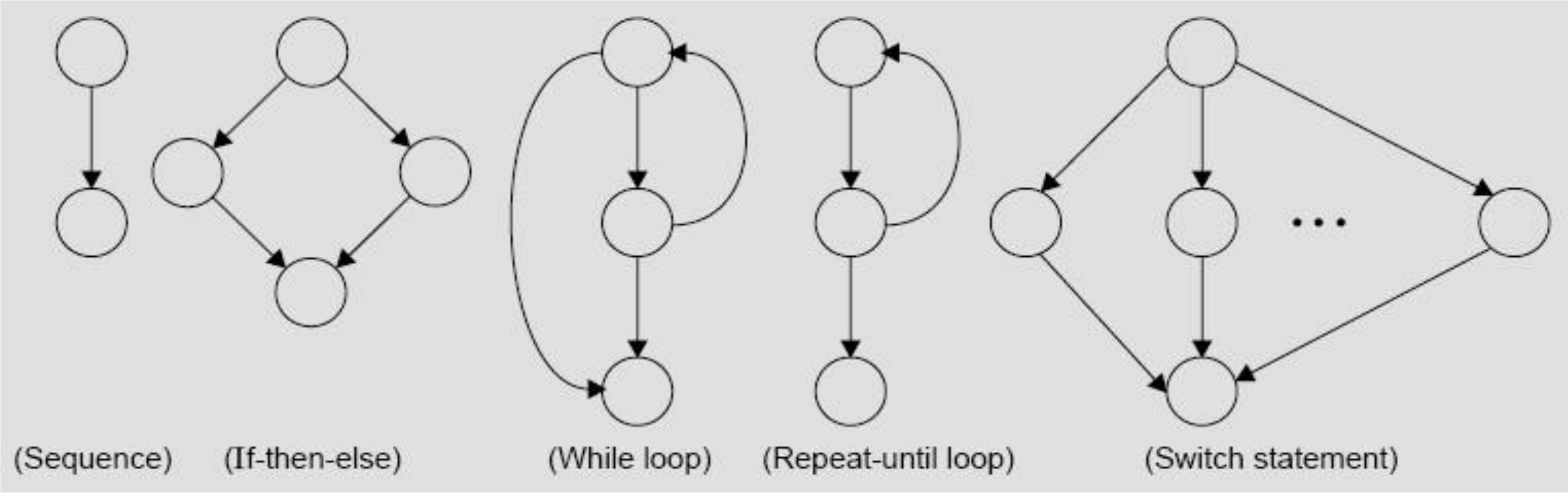


Fig. 14: The basic construct of the flow graph

```c
/* Program to generate the previous date given a date, assumes data
given as dd mm yyyy separated by space and performs error checks on the
validity of the current date entered. */

#include <stdio.h>
#include <conio.h>

int main()
{
    int day, month, year, validDate = 0;
  /*Date Entry*/
    printf(“Enter the day value: ”);
    scanf(“%d”, &day);
    printf(“Enter the month value: ”);
    scanf(“%d”, &month);
    printf(“Enter the year value: ”);
    scanf(“%d”,&year);
  /*Check Date Validity */
    if (year >= 1900 && year <= 2025) {
      if (month == 1 || month == 3 || month == 5 || month == 7 ||
        month == 8 || month == 10 || month == 12) {
```

(Contd.)...

```
          if (day >= 1 && day <= 31) {
              validDate = 1;
          }
          else {
            validDate = 0;
          }
        }
        else if (month == 2) {
          int rVal=0;
          if (year%4 == 0) {
            rVal=1;
            if ((year%100)==0 && (year % 400) !=0) {
              rVal=0;
            }
          }
          if (rVal ==1 && (day >=1 && day <=29) ) {
            validDate = 1;
          }
          else if (day >=1 && day <= 28 ) {
            validDate = 1;
          }
```

(Contd.)...

```
      else {
        validDate = 0;
      }
    }
    else if ((month >= 1 && month <= 12) && (day >= 1 && day <= 30)) {
      validDate = 1;
    }
    else {
      validDate = 0;
    }
  }
  /*Prev Date Calculation*/
  if (validDate) {
    if (day == 1) {
      if (month == 1) {
        year--;
        day=31;
        month=12;
      }
      else if (month == 3) {
        int rVal=0;
```

(Contd.)...

```
      if (year%4 == 0) {
        rVal=1;
        if ((year%100)==0 && (year % 400) !=0) {
      rVal=0;
    }
      }
      if (rVal ==1) {
        day=29;
        month--;
      }
      else {
        day=28;
        month--;
      }
    }
    else if (month == 2 || month == 4 || month == 6 || month == 9 || month == 11) {
      day = 31;
      month--;
```

```
        }
        else {
          day=30;
          month--;
        }
      }
      else {
        day--;
      }
      printf("The next date is: %d-%d-%d",day,month,year);
    }
    else {
      printf("The entered date ( %d-%d-%d ) is invalid",day,month, year);
    }
    getche ();
    return 1;
}
```

