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.

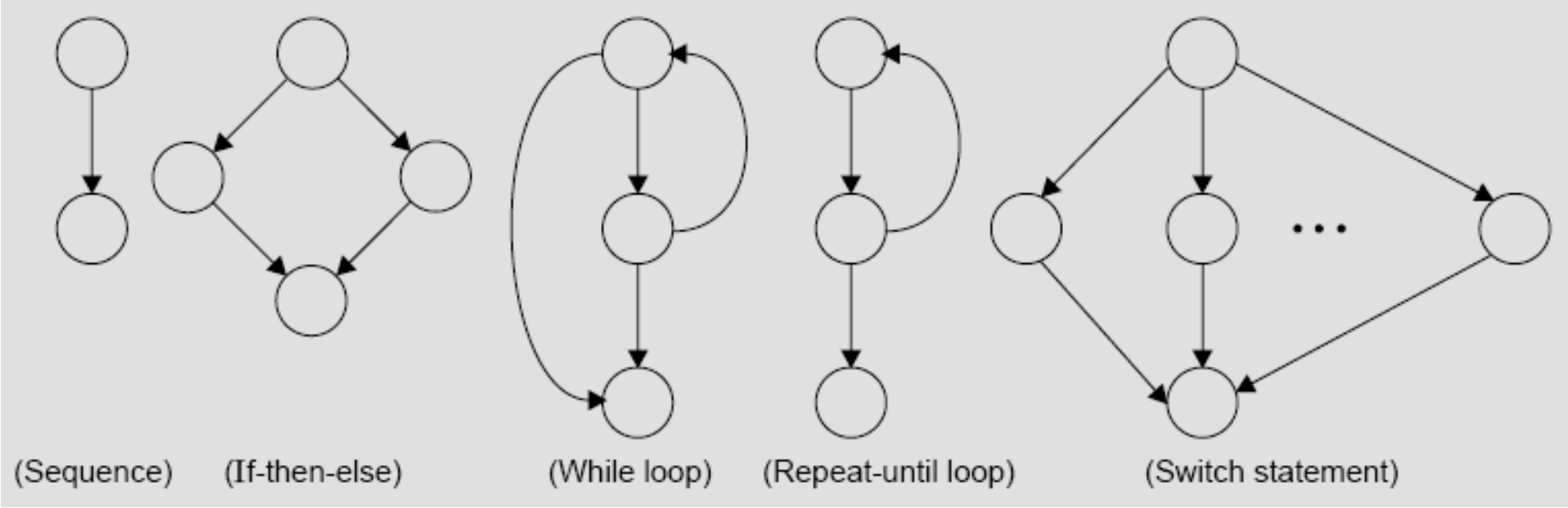


Fig. 14: The basic construct of the flow graph

Software Testing

```
/* 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>

 1  int main()
 2  {
 3    int day, month, year, validDate = 0;
    /*Date Entry*/
 4    printf("Enter the day value: ");
 5    scanf("%d", &day);
 6    printf("Enter the month value: ");
 7    scanf("%d", &month);
 8    printf("Enter the year value: ");
 9    scanf("%d",&year);
    /*Check Date Validity */
10    if (year >= 1900 && year <= 2025) {
11      if (month == 1 || month == 3 || month == 5 || month == 7 ||
          month == 8 || month == 10 || month == 12) {
```

(Contd.)...

Software Testing

```
12          if (day >= 1 && day <= 31) {
13              validDate = 1;
14          }
15          else {
16            validDate = 0;
17          }
18        }
19        else if (month == 2) {
20          int rVal=0;
21          if (year%4 == 0) {
22            rVal=1;
23            if ((year%100)==0 && (year % 400) !=0) {
24              rVal=0;
25            }
26          }
27          if (rVal ==1 && (day >=1 && day <=29) ) {
28            validDate = 1;
29          }
30          else if (day >=1 && day <= 28 ) {
31            validDate = 1;
32          }
```

(Contd.)...

Software Testing

```
33          else {
34            validDate = 0;
35          }
36        }
37        else if ((month >= 1 && month <= 12) && (day >= 1 && day <= 30)) {
38          validDate = 1;
39        }
40        else {
41          validDate = 0;
42        }
43      }
        /*Prev Date Calculation*/
44      if (validDate) {
45        if (day == 1) {
46          if (month == 1) {
47            year--;
48            day=31;
49            month=12;
50          }
51          else if (month == 3) {
52            int rVal=0;
```

(Contd.)...

