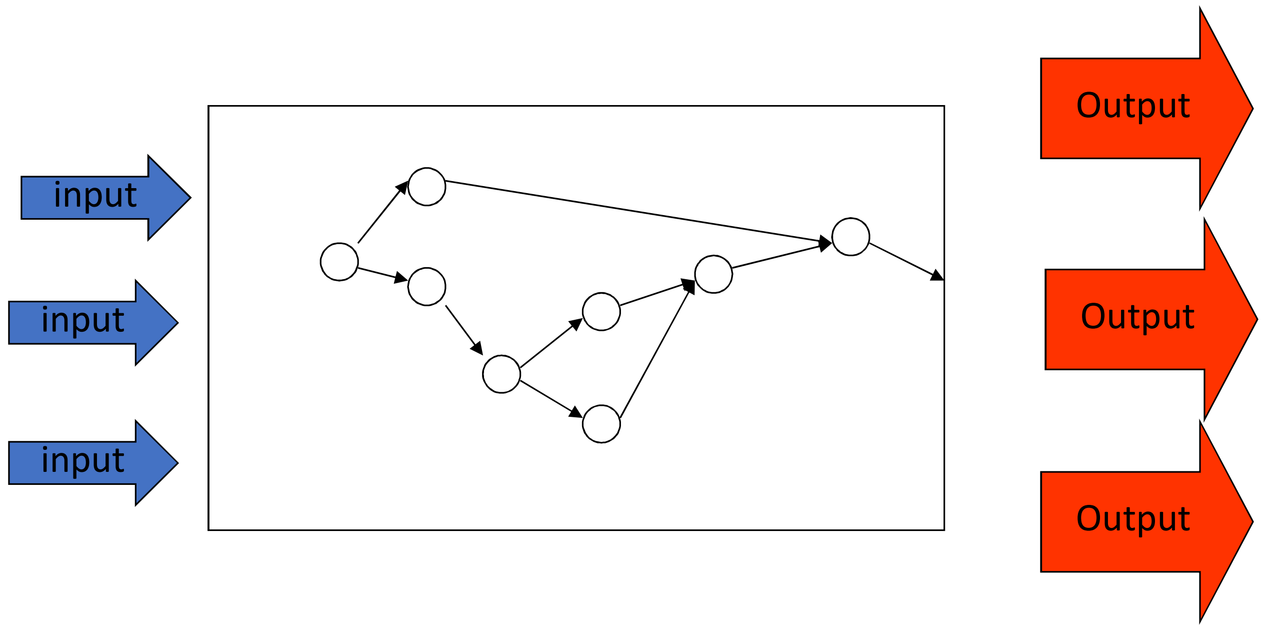
Test Theory
White Box Testing

SE323 Software Construction Testing and Maintenance

What is White Box?

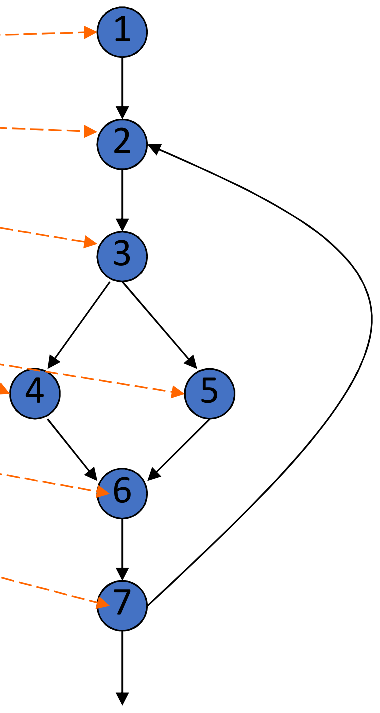
- Opposite of Black box testing
- We see what is inside

