Announcements

- Last extra credit opportunity today with quizizz
- Todays quizizz is for participation as well
- In class activity today
- TA Workshop next Tuesday on “Modern Payment Systems and Serverless Cloud in E-commerce”

Feedback

Good

- Project based approach to the course.
- Open to any tech stack
- AI integration but not major focus
- Demos in class for new mostly used tools
    - Effectiveness average: 4.1/5
- Quizizz to focus attention

Mixed

More focused small assignments to reinforce knowledge/Less assignments to focus on project – It’s a balance

Feedback

Can be improved

- More time to work on project in class – **scheduling constraints,summer semester, noticed if give half an hour to 1 hour – nobody stays back in class**
- Reduce documentation based assignments – **agree to some extent but have to abide by course syllabus and requirements. They are important for holistic and large scale organizational pipeline knowledge**
- Release lecture notes before class – I always do.
- Hiccups during demo **– Agree! Should include TAs! And also record to provide to students later on (but that also defeats attendance)**
- Timings and due date unclear **–** always on course website and canvas.

CS3300 Introduction to Software Engineering

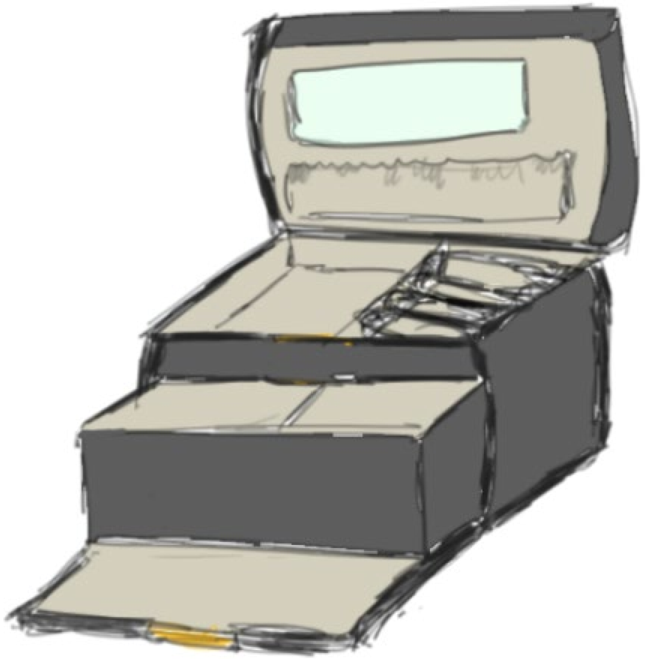
Lecture 10: White-Box Testing

Dr. Nimisha Roy

- nroy9@gatech.edu

Slides adapted from Alessandro Orso

White- Box Testing

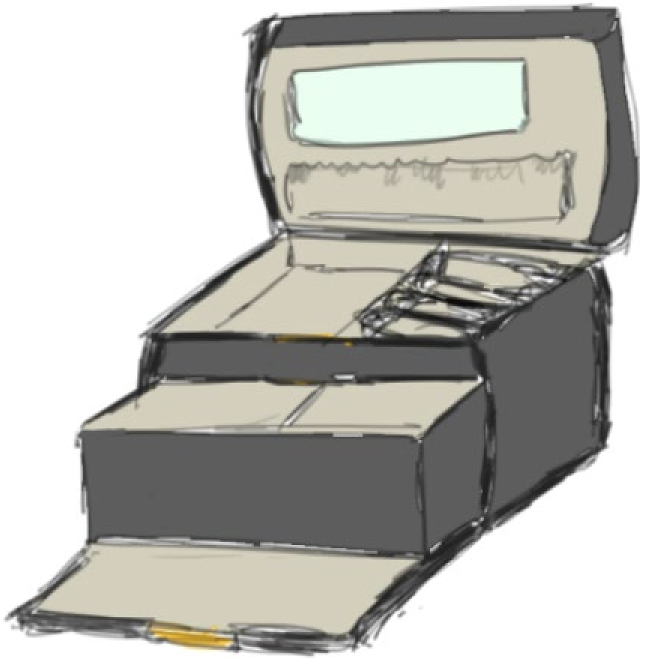


Basic Assumption

Executing the faulty statement is a necessary condition for revealing a fault

Slide adapted from Alessandro Orso

White- Box Testing

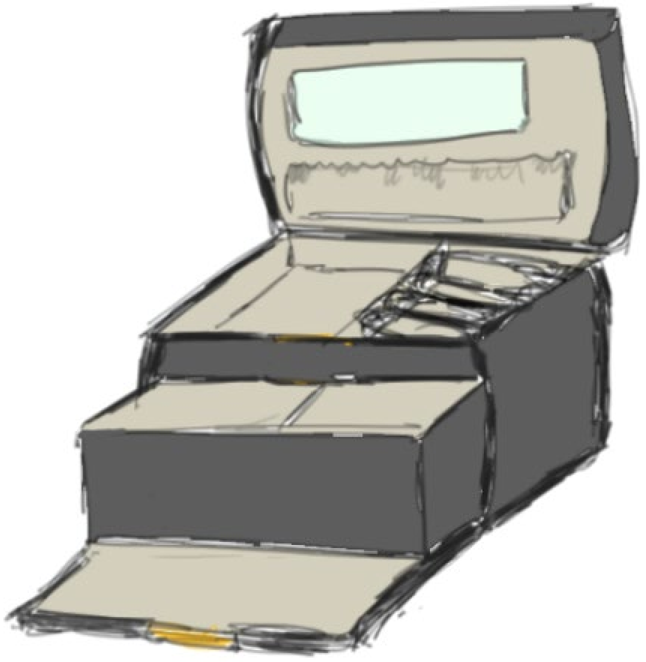


Advantages

- Based on the code
  - Can be measured objectively
  - Can be measured automatically
- Can be used to compare test suites
- Allows for covering the coded behavior

Slide adapted from Alessandro Orso

White- Box Testing



Different Kinds

- Control-Flow Based
- Data-flow based
- Fault based

Slide adapted from Alessandro Orso

Let’s Consider Program printSum()

```
printSum (int a, int b) {
   int result = a+b;
   if (result > 0)
      printcol(“red”, result);
   else if (result < 0)
      printcol(“blue”, result);
}
```

Slide adapted from Alessandro Orso

Coverage Criteria

Defined in terms of

Test requirements - Elements/entities in the code that we need to execute

Result in

Test specifications

Test cases

Slide adapted from Alessandro Orso

printSum: Test Requirements

```
printSum (int a, int b) {
   int result = a+b;
   if (result > 0)
      printcol("red", result);
   else if (result < 0)
      printcol("blue", result);
}
```

Req #1

Req #2

Slide adapted from Alessandro Orso

printSum: Test Specifications

```
printSum (int a, int b) {
   int result = a+b;
   if (result > 0)
      printcol("red", result);
   else if (result < 0)
      printcol("blue", result);
}
```

Test Spec #1
a + b > 0

Test Spec #2
a + b < 0

Slide adapted from Alessandro Orso

printSum: Test Cases

```
printSum (int a, int b) {
   int result = a+b;
   if (result > 0)
      printcol("red", result);
   else if (result < 0)
      printcol("blue", result);
}
```

Test Spec #1
a + b > 0

Test Spec #2
a + b < 0

#1 ((a = [5], b = [-4]), (output color = [red], output value = [1]

#2 ((a = [0], b = [-1]), (output color = [blue], output value = [-1]

Slide adapted from Alessandro Orso

Coverage Criteria: Statement Coverage

Test Requirements

Statements in the program

Coverage Measure

$$\frac{\text{Number of executed Statements}}{\text{Total number of Statements}}$$

Slide adapted from Alessandro Orso

printSum: statement coverage

TC #1
a == 5
b == -4

```
printSum (int a, int b) {
   int result = a+b;
   if (result >= 0)
      printcol(“red”, result);
   else if (result < 0)
      printcol(“blue”, result);
}
```

Coverage: 0%

printSum: statement coverage

TC #1
a == 5
b == -4

```
printSum (int a, int b) {
   int result = a+b;
   if (result >= 0)
      printcol(“red”, result);
   else if (result < 0)
      printcol(“blue”, result);
}
```