Software Testing

```
53          if (year%4 == 0) {
54            rVal=1;
55            if ((year%100)==0 && (year % 400) !=0) {
56          rVal=0;
57        }
58          }
59          if (rVal ==1) {
60            day=29;
61            month--;
62          }
63          else {
64            day=28;
65            month--;
66          }
67        }
68        else if (month == 2 || month == 4 || month == 6 || month == 9 ||
          month == 11) {
69          day = 31;
70          month--;
```

(Contd.)...

```
71          }
72          else {
73             day=30;
74             month--;
75          }
76       }
77       else {
78          day--;
79       }
80       printf("The next date is: %d-%d-%d",day,month,year);
81    }
82    else {
83       printf("The entered date ( %d-%d-%d ) is invalid",day,month, year);
84    }
85    getche ();
86    return 1;
87 }
```

Fig. 15: Program for previous date problem

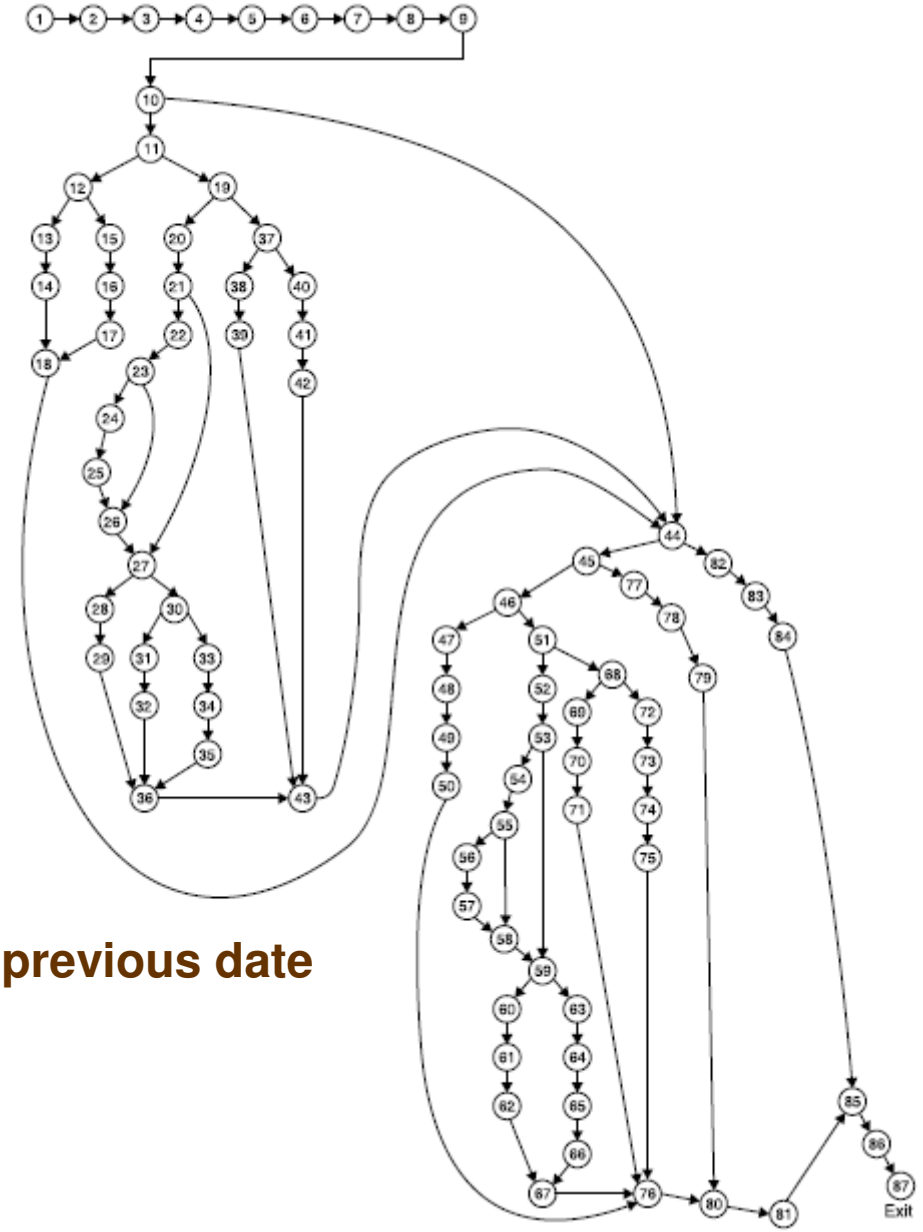


Fig. 16: Flow graph of previous date problem

DD Path Graph

Table 7: Mapping of flow graph nodes and DD path nodes

Flow graph nodes	DD Path graph corresponding node	Remarks
1 to 9	n_1	There is a sequential flow from node 1 to 9
10	n_2	Decision node, if true go to 13 else go to 44
11	n_3	Decision node, if true go to 12 else go to 19
12	n_4	Decision node, if true go to 13 else go to 15
13,14	n_5	Sequential nodes and are combined to form new node n_5
15,16,17	n_6	Sequential nodes
18	n_7	Edges from node 14 to 17 are terminated here
19	n_8	Decision node, if true go to 20 else go to 37
20	n_9	Intermediate node with one input edge and one output edge
21	n_{10}	Decision node, if true go to 22 else go to 27
22	n_{11}	Intermediate node
23	n_{12}	Decision node, if true go to 24 else go to 26

Flow graph nodes	DD Path graph corresponding node	Remarks
24,25	n_{13}	Sequential nodes
26	n_{14}	Two edges from node 25 & 23 are terminated here
27	n_{15}	Two edges from node 26 & 21 are terminated here. Also a decision node
28,29	n_{16}	Sequential nodes
30	n_{17}	Decision node, if true go to 31 else go to 33
31,32	n_{18}	Sequential nodes
33,34,35	n_{19}	Sequential nodes
36	n_{20}	Three edge from node 29,32 and 35 are terminated here
37	n_{21}	Decision node, if true go to 38 else go to 40
38,39	n_{22}	Sequential nodes
40,41,42	n_{23}	Sequential nodes
43	n_{24}	Three edge from node 36,39 and 42 are terminated here

Cont….