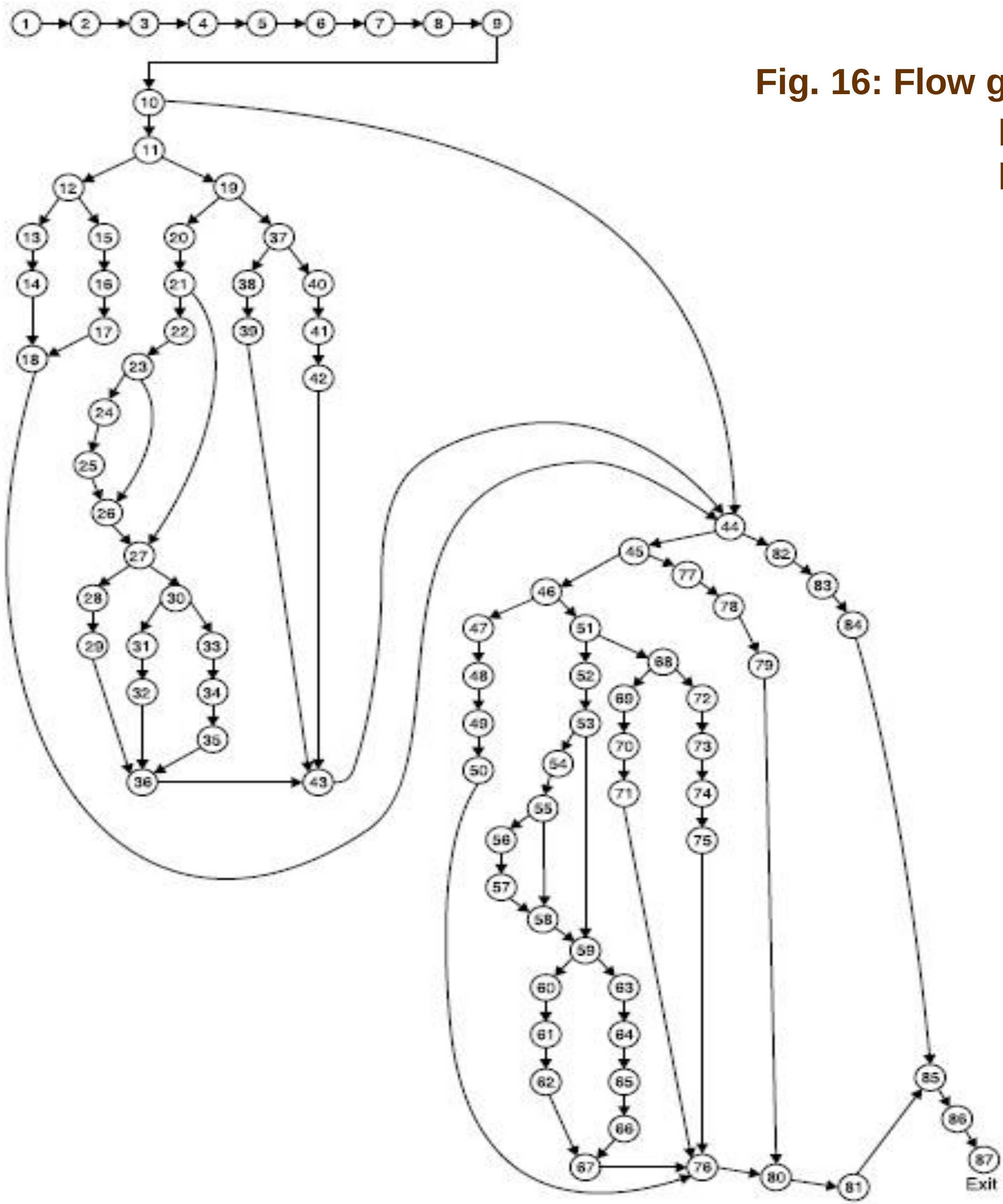


Fig. 16: Flow graph of previous date problem

DD (Decision to decision) Path Graph

Flow graph nodes	DD Path graph corresponding node	Remarks
1 to 9	n1	There is a sequential flow from node 1 to 9
10	n2	Decision node, if true go to 13 else go to 44
11	n3	Decision node, if true go to 12 else go to 19
12	n4	Decision node, if true go to 13 else go to 15
13,14	n5	Sequential nodes and are combined to form new node n_5
15,16,17	n6	Sequential nodes
18	n7	Edges from node 14 to 17 are terminated here
19	n8	Decision node, if true go to 20 else go to 37
20	n9	Intermediate node with one input edge and one output edge
21	n10	Decision node, if true go to 22 else go to 27
22	n11	Intermediate node
23	n12	Decision node, if true go to 24 else go to 26

Flow graph nodes	DD Path graph correspondin g node	Remarks
24,25	n13	Sequential nodes
26	n14	Two edges from node 25 & 23 are terminated here
27	n15	Two edges from node 26 & 21 are terminated here. Also a decision node
28,29	n16	Sequential nodes
30	n17	Decision node, if true go to 31 else go to 33
31,32	n18	Sequential nodes
33,34,35	n19	Sequential nodes
36	n20	Three edge from node 29,32 and 35 are terminated here
37	n21	Decision node, if true go to 38 else go to 40
38,39	n22	Sequential nodes
40,41,42	n23	Sequential nodes
43	n24	Three edge from node 36,39 and 42 are terminated here