Code-based Testing: White Box Testing



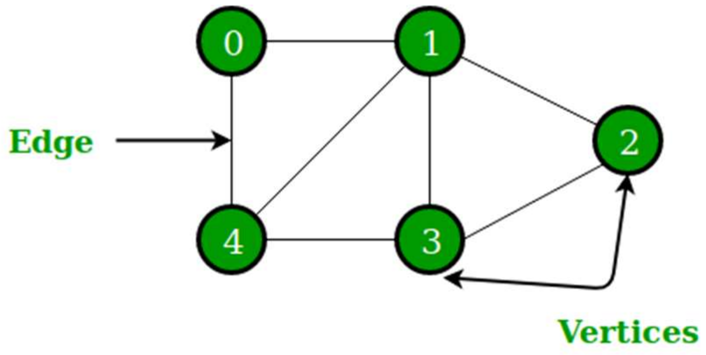
Control Flow Graph

```
i = 0
For j = 0 to 10 do
If i mod 2 = 0 then
       print("hello")
Else
       print("world")
Endif
endfor
```



Graph Data structure

- A Graph is a non-linear data structure consisting of nodes and edges. The nodes are sometimes also referred to as vertices and the edges are lines or arcs that connect any two nodes in the graph. More formally a Graph can be defined as,
- *A Graph consists of a finite set of vertices(or nodes) and set of Edges which connect a pair of nodes.*

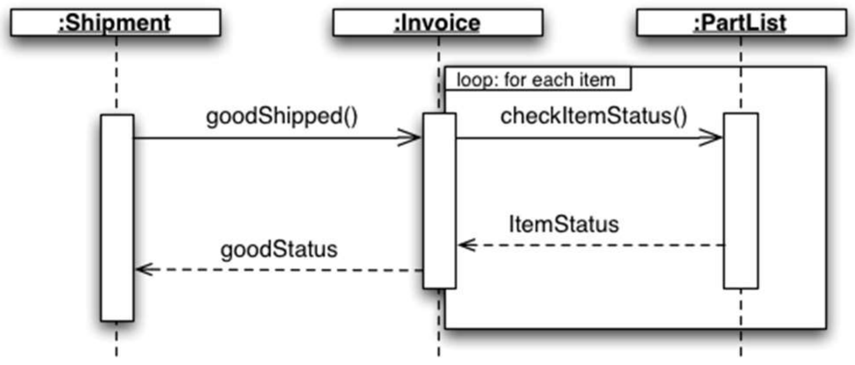


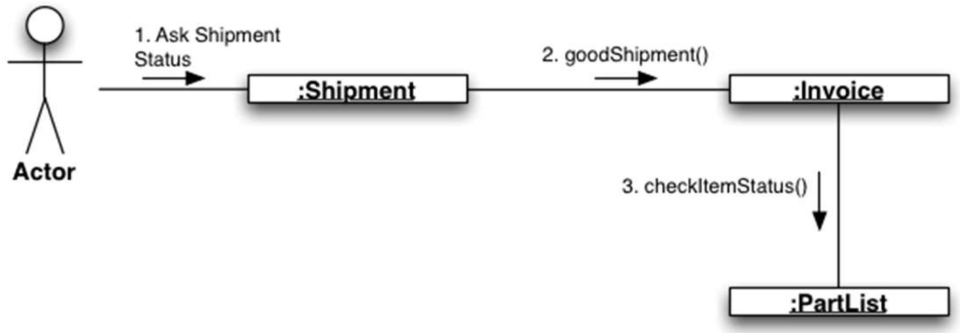
Control Flow graph

- Define each vertice(or node) or node as the statement
- The edge is the control flow which lead from one statement to other statement

