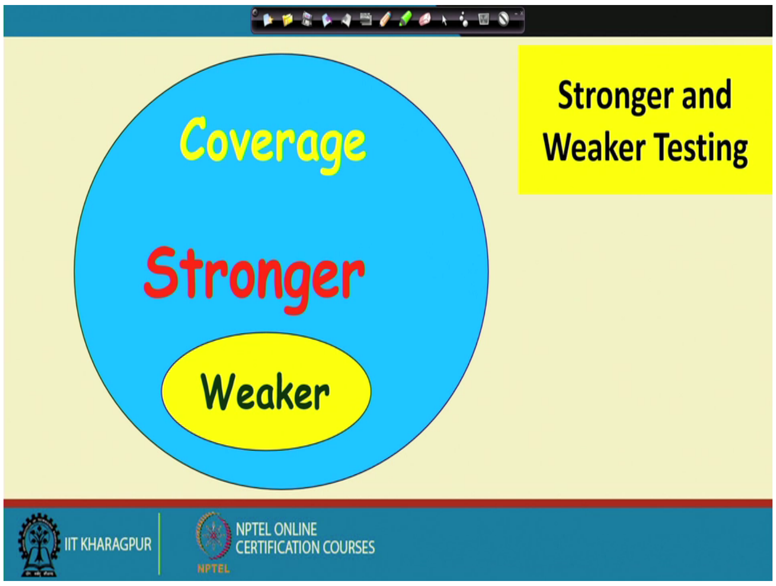
Software Engineering

Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 55
White box Testing

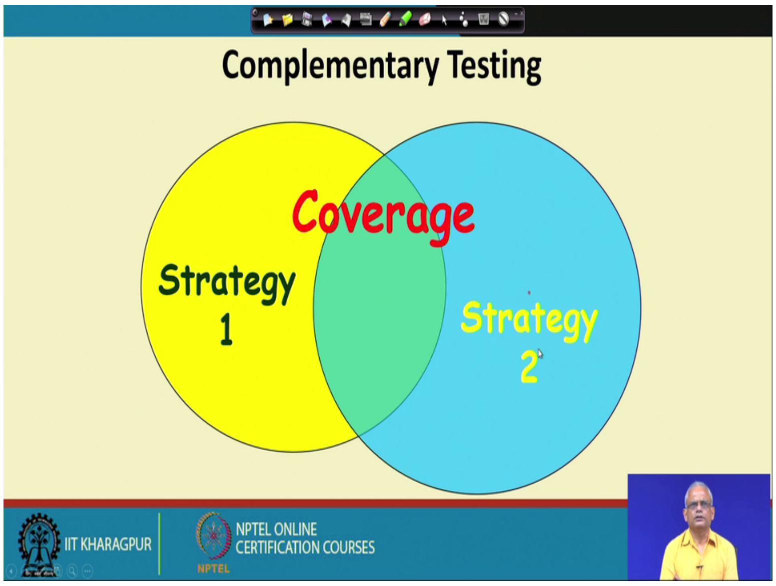
We had discussed that there are many white box testing techniques, but we need to discuss one important concept, that whether we need to test all of these different testing techniques. Whether, we need to do testing using all this half a dozen or a dozen testing techniques. For that we must understand the concept of a stronger testing and a weaker testing.

(Refer Slide Time: 00:46)


Stronger and Weaker Testing
Coverage
Stronger
Weaker
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

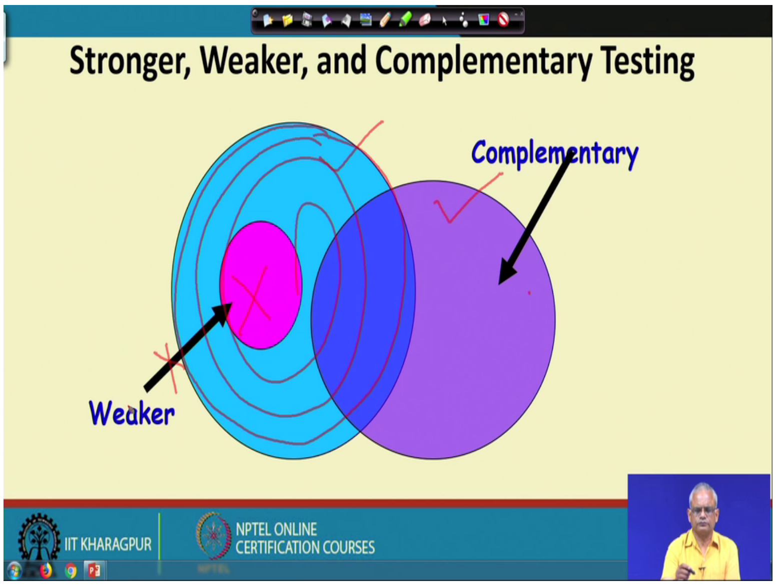
The idea here is that different testing techniques they cover certain program elements. A stronger testing covers all the elements that have been covered by a weaker testing. That means, if you are doing a stronger testing all the program elements that have been executed by a stronger testing includes the weaker testing or in other words if, you are doing a stronger testing, weaker testing is not necessary. We need not again design test cases for weaker testing. As, long as we do a stronger testing weaker testing need not be done it is ensured automatically by a stronger testing.