Flow graph nodes	DD Path graph corresponding node	Remarks
44	n_{25}	Decision node, if true go to 45 else go to 82. Three edges from 18,43 & 10 are also terminated here.
45	n_{26}	Decision node, if true go to 46 else go to 77
46	n_{27}	Decision node, if true go to 47 else go to 51
47,48,49,50	n_{28}	Sequential nodes
51	n_{29}	Decision node, if true go to 52 else go to 68
52	n_{30}	Intermediate node with one input edge & one output ege
53	n_{31}	Decision node, if true go to 54 else go to 59
54	n_{32}	Intermediate node
55	n_{33}	Decision node, if true go to 56 else go to 58
56,57	n_{34}	Sequential nodes
58	n_{35}	Two edge from node 57 and 55 are terminated here
59	n_{36}	Decision node, if true go to 60 else go to 63. Two edge from nodes 58 and 53 are terminated.

Cont….

Flow graph nodes	DD Path graph corresponding node	Remarks
60,61,62	n_{37}	Sequential nodes
63,64,65,66	n_{38}	Sequential nodes
67	n_{39}	Two edge from node 62 and 66 are terminated here
68	n_{40}	Decision node, if true go to 69 else go to 72
69,70,71	n_{41}	Sequential nodes
72,73,74,75	n_{42}	Sequential nodes
76	n_{43}	Four edges from nodes 50, 67, 71 and 75 are terminated here.
77,78,79	n_{44}	Sequential nodes
80	n_{45}	Two edges from nodes 76 & 79 are terminated here
81	n_{46}	Intermediate node
82,83,84	n_{47}	Sequential nodes
85	n_{48}	Two edges from nodes 81 and 84 are terminated here
86,87	n_{49}	Sequential nodes with exit node

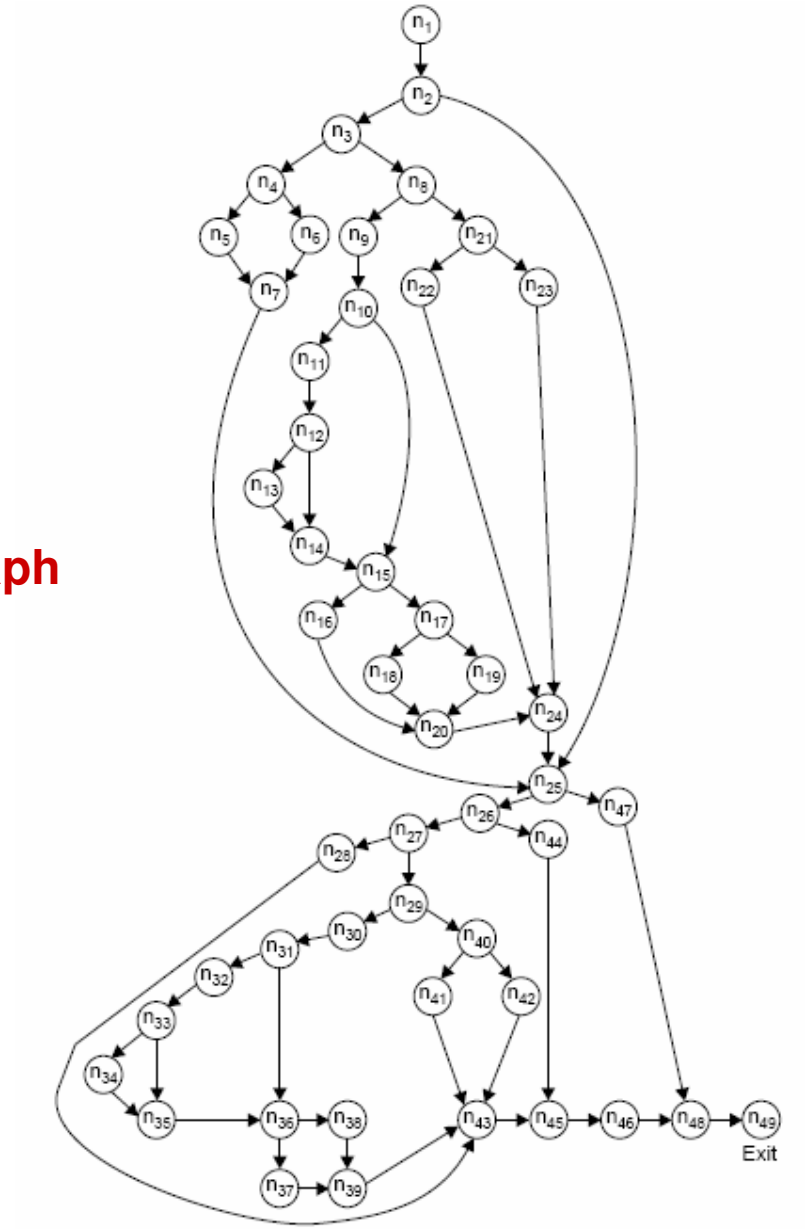


Fig. 17: DD path graph of previous date problem

	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

Example 8.13

Consider the problem for the determination of the nature of roots of a quadratic equation. Its input a triple of positive integers (say a,b,c) and value may be from interval [0,100].

