Differences Between Black Box Testing and White Box Testing are listed below.

Criteria	Black Box Testing	White Box Testing
Definition	Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is NOT known to the tester	White Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.
Levels Applicable To	Mainly applicable to higher levels of testing:Acceptance Testing System Testing	Mainly applicable to lower levels of testing:Unit Testing Integration Testing
Responsibility	Generally, independent Software Testers	Generally, Software Developers
Programming Knowledge	Not Required	Required
Implementation Knowledge	Not Required	Required
Basis for Test Cases	Requirement Specifications	Detail Design

WHITE BOX TESTING (also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing) is a software testing method in which the internal structure/design/implementation of the item being tested is known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. Programming know-how and the implementation knowledge is essential. White box testing is testing beyond the user interface and into the nitty-gritty of a system.

This method is named so because the software program, in the eyes of the tester, is like a white/transparent box; inside which one clearly sees.

Levels Applicable To

White Box Testing method is applicable to the following levels of software testing:

- Unit Testing: For testing paths within a unit.
- Integration Testing: For testing paths between units.
- System Testing: For testing paths between subsystems.

However, it is mainly applied to Unit Testing.

Advantages

- Testing can be commenced at an earlier stage. One need not wait for the GUI to be available.
- Testing is more thorough, with the possibility of covering most paths.

Disadvantages

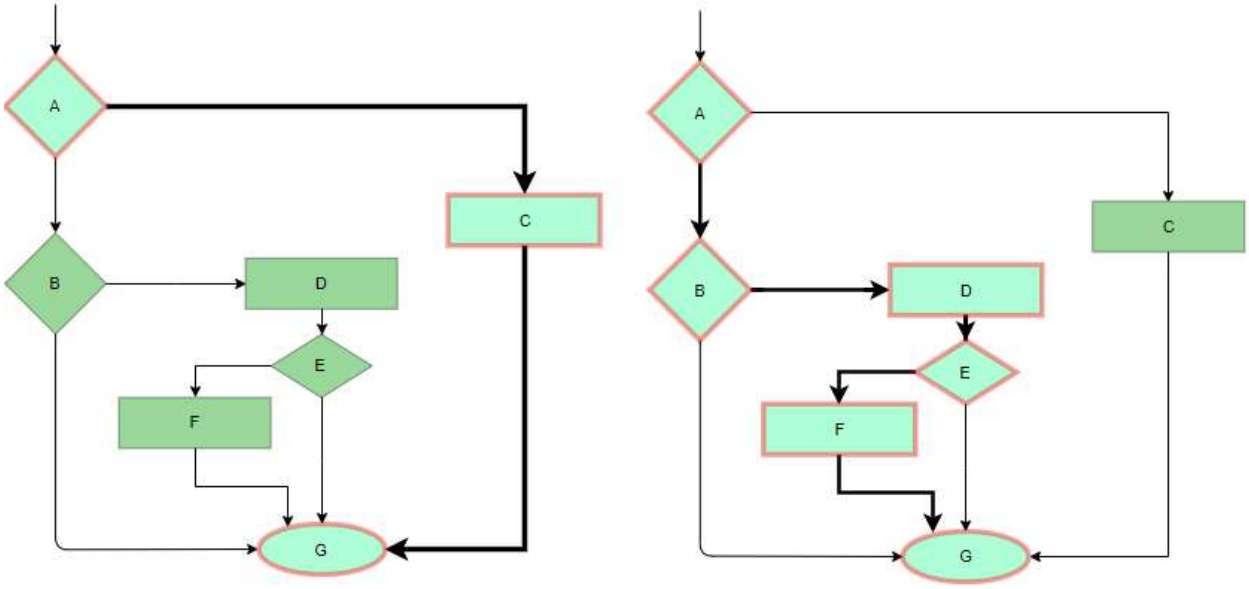
- White box testing can be quite complex and expensive.
- Developers who usually execute white box test cases detest it. The white box testing by developers is not detailed can lead to production errors.
- White box testing requires professional resources, with a detailed understanding of programming and implementation.
- White-box testing is time-consuming, bigger programming applications take the time to test fully.