(Refer Slide Time: 01:52)


Complementary Testing

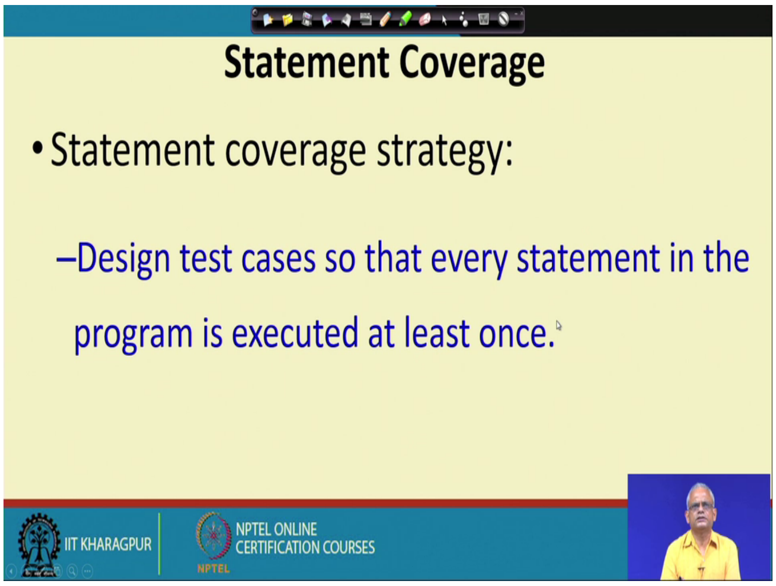
It may be possible that two white box test strategies are complementary, that is they execute some program elements, which are common strategy 1, test strategy 1 it executes elements some of which are overlapping with strategy 2. If 2 test strategies are complementary, then we need to conduct testing using both strategies.

(Refer Slide Time: 02:30)


Stronger, Weaker, and Complementary Testing

If, we represent the stronger, weaker, and complementary, we can see that the test strategy, which is covered by the green all the elements here by the green strategy. They cover the weaker strategy which is given in pink. Therefore, as long as we are doing the testing for the blue coverage we need not do the weaker testing, but see here this is a complementary test strategy, which execute some elements here for the other strategy, but it also executes additional elements. And therefore, we need to do testing with both this strategies; we only can eliminate the weaker strategy.

(Refer Slide Time: 03:34)

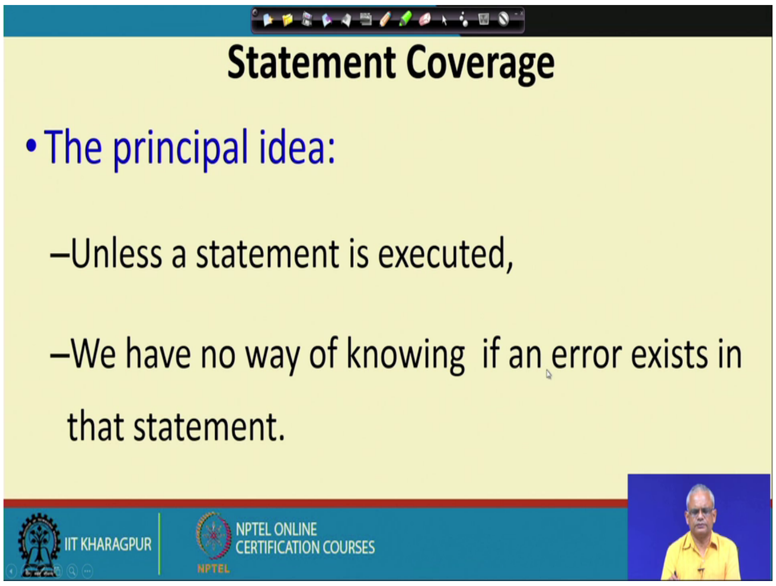


Statement Coverage

- Statement coverage strategy:
  - Design test cases so that every statement in the program is executed at least once.

Now, based on the basic concepts that we discussed, now let us look at the different white box testing strategies. The simplest strategy is statement coverage. The idea here is that we need to execute every statement in the program at least once. We just look at the source code and design test cases such that every statement is executed at least once.

(Refer Slide Time: 04:12)



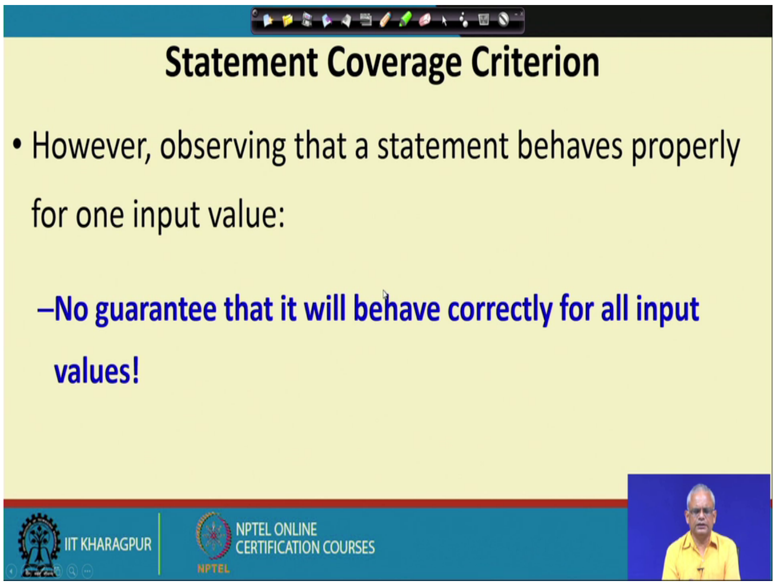
Statement Coverage

- The principal idea:
  - Unless a statement is executed,
  - We have no way of knowing if an error exists in that statement.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

The principal idea behind this technique is that, if some statements are not executed we never know if there is a bug in that. And, that is a reason why we want every statement to be executed by the test cases. Such that, if there is a bug in the test case it will probably get expressed when we execute that test cases.