Coverage: 71%

printSum: statement coverage

TC #1
a == 5
b == -4

TC #2
a == 0
b == -1

```
printSum (int a, int b) {
   int result = a+b;
   if (result >= 0)
      printcol(“red”, result);
   else if (result < 0)
      printcol(“blue”, result);
}
```

Coverage: 100%

Statement coverage in Practice

Most used in Industry

"Typical coverage" target is 80 – 90%

Why don't we aim at 100%

[Unreachable code, dead code, complex sequences,]
[Not enough resources]

printSum: statement coverage

TC #1
a == 5
b == -4

TC #2
a == 0
b == -1

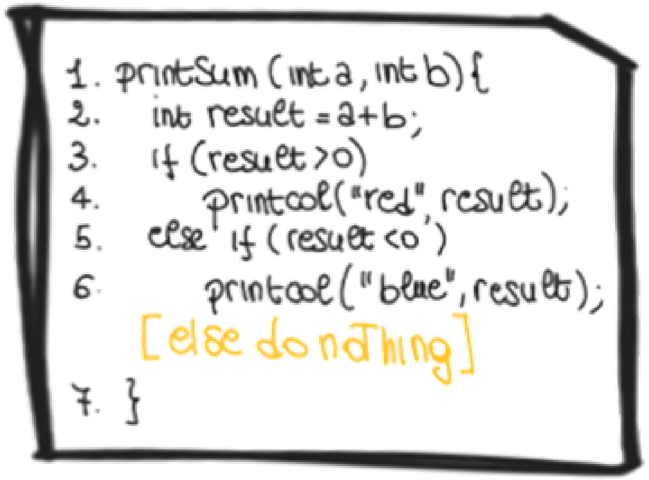
```
printSum (int a, int b) {
   int result = a+b;
   if (result >= 0)
      printcol("red", result);
   else if (result < 0)
      printcol("blue", result);
  else
     print("no result");
}
```

Coverage is never 100%

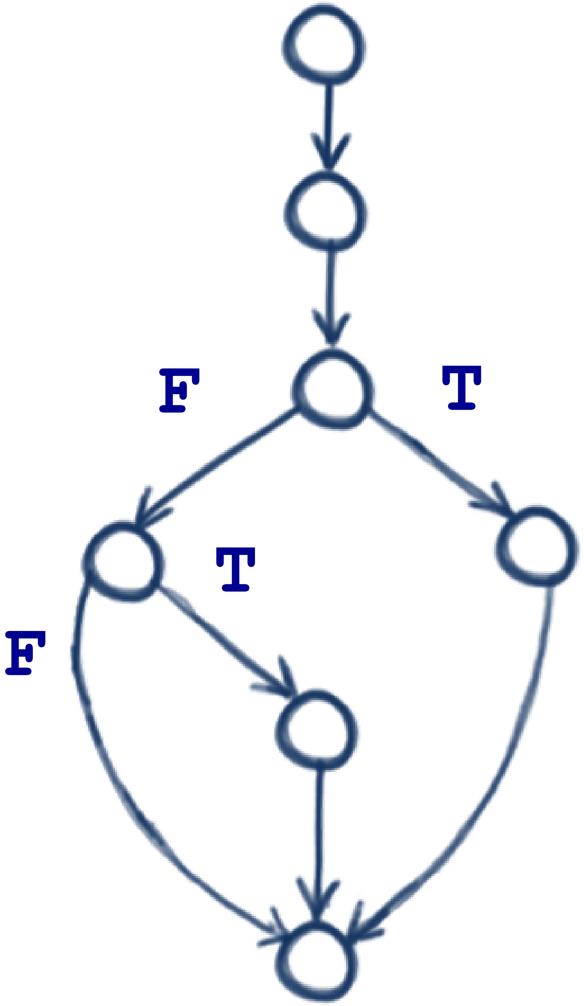
Control Flow Graphs

Representation for the code that is very convenient when we run our reason about the code and its structure.

Represents statement with nodes and the flow of control within the code with edges.



```
printSum(int a, int b){
   int result = a+b;
   if (result >0)
      printcol("red", result);
   else if (result <0)
      printcol("blue",result);
   [else donothing]
}
```



Coverage Criteria: Branch Coverage

Test Requirements

Branches in the program: outgoing edges from a decision point

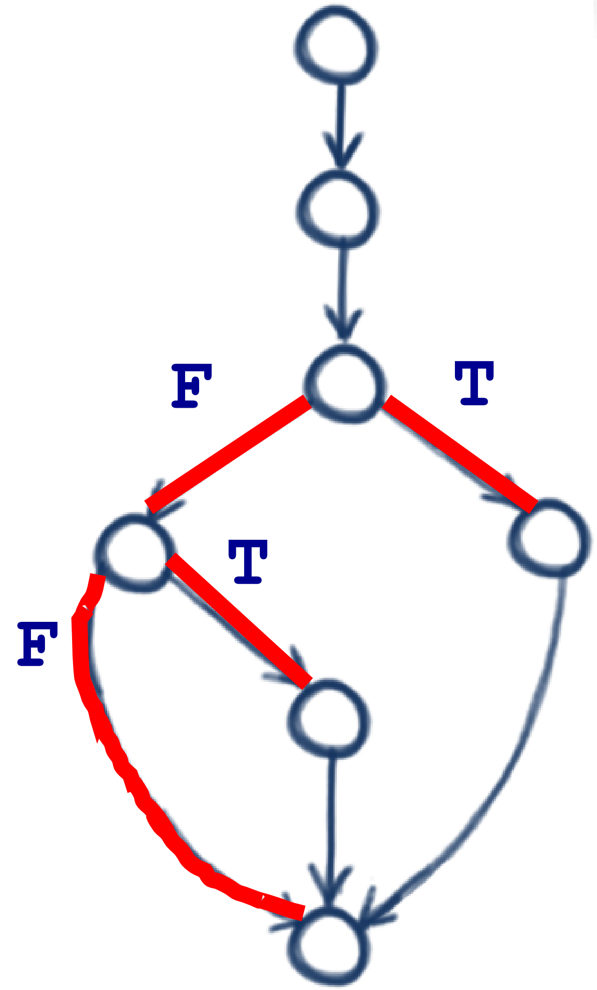
Coverage Measure

$$\frac{\text{Number of executed Branches}}{\text{Total number of Branches}}$$

Slide adapted from Alessandro Orso

printSum: Branch coverage

```
printSum (int a, int b) {
   int result = a+b;
   if (result > 0)
      printcol("red", result);
   else if (result < 0)
      printcol("blue", result);
   [else DO NOTHING]
}
```

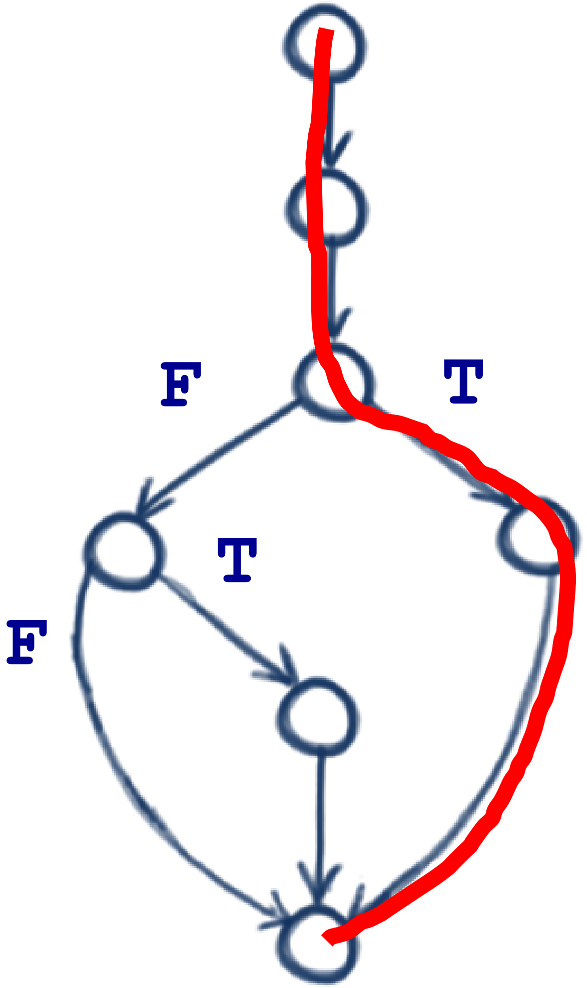


How many branches? [4]

printSum: Branch coverage

TC #1
a == 5
b == -4

```
printSum (int a, int b) {
    int result = a+b;
    if (result > 0)
        printcol(“red”, result);
    else if (result < 0)
        printcol(“blue”, result);
    [else DO NOTHING]
}
```



