White-Box Testing

White-Box Testing

White-box testing is an important type of unit testing. A large number of white-box testing strategies exist. Each testing strategy essentially designs test cases based on analysis of some aspect of source code and is based on some heuristic.

A white-box testing strategy can either be coverage-based or fault-based.

Fault-based testing: A fault-based testing strategy targets to detect certain types of faults. An example of a fault-based strategy is mutation testing.

Coverage-based testing: A coverage-based testing strategy attempts to execute (or cover) certain elements of a program. Popular examples of coverage-based testing strategies are statement coverage, branch coverage, multiple condition coverage, and path coverage-based testing.

Coverage-based Testing

Testing criterion for coverage-based testing:

A coverage-based testing strategy typically targets to execute (i.e., cover) certain program elements for discovering failures.

For example, if a testing strategy requires all the statements of a program to be executed at least once, then we say that the testing criterion of the strategy is statement coverage.

A test suite is adequate with respect to a criterion, if it covers all program elements of the domain defined by that criterion.

Stronger versus weaker testing

When none of two testing strategies fully covers the program elements exercised by the other, then the two are called complementary testing strategies.

The concepts of stronger, weaker, and complementary testing are schematically illustrated in the given figure.

In the Figure(a), testing strategy A is stronger than B since B covers only a proper subset of elements covered by A. On the other hand, Figure(b) shows A and B are complementary testing strategies since some elements of A are not covered by B and vice versa.

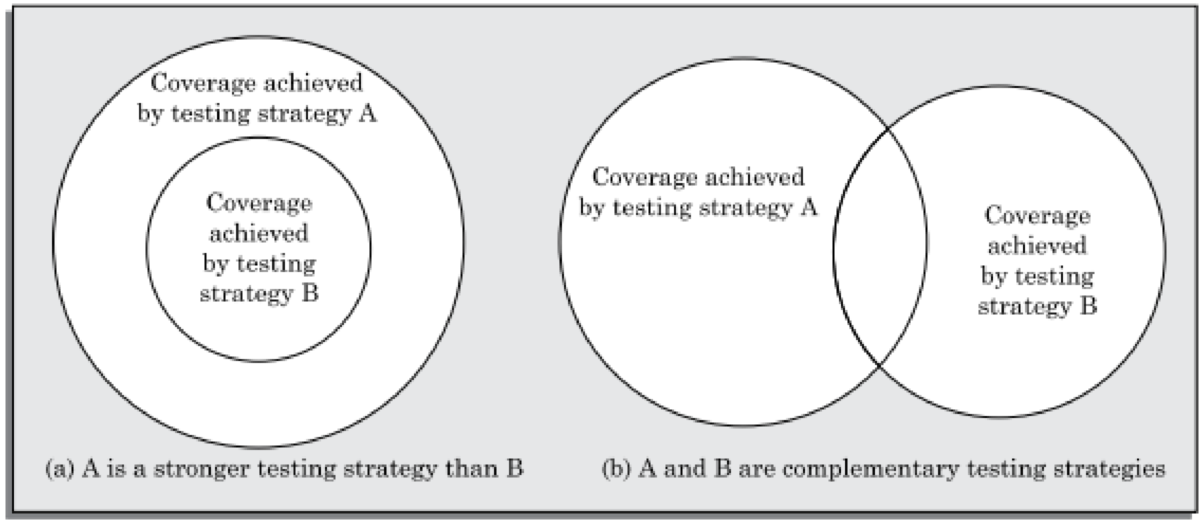

Coverage achieved by testing strategy A
Coverage achieved by testing strategy B
(a) A is a stronger testing strategy than B
Coverage achieved by testing strategy A
Coverage achieved by testing strategy B
(b) A and B are complementary testing strategies

Figure: Illustration of stronger, weaker, and complementary testing strategies.

Coverage-based Testing

Testing criterion for coverage-based testing:

A coverage-based testing strategy typically targets to execute (i.e., cover) certain program elements for discovering failures.

For example, if a testing strategy requires all the statements of a program to be executed at least once, then we say that the testing criterion of the strategy is statement coverage.

A test suite is adequate with respect to a criterion, if it covers all program elements of the domain defined by that criterion.

Control Flow Testing

- Control flow testing is a testing technique to determine the execution order of statements or instructions of the program through a control structure.
- The control structure of a program is used to develop a test case for the program.
- In this technique, a particular part of a large program is selected by the tester to set the testing path.
- It is mostly used in unit testing.
- Test cases represented by the control graph of the program.

Control Flow Graph

Control Flow Graph is formed from the node, edge, decision node, junction node to specify all possible execution path.

Notations used for Control Flow Graph

- ➢Node
- ➢Edge
- ➢Decision Node
- ➢Junction node

Node

- Nodes in the control flow graph are used to create a path of procedures. Basically, it represents the sequence of procedures which procedure is next to come.