The program is given in fig. 19. The output may have one of the following words:

[Not a quadratic equation; real roots; Imaginary roots; Equal roots]

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

Software Testing

```
#include <conio.h>
#include <math.h>
1       int main()
2     {
3       int a,b,c,validInput=0,d;
4       double D;
5       printf("Enter the 'a' value: ");
6       scanf("%d",&a);
7       printf("Enter the 'b' value: ");
8       scanf("%d",&b);
9       printf("Enter the 'c' value: ");
10      scanf("%d",&c);
11      if ((a >= 0) && (a <= 100) && (b >= 0) && (b <= 100) && (c >= 0)
          && (c <= 100)) {
12        validInput = 1;
13        if (a == 0) {
14          validInput = -1;
15        }
16      }
17      if (validInput==1) {
18        d = b*b - 4*a*c;
19        if (d == 0) {
20          printf("The roots are equal and are r1 = r2 = %f\n",
                  -b/(2*(float) a));
```

Cont....

```
21        }
22        else if ( d > 0 ) {
23          D=sqrt(d);
24          printf("The roots are real and are r1 = %f and r2 = %f\n",
                    (-b-D)/(2* a), (-b+D)/(2* a));
25        }
26        else {
27          D=sqrt(-d)/(2*a);
28          printf("The roots are imaginary and are r1 = (%f,%f) and
                    r2 = (%f,%f)\n", -b/(2.0*a),D,-b/(2.0*a),-D);
29        }
30      }
31      else if (validInput == -1) {
32        printf("The vlaues do not constitute a Quadratic equation.");
33      }
34      else {
35        printf("The inputs belong to invalid range.");
36      }
37      getche();
38      return 1;
39    }
```