White box testing involves the testing of the software code for the following:

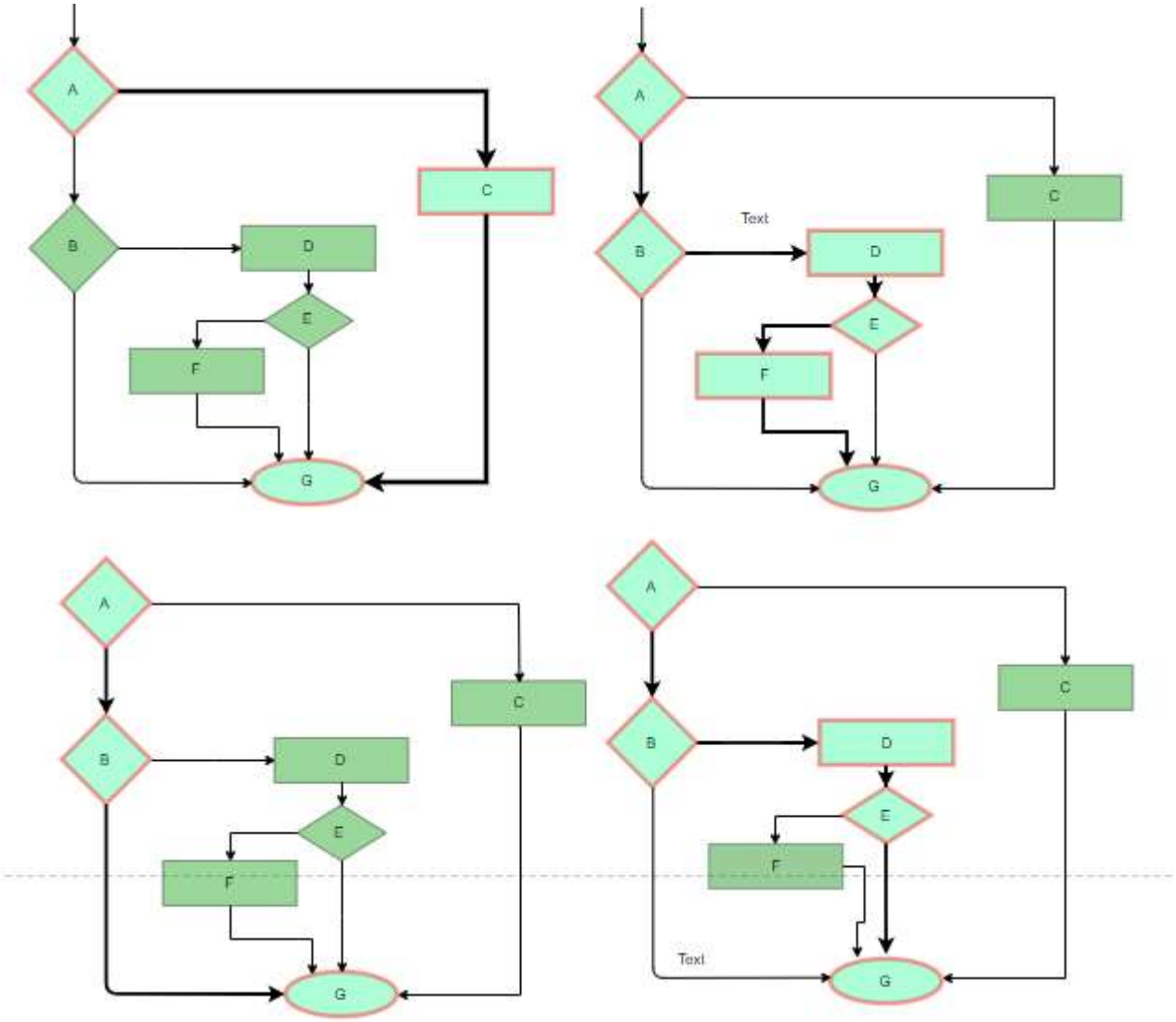
- Internal security holes
- Broken or poorly structured paths in the coding processes
- The flow of specific inputs through the code
- Expected output
- The functionality of conditional loops
- Testing of each statement, object, and function on an individual basis

Testing techniques:

- **Statement coverage:** In this technique, the aim is to traverse all statement at least once. Hence, each line of code is tested. In case of a flowchart, every node must be traversed at least once. Since all lines of code are covered, helps in pointing out faulty code.