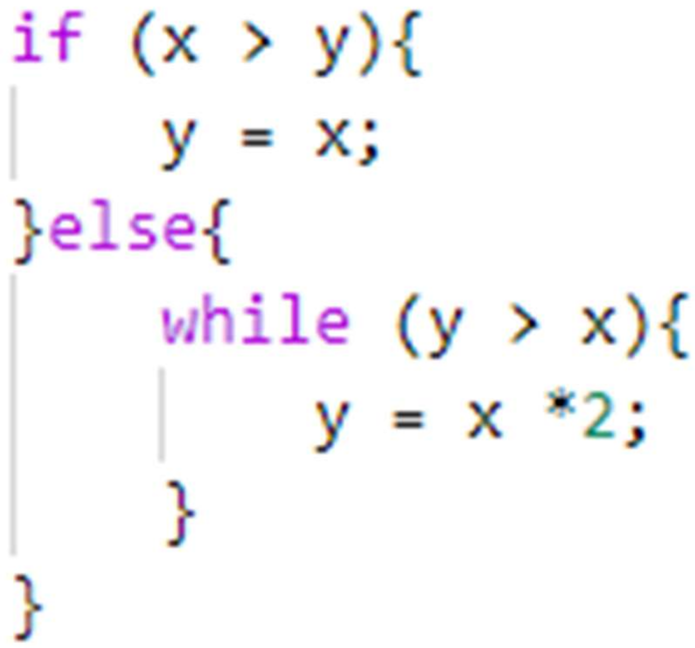
In the integration test

- The sequence diagram or collaboration diagram can be shown as control flow graph
- Each node represent the method the be called instead of statements





Draw the control flow of



```
if (x > y){
    y = x;
}else{
    while (y > x){
        y = x *2;
    }
}
```

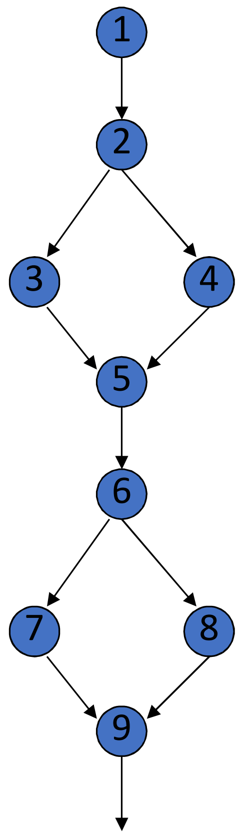
Quality of the test

- The criteria for Blackbox testing
  - Try to cover all possible cases
- The white box testing
  - Try to cover all possible graphs
  - What are the possible cover graph?

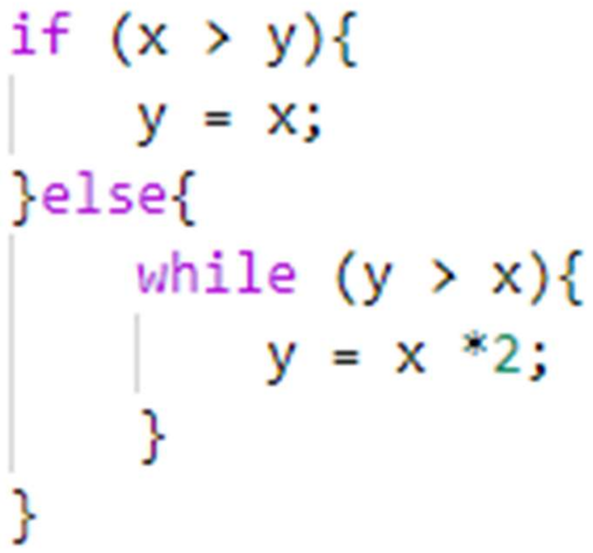
Statement coverage

- Every statement should be executed

1-2-4-5-6-7-9-10
1-2-3-5-6-8-9-10



Find the statement coverage



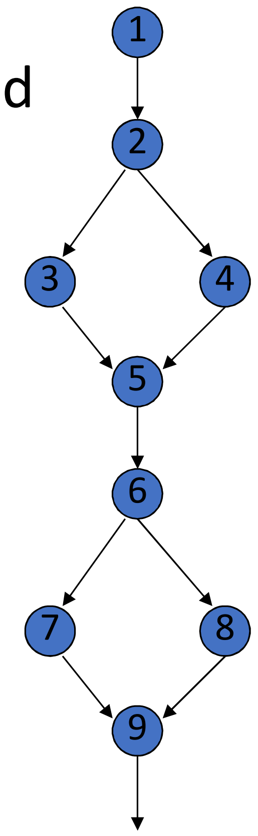
```
if (x > y){
    y = x;
}else{
    while (y > x){
        y = x *2;
    }
}
```

Statement coverage = ?

