A major White box testing technique is **Code Coverage analysis.** Code Coverage analysis eliminates gaps in a [Test Case](https://www.guru99.com/test-case.html) suite.

Following are major code coverage methods

* Statement Coverage
* Decision Coverage
* Branch Coverage
* Toggle Coverage
* FSM Coverage

**Statement Coverage** is a white box testing technique in which all the executable statements in the source code are executed at least once. It is used for calculation of the number of statements in source code which have been executed. The main purpose of Statement Coverage is to cover all the possible paths, lines and statements in source code.

Statement coverage is used to derive scenario based upon the structure of the code under test.

A red line between green text

Description automatically generated

**What is covered by Statement Coverage?**

* Unused Statements
* Dead Code
* Unused Branches
* Missing Statements

**Decision Coverage** is a white box testing technique which reports the true or false outcomes of each boolean expression of the source code. The goal of decision coverage testing is to cover and validate all the accessible source code by checking and ensuring that each branch of every possible decision point is executed at least once.In this coverage, expressions can sometimes get complicated. Therefore, it is very hard to achieve 100% coverage.



**Branch Coverage** is a white box testing method in which every outcome from a code module(statement or loop) is tested. The purpose of branch coverage is to ensure that each decision condition from every branch is executed at least once. It helps to measure fractions of independent code segments and to find out sections having no branches.

For example, if the outcomes are binary, you need to test both True and False outcomes.

A black text on a white background

Description automatically generated

**Condition Coverage** or expression coverage is a testing method used to test and evaluate the variables or sub-expressions in the conditional statement. The goal of condition coverage is to check individual outcomes for each logical condition. Condition coverage offers better sensitivity to the control flow than decision coverage. In this coverage, expressions with logical operands are only considered.

For example, if an expression has Boolean operations like AND, OR, XOR, which indicates total possibilities. Condition coverage does not give a guarantee about full decision coverage.

A black text on a white background

Description automatically generated

Code coverage tools:

* Cobertura
* Clover
* DevPartner
* Emma
* KaliStick
* Coview and Coant
* Bullseye for c++
* Sonar

**Path testing** is a structural testing method that involves using the source code of a program in order to find every possible executable path. It helps to determine all faults lying within a piece of code. This method is designed to execute all or selected path through a computer program.

Any software program includes, multiple entry and exit points. Testing each of these points is a challenging as well as time-consuming. In order to reduce the redundant tests and to achieve maximum test coverage, basis path testing is used.

**Basis Path Testing** in software engineering is a [White Box Testing](https://www.guru99.com/white-box-testing.html) method in which test cases are defined based on flows or logical paths that can be taken through the program. The objective of basis path testing is to define the number of independent paths, so the number of test cases needed can be defined explicitly to maximize test coverage.

In [software engineering](https://www.guru99.com/what-is-software-engineering.html), Basis path testing involves execution of all possible blocks in a program and achieves maximum path coverage with the least number of test cases. It is a hybrid method of branch testing and path testing methods.

The basic steps involved in basis path testing include

* Draw a control graph (to determine different program paths)
* Calculate [Cyclomatic complexity](https://www.guru99.com/cyclomatic-complexity.html) (metrics to determine the number of independent paths)
* Find a basis set of paths
* Generate test cases to exercise each path

**How To Calculate Statement Coverage, Branch Coverage, Path Coverage**

First let’s learn what are nodes and edges

**Nodes:** Nodes represent entries, each statement of the code, decisions, and exits

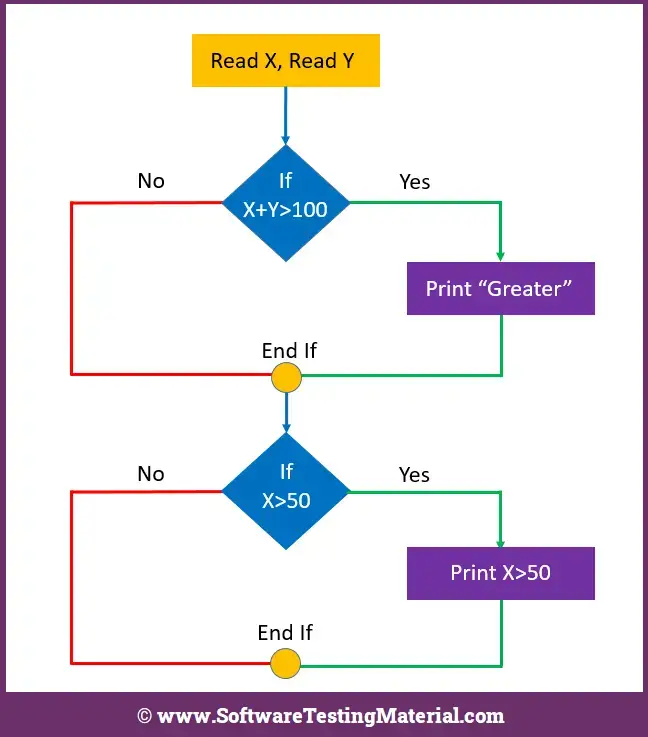
**Edges:** Edges represent branching, and non-branching links between nodes.

Let’s take a sample program and calculate the statement coverage, branch coverage, and path coverage.

***Read X***  
***Read Y***  
***IF X+Y > 100 THEN***  
***Print “Greater”***  
***ENDIF***  
***If P > 50 THEN***  
***Print “X>50”***  
***ENDIF***

Solution:

Here is the flow chart for the above program



Let’s specify nodes and edges of above flow chart

A diagram of a software

Description automatically generated

**Statement Coverage**

It is to find the shortest number of paths by covering all the nodes.

With this one path 1A-2C-3D-E-4G-5H, we have covered all the nodes such as 1,2,3,4,5

Total number of statements covered are 1.

**Branch Coverage**

It is to find the minimum number of paths by covering all the edges. Here our focus is to cover all the possible true and false decisions.

With this path 1A-2C-3D-E-4G-5H, we have covered edges A,C,D,E,G, and H but we missed edges B and F.  
With this path 1A-2B-E-4F, we have covered edges A,B,E, and F.

By combining the above two paths we have covered all the edges A,B,C,D,E,F,G, and H.

Total number of branches covered are 2.

**Path Coverage**

It ensures covering all the possible paths right from start to end

1A-2B-E-4G-5H  
1A-2B-E-4F  
1A-2C-3D-E-4G-5H  
1A-2C-3D-E-4F

Total number of paths covered are 4.

**Output from the above example is**  
Statement Coverage = 1  
Branch Coverage = 2  
Path Coverage = 4

Also remember the following

* 100% LCSAJ coverage will imply 100% Branch/Decision coverage
* 100% Path coverage will imply 100% Statement coverage
* 100% Branch/Decision coverage will imply
* 100% Statement coverage
* 100% Path coverage will imply 100% Branch/Decision coverage

**Note:**

* Both decision coverage and branch coverage are same
* Decision coverage gives more coverage compared to Statement Coverage.
* 100% Decision coverage guarantees 100% Statement coverage but not vice versa.
* LCSAJ means Linear Code Sequence and Jump