(Refer Slide Time: 04:43)



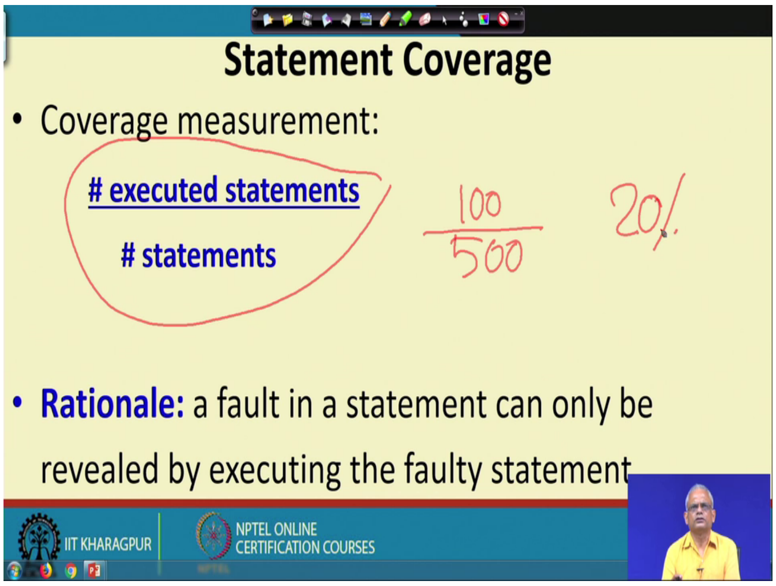
Statement Coverage Criterion

- However, observing that a statement behaves properly for one input value:
  - **No guarantee that it will behave correctly for all input values!**

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

But, we must also understand that the statement coverage criterion has deficiency, because just by executing a test case once is no guarantee that all the bugs have been expressed, but then it is better than not executing all statements. So, we must ensure that all statements are at least executed once, but what we are trying to say here that just statement coverage may not be enough, we met achieve statement coverage, but then it may not guarantee that good number of bugs majority of the bugs have been exposed. Because just by executing it is statement once may not be exposing the bugs present in that statement.

(Refer Slide Time: 05:43)



Statement Coverage

- Coverage measurement:

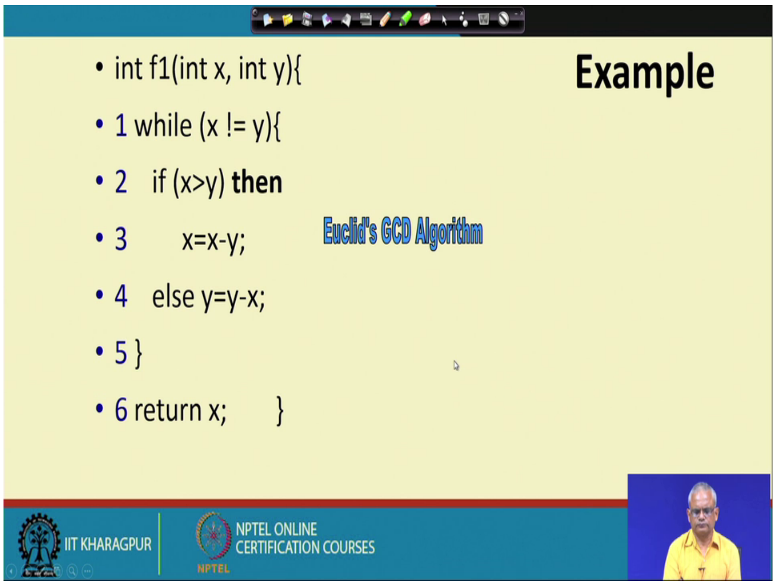
$$\frac{\text{\# executed statements}}{\text{\# statements}} \qquad \frac{100}{500} \qquad 20\%$$

- **Rationale:** a fault in a statement can only be revealed by executing the faulty statement

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

Now, let us see how the statement coverage is computed? Here, we count the total number of statements in the program and we track how many statements are executed by the test cases. So, the percentage of executed statements by the total number of statements, that gives us what is the coverage achieved. If we have 500 is the total number of statements. And, we have executed let us say 100 statements by the test cases we have achieved only 20 percent statement coverage. It is a very simple strategy, but then it is one of the basic strategies and other strategies that will see, they will achieve a more stronger testing than statement coverage let us discuss those techniques.

(Refer Slide Time: 06:55)


Example

- int f1(int x, int y){
- 1 while (x != y){
- 2 if (x>y) then
- 3 x=x-y;
- 4 else y=y-x;
- 5 }
- 6 return x; }

Euclid's GCD Algorithm

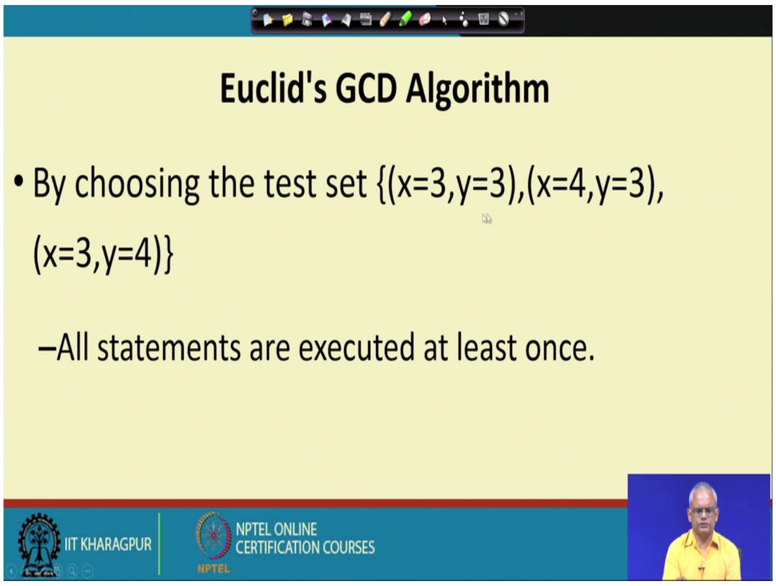
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

Before, that let us take an example we have this code here simple code, while x is not equal to y, if x greater than y then x equal to x minus y else y equal to y minus x. If you remember this is the GCD computation Euler's GCD computation code. And, we want to see, what are the input values? For which statement coverage will be achieved. There is a decision here only if x is greater than y, then x this statement will get executed.