Fig. 19: Code of quadratic equation problem

Solution

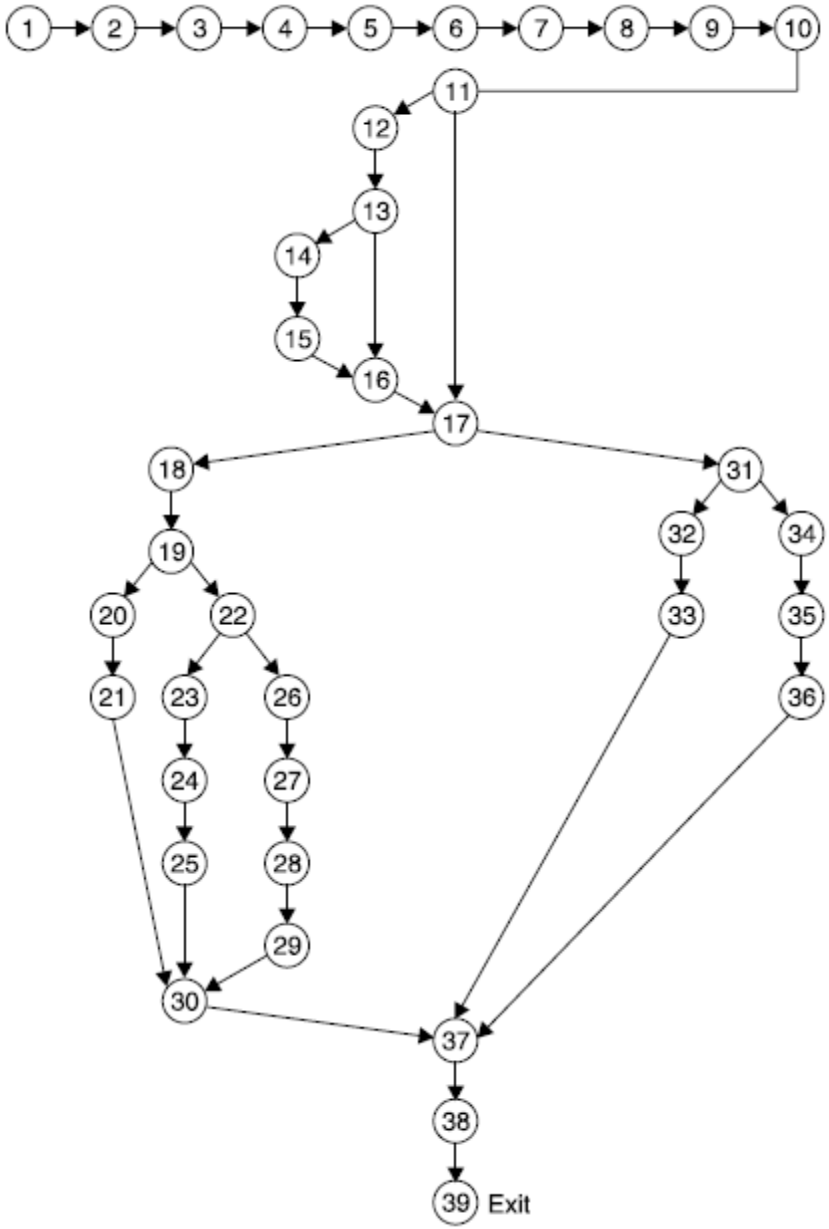


Fig. 19 (a) : Program flow graph

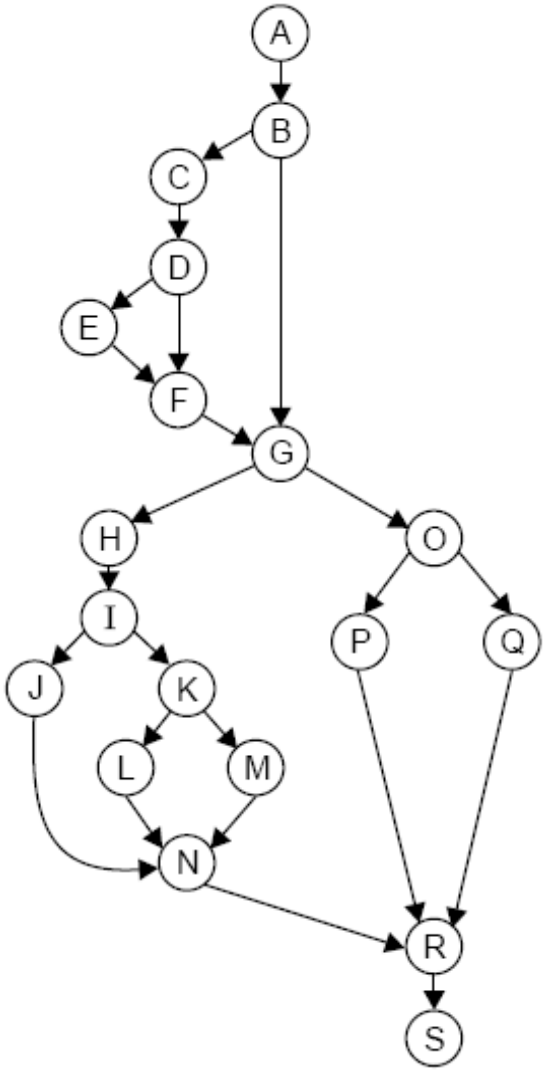


Fig. 19 (b) : DD Path graph

The mapping table for DD path graph is:

Flow graph nodes	DD Path graph corresponding node	Remarks
1 to 10	A	Sequential nodes
11	B	Decision node
12	C	Intermediate node
13	D	Decision node
14,15	E	Sequential node
16	F	Two edges are combined here
17	G	Two edges are combined and decision node
18	H	Intermediate node
19	I	Decision node
20,21	J	Sequential node
22	K	Decision node
23,24,25	L	Sequential node

Cont….

Flow graph nodes	DD Path graph corresponding node	Remarks
26,27,28,29	M	Sequential nodes
30	N	Three edges are combined
31	O	Decision node
32,33	P	Sequential node
34,35,36	Q	Sequential node
37	R	Three edges are combined here
38,39	S	Sequential nodes with exit node

Independent paths are:

(i) ABGOQRS

(ii) ABGOPRS

(iii) ABCDFGOQRS

(iv) ABCDEFGOPRS

(v) ABGHIJNRS

(vi) ABGHIKLNRS

(vi) ABGHIKMNRS

Example 8.14

Consider a program given in Fig.8.20 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.

Software Testing

```
#include <stdio.h>
#include <conio.h>
1    int main()
2    {
3      int a,b,c,validInput=0;
4      printf("Enter the side 'a' value: ");
5      scanf("%d",&a);
6      printf("Enter the side 'b' value: ")
7      scanf("%d",&b);
8      printf("Enter the side 'c' value:");
9      scanf("%d",&c);
10     if ((a > 0) && (a <= 100) && (b > 0) && (b <= 100) && (c > 0)
           && (c <= 100)) {
11       if ( (a + b) > c) && ((c + a) > b) && ((b + c) > a)) {
12         validInput = 1;
13       }
14     }
15     else {
16       validInput = -1;
17     }
18     If (validInput==1) {
19       If ((a==b) && (b==c)) {
20         printf("The trinagle is equilateral");
21       }
22       else if ( (a == b) || (b == c) || (c == a) ) {
```

(Contd.)...

```
23          printf("The triangle is isosceles");
24        }
25        else {
26          printf("The trinagle is scalene");
27        }
28      }
29      else if (validInput == 0) {
30        printf("The values do not constitute a Triangle");
31      }
32      else {
33        printf("The inputs belong to invalid range");
34      }
35      getche();
36      return 1;
37    }
```

Fig. 20 : Code of triangle classification problem

Solution :

Flow graph of triangle problem is:

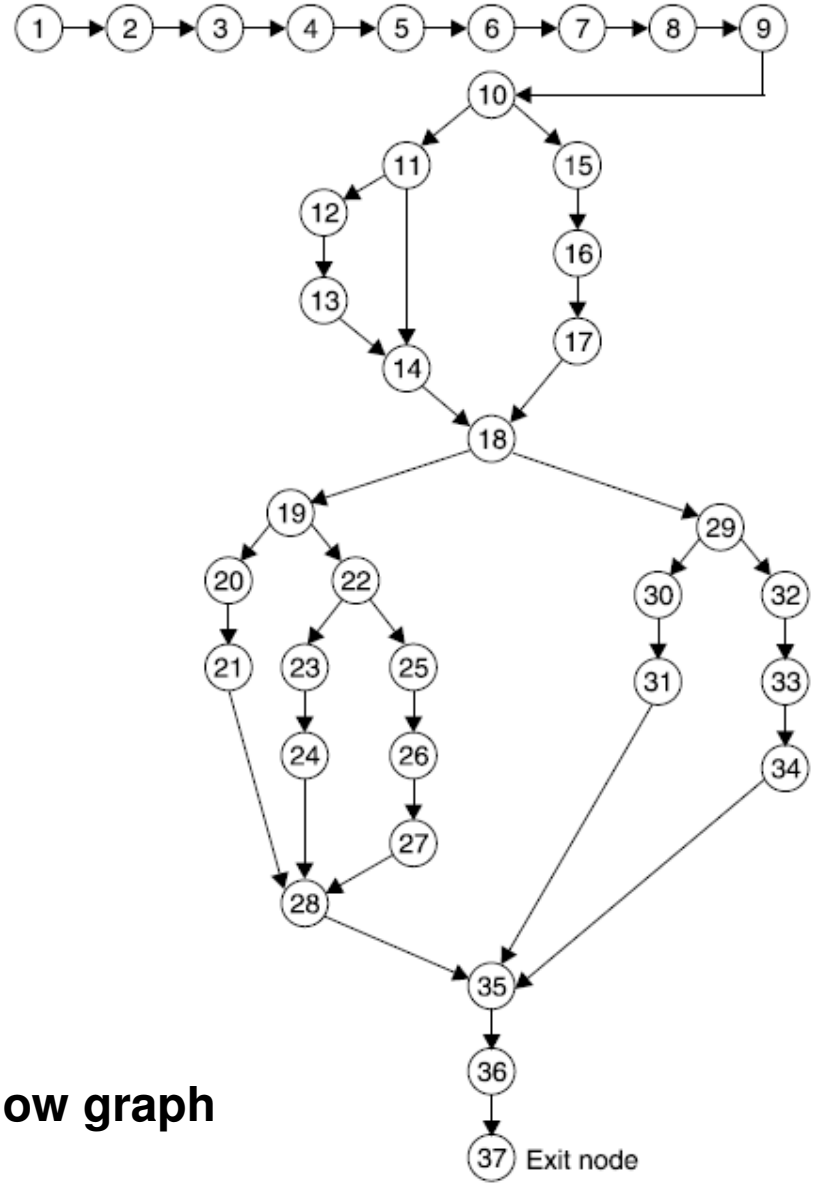


Fig.8. 20 (a): Program flow graph

The mapping table for DD path graph is:

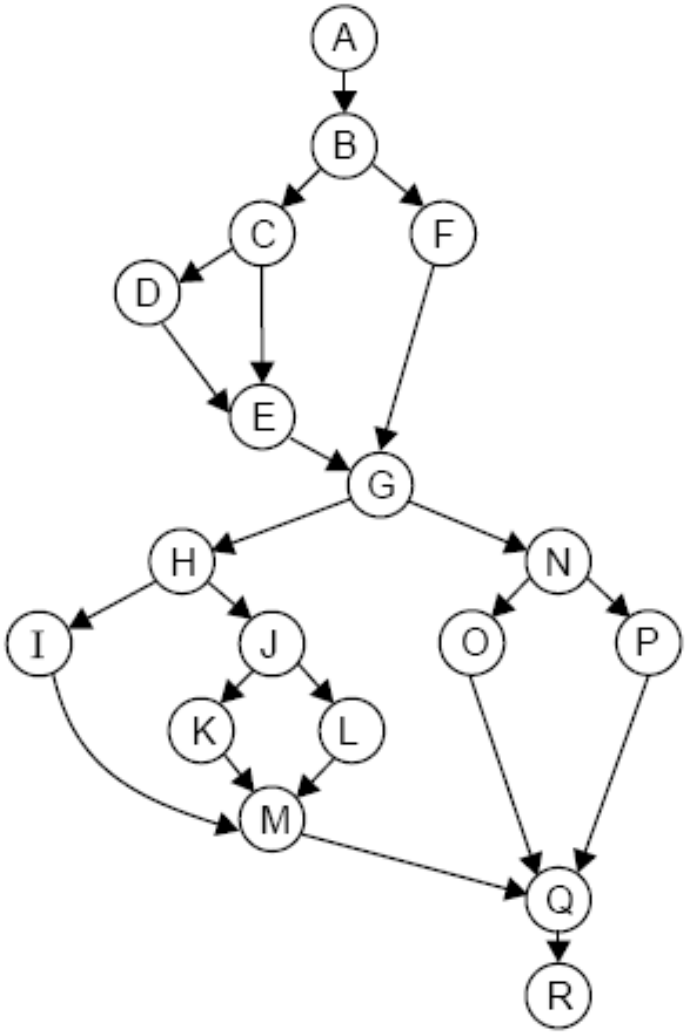
Flow graph nodes	DD Path graph corresponding node	Remarks
1 TO 9	A	Sequential nodes
10	B	Decision node
11	C	Decision node
12, 13	D	Sequential nodes
14	E	Two edges are joined here
15, 16, 17	F	Sequential nodes
18	G	Decision nodes plus joining of two edges
19	H	Decision node
20, 21	I	Sequential nodes
22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