Branch Coverge: In this technique, test cases are designed so that each branch from all decision points are traversed at least once. In a flowchart, all edges must be traversed at least once.



- **condition Coverage:** In this technique, all individual conditions must be covered as shown in the following example:
  1. READ X, Y
  2. IF(X == 0 || Y == 0)
  3. PRINT '0'

  In this example, there are 2 conditions: X == 0 and Y == 0. Now, test these conditions get TRUE and FALSE as their values. One possible example would be:

  - #TC1 – X = 0, Y = 55
  - #TC2 – X = 5, Y = 0
- **Multiple Condition Coverage:** In this technique, all the possible combinations of the possible outcomes of conditions are tested at least once. Let's consider the following example:
  0. READ X, Y
  1. IF(X == 0 || Y == 0)
  2. PRINT '0'

  - #TC1: X = 0, Y = 0
  - #TC2: X = 0, Y = 5

- #TC3: X = 55, Y = 0
- #TC4: X = 55, Y = 5

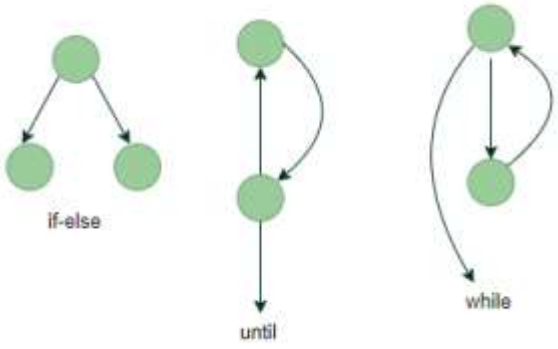
Hence, four test cases required for two individual conditions. Similarly, if there are n conditions then 2^n test cases would be required.

Basis Path Testing: In this technique, control flow graphs are made from code or flowchart and then Cyclomatic complexity is calculated which defines the number of independent paths so that the minimal number of test cases can be designed for each independent path.

Steps:

1. Make the corresponding control flow graph
2. Calculate the cyclomatic complexity
3. Find the independent paths
4. Design test cases corresponding to each independent path

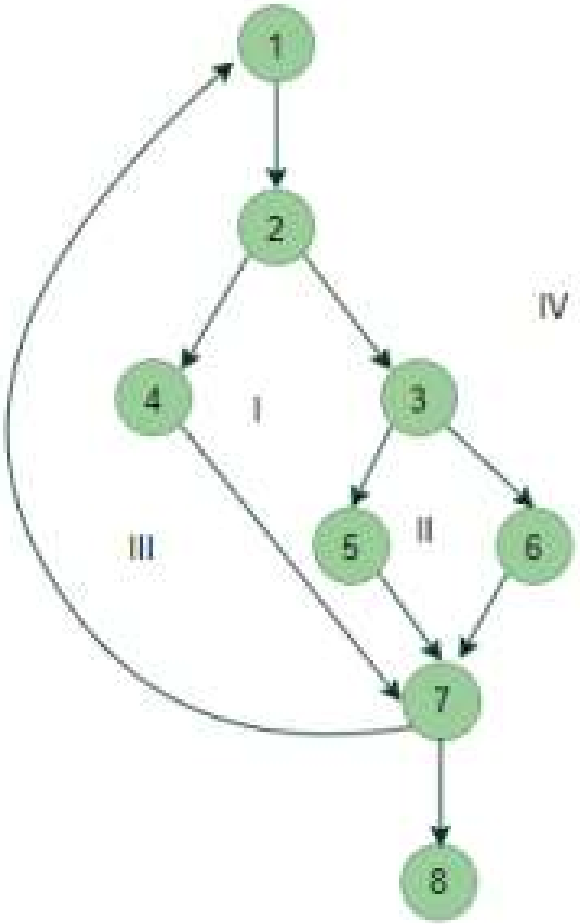
Flow graph notation: It is a directed graph consisting of nodes and edges. Each node represents a sequence of statements, or a decision point. A predicate node is the one that represents a decision point that contains a condition after which the graph splits. Regions are bounded by nodes and edges.