And, if x is not equal to y this statement will get executed. And, this will be executed only if x is not equal to y, if we test it with x equal to y, then all this will not be tested. So, first need to check whether x is not equal to y and then we must have x greater than y and x less than y.

So, this is the Euclid's GCD Algorithm and we know the specific values for which statement coverage will be achieved, but then given a large enough program, it is very hard to identify what are the specific values for which the statement coverage is achieved. And fortunately for white box testing, we do not really design test cases by looking at the specific values and see what will execute which one and so on. We have software to measure coverage statement coverage let us say and then we just keep on giving values to this test cases to the software. And, then the coverage tool will tell how much cover is achieved? We keep on testing until we achieve sufficient coverage.

(Refer Slide Time: 09:41)



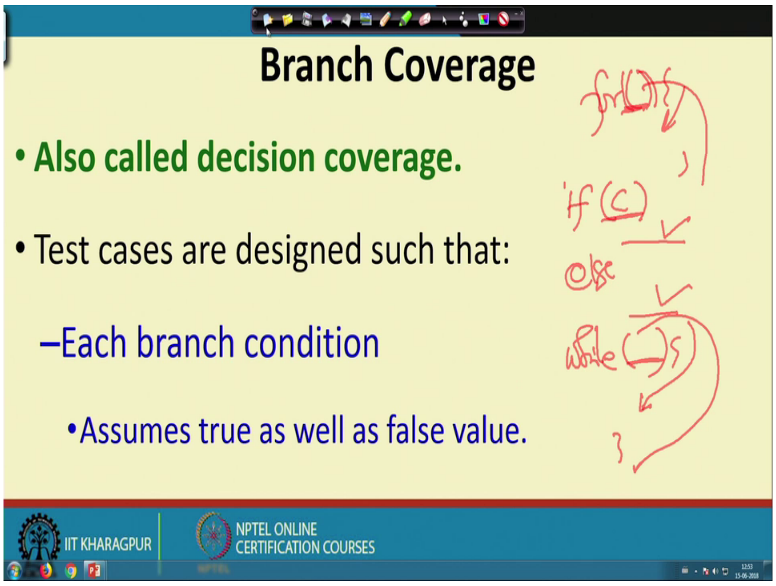
Euclid's GCD Algorithm

- By choosing the test set {(x=3,y=3),(x=4,y=3),(x=3,y=4)}
  - All statements are executed at least once.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

Now, for this specific example the Euclid's GCD program, if we choose these are the different values then statement coverage achieved and as I saying that in reality for large programs, we do not have to identify the values for which statement coverage is achieved, but to understand the concept. What is statement coverage maybe for a small program? We might have to identify value such that statement coverage achieved or given a some values or test cases we should be able to tell that, whether statement coverage will be achieved.

(Refer Slide Time: 10:22)



Branch Coverage

- Also called decision coverage.
- Test cases are designed such that:
  - Each branch condition
    - Assumes true as well as false value.

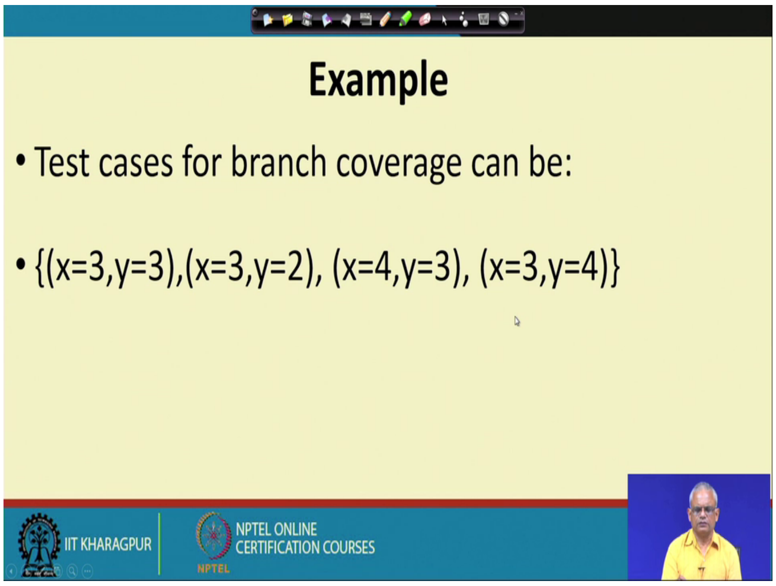
for(){
}
if (C)
else
while ( ) S
}
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

The next white box strategy that we will discuss is the branch coverage. Another, name for branch coverage is decision coverage. The main idea here is that in very program there are many decision statements, the decision statements can be of the form if some condition execute some statement else, execute some statement or it may be while something keep doing something or it may be for and so on. These are examples of decision statements. And here the idea in branch or decision coverage is that every decision statement here. They must be taking true and false values. So, it should at least the test cases ensure that it goes inside the loop and also it exits the loop.

Similarly, for if both this take place that is the condition is true and false. And, similarly for the for it should enter into the loop and also for some test input it should not enter into the loop. So, every condition in a conditional statement every branch condition must take true and false values that is our test cases should ensure that. Now, if we look at the same function here the Euclid’s GCD algorithm, we find that the decision statements are here and here. There are 2 decision statements and our test cases must make this once true make this once false. And, similarly this should be true and false and for very small programs we can design the branch coverage test cases.

But, in normal practical situation we do not have to design the test cases branch coverage test cases, we just keep on giving input and coverage tool will tell us, what is the extent of branch coverage that is achieved. And, we keep on giving data until you get sufficient branch coverage.

