The cyclomatic complexity of a code section is the quantitative measure of the number of linearly independent paths in it. It is a software metric used to indicate the complexity of a program. It is computed using the Control Flow Graph of the program. The nodes in the graph indicate the smallest group of commands of a program, and a directed edge in it connects the two nodes i.e. if the second command might immediately follow the first command.

For example, if the source code contains no control flow statement then its cyclomatic complexity will be 1, and the source code contains a single path in it. Similarly, if the source code contains one if condition then cyclomatic complexity will be 2 because there will be two paths one for true and the other for false.

Mathematically, for a structured program, the directed graph inside the control flow is the edge joining two basic blocks of the program as control may pass from first to second.

So, cyclomatic complexity M would be defined as,

$M = E – N + 2P$ where E = the number of edges in the control flow graph

N = the number of nodes in the control flow graph

P = the number of connected components

In case, when the exit point is directly connected back to the entry point. Here, the graph is strongly connected, and cyclomatic complexity is defined as

$M = E – N + P$

where

E = the number of edges in the control flow graph

N = the number of nodes in the control flow graph

P = the number of connected components

In the case of a single method, P is equal to 1. So, for a single subroutine, the formula can be defined as

$M = E – N + 2$

where

E = the number of edges in the control flow graph

N = the number of nodes in the control flow graph

P = the number of connected components

How to Calculate Cyclomatic Complexity?

Steps that should be followed in calculating cyclomatic complexity and test cases design are:

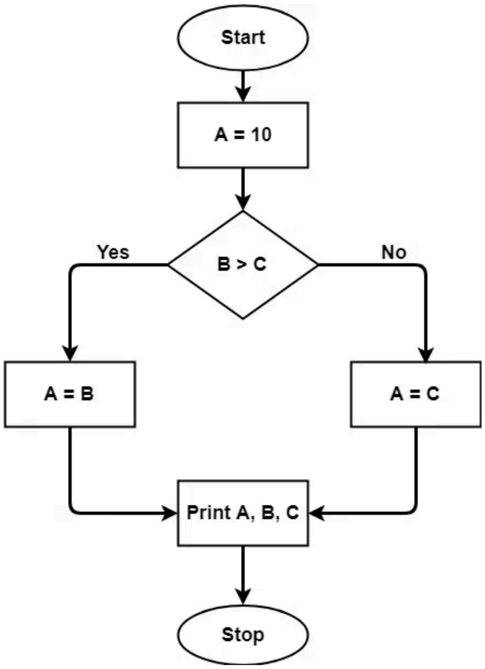
Construction of graphs with nodes and edges from code.

- Identification of independent paths.
- Cyclomatic Complexity Calculation
- Design of Test Cases

Let a section of code as such:

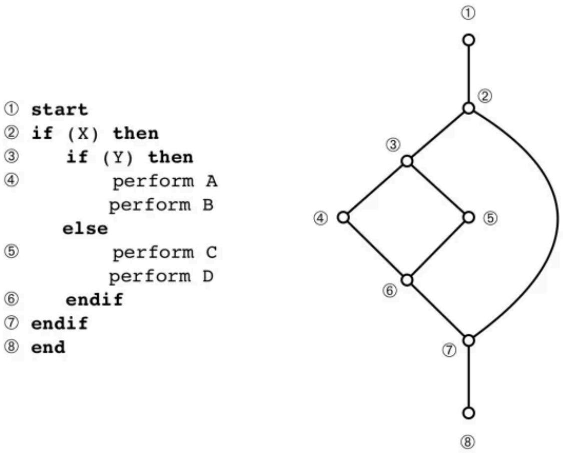
```
A = 10
  IF B > C THEN
    A = B
  ELSE
    A = C
  ENDIF
Print A
Print B
Print C
```

Control Flow Graph of the above code



The cyclomatic complexity calculated for the above code will be from the control flow graph. The graph shows seven shapes(nodes), and seven lines(edges), hence cyclomatic complexity is 7-7+2 = 2.

Consider the following program segment, and associated cyclomatic graph:



```
① start
② if (X) then
③    if (Y) then
④        perform A
         perform B
     else
⑤        perform C
         perform D
⑥    endif
⑦ endif
⑧ end
```

In this case the number of nodes is 8, the number of edges 9, C=9-8+2= 3