Flow graph nodes	DD Path graph correspondi ng node	Remarks
44	n25	Decision node, if true go to 45 else go to 82. Three edges from 18,43 & 10 are also terminated here.
45	n26	Decision node, if true go to 46 else go to 77
46	n27	Decision node, if true go to 47 else go to 51
47,48,49,50	n28	Sequential nodes
51	n29	Decision node, if true go to 52 else go to 68
52	n30	Intermediate node with one input edge & one output ege
53	n31	Decision node, if true go to 54 else go to 59
54	n32	Intermediate node
55	n33	Decision node, if true go to 56 else go to 58
56,57	n34	Sequential nodes
58	n35	Two edge from node 57 and 55 are terminated here
59	n36	Decision node, if true go to 60 else go to 63. Two edge from nodes 58 and 53 are terminated.

Flow graph nodes	DD Path graph correspondin g node	Remarks
60,61,62	n37	Sequential nodes
63,64,65,66	n38	Sequential nodes
67	n39	Two edge from node 62 and 66 are terminated here
68	n40	Decision node, if true go to 69 else go to 72
69,70,71	n41	Sequential nodes
72,73,74,75	n42	Sequential nodes
76	n43	Four edges from nodes 50, 67, 71 and 75 are terminated here.
77,78,79	n44	Sequential nodes
80	n45	Two edges from nodes 76 & 79 are terminated here
81	n46	Intermediate node
82,83,84	n47	Sequential nodes
85	n48	Two edges from nodes 81 and 84 are terminated here
86,87	n49	Sequential nodes with exit node

