DA-IICT

IT 314: Software Engineering

White-Box Test Case Design
Control Flow Analysis

Saurabh Tiwari

White-Box Test Case Design Techniques

Control Flow Testing

- Statement coverage
- Decision (Branch) coverage
- Condition coverage
- Decision-Condition coverage
- Multiple condition coverage
- Basis Path Testing
- Loop testing

Data Flow Testing

- All p-use
- All c-use
- All d-use
- All uses

Overview

- The most common application of graph criteria is to program source
- Graph : Usually the control flow graph (CFG)
- Node coverage : Execute every statement
- Edge coverage : Execute every branch
- Loops : Looping structures such as for loops, while loops, etc.
- Data flow coverage : Augment the CFG
  - defs are statements that assign values to variables
  - uses are statements that use variables

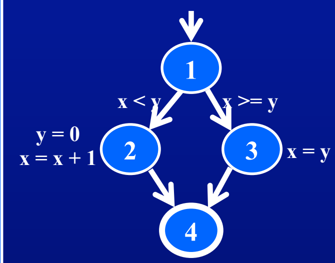
Control Flow Graphs

- A CFG models all executions of a method by describing control structures
- Nodes : Statements or sequences of statements (basic blocks)
- Edges : Transfers of control
- Basic Block : A sequence of statements such that if the first statement is executed, all statements will be (no branches)

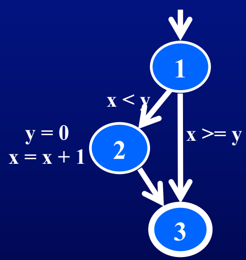
- CFGs are sometimes annotated with extra information
  - branch predicates
  - defs
  - uses
- Rules for translating statements into graphs …

CFG : The if Statement

```
if (x < y)
{
   y = 0;
   x = x + 1;
}
else
{
   x = y;
}
```

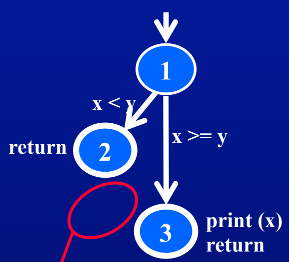



```
if (x < y)
{
   y = 0;
   x = x + 1;
}
```



CFG : The if-Return Statement

```
if (x < y)
{
   return;
}
print (x);
return;
```



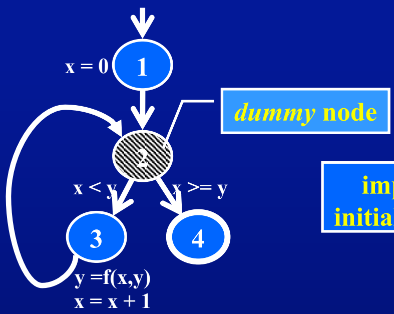
No edge from node 2 to 3.
The return nodes must be distinct.

Loops

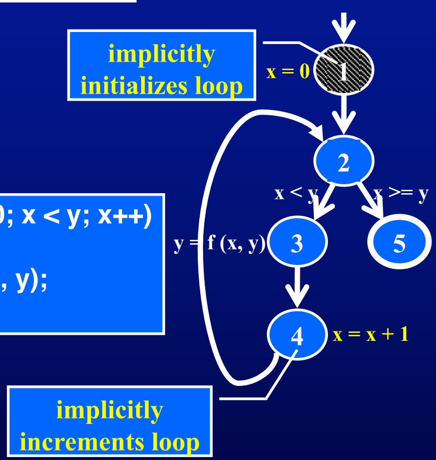
- Loops require “*extra*” nodes to be added
- Nodes that <u>do not</u> represent statements or basic blocks

CFG : while and for Loops

```
x = 0;
while (x < y)
{
   y = f (x, y);
   x = x + 1;
}
```