Control Flow Graph

Example:

```
public class VoteEligiblityAge{
public static void main(String []args){
int n=45;
if(n>=18)
{
System.out.println("You are eligible for voting");
}  else
{
System.out.println("You are not eligible for voting");
}
}
}
```

Control Flow Graph

Example:

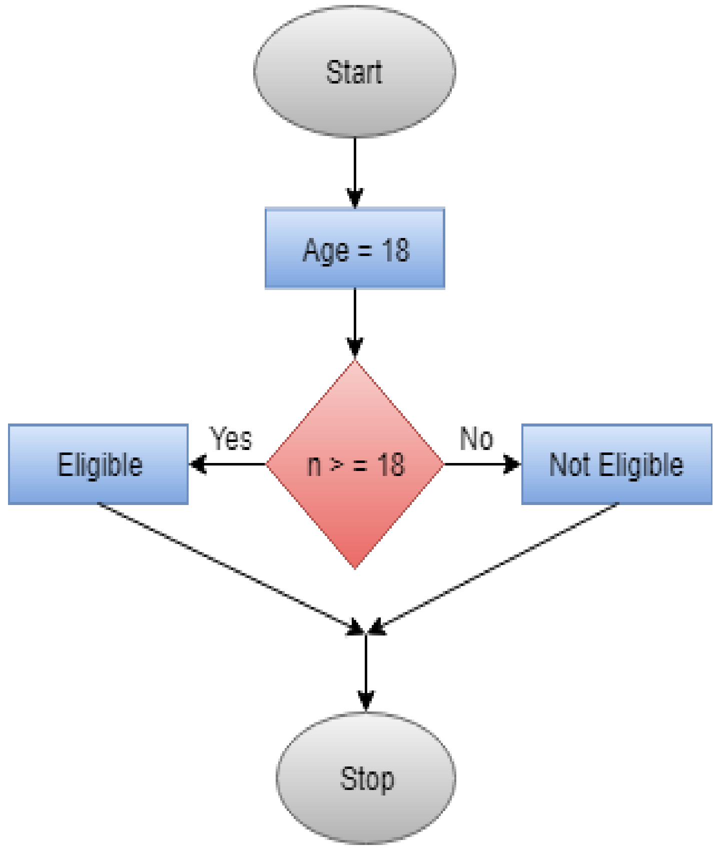
- In the example, the first node represent the start procedure and the next procedure is to assign the value of n.
- After assigning the value there is decision node to decide next node of procedure as per the value of n if it is 18 or more than 18 so Eligible procedure will execute otherwise if it is less than 18 Not Eligible procedure executes.
- The next node is the junction node, and the last node is stop node to stop the procedure.

Control Flow Graph

The example shows eligibility criteria of age for voting where if age is 18 or more than 18 so print message "You are eligible for voting" if it is less than 18 then print "You are not eligible for voting."

In the control flow graph, start, age, eligible, not eligible and stop are the nodes, n>=18 is a decision node to decide which part (if or else) will execute as per the given value. Connectivity of the eligible node and not eligible node is there on the stop node.

Test cases are designed through the flow graph of the programs to determine the execution path is correct or not. All nodes, junction, edges, and decision are the essential parts to design test cases.



Statement Coverage Testing

- ➢This technique involves execution of all statements of the source code at least once.
- ➢It is used to calculate the total number of executed statements in the source code out of total statements present in the source code.
- ➢Statement coverage derives scenario of test cases under the white box testing process which is based upon the structure of the code.

Statement Coverage= (Number of executed statements/Total number of statements)*100

- ➢Generally, in the internal source code, there is a wide variety of elements like operators, methods, arrays, looping, control statements, exception handlers, etc. Based on the input given to the program, some code statements are executed and some may not be executed. The goal of statement coverage technique is to cover all the possible executing statements and path lines in the code.

Statement Coverage Testing

Example:

```
print (int a, int b) {
int sum = a+b;
if (sum>0)
print ("This is a positive result")
else
print ("This is negative result")
}
```

Statement Coverage Testing

Example:

Test suite-1:

If a = 5, b = 4

- In this scenario, the value of sum will be 9 that is greater than 0 and as per the condition result will be "**This is a positive result".**
- To calculate statement coverage of this scenario, take the total number of statements that is 7 and the number of executed statements is 5.
- Total number of statements = 7
- Number of executed statements = 5
- So, statement coverage=(5/7)*100, that is, 71%

Statement Coverage Testing

Example:

Test suite-1:

If a = -2, b = -7

- In scenario 2, we can see the value of sum will be -9 that is less than 0 and as per the condition, result will be "**This is a negative result.**"
- To calculate statement coverage of this scenario, take the total number of statements that is 7 and the number of executed statements is 6.
- Total number of statements = 7
- Number of executed statements = 6
- So, statement coverage=(6/7)*100, that is, 85%

Branch Coverage Testing

- Branch coverage technique is used to cover all branches of the control flow graph.
- It covers all the possible outcomes (true and false) of each condition of decision point at least once.
- Branch coverage technique is a whitebox testing technique that ensures that every branch of each decision point must be executed.
- However, branch coverage technique and decision coverage technique are very similar, but there is a key difference between the two. Decision coverage technique covers all branches of each decision point whereas branch testing covers all branches of every decision point of the code.
- The most basic metrics for finding the percentage of program and paths of execution during the execution of the program.
- It uses a control flow graph to calculate the number of branches.