Cont….

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) ABFGHJMQR

Fig. 20 (b): DD Path graph

Cyclomatic Complexity

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

For example, a flow graph shown in in Fig. 21 with entry node '*a*' and exit node '*f*'.

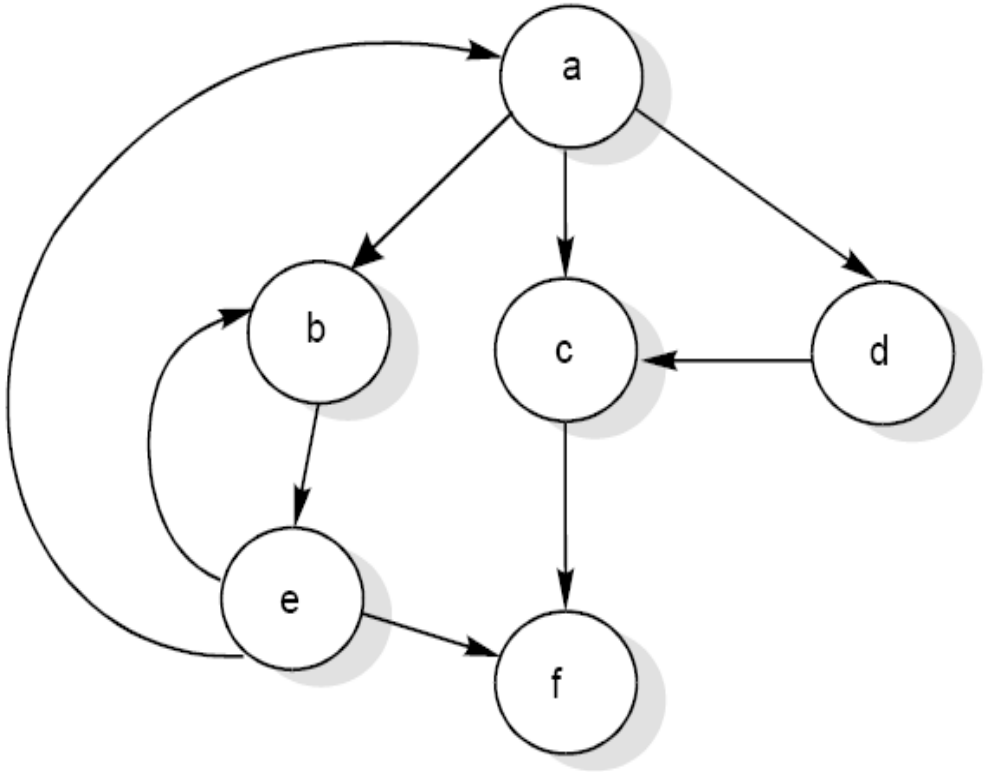


Fig. 21: Flow graph

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*

Several properties of cyclomatic complexity are stated below:

1. V(G) ≥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. Inserting a new row in G increases V(G) by unity.
6. V(G) depends only on the decision structure of G.

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. One could, however, imagine a main program M and two called subroutines A and B having a flow graph shown in Fig. 22.

