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.**

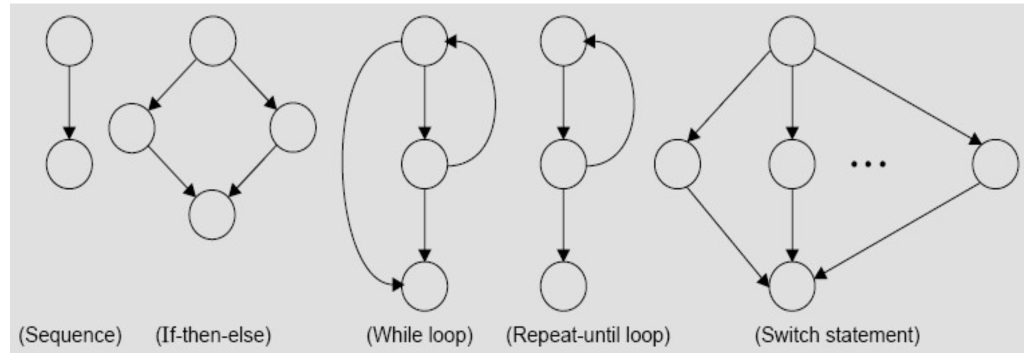


Fig. 14: The basic construct of the flow graph

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.


```
#include <conio.h>
#include <math.h>
    int main()
  {
    int a,b,c,validInput=0,d;
    double D;
    printf("Enter the 'a' value: ");
    scanf("%d",&a);
    printf("Enter the 'b' value: ");
    scanf("%d",&b);
    printf("Enter the 'c' value: ");
    scanf("%d",&c);
    if ((a >= 0) && (a <= 100) && (b >= 0) && (b <= 100) && (c >= 0)
      && (c <= 100)) {
      validInput = 1;
      if (a == 0) {
        validInput = -1;
      }
    }
    if (validInput==1) {
      d = b*b - 4*a*c;
      if (d == 0) {
        printf("The roots are equal and are r1 = r2 = %f\n",
               -b/(2*(float) a));
```

```
      }
      else if ( d > 0 ) {
        D=sqrt(d);
        printf("The roots are real and are r1 = %f and r2 = %f\n",
               (-b-D)/(2* a), (-b+D)/(2* a));
      }
      else {
        D=sqrt(-d)/(2*a);
        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);
      }
    }
    else if (validInput == -1) {
      printf("The vlaues do not constitute a Quadratic equation.");
    }
    else {
      printf("The inputs belong to invalid range.");
    }
    getche();
    return 1;
}
```