Cyclomatic Complexity: It is a measure of the logical complexity of the software and is used to define the number of independent paths. For a graph G, V(G) is its cyclomatic complexity. Calculating V(G):

1. V(G) = P + 1, where P is the number of predicate nodes in the flow graph
2. V(G) = E – N + 2, where E is the number of edges and N is the total number of nodes
3. V(G) = Number of non-overlapping regions in the graph

Example:

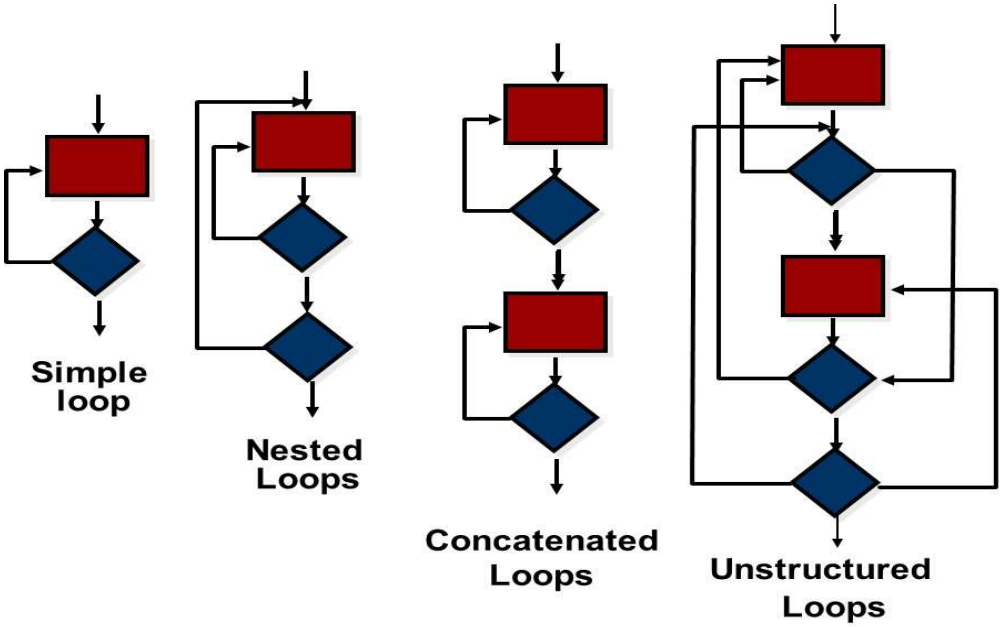


- V(G) = 4 (Using any of the above formulae)

  No of independent paths = 4

  - #P1: 1 – 2 – 4 – 7 – 8
  - #P2: 1 – 2 – 3 – 5 – 7 – 8
  - #P3: 1 – 2 – 3 – 6 – 7 – 8
  - #P4: 1 – 2 – 4 – 7 – 1 – . . . – 7 – 8

- **Loop Testing:** Loops are widely used and these are fundamental to many algorithms hence, their testing is very important. Errors often occur at the beginnings and ends of loops.



1. **Simple loops:** For simple loops of size n, test cases are designed that:
   - Skip the loop entirely
   - Only one pass through the loop
   - 2 passes
   - m passes, where m < n
   - n-1 ans n+1 passes
2. **Nested loops:** For nested loops, all the loops are set to their minimum count and we start from the innermost loop. Simple loop tests are conducted for the innermost loop and this is worked outwards till all the loops have been tested.
3. **Concatenated loops:** Independent loops, one after another. Simple loop tests are applied for each. If they're not independent, treat them like nesting.