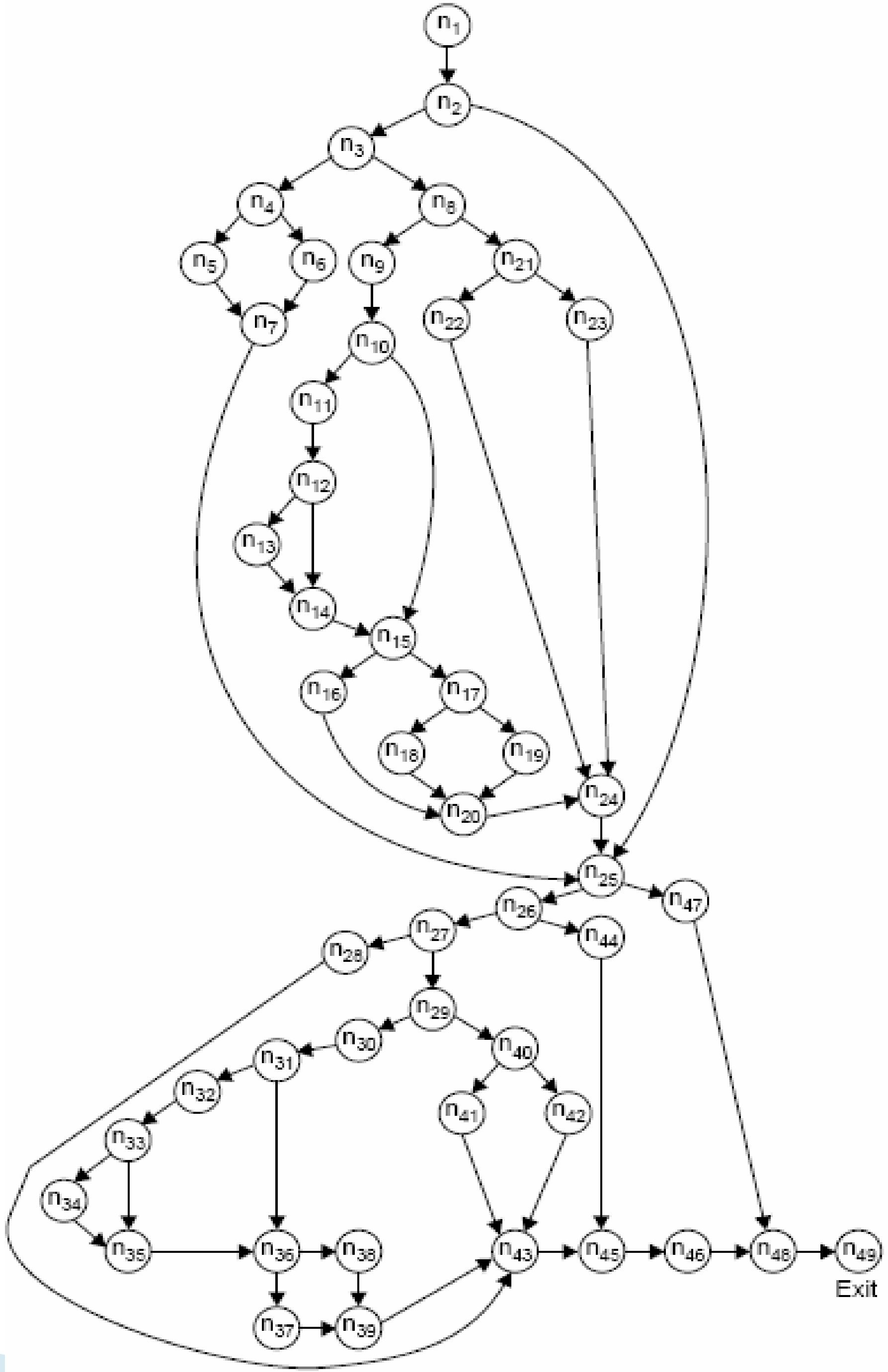


Fig. 17: DD path graph of previous date problem

	Independent paths of previous date problem
1	$n_1, n_2, n_{25}, n_{47}, n_{48}, n_{49}$
2	$n_1, n_2, n_3, n_4, n_5, n_7, n_{25}, n_{47}, n_{48}, n_{49}$
3	$n_1, n_2, n_3, n_4, n_6, n_7, n_{25}, n_{47}, n_{48}, n_{49}$
4	$n_1, n_2, n_3, n_8, n_{21}, n_{22}, n_{24}, n_{25}, n_{47}, n_{48}, n_{49}$
5	$n_1, n_2, n_3, n_8, n_{21}, n_{23}, n_{24}, n_{25}, n_{47}, n_{48}, n_{49}$
6	$n_1, n_2, n_3, n_8, n_9, n_{10}, n_{15}, n_{17}, n_{19}, n_{20}, n_{24}, n_{25}, n_{47}, n_{48}, n_{49}$
7	$n_1, n_2, n_3, n_8, n_9, n_{10}, n_{15}, n_{17}, n_{18}, n_{20}, n_{24}, n_{25}, n_{47}, n_{48}, n_{49}$
8	$n_1, n_2, n_3, n_8, n_9, n_{10}, n_{11}, n_{12}, n_{13}, n_{14}, n_{15}, n_{17}, n_{18}, n_{20}, n_{24}, n_{25}, n_{47}, n_{48}, n_{49}$
9	$n_1, n_2, n_3, n_8, n_9, n_{10}, n_{11}, n_{12}, n_{14}, n_{15}, n_{17}, n_{18}, n_{20}, n_{24}, n_{25}, n_{47}, n_{48}, n_{49}$
10	$n_1, n_2, n_3, n_8, n_9, n_{10}, n_{15}, n_{16}, n_{20}, n_{24}, n_{25}, n_{47}, n_{48}, n_{49}$
11	$n_1, n_2, n_3, n_8, n_9, n_{10}, n_{15}, n_{16}, n_{20}, n_{24}, n_{25}, n_{26}, n_{44}, n_{45}, n_{46}, n_{48}, n_{49}$
12	$n_1, n_2, n_3, n_8, n_9, n_{11}, n_{12}, n_{14}, n_{15}, n_{16}, n_{20}, n_{24}, n_{25}, n_{26}, n_{27}, n_{28}, n_{43}, n_{45}, n_{46}, n_{48}, n_{49}$
13	$n_1, n_2, n_3, n_8, n_9, n_{10}, n_{11}, n_{12}, n_{14}, n_{15}, n_{16}, n_{20}, n_{24}, n_{25}, n_{26}, n_{27}, n_{29}, n_{40}, n_{41}, n_{43}, n_{45}, n_{46}, n_{48}, n_{49}$
14	$n_1, n_2, n_3, n_8, n_9, n_{10}, n_{11}, n_{12}, n_{14}, n_{15}, n_{16}, n_{20}, n_{24}, n_{25}, n_{26}, n_{27}, n_{29}, n_{40}, n_{42}, n_{43}, n_{45}, n_{46}, n_{48}, n_{49}$
15	$n_1, n_2, n_3, n_8, n_{21}, n_{22}, n_{24}, n_{25}, n_{26}, n_{27}, n_{29}, n_{30}, n_{31}, n_{36}, n_{38}, n_{39}, n_{43}, n_{45}, n_{46}, n_{48}, n_{49}$
16	$n_1, n_2, n_3, n_8, n_{21}, n_{22}, n_{24}, n_{25}, n_{26}, n_{27}, n_{29}, n_{30}, n_{31}, n_{36}, n_{37}, n_{39}, n_{43}, n_{45}, n_{46}, n_{48}, n_{49}$
17	$n_1, n_2, n_3, n_8, n_{21}, n_{22}, n_{24}, n_{25}, n_{26}, n_{27}, n_{29}, n_{30}, n_{31}, n_{32}, n_{33}, n_{34}, n_{35}, n_{36}, n_{37}, n_{39}, n_{43}, n_{45}, n_{46}, n_{48}, n_{49}$
18	$n_1, n_2, n_3, n_8, n_{21}, n_{22}, n_{24}, n_{25}, n_{26}, n_{27}, n_{29}, n_{30}, n_{31}, n_{32}, n_{33}, n_{35}, n_{36}, n_{37}, n_{39}, n_{43}, n_{45}, n_{46}, n_{48}, n_{49}$

Fig. 18: Independent paths of previous date problem

Software Testing

Example

Consider the program given in for the classification of a triangle. Its input is a triple of positive integers (say a,b,c) from the interval [1,100]. The output may be [Scalene, Isosceles, Equilateral, Not a triangle].