Fig. 19: Code of quadratic equation problem

Solution

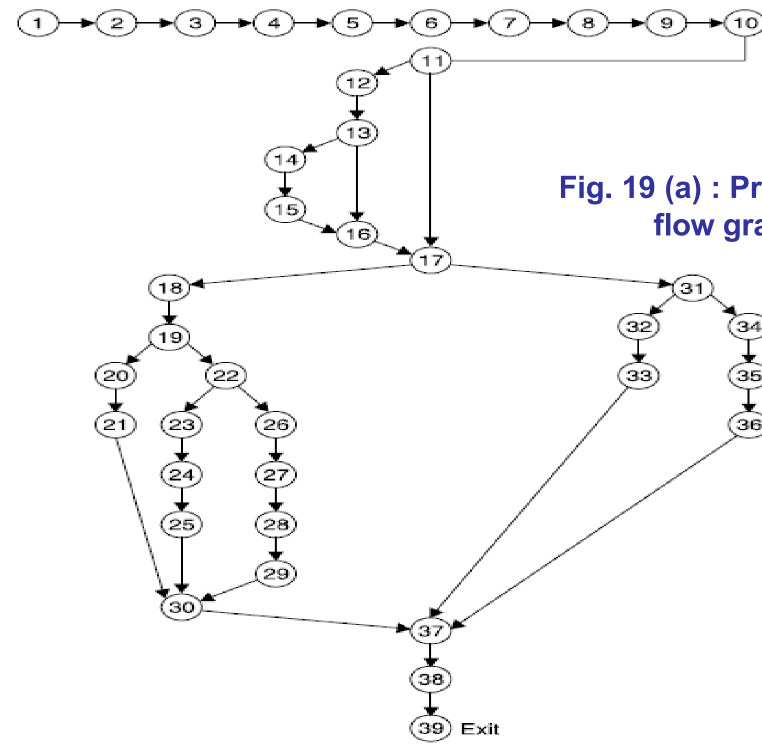


Fig. 19 (a) : Program flow graph

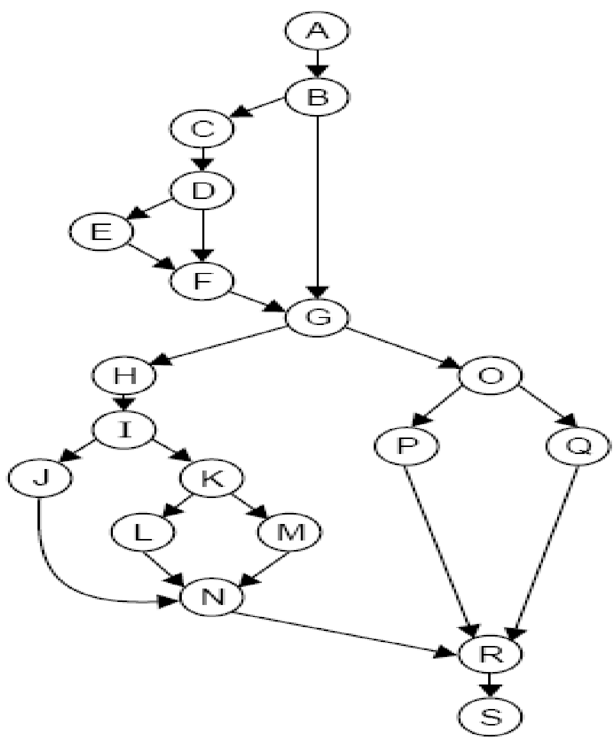


Fig. 19 (b) : DD Path graph

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

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

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*'.**

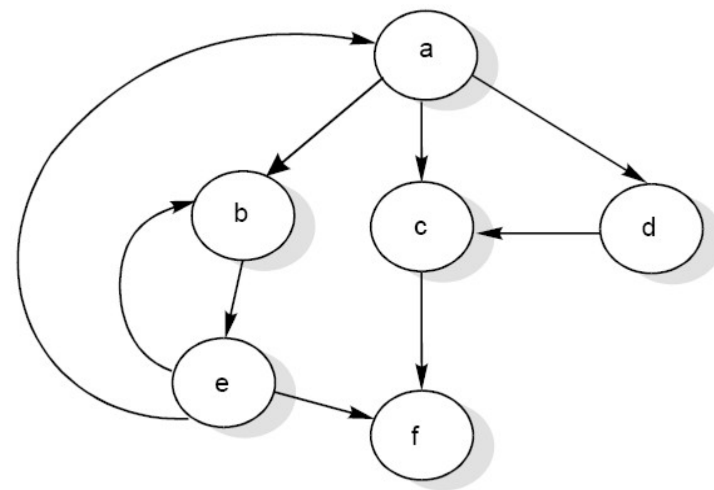


Fig. 21: Flow graph

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*
Path 5 : *a b e f 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.**

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**.

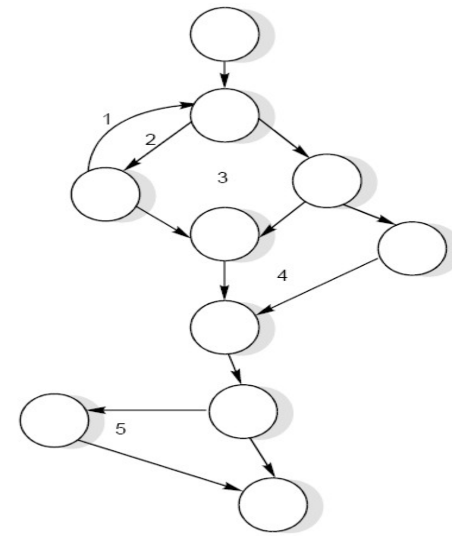
   $V(G)=\pi+1$

   Where: π **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.



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 is 5.