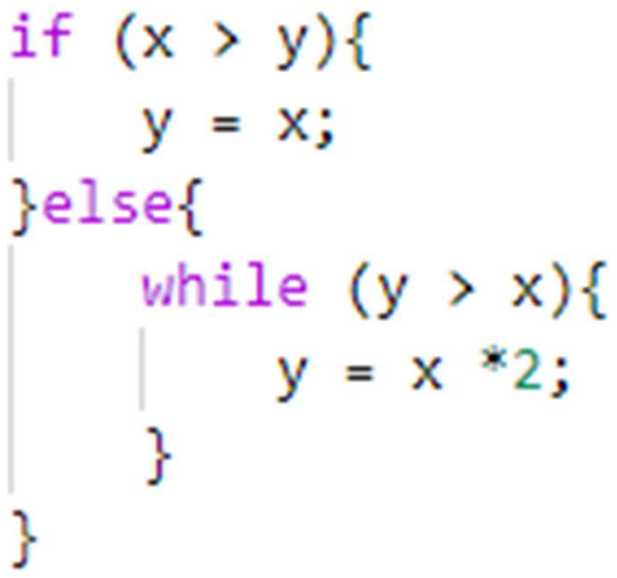
Condition Coverage

- Every branch of condition should be executed

1-2-3-5-6-7-9-10
1-2-4-5-6-8-9-10



Condition Coverage for the given code ?

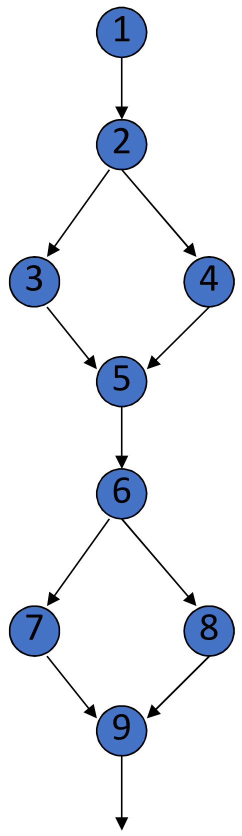


```
if (x > y){
    y = x;
}else{
    while (y > x){
        y = x *2;
    }
}
```

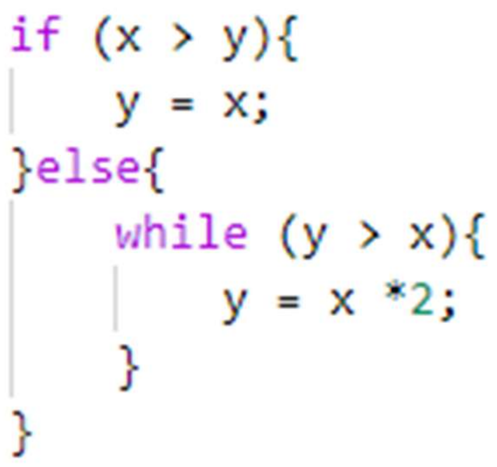
Path Coverage

- All possible should be coverage

1-2-3-5-6-7-9-10
1-2-4-5-6-7-9-10
1-2-3-5-6-8-9-10
1-2-4-5-6-8-9-10



Path Coverage for the code ?



```
if (x > y){
    y = x;
}else{
    while (y > x){
        y = x *2;
    }
}
```

State Explosion problem

- In your loop,