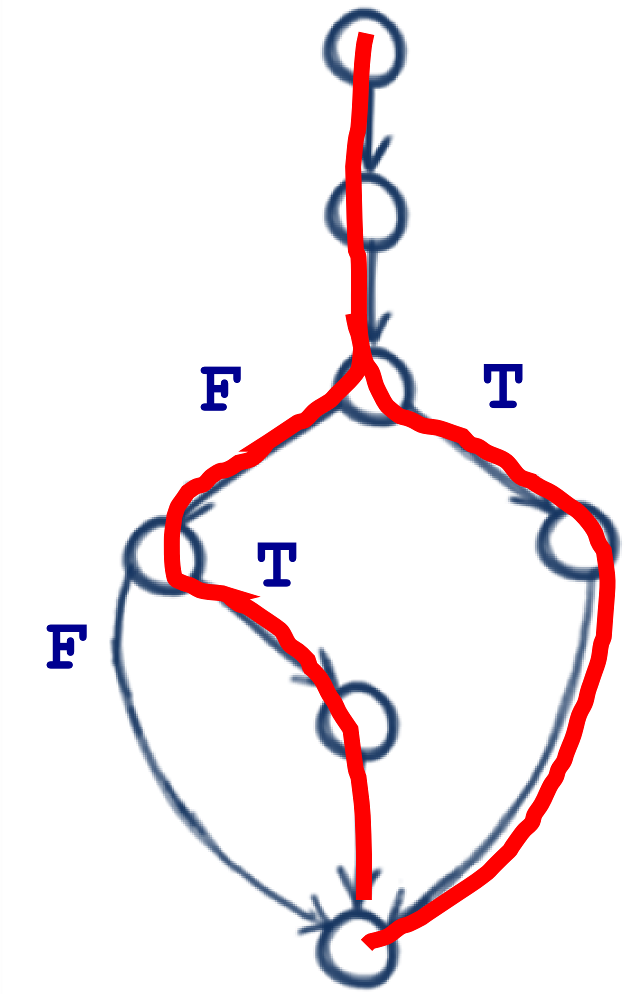
Coverage [25 %]

printSum: Branch coverage

TC #1
a == 5
b == -4

TC #2
a == 0
b == -1

```
1.  printSum (int a, int b) {
2.      int result = a+b;
3.      if (result > 0)
4.          printcol("red", result);
5.      else if (result < 0)
6.          printcol("blue", result);
7.      [else DO NOTHING]
8.  }
```



Coverage [75 %]

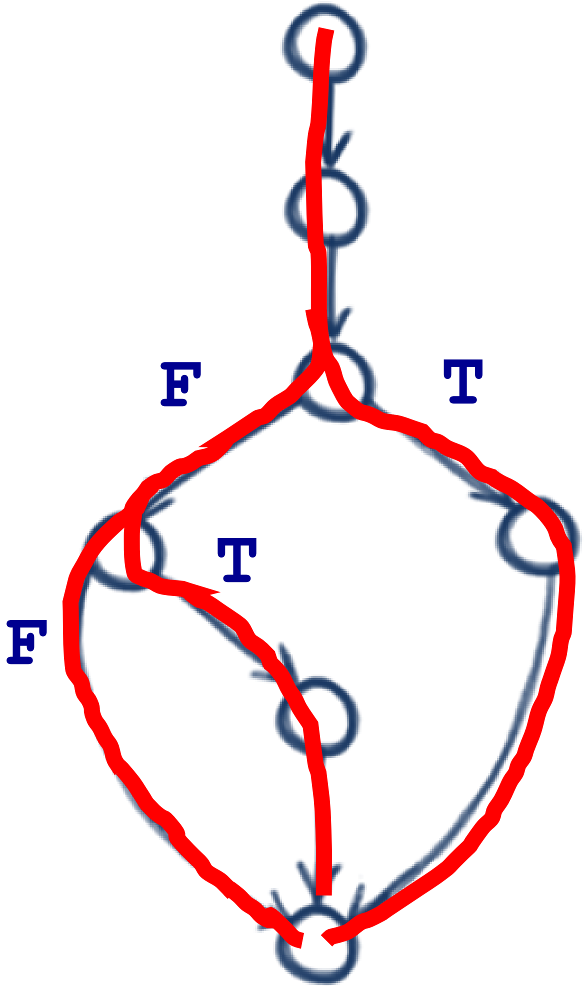
printSum: Branch coverage

TC #1
a == 5
b == -4

TC #2
a == 0
b == -1

TC #3
a == 0
b == 0

```
printSum (int a, int b) {
    int result = a+b;
    if (result > 0)
        printcol("red", result);
    else if (result < 0)
        printcol("blue", result);
    [else DO NOTHING]
}
```

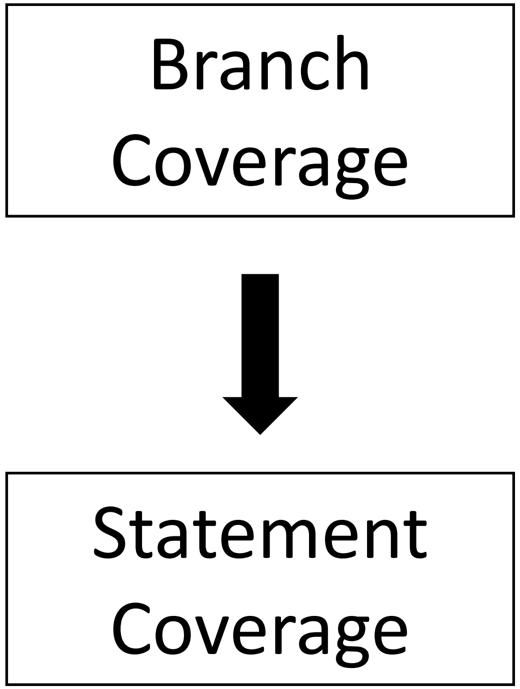


Coverage [100 %]

Note: 100% coverage does not provide any guarantee of finding the problems in the code.

Test Criteria Subsumption

One test criteria subsumes another criteria when all the test suites that satisfy that criteria will also satisfy the other one



Branch Coverage

Statement Coverage

Branch Coverage is a stronger criteria than Statement Coverage. There is no way of covering all branches but leaving out some statements.