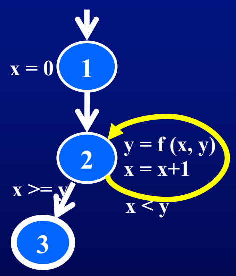



```
for (x = 0; x < y; x++)
{
   y = f (x, y);
}
```

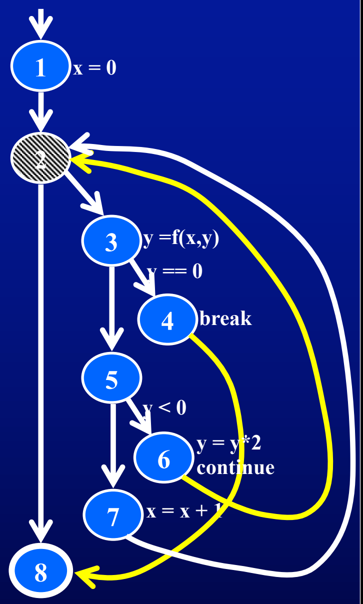


CFG : do Loop, break and continue

```
x = 0;
do
{
   y = f (x, y);
   x = x + 1;
} while (x < y);
println (y)
```

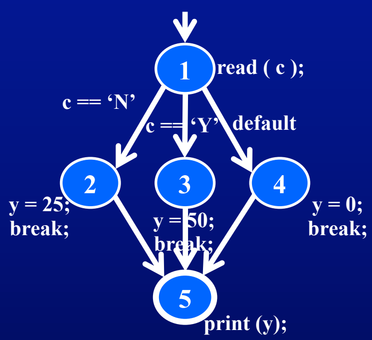



```
x = 0;
while (x < y)
{
   y = f (x, y);
   if (y == 0)
   {
      break;
   } else if y < 0)
   {
      y = y*2;
      continue;
   }
   x = x + 1;
}
print (y);
```



CFG : The case (switch) Structure

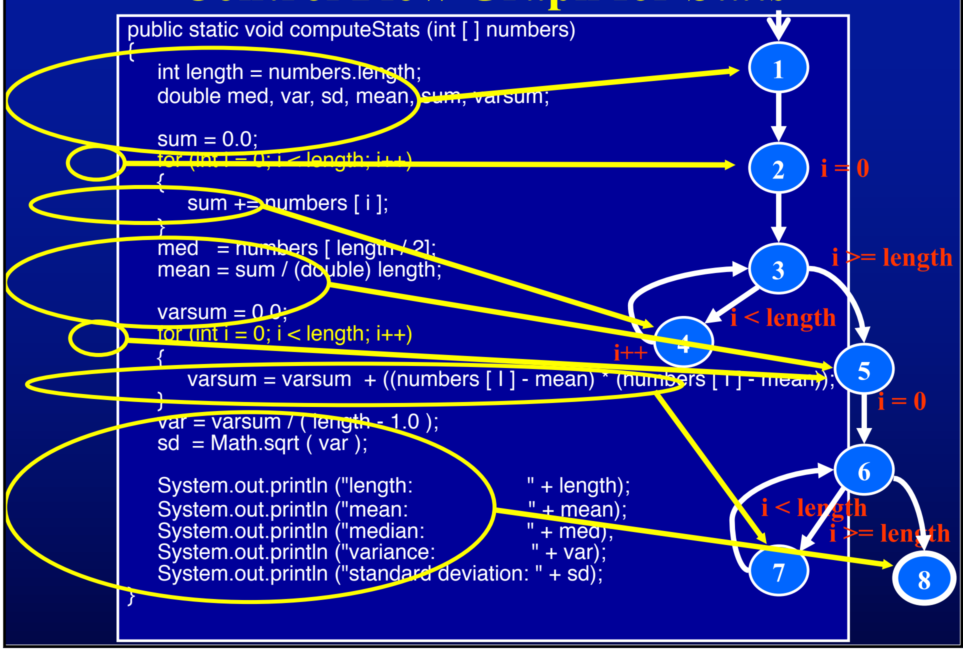
```
read ( c) ;
switch ( c )
{
   case 'N':
      y = 25;
      break;
   case 'Y':
      y = 50;
      break;
   default:
      y = 0;
      break;
}
print (y);
```