Draw the flow graph & DD Path graph. Also find the independent paths from the DD Path graph.

```
#include <stdio.h>
#include <conio.h>
int main()
{
  int  a,b,c,validInput=0;
  printf("Enter the side 'a' value: ");
  scanf("%d",&a);
  printf("Enter the side 'b' value: ")
  scanf("%d",&b);
  printf("Enter the side 'c' value:");
  scanf("%d",&c);
  if ((a > 0) && (a <= 100) && (b > 0) && (b <= 100) && (c > 0)
      && (c <= 100)) {
    if ( (a + b) > c) && ((c + a) > b) && ((b + c) > a)) {
      validInput = 1;
    }
  }
  else {
    validInput = -1;
  }
  If (validInput==1) {
    If ((a==b) && (b==c)) {
      printf("The trinagle is equilateral");
    }
    else if ( (a == b) || (b == c) || (c == a) ) {
```

```
          printf(“The triangle is isosceles”);
        }
        else {
          printf(“The trinagle is scalene”);
        }
      }
      else if (validInput == 0) {
        printf(“The values do not constitute a Triangle”);
      }
      else {
        printf(“The inputs belong to invalid range”);
      }
      getche();
      return 1;
    }
```

Solution :

Flow graph of triangle problem is:

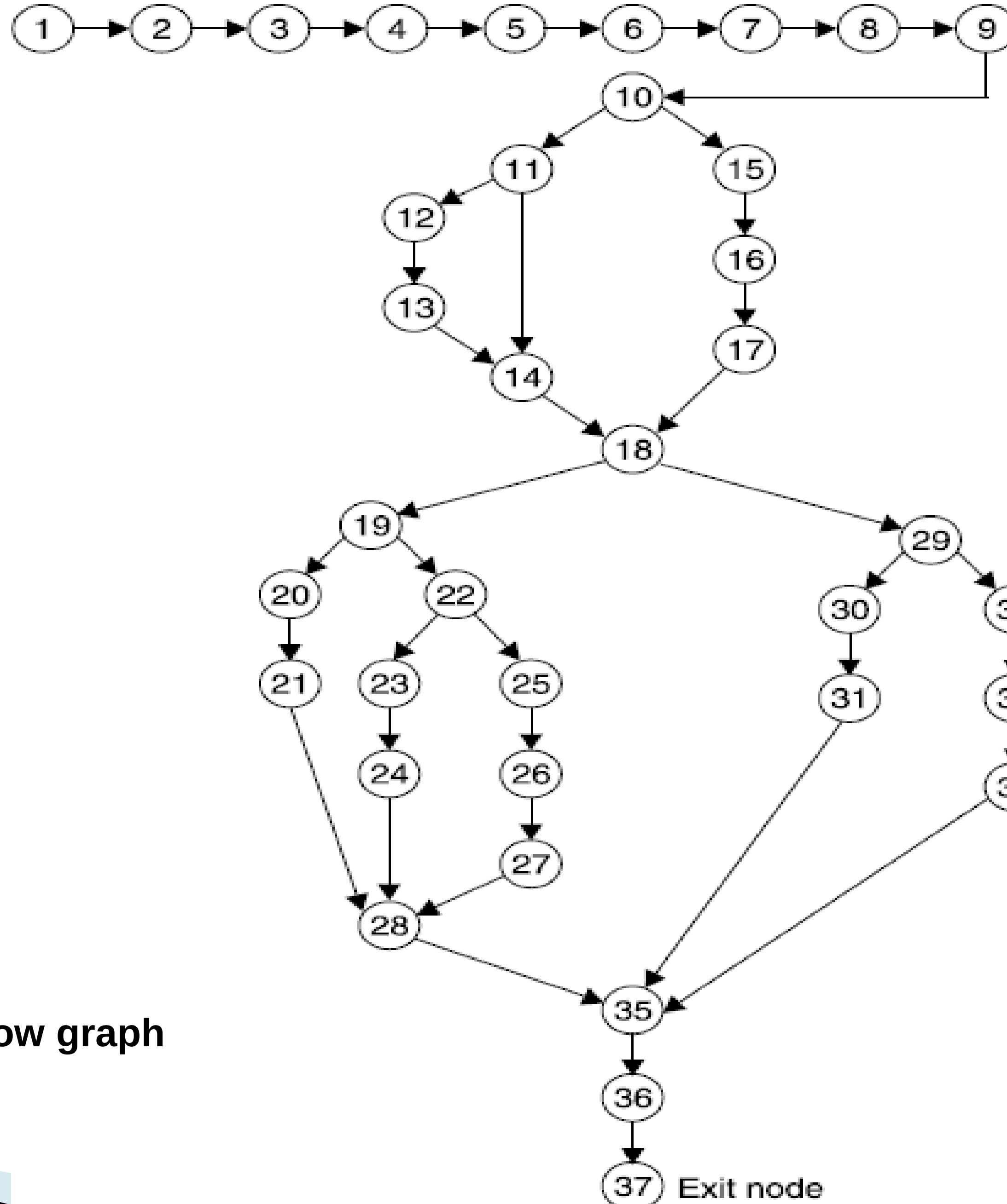


Fig.8. 20 (a): Program flow graph

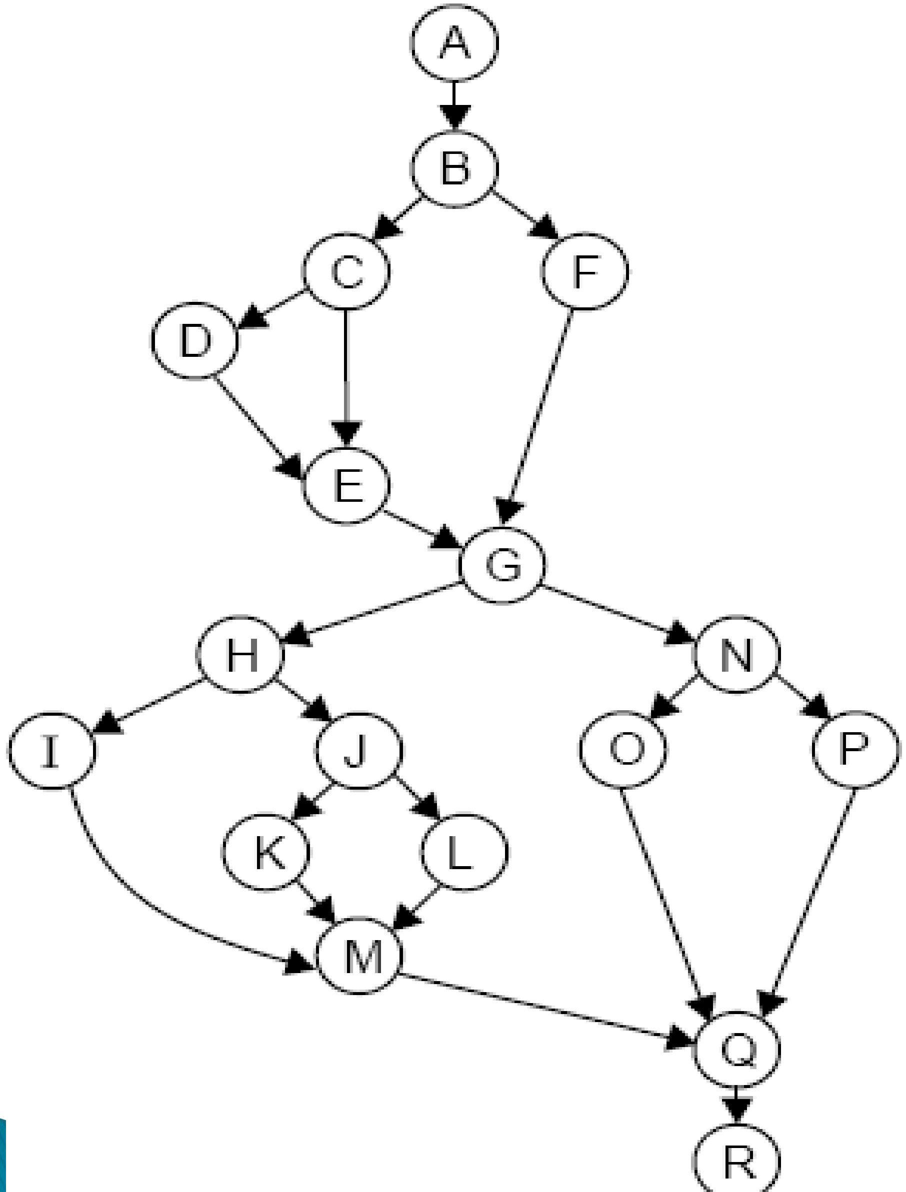