How to calculate Branch coverage?

- There are several methods to calculate Branch coverage, but pathfinding is the most common method.
- In path finding method, the number of paths of executed branches is used to calculate Branch coverage. Branch coverage technique can be used as the alternative of decision coverage.

Example:

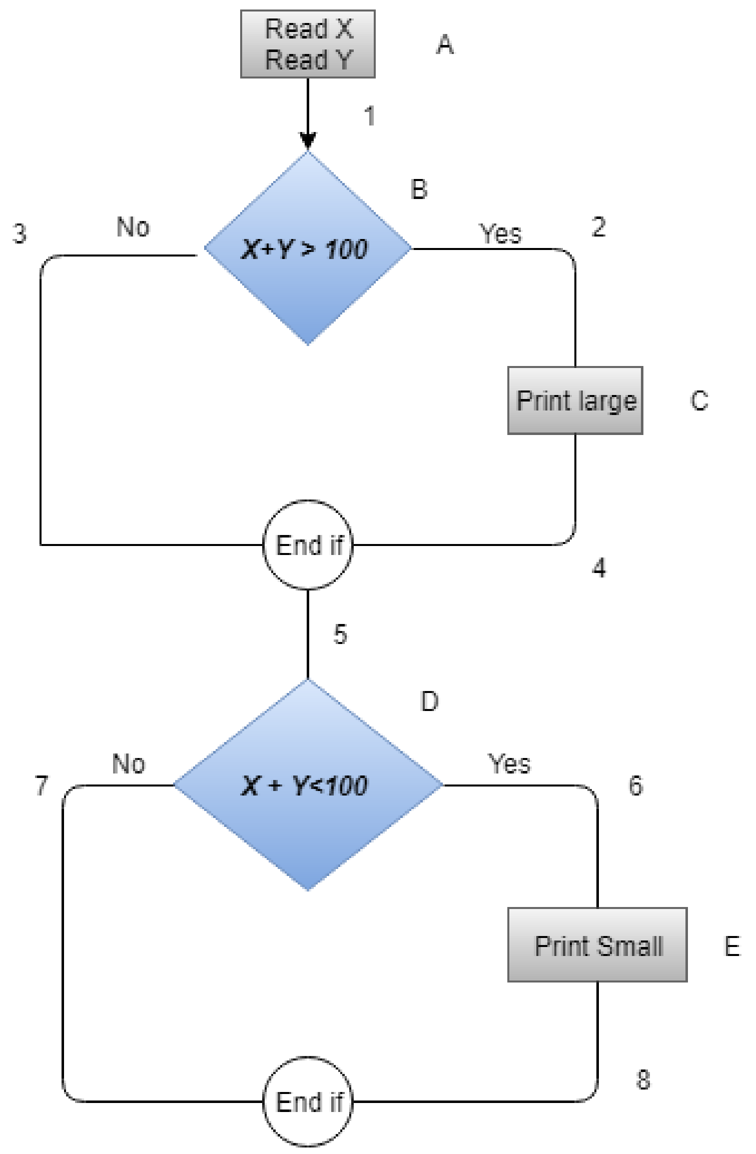
```
Read X
Read Y
IF X+Y > 100 THEN
Print "Large"
ENDIF
If X + Y<100 THEN
Print "Small"
ENDIF
```

This is the basic code structure where we took two variables X and Y and two conditions. If the first condition is true, then print "Large" and if it is false, then go to the next condition. If the second condition is true, then print "Small."

How to calculate Branch coverage?

Control flow graph of code structure

In the adjacent diagram, control flow graph of the given code is depicted. In the first case traversing through "Yes "decision, the path is **A1-B2-C4-D6-E8**, and the number of covered edges is 1, 2, 4, 5, 6 and 8 but edges 3 and 7 are not covered in this path. To cover these edges, we have to traverse through "No" decision. In the case of "No" decision the path is A1-B3-5-D7, and the number of covered edges is 3 and 7. So by traveling through these two paths, all branches have covered.



How to calculate Branch coverage?

Path 1 - A1-B2-C4-D6-E8

Path 2 - A1-B3-5-D7

Branch Coverage (BC) = Number of paths =2

Case	Covered Branches	Path	Branch coverage
Yes	1, 2, 4, 5, 6, 8	A1-B2-C4-D6-E8	2
No	3,7	A1-B3-5-D7	