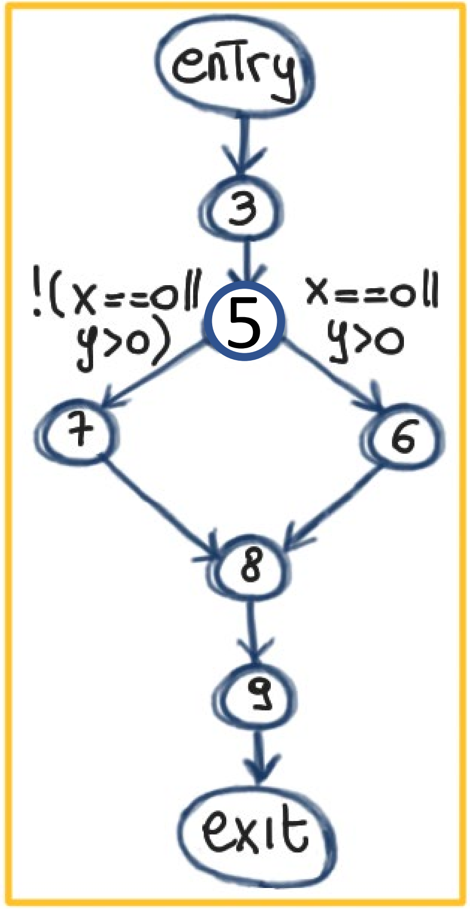
Slide adapted from Alessandro Orso

Lets consider another example

```
void main () {
   float x, y;
   read (x);
   read (y);
   if ((x== 0) || (y > 0))
      y = y/x;
   else         x = y+2;
   write (x);
   write(y);
}
```

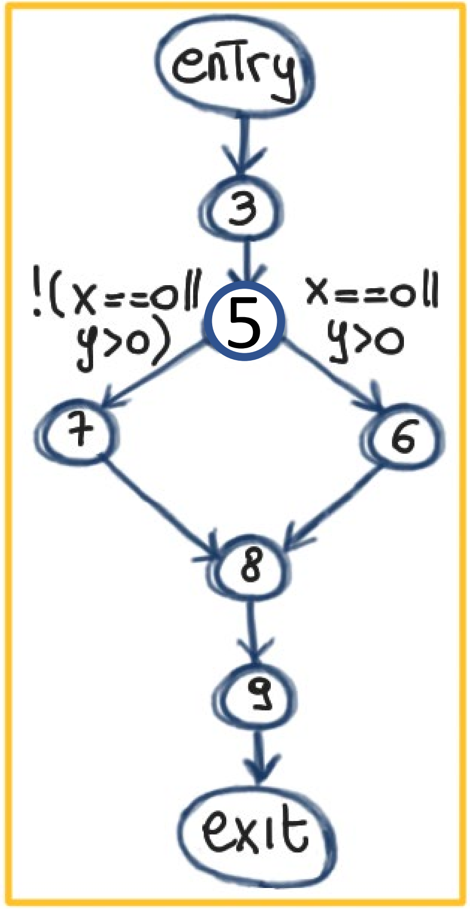
Lets consider another example

```
1. void main () {
2.    float x, y;
3.    read (x);
4.    read (y);
5.    if ((x== 0) || (y > 0))
6.        y = y/x;
7.    else            x = y+2;
8.    write (x);
9.    write(y);
10.}
```



Lets consider another example

```
1. void main () {
2.    float x, y;
3.    read (x);
4.    read (y);
5.    if ((x== 0) || (y > 0))
6.        y = y/x;
7.    else          x = y+2;
8.    write (x);
9.    write(y);
10.}
```



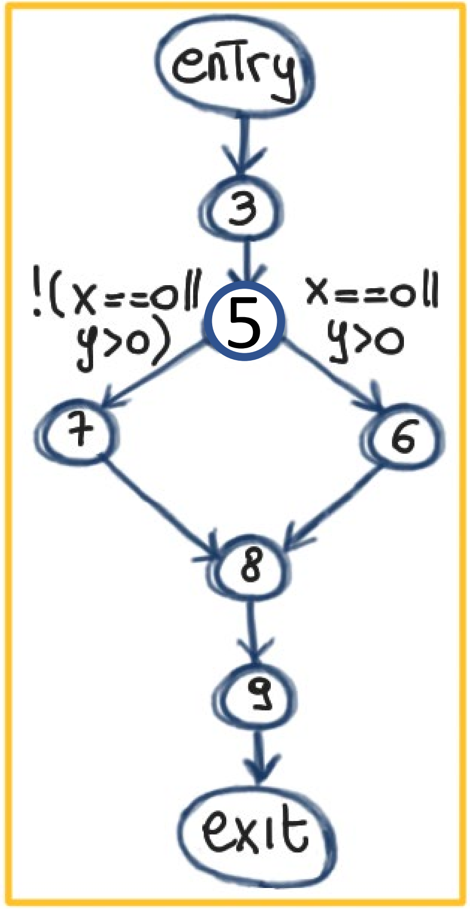
x = 5; y = 5;
x = 5; y = -5;

Branch Coverage: ?

100%

Lets consider another example

```
1. void main () {
2.    float x, y;
3.    read (x);
4.    read (y);
5.    if ((x== 0) || (y > 0))
6.        y = y/x;
7.    else         x = y+2;
8.    write (x);
9.    write(y);
10.}
```



x = 5; y = 5;
x = 5; y = -5;

Branch Coverage:

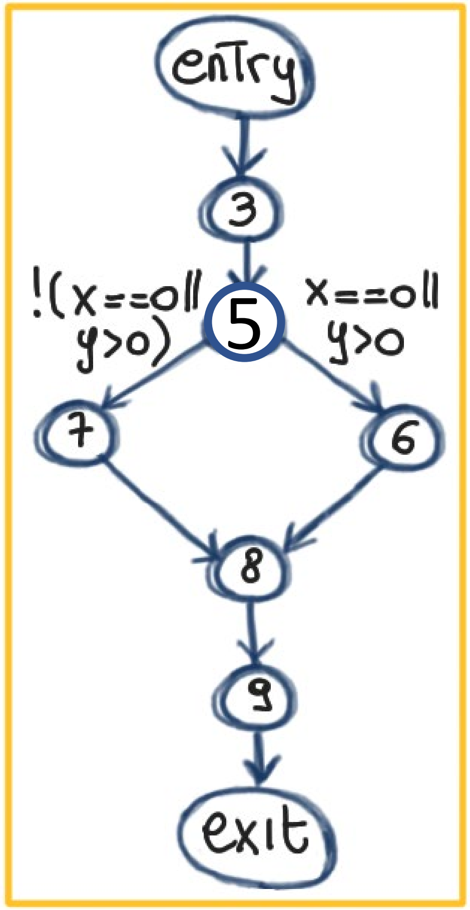
100%

Identify a test case when code can fail:

x = 0, y can be anything

Lets consider another example

```
1. void main () {
2.    float x, y;
3.    read (x);
4.    read (y);
5.    if ((x== 0) || (y > 0))
6.        y = y/x;
7.    else          x = y+2;
8.    write (x);
9.    write(y);
10.}
```



x = 5; y = 5;
x = 5; y = -5;

Branch Coverage:

100%

Identify a test case when code can fail:

x = 0

How can we be more thorough?

Each condition T and F

Coverage Criteria: Condition Coverage

Test Requirements

Individual Conditions in the program

Coverage Measure

$$\frac{\text{Number of conditions that are both T and F}}{\text{Total number of Conditions}}$$

Has each condition evaluated to true and false?

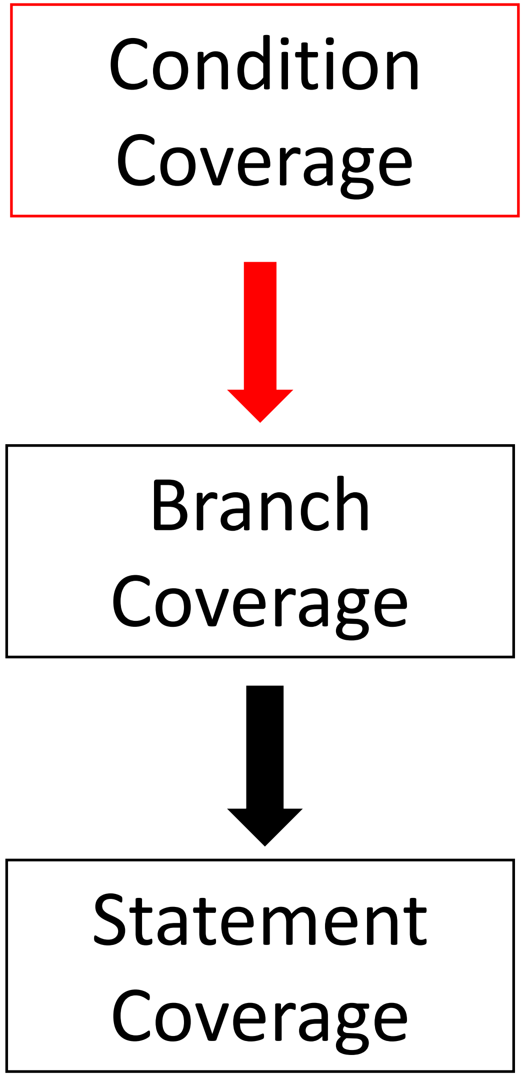
Slide adapted from Alessandro Orso

Subsumption

Does Condition Coverage imply branch coverage?