Example 8.17

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

Solution

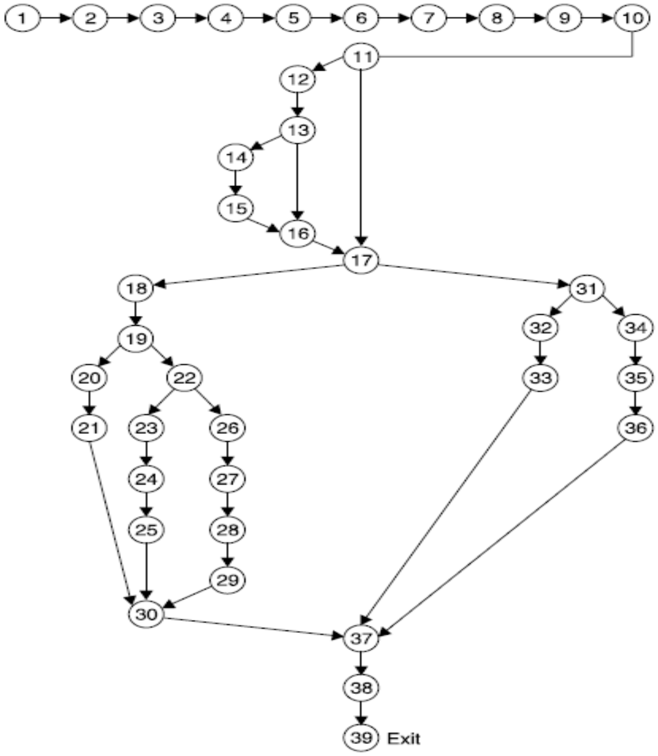
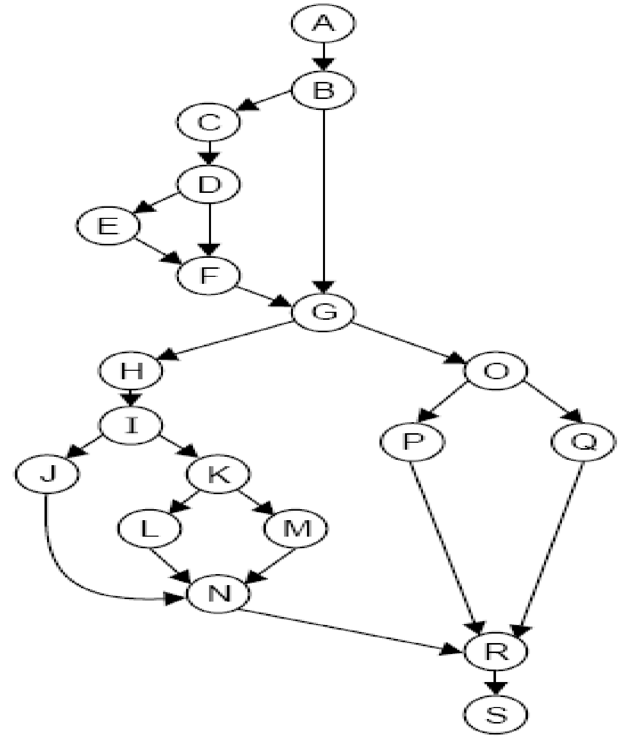


Fig. 19 (a) : Program flow graph

Fig. 19 (b) : DD Path graph



Solution

Number of nodes (n) = 19

Number of edges (e) = 24

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

(ii) V(G) = π + 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 :

Flow graph of triangle problem is:

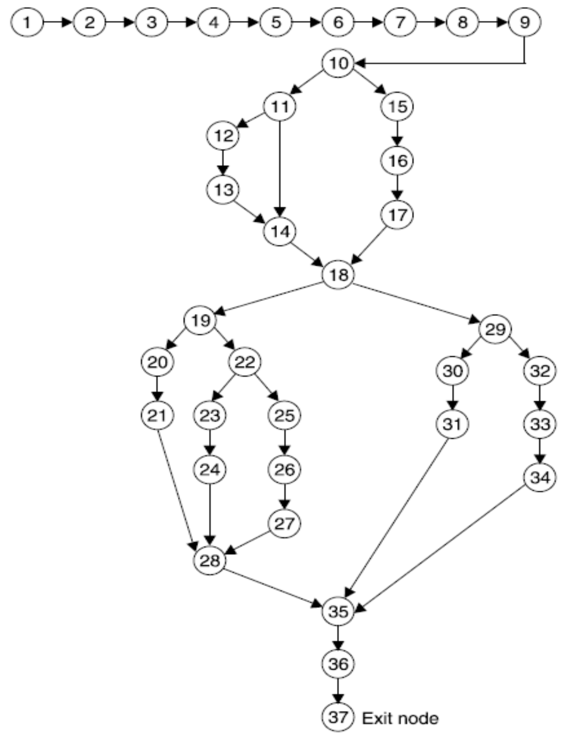


Fig.8. 20 (a): Program flow graph

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

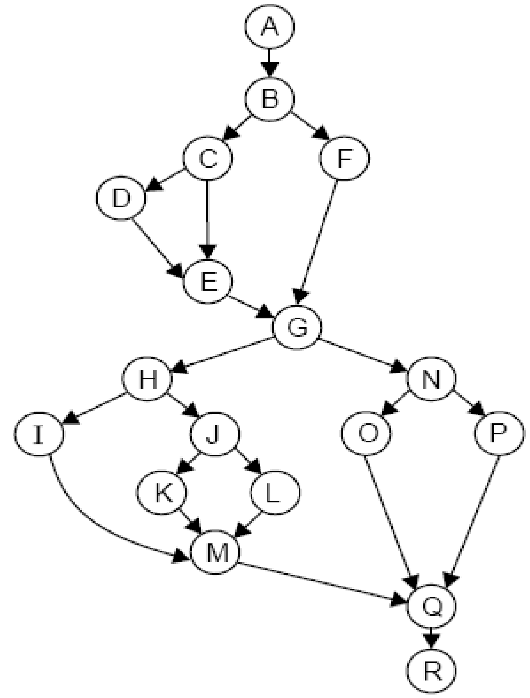


Fig. 20 (b): DD Path graph

Independent paths are:

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

Solution

Number of edges (e) =

23

Number of nodes (n) =18

(i) V(G) = $e - n$ + 2P = 23 – 18 + 2

= 7

(ii) V(G) = π + 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.