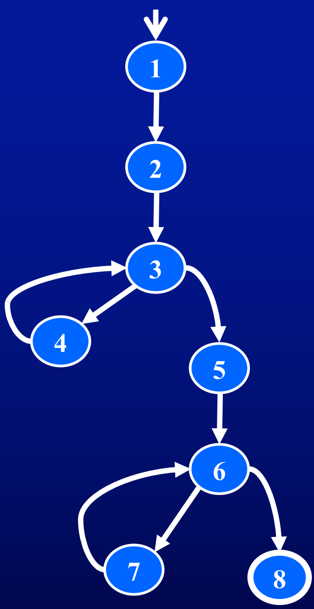
Example Control Flow – Stats

```
public static void computeStats (int [ ] numbers)
{
    int length = numbers.length;
    double med, var, sd, mean, sum, varsum;

    sum = 0.0;
    for (int i = 0; i < length; i++)
    {
        sum += numbers [ i ];
    }
    med   = numbers [ length / 2];
    mean = sum / (double) length;

    varsum = 0.0;
    for (int i = 0; i < length; i++)
    {
        varsum = varsum  + ((numbers [ I ] - mean) * (numbers [ I ] - mean));
    }
    var = varsum / ( length - 1.0 );
    sd  = Math.sqrt ( var );

    System.out.println ("length:                  " + length);
    System.out.println ("mean:                    " + mean);
    System.out.println ("median:                  " + med);
    System.out.println ("variance:                " + var);
    System.out.println ("standard deviation: " + sd);
}
```

Control Flow Graph for Stats

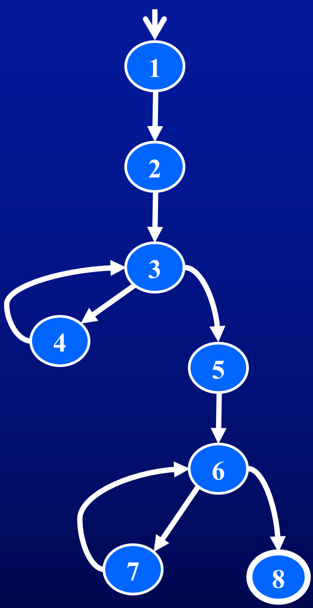


Control Flow TRs and Test Paths – EC



Edge Coverage	
TR	**Test Path**
A. [1, 2] B. [2, 3] C. [3, 4] D. [3, 5] E. [4, 3] F. [5, 6] G. [6, 7] H. [6, 8] I. [7, 6]	[1, 2, 3, 4, 3, 5, 6, 7, 6, 8]

Control Flow TRs and Test Paths – EPC



Edge-Pair Coverage	
TR	**Test Paths**
A. [1, 2, 3] B. [2, 3, 4] C. [2, 3, 5] D. [3, 4, 3] E. [3, 5, 6] F. [4, 3, 5] G. [5, 6, 7] H. [5, 6, 8] I. [6, 7, 6] J. [7, 6, 8] K. [4, 3, 4] L. [7, 6, 7]	i. [1, 2, 3, 4, 3, 5, 6, 7, 6, 8] ii. [1, 2, 3, 5, 6, 8] iii. [1, 2, 3, 4, 3, 4, 3, 5, 6, 7, 6, 7, 6, 8]

TP	TRs toured	*sidetrips*
i	A, B, D, E, F, G, I, J	C, H
ii	A, C, E, H	
iii	A, B, D, E, F, G, I, J, K, L	C, H

White-Box Test Case Design

Statement coverage

write enough test cases to execute every statement at least once

TER (Test Effectiveness Ratio)
TER = Coverage achieved
= statements exercised / total statements