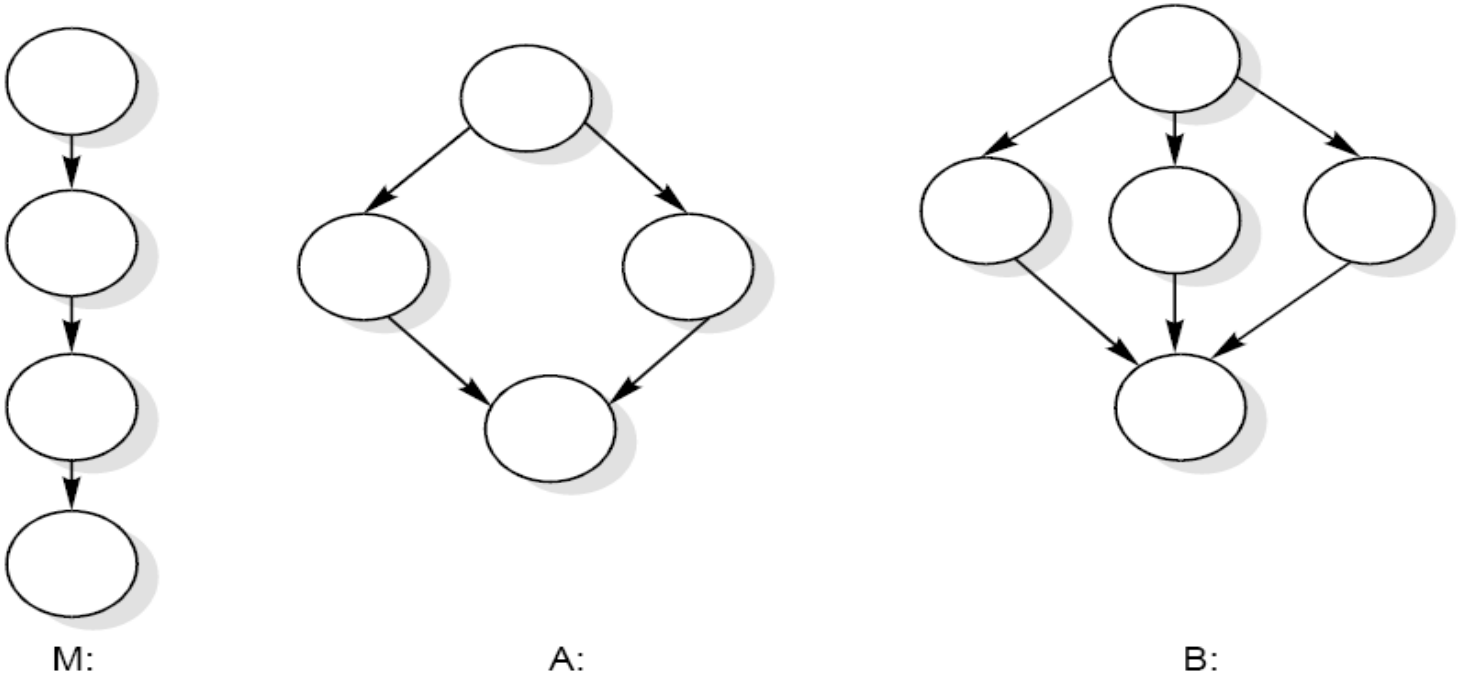


Fig. 22

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

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

$$= 13\text{-}13+2*3$$

$$= 6$$

This method with P ≠ 1 can be used to calculate the complexity of a collection of programs, particularly a hierarchical nest of subroutines.

Notice that $V(M \cup A \cup B) = V(M) + V(A) + V(B) = 6$. 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, $1 \leq I \leq K$ 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.

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

Example 8.15

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

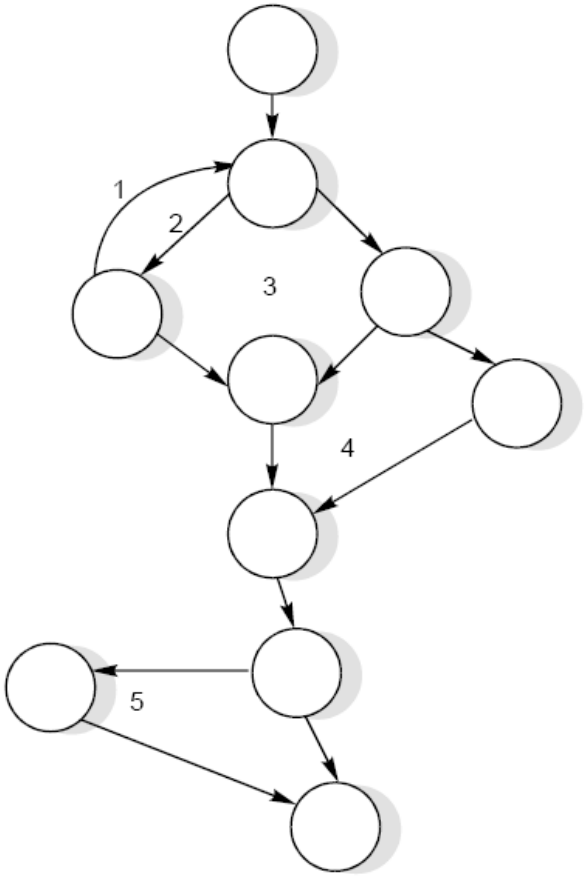


Fig. 23

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.

Example 8.16

Consider the previous date program with DD path graph given in Fig. 17. Find cyclomatic complexity.

Solution

Number of edges (e) = 65

Number of nodes (n) =49

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

(ii) $V(G) = \pi + 1 = 17 + 1 = 18$

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

The cyclomatic complexity is 18.

Example 8.17

Consider the quadratic equation problem given in example 8.13 with its DD Path graph. Find the cyclomatic complexity:

Solution

Number of nodes (n) = 19

Number of edges (e) = 24

(i) $V(G) = e - n + 2P = 24 - 19 + 2 = 7$

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

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

Hence cyclomatic complexity is 7 meaning thereby, seven independent paths in the DD Path graph.

Example 8.18

Consider the classification of triangle problem given in example 8.14. Find the cyclomatic complexity.

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 as given in example 8.14.