The mapping table for DD path graph is:

Flow graph nodes	DD Path graph corresponding node	Remarks
1 TO 9	A	Sequential nodes
10	B	Decision node
11	C	Decision node
12, 13	D	Sequential nodes
14	E	Two edges are joined here
15, 16, 17	F	Sequential nodes
18	G	Decision nodes plus joining of two edges
19	H	Decision node
20, 21	I	Sequential nodes
22	J	Decision node
23, 24	K	Sequential nodes
25, 26, 27	L	Sequential nodes

Flow graph nodes	DD Path graph corresponding node	Remarks
28	M	Three edges are combined here
29	N	Decision node
30, 31	O	Sequential nodes
32, 33, 34	P	Sequential nodes
35	Q	Three edges are combined here
36, 37	R	Sequential nodes with exit node

Fig. 20 (b): DD Path graph

DD Path graph is given in Fig. 20 (b)



Independent paths are:

(i) ABFGNPQR
(ii) ABFGNOQR
(iii) ABCEGNPQR
(iv) ABCDEGNOQR
(v) ABFGHIMQR
(vi)ABFGHJKMQR
(vii)ABFGHLJMQR

Fig. 20 (b): DD Path graph

Cyclomatic Complexity

Software Testing

- **Cyclomatic Complexity**
  - Also known as structural complexity
  - Complexity measure is defined in terms of independent paths
  - Provides an upper bound on the number of test cases that must be conducted to ensure that all the statements have been tested at least once and every condition has been tested on its true side and false side.