[ ] Yes

[✓] No



- **Condition Coverage**: Each **boolean sub-expression** (e.g., A, B) within a compound condition is tested to be both true and false.
- **Branch Coverage**: Each **entire decision outcome** (e.g., the result of if (A && B)) is tested as both true and false.

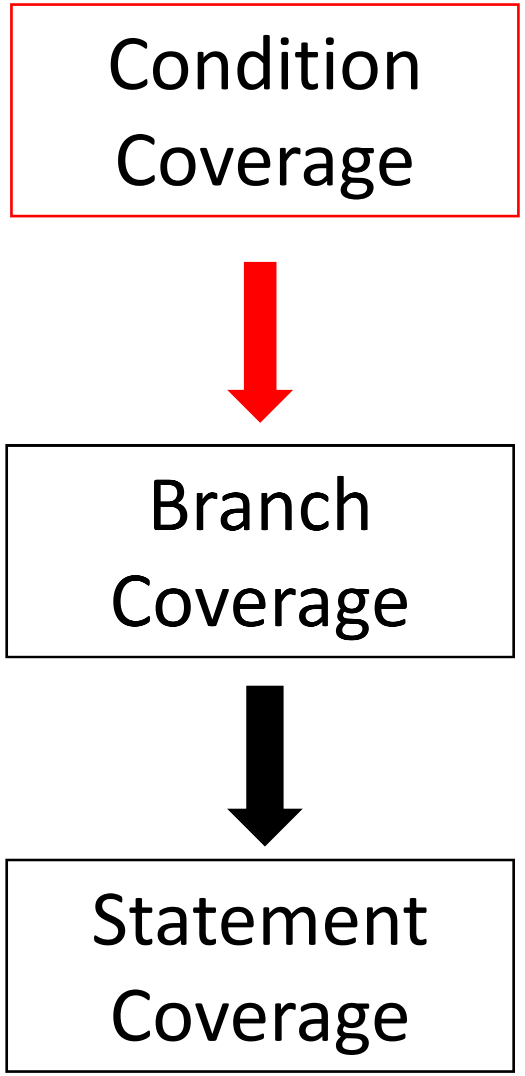
Slide adapted from Alessandro Orso

Subsumption

```
if (A && B) {
    doSomething();
}
```

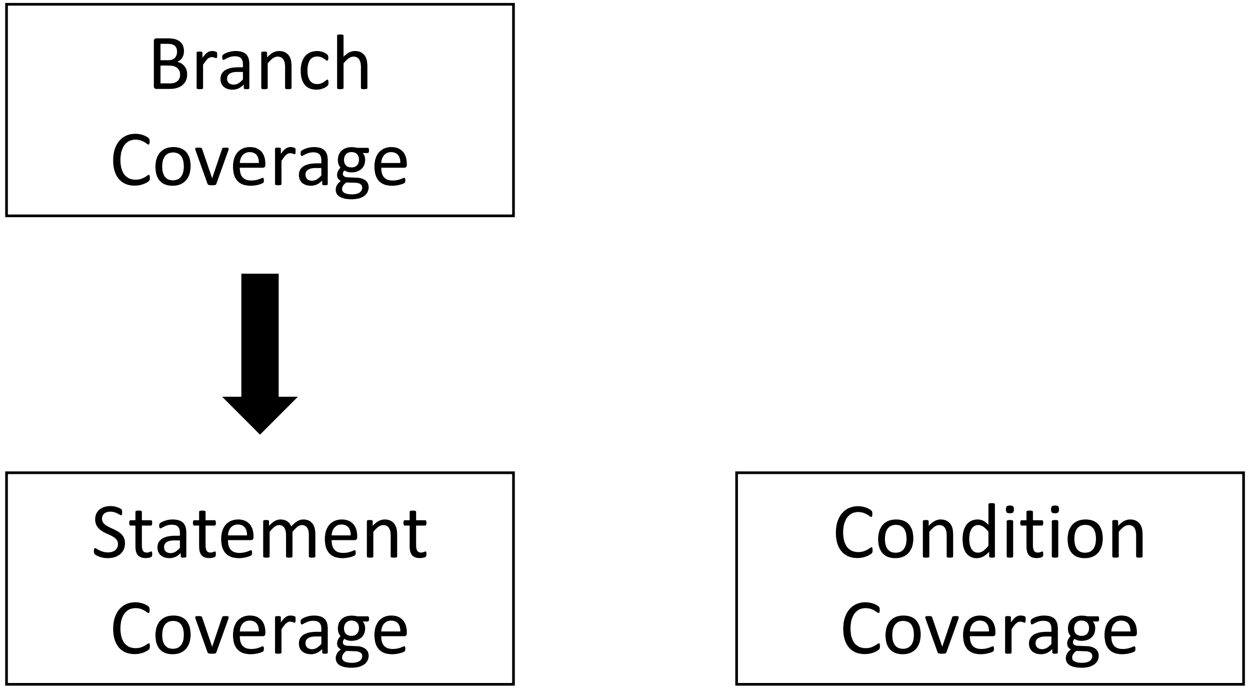
T1: A = true, B = false
T2: A = false, B = true

100% condition coverage and 50% branch coverage



Slide adapted from Alessandro Orso

Test Criteria Subsumption



Branch Coverage

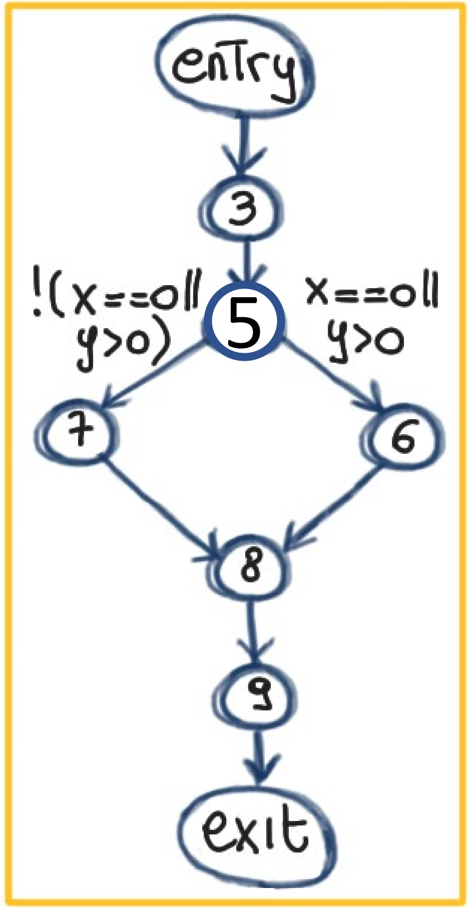
Statement Coverage

Condition Coverage

Slide adapted from Alessandro Orso

Lets consider the previous example

```
1. void main () {
2.    float x, y;
3.    read (x);
4.    read (y);
5.    if ((x== 0) || (y > 0))
6.        y = y/x;
7.    else        x = y+2;
8.    write (x);
9.    write(y);
10.}
```



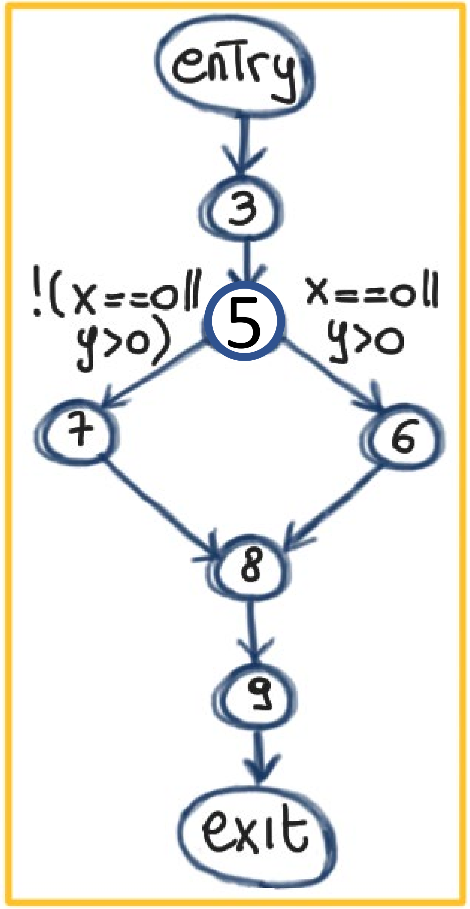
x = 0; y = -5;
x = 5; y = 5;

Condition Coverage: ?