(Refer Slide Time: 13:18)

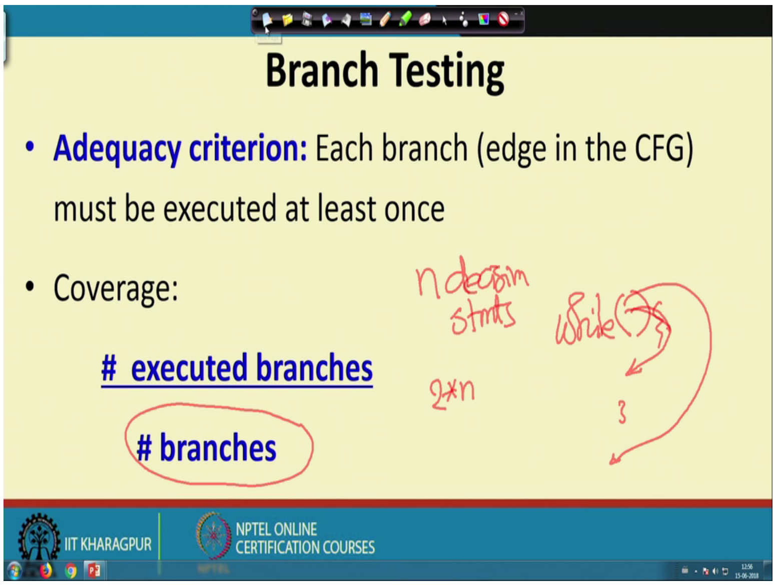


Example

- Test cases for branch coverage can be:
- {(x=3,y=3),(x=3,y=2), (x=4,y=3), (x=3,y=4)}

So, for that simple program Euclid's GCD we can design some test cases which will achieve the branch coverage.

(Refer Slide Time: 13:33)



Branch Testing

- **Adequacy criterion:** Each branch (edge in the CFG) must be executed at least once
- Coverage:

$$\frac{\#\ \text{executed branches}}{\#\ \text{branches}}$$

n decision stmts

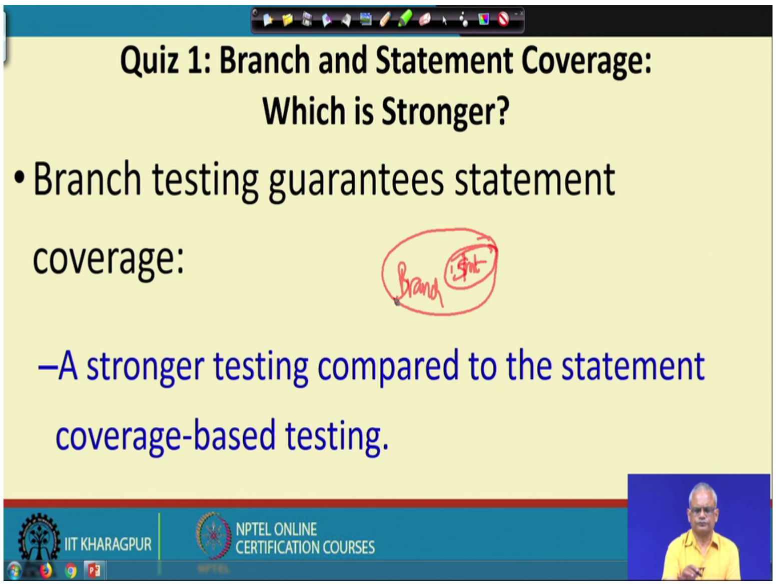
2*n

While(x)

3

But, how do we measure coverage? If, we are giving executing the program using test cases how does the coverage tool that will report the extent of coverage achieved.

(Refer Slide Time: 13:52)



Quiz 1: Branch and Statement Coverage: Which is Stronger?

- Branch testing guarantees statement coverage:
  - A stronger testing compared to the statement coverage-based testing.

It will find out all the possible branches that are there on the code number of branches and how many branches have been executed? If, it is a while condition then there are 2 branches; one branch is that it is true, it enters into the loop and false exits. Similarly every condition conditional statement must achieve true and false values, during the execution of the test cases.

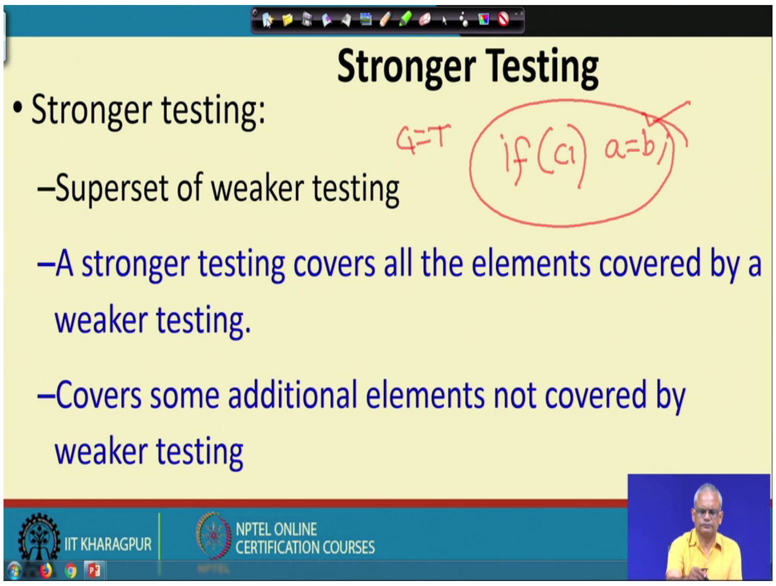
So, each of this will be computed 2 here, if we have n decision statements, then the number of branches will be 2 into n. And, then the coverage tool can find out how many of these branches true and false are taken and, then it will report the extent of branch coverage achieved.

But, then one thing we must be clear is that, which is a stronger test is branch coverage or statement coverage, because if we can say that which is stronger testing we need not do the other testing, but then if we say that something is stronger, we must be able to show that it is stronger testing. Now, here between branch coverage and statement coverage, we can say that branch coverage is stronger than statement coverage, because every statement must be there on some branch. So, that is our argument here that if there is a statement in a program it must be there on some branch. And therefore, if we are covering the branches, then all statements must have been covered. So, branch coverage ensures statement coverage.

But, then the question comes that is it possible that branch coverage ensures statement coverage, but is it possible that branch coverage guarantees statement coverage that we could show now, but we have to also show that there are some branches, which are not ensured by statement coverage. Otherwise they will become identical to show that it is stronger we have to show not only that branch coverage ensures statement coverage, but we have to also show that statement coverage does not ensure branch coverage. Or in other words, there are some branches, which may not get executed even though we achieve statement coverage, how do we show that?

One way we could show that, branch coverage guarantees statement coverage because every statement must lie on some branch.

(Refer Slide Time: 17:47)

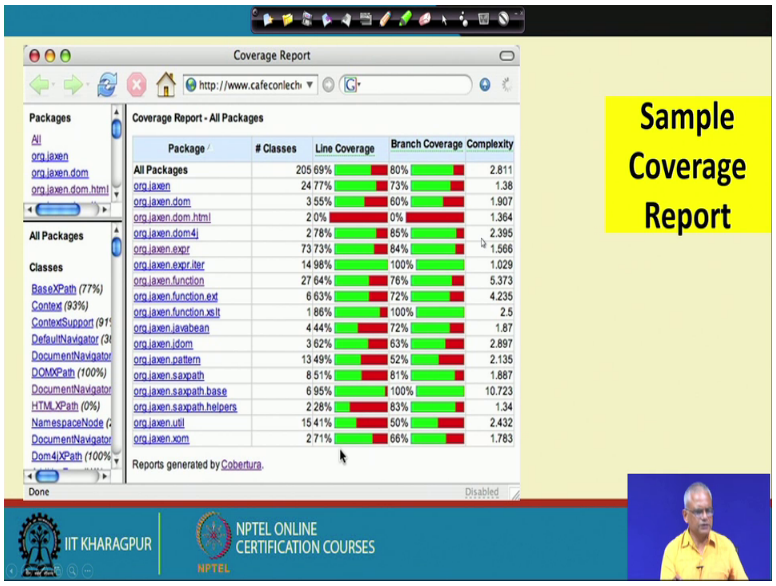


Stronger Testing

- Stronger testing:
  - Superset of weaker testing
  - A stronger testing covers all the elements covered by a weaker testing.
  - Covers some additional elements not covered by weaker testing

So, a stronger testing as we said is a superset of weaker testing, if we are doing stronger testing, we need not do the weaker testing if we are doing the branch testing, we need not do the statement coverage testing, but then so far we have showed that state branch coverage ensure statement coverage, but we have to also show that statement coverage does not achieve branch coverage.

(Refer Slide Time: 18:21)



Sample Coverage Report

Coverage Report

Coverage Report - All Packages

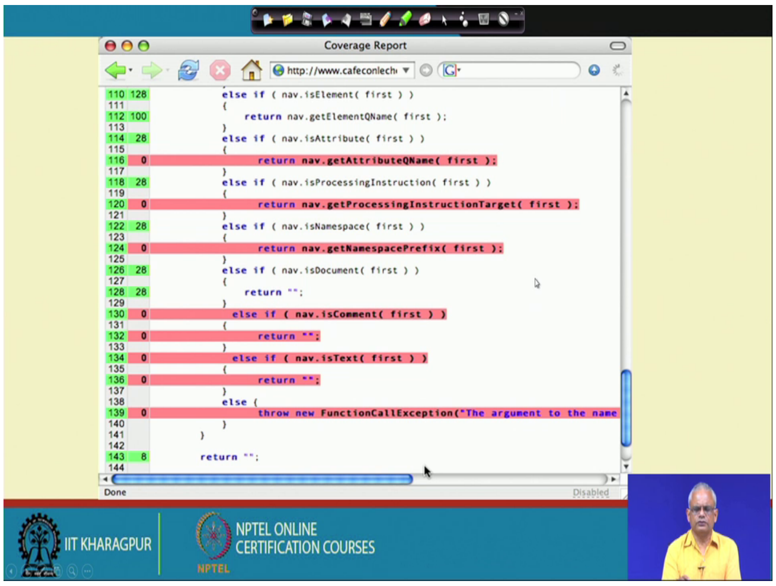
Package	# Classes	Line Coverage	Branch Coverage	Complexity
All Packages	205	69%	80%	2.811
org.jaxen	24	77%	73%	1.38
org.jaxen.dom	3	55%	60%	1.907
org.jaxen.dom.html	2	0%	0%	1.364
org.jaxen.dom4j	2	78%	85%	2.395
org.jaxen.expr	73	73%	84%	1.566
org.jaxen.expr.iter	14	98%	100%	1.029
org.jaxen.function	27	64%	76%	5.373
org.jaxen.function.ext	6	63%	72%	4.235
org.jaxen.function.xslt	1	86%	100%	2.5
org.jaxen.javabean	4	44%	72%	1.87
org.jaxen.jdom	3	62%	63%	2.897
org.jaxen.pattern	13	49%	52%	2.135
org.jaxen.saxpath	8	51%	81%	1.887
org.jaxen.saxpath.base	6	95%	100%	10.723
org.jaxen.saxpath.helpers	2	28%	83%	1.34
org.jaxen.util	15	41%	50%	2.432
org.jaxen.xom	2	71%	66%	1.783

Reports generated by Cobertura.

So let me just give one example here that will ensure that statement coverage does not achieve branch coverage. Let say we have a statement if c 1 then a is equal to b. Now, let us say that c 1 is true, then we achieve the branch coverage here sorry the statement coverage c 1 is equal to true achieves statement coverage, but it does not achieve branch coverage because c 1 is false has not been ensured. So, just achieving statement coverage and this code is not ensuring branch coverage and the code. So, this a simple example, which says that branch coverage does not ensure statement coverage.

There are many coverage tools and coverage tool as the test case are executed, it displays coverage report. In the coverage report it displays that, what are the statement coverage, branch coverage, etcetera achieved for different functions.

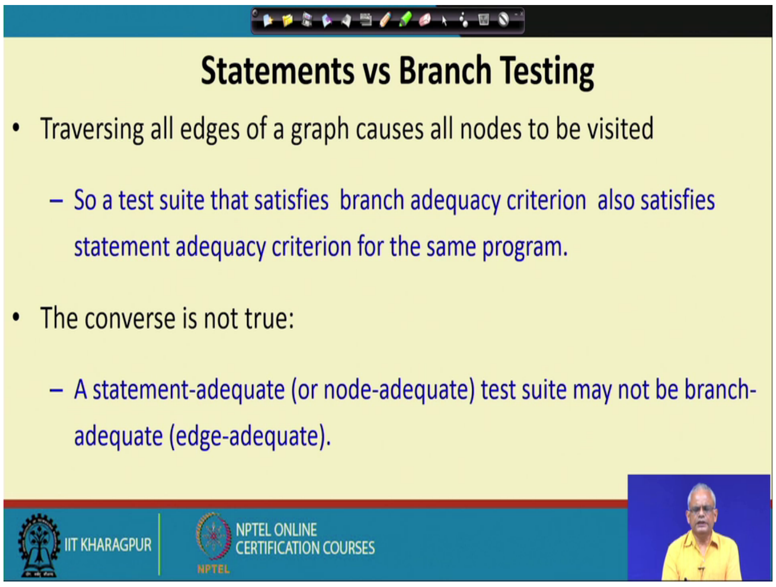
(Refer Slide Time: 20:14)



```
110 128          else if ( nav.isElement( first ) )
111              {
112 100              return nav.getElementQName( first );
113              }
114  28          else if ( nav.isAttribute( first ) )
115              {
116   0              return nav.getAttributeQName( first );
117              }
118  28          else if ( nav.isProcessingInstruction( first ) )
119              {
120   0              return nav.getProcessingInstructionTarget( first );
121              }
122  28          else if ( nav.isNamespace( first ) )
123              {
124   0              return nav.getNamespacePrefix( first );
125              }
126  28          else if ( nav.isDocument( first ) )
127              {
128  28              return "";
129              }
130   0          else if ( nav.isComment( first ) )
131              {
132   0              return "";
133              }
134   0          else if ( nav.isText( first ) )
135              {
136   0              return "";
137              }
138              else {
139   0              throw new FunctionCallException("The argument to the name
140              }
141          }
142
143   8      return "";
144
```

And also, many tools that display what are the statements that are still not got executed. So, these are shown in the red and we know that statement coverage has not been achieved and these are the statements that have not been executed so far.

(Refer Slide Time: 20:38)

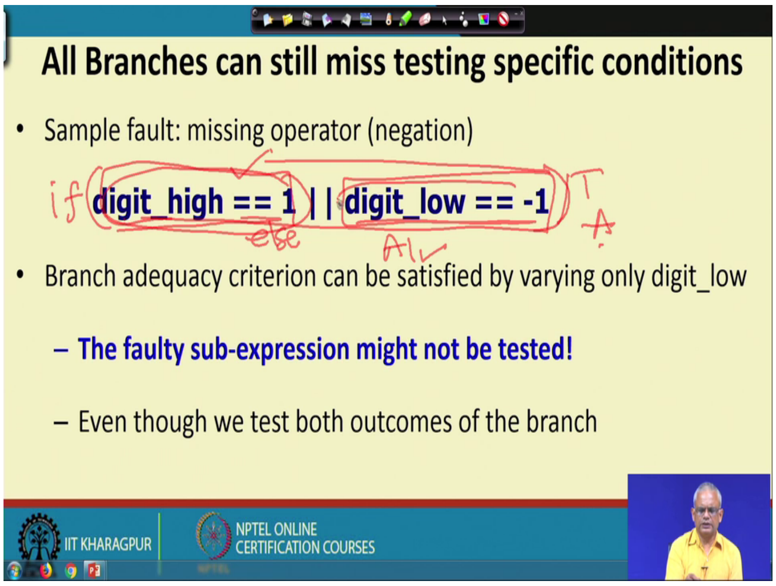

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

Statements vs Branch Testing

- Traversing all edges of a graph causes all nodes to be visited
  - So a test suite that satisfies branch adequacy criterion also satisfies statement adequacy criterion for the same program.
- The converse is not true:
  - A statement-adequate (or node-adequate) test suite may not be branch-adequate (edge-adequate).

So, we have seen that branch coverage is stronger testing, a branch coverage achieve statement coverage, but the converse is not true. The statement coverage, if we achieve statement coverage that does not mean that branch coverage has been achieved.

(Refer Slide Time: 21:01)



All Branches can still miss testing specific conditions

- Sample fault: missing operator (negation)

  if (digit_high == 1 || digit_low == -1) A

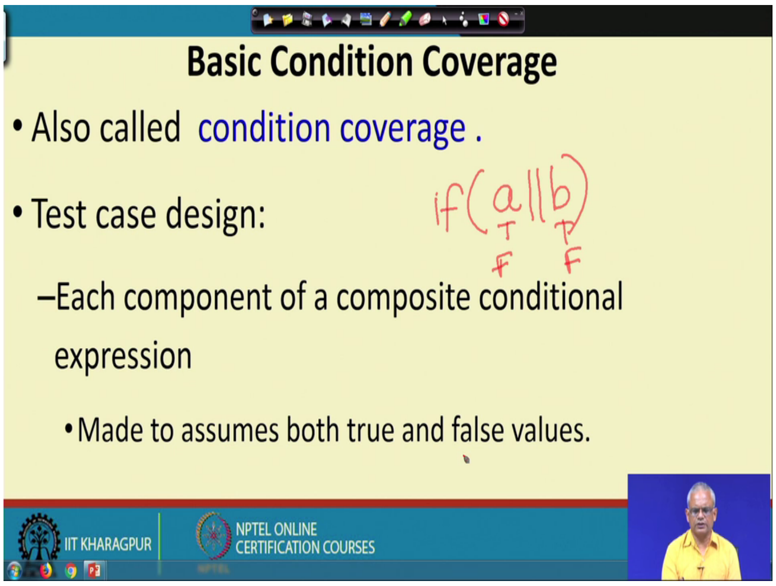
- Branch adequacy criterion can be satisfied by varying only digit_low
  - **The faulty sub-expression might not be tested!**
  - Even though we test both outcomes of the branch

Now, so far we have looked at the statement coverage and branch coverage, but is a branch coverage is a good enough testing connect miss some bugs, that is we achieve branch coverage, but then some types of bugs are not exposed. And, this is what we are going to discuss now. That we have a certain branch condition, if digit is high, if digit high is 1 or digit low is minus 1 then do some action A.

Now, the problem here is that we achieve. If digit high is 1 or digit low is minus 1, then the action is A else action is A 1 and so on. Now, let us say we achieve branch coverage here, that we have this branch condition is true. So, A is executed branch condition is false and then A 1 is executed, but then just observe here that to get this true, we might have digit high equal to 1, that will make it true irrespective of digit low is minus 1. And, let us say we have digit high is 0 and digit low is not minus 1 and then this becomes false. So, we have achieved both high both true and false for this specific branch.

But, then let us say that we have some action here, where the which will encounter a failure only when digit low is minus 1. In that case we will not be able to discover that bug by testing. And therefore, not only that we must ensure that the decision is both true and false, we must ensure that all component conditions here are clauses are true and false. So, branch coverage even though it is a strong testing, but then it may not achieve it may not expose many types of bugs. And, we might have to do a stronger testing and a stronger testing is component clause in the decision statement must achieve both true and false values.

(Refer Slide Time: 24:24)



Basic Condition Coverage

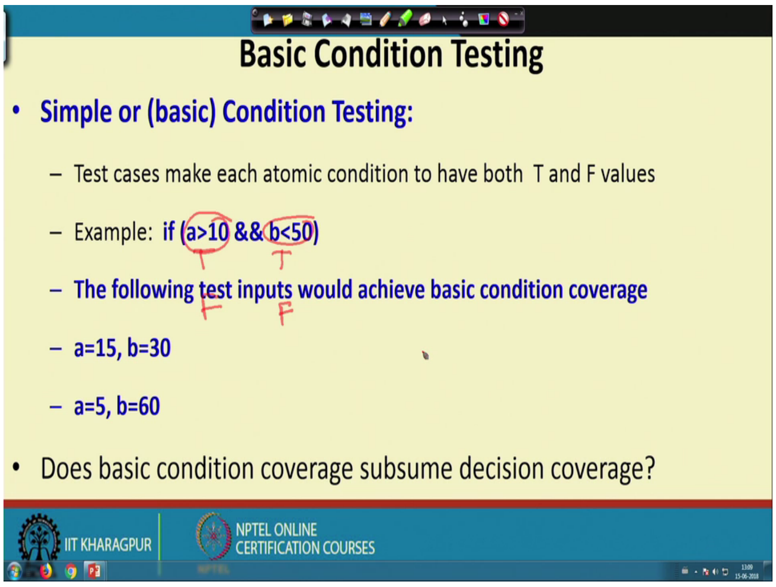
- Also called condition coverage .
- Test case design:
  - Each component of a composite conditional expression
    - Made to assumes both true and false values.

$\text{if}(\underset{F}{\underset{T}{a}} \| \underset{F}{\underset{T}{b}})$

And, that brings us to the basic condition coverage; in the basic condition coverage each component condition must achieve true and false values. So, if we have if a or b, then a must be true and false the test cases must ensure that a test true and false values be also test true and false values.

In the basic condition coverage, we must ensure that the component clauses here on the decision statement take true and false values. So, each component of a composite conditional expression must take true and false values.

(Refer Slide Time: 25:35)



Basic Condition Testing

- Simple or (basic) Condition Testing:
  - Test cases make each atomic condition to have both T and F values
  - Example: if (a>10 && b<50)
  - The following test inputs would achieve basic condition coverage
  - a=15, b=30
  - a=5, b=60
- Does basic condition coverage subsume decision coverage?

T
T
F
F
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

Let us consider the example if a greater than 10 and b less than 50. Now, if a equal to 15 and b equal to 30. So, a is 15 this becomes true and b is 30 this becomes true. Now, for the other test case a equal to 5 b equal to 60 a is 5 5 less greater than 10 is false, and b is 60 60 less than 50 is false.

So, these two test cases ensure that both the component clauses, ensure achieve the true and false values. And, if there is any expression in including dozens of clauses, it is possible that only 2 test cases may be able to achieve basic condition coverage. Because one test case may give all true and another test case may assign false to every condition.

And therefore, it is possible that in some large decision statement, consisting a many clauses just 2 test cases may be able to achieve basic condition testing. And, that gives us a hint that this may not be a very strong coverage criterion, but then does it ensure if we achieve basic condition testing, does it ensure branch coverage we need to answer that question is the basic condition testing a stronger testing than the decision coverage or branch coverage, or is it a weaker testing or is it a complementary testing. Now, we are almost at the end of the lecture, we will discuss this point and other white box testing strategies in the next lecture.

Thank you.