Example

```
void function eval (int A, int B,
int X )
{
if ( A > 1) and ( B = 0 )
then X = X / A;
if ( A = 2 ) or ( X > 1)
then X = X + 1;
}
```

Statement coverage test cases:
1) A = 2, B = 0, X = ? (X can be assigned any value)

White-Box Test Case Design

Decision coverage (Branch coverage)

write test cases to exercise the true and false outcomes of every decision

TER = branches exercised / total branches

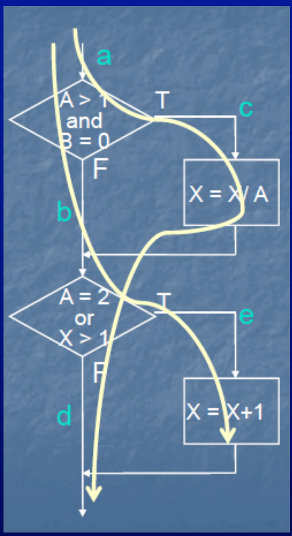
Condition coverage (Predicate coverage)

write test cases such that each condition in a decision takes on all possible outcomes at least once

may not always satisfy decision coverage

Example

void function eval (int A, int B,
int X)
{
if (A > 1) and (B = 0)
then X = X / A;
if (A = 2) or (X > 1)
then X = X + 1;
}



Decision coverage test cases:

1) **A = 3 B = 0 X = 1 (acd)**
2) **A = 2, B = 1, X = ? (abe)**

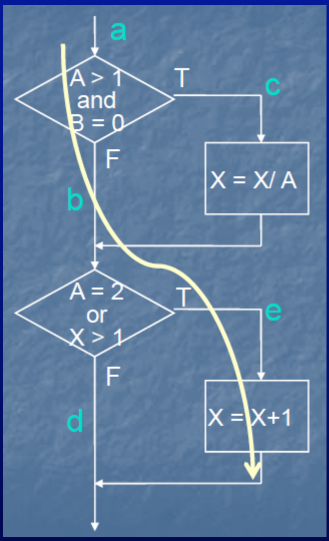
Example

Condition coverage test cases must cover conditions

A > 1, A <=1, B = 0 , B!=0
A = 2, A!=2, X > 1, X <=1

Test Cases:
1) A = 1, B = 0, X = 3 (abe)
2) A =2, B = 1,   X = 1 (abe)

Doesn't satisfy decision coverage



White-Box Test Case Design

Decision Condition coverage

write test cases such that each condition in a decision takes on all possible outcomes at least once and each decision takes on all possible outcomes at least once

Multiple Condition coverage (Full Predicate)

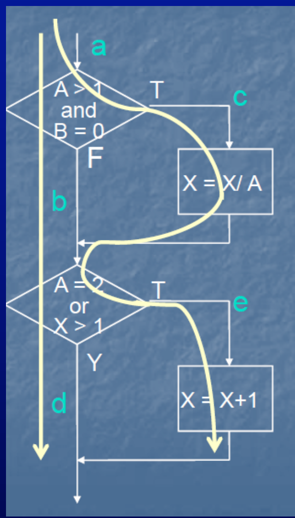
write test cases to exercise all *possible combinations* of True and False outcomes of conditions within a decision

Example

Decision Condition coverage test cases must cover conditions

A > 1, A <=1, B = 0 , B!=0
A = 2, A!=2, X > 1, X <=1
And
also (A > 1 and B = 0) T, F
      (A = 2 or X > 1) T, F

Test Cases:
1) A = 2, B = 0, X = 4 (ace)
2) A =1, B = 1,   X = 1 (abd)