100%

Lets consider the previous example

```
1. void main () {
2.    float x, y;
3.    read (x);
4.    read (y);
5.    if ((x== 0) || (y > 0))
6.        y = y/x;
7.    else          x = y+2;
8.    write (x);
9.    write(y);
10.}
```



x = 0; y = -5;
x = 5; y = 5;

Condition Coverage: ?

100%

Branch Coverage:?

50 %

Coverage Criteria: Branch and Condition Coverage

Test Requirements	Branches and Individual Conditions in the program
Coverage Measure	Computed using both coverage measures

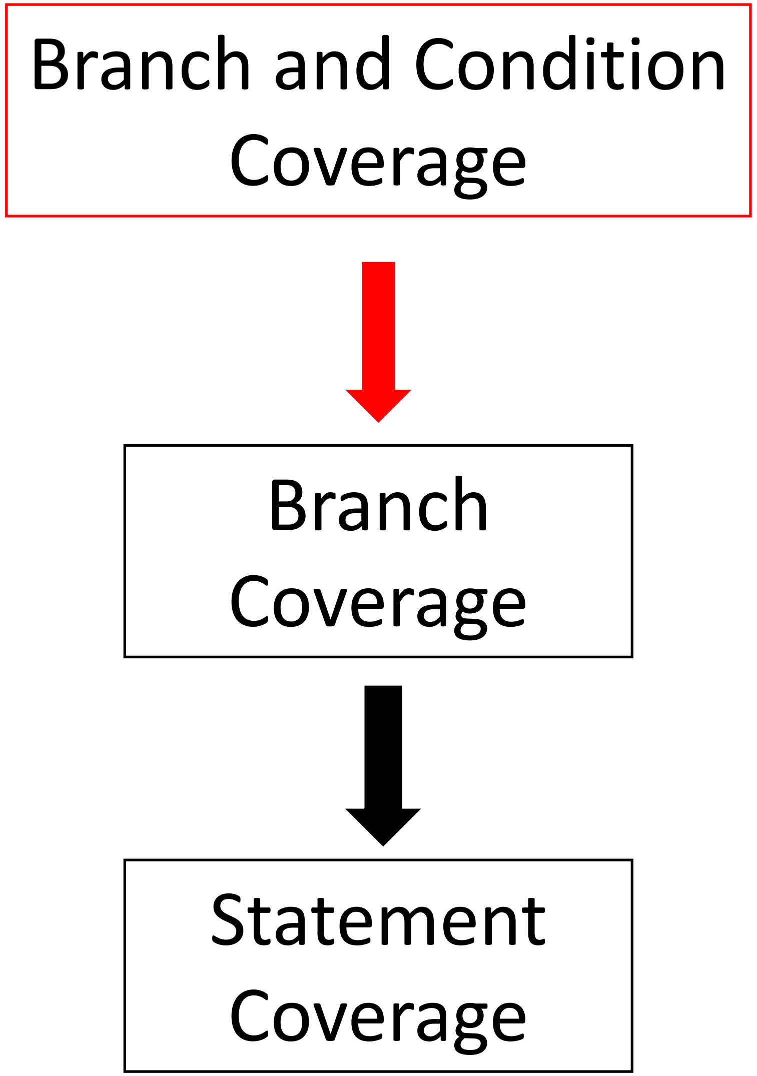
Slide adapted from Alessandro Orso

Subsumption

Does Branch and Condition Coverage imply branch coverage?

[✓] Yes

[ ] No



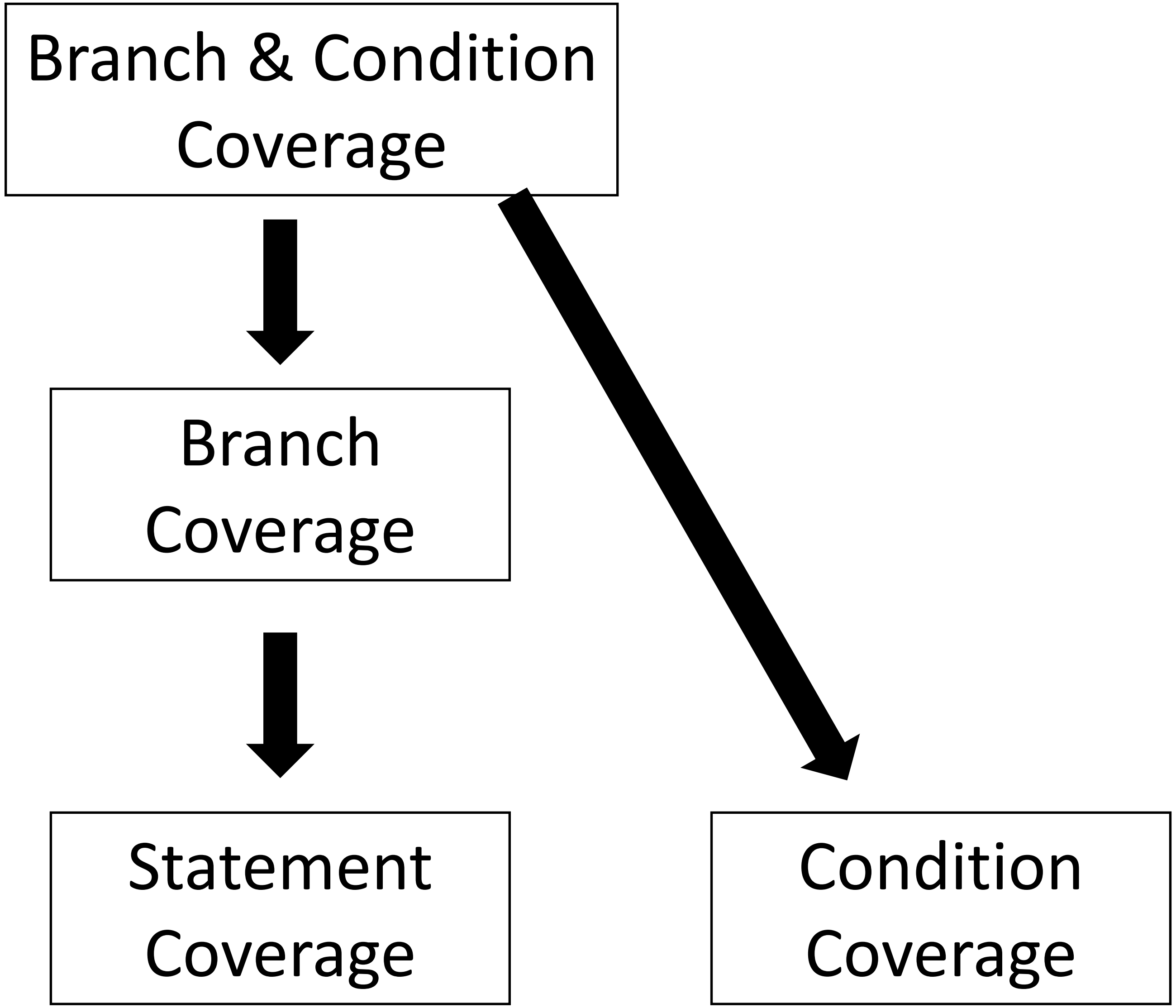
Branch and Condition Coverage

Branch Coverage

Statement Coverage

Slide adapted from Alessandro Orso

Test Criteria Subsumption



Branch & Condition Coverage

Branch Coverage

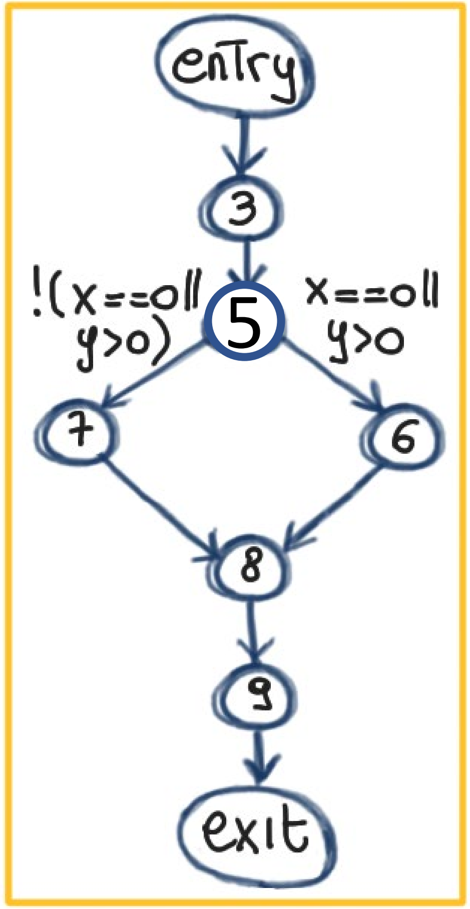
Statement Coverage

Condition Coverage

Slide adapted from Alessandro Orso

Achieving 100% B&C Coverage

```
1. void main () {
2.    float x, y;
3.    read (x);
4.    read (y);
5.    if ((x== 0) || (y > 0))
6.        y = y/x;
7.    else          x = y+2;
8.    write (x);
9.    write(y);
10.}
```



x = 0; y = -5;
x = 5; y = 5;

Add a test case to achieve 100% B&C Coverage

x = 3, y = -2

Multiple Condition Coverage – permutation-combination of conditions in a decision statement

Coverage Criteria: Modified Condition/Decision Coverage

Very Important Criteria; Often required for safety critical applications. For example: FAA requires SW that runs on commercial airplanes to be tested according to this criteria

Key Idea: Test important combinations of conditions and limited testing costs

Extend Branch and Condition Coverage with the requirement that each condition should affect the decision outcome independently

Slide adapted from Alessandro Orso

MC/DC Example

a && b && c

	Test Case	A	B	C	Outcome
✓	1	True	True	True	True
	2	True	True	False	False
	3	True	False	True	False
	4	True	False	False	False
✓	5	False	True	True	False
	6	False	True	False	False
	7	False	False	True	False
	8	False	False	False	False

1	True	True	True	True
5	False	True	True	False

MC/DC Example

a && b && c

	Test Case	A	B	C	Outcome
✓	1	True	True	True	True
	2	True	True	False	False
✓	3	True	False	True	False
	4	True	False	False	False
	5	False	True	True	False
	6	False	True	False	False
	7	False	False	True	False
	8	False	False	False	False

1	True	True	True	True
5	False	True	True	False
3	True	False	True	False

MC/DC Example

a && b && c

	Test Case	A	B	C	Outcome
✓	1	True	True	True	True
✓	2	True	True	False	False
	3	True	False	True	False
	4	True	False	False	False
	5	False	True	True	False
	6	False	True	False	False
	7	False	False	True	False
	8	False	False	False	False

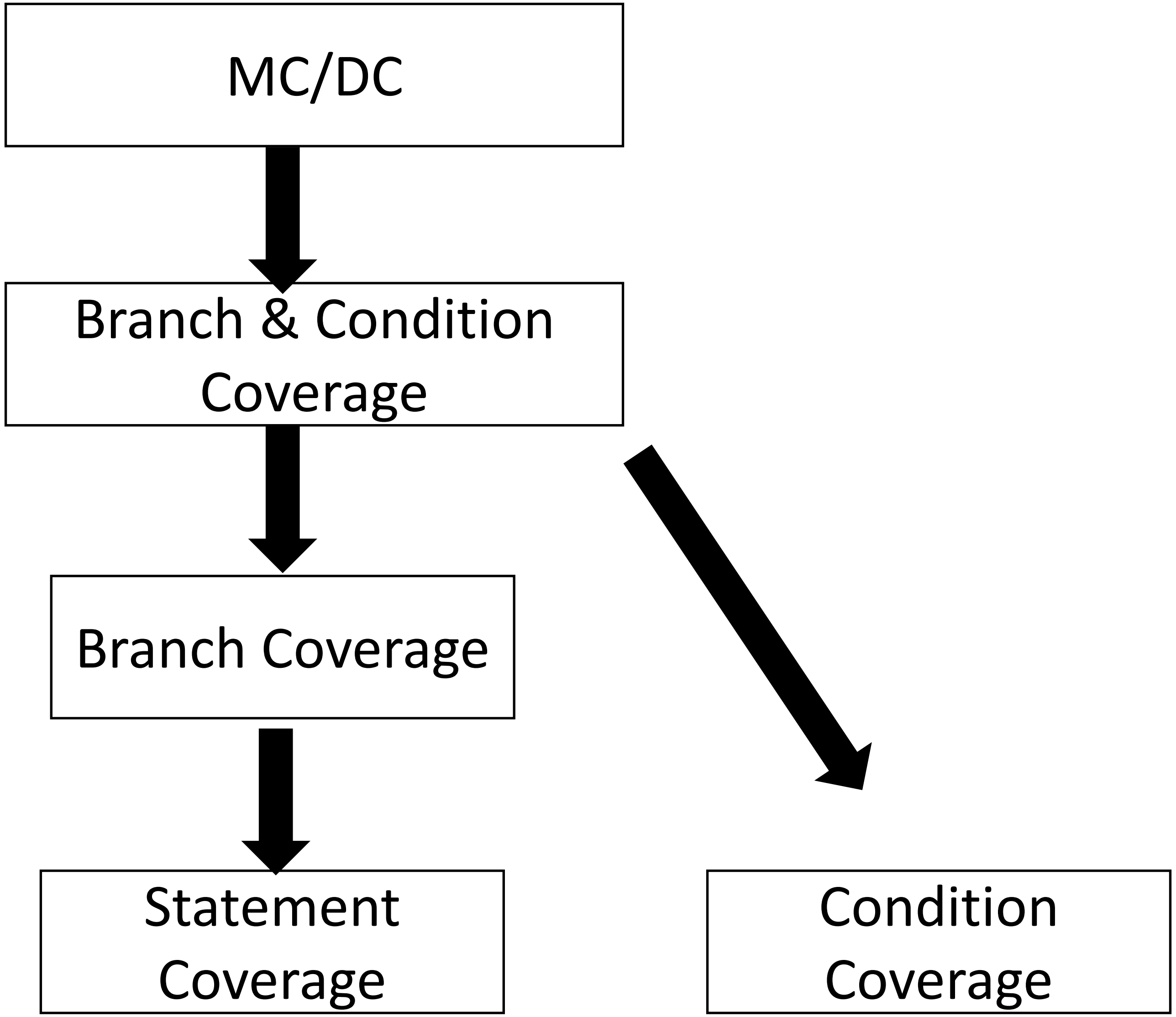
8 TC

To

4 TC

1	True	True	True	True
5	False	True	True	False
3	True	False	True	False
2	True	True	False	False

Test Criteria Subsumption



MC/DC

Branch & Condition Coverage

Branch Coverage

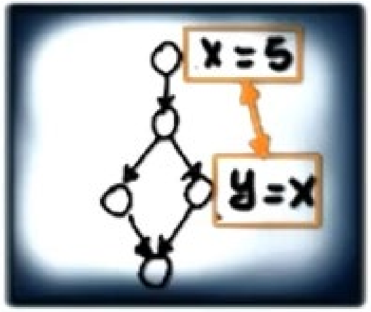
Statement Coverage

Condition Coverage

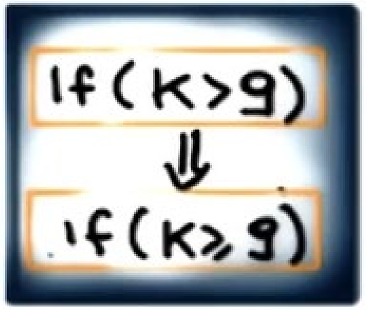
Slide adapted from Alessandro Orso

Other Criteria

Path Coverage (all paths are covered- incredibly expensive)