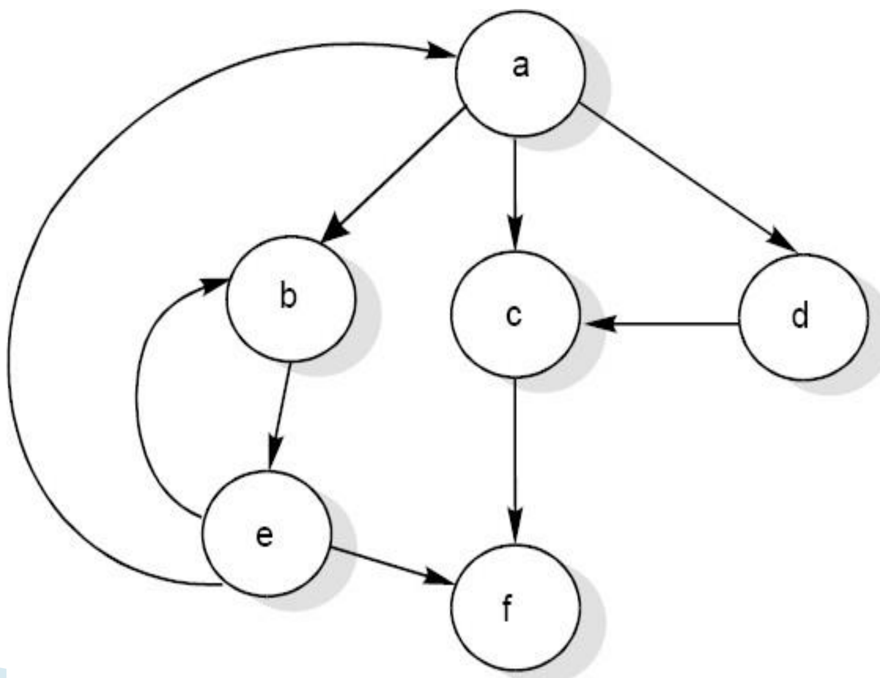
Software Testing

Cyclomatic Complexity

McCabe's cyclomatic metric V(G) = *e* – *n* + 2P.

Where e = number of edges; n = number of nodes; p = number of connected components

For example, a flow graph shown under with entry node '*a*' and exit node '*f*'.



Software Testing

The value of cyclomatic complexity can be calculated as :

$$V(G) = 9 - 6 + 2 = 5$$

Here $e = 9$, $n = 6$ and $P = 1$

There will be five independent paths for the flow graph illustrated in Fig. 21.

Path 1 : *a c f*

Path 2 : *a b e f*

Path 3 : *a d c f*

Path 4 : *a b e a c f* **or** *a b e a b e f*

Path 5 : *a b e b e f*

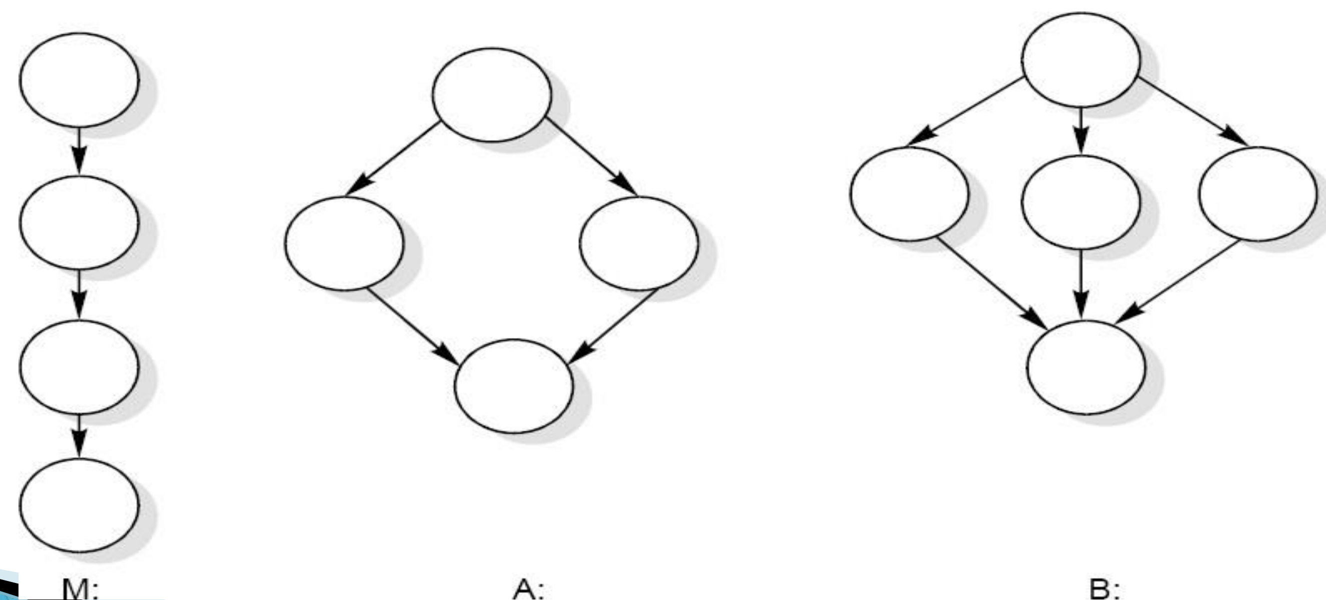
Software Testing

Several properties of cyclomatic complexity are stated below:

1. $V(G) \geq 1$
2. V (G) is the maximum number of independent paths in graph G.
3. Inserting & deleting functional statements to G does not affect V(G).
4. G has only one path if and only if V(G)=1.
5. V(G) depends only on the decision structure of G.

Software Testing

The role of P in the complexity calculation V(G)=e-n+2P is required to be understood correctly. We define a flow graph with unique entry and exit nodes, all nodes reachable from the entry, and exit reachable from all nodes. This definition would result in all flow graphs having only one connected component for a program but this is useful when combined complexity of multiple programs is to be calculated.



Software Testing

Let us denote the total graph above with 3 connected components as

$$V(M \cup A \cup B) = e - n + 2P$$

$$= 13-13+2*3$$

$$= 6$$

This method with P $\neq$ 1 can be used to calculate the complexity of a collection of programs

Software Testing

Notice that $\boldsymbol{V(M \cup A \cup B) = V(M) + V(A) + V(B) = 6}$. In general, the complexity of a collection C of flow graphs with K connected components is equal to the summation of their complexities. To see this let C_i,$1 \le i \le K$ denote the k distinct connected component, and let e_i and n_i be the number of edges and nodes in the ith-connected component. Then

$$V(C) = e - n + 2p = \sum_{i=1}^{k} e_i - \sum_{i=1}^{k} n_i + 2K$$

$$= \sum_{i=1}^{k} (e_i - n_i + 2) = \sum_{i=1}^{k} V(C_i)$$

Software Testing

Two alternate methods are available for the complexity calculations.

1. Cyclomatic complexity V(G) of a flow graph G is equal to the number of predicate (decision) nodes plus one.

   $V(G)=\prod+1$

   Where $\prod$ is the number of predicate nodes contained in the flow graph G.

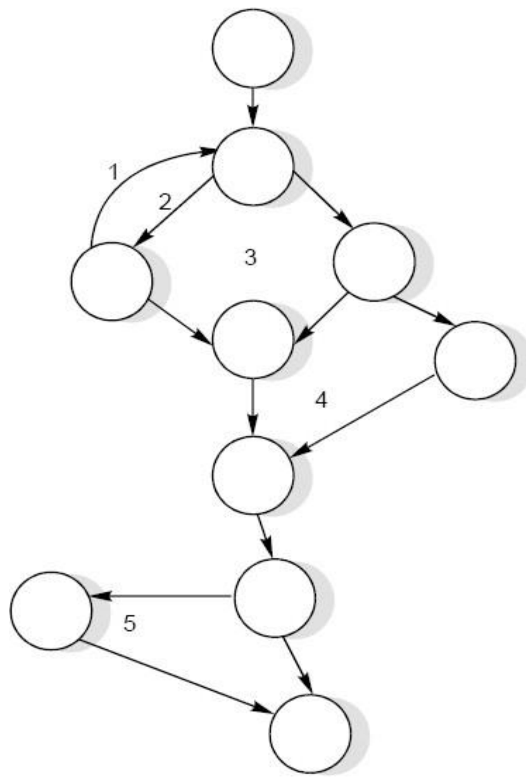
   Restriction is that every predicate node must have only two conditions. If not, it needs to be converted so

2. Cyclomatic complexity is equal to the number of regions of the flow graph.

Software Testing

Example

Consider a flow graph given in Fig. 23 and calculate the cyclomatic complexity by all three methods.



Software Testing

Solution

Cyclomatic complexity can be calculated by any of the three methods.

1. V(G) $= e - n + 2P$

   $= 13 - 10 + 2 = 5$

2. V(G) $= \pi + 1$

   $= 4 + 1 = 5$

3. V(G) = number of regions

   = 5

Therefore, complexity value of a flow graph in Fig. 23 is 5.

Software Testing

Example

Consider the previous date program with DD path graph. Find cyclomatic complexity.

Software Testing

Solution

Number of edges (e) = 65

Number of nodes (n) =49

(i) V(G) = e – n + 2P = 65 – 49 + 2 = 18

(ii) V(G) = π + 1 = 17 + 1 = 18

(iii) V(G) = Number of regions = 18

The cyclomatic complexity is 18.

Software Testing

Example 8.18

Consider the classification of triangle problem. Find the cyclomatic complexity.

Software Testing

Solution

Number of edges (e) = 23

Number of nodes (n) =18

(i) V(G) = $e - n + 2P = 23 - 18 + 2 = 7$

(ii) V(G) = $\pi + 1 = 6 + 1 = 7$

(iii) V(G) = Number of regions =7

The cyclomatic complexity is 7. Hence, there are seven independent paths.