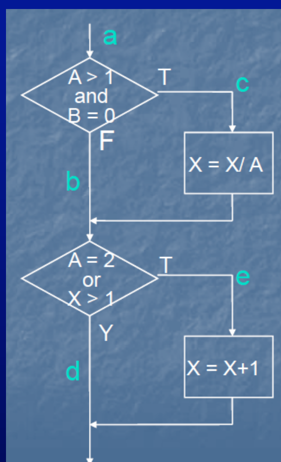


Example

Multiple Condition coverage must cover conditions

1) A >1, B =0
2) A >1, B !=0
3) A<=1, B=0
4) A <=1, B!=0
5) A=2, X>1
6) A=2, X <=1
7) A!=2, X > 1
8) A !=2, X<=1

Test Cases:
1) A =2, B = 0, X = 4 (covers 1,5)
2) A =2, B = 1, X = 1 (covers 2,6)
3) A =1, B = 0, X = 2 (covers 3,7)
4) A =1, B = 1, X = 1 (covers 4,8)



Basis Path Testing

1. Draw control flow graph of program from the program detailed design or code.

2. Compute the Cyclomatic complexity V(G) of the flow graph using any of the formulas:

V(G) = #Edges - #Nodes + 2
or V(G) = #regions in flow graph
or V(G) = #predicates + 1

Basis Path Testing (cont…)

3. Determine a basis set of linearly independent paths.

4. Prepare test cases that will force execution of each path in the Basis set.

The value of Cyclomatic complexity provides an upper bound on the number of tests that must be designed to guarantee coverage of all program statements.

Determining Metrics

- Quality characteristics can be measured with metrics
- The intention is to gain a quantitative measure of software whose nature is abstract

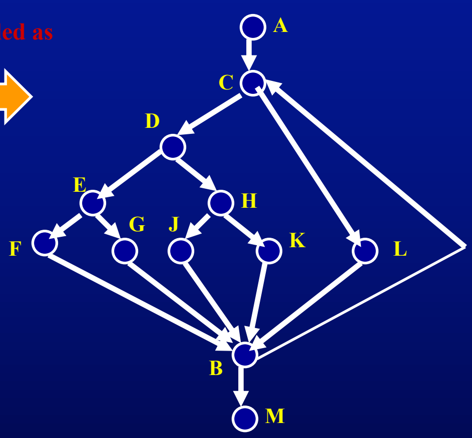
- Example:
  - McCabe’s metric or cyclomatic complexity, V
  - Measures the structural complexity of program code
  - Based on CFG
  - $V(G) = e - n + 2$
    where V(G) is Cyclomatic number of graph G
    e = number of edges in G
    n = number of nodes in G

25

Determining Metrics

```
A
DO
        IF C THEN
          IF D THEN
            IF E THEN F
            ELSE G
          ELSE IF H THEN J
                  ELSE K
        ELSE L
WHILE B
M
```

Modeled as



26

Determining Metrics

Example: for CFG in previous slide

V(G) = e - n + 2 = 16 – 12 + 2 = 6

V(G) higher than 10 can not be tolerated and rework of the source code has to take place

- V(G) can be used to estimate the testability and maintainability
- V(G) specifies the number of linearly independent paths in the program
-

Summary

- Applying the graph test criteria to control flow graphs is relatively straightforward
  - Most of the developmental research work was done with CFGs
- A few subtle decisions must be made to translate control structures into the graph

Questions?

- What is the relationship between Statement and Branch Coverage?

  Possible relationships:

1. None.
2. Statement Coverage subsumes Branch Coverage ("statement => branch").
3. Branch Coverage subsumes Statement Coverage ("branch => statement").

Questions?

- In general, how many different combinations of condition values must be considered when a branch predicate has N conditions?

In General…

Number of program Paths	≥	Number of Basis Paths	≥	Number of test cases required for branch coverage
Path Coverage	=>	Basis Paths Coverage	=>	Branch Coverage

Exercise

1. Prove that Path and Compound Condition Coverage are independent.

   (Hint: consider the proof that Branch and Condition Coverage are independent.)

2. Prove that Branch Testing guarantees statement coverage

3. Condition Testing: Stronger testing than branch testing

4. Which code coverage criteria is strongest among testing strategies? Why?

Questions?

Next Lecture….
Data Flow Analysis, Model-based Testing...