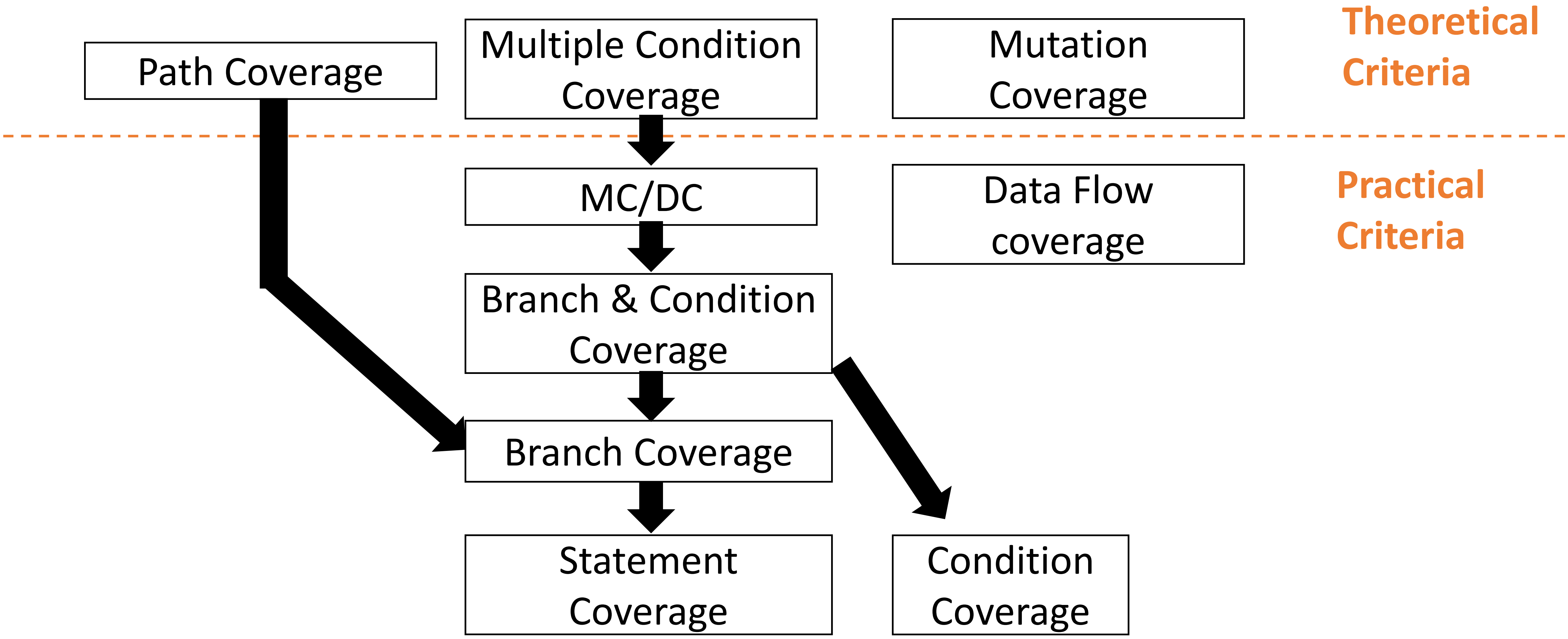


Data-Flow Coverage Instead of focusing on control structures, you look at how data moves. You identify every point where a variable is defined (assigned) and every point it's used, and make sure tests cover at least one path from each definition to each use.It catches bugs like uninitialized variables, overwrites etc.



Mutation Coverage (evaluate goodness of test by modifying the code; The more mutants identified by test, the better they are at identifying real faults)

Test Criteria Subsumption



Slide adapted from Alessandro Orso

White box testing Quiz

```
1. int i;
2. read (i);
3. print (10/((i-3))
```

Test Suite: (1, -5), (-1, 2.5), (0, -3.3)

Does it achieve path coverage?

Yes

Does it reveal the fault at line 3?

No

Even path coverage couldn’t detect the fault. Exhaustive testing is the only way to ensure all possible test cases.

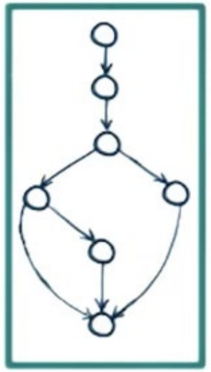
White box testing Quiz

```
int i = 0;
int j;
read (j);
if ((j > 5) && (i > 0))
   print (i)
```

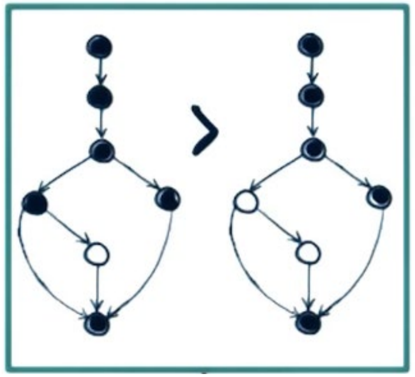
Can you create a test suite to adhere statement coverage?

No; Dead/ Unreachable Code.
infeasible paths, inexecutable statements, conditions that can never be true all are present in codes. Hence industry targets ~80% coverage

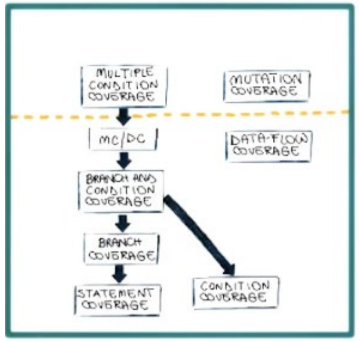
White-box testing Summary



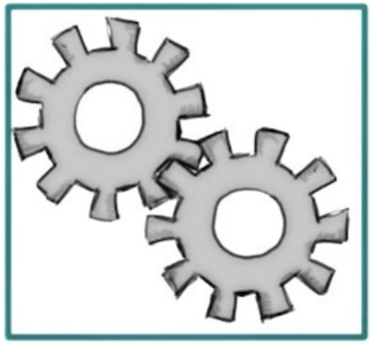
Works on a formal Model - No subjective decisions on level of abstraction needed



Comparable-coverage percentage as objective measure



2 broad classes: Practical and Theoretical



Fully Automatable

Industry Standard Today

- Junit
  - unit testing framework that supports test automation in Java Programming Language, provides the Test coverage report as well; licensed under Eclipse Public License
- Nunit
  - open source unit testing framework that supports all .NET languages
- Fiddler
  - Popular framework for web applications; logs and scrutinizes all HTTP(s) traffic between your system and the Internet.
- Bugzilla
  - popular defect tracking system; records the steps that lead up to reproduce the bug, so developers have all the information they need to fix it.
- Parasoft Jtest
  - Used to test and improve Java codebase on both development and production systems.maintain Junit tests
- Security vulnerabilities - Wireshark (network protocol analyzer), ZAP, Nmap

Industry Standard Today

Code Coverage Measurement in IDEs: IDEs like IntelliJ IDEA, Eclipse, and Visual Studio, are equipped with tools like JaCoCo (for Java) or Coverage.py (for Python) to measure code coverage by identifying which parts of the code have been executed through the test cases.

Coverage Criterion	Prevalence	Common Contexts
Statement Coverage	✅ Very Common	Most general-purpose software, CI tools
Branch Coverage	✅ Common	Quality-conscious teams, logic-heavy modules
Condition/MC/DC	🔶 Niche but Critical	Aerospace, automotive, medical, regulated software
Path Coverage	❌ Rare	Formal verification, academic tools

Industry Standard Today

- Most teams aim for **80%+ statement or branch coverage** in unit tests.
- **High-integrity systems** (e.g., autopilots, pacemakers) legally **require** MC/DC or more.
- **Coverage metrics are useful**, but they **do not guarantee correctness** — they tell you what was tested, not whether the test was good.

Assignment Today

Given code base. Write test cases manually vs using AI. Measure coverage in VS Code and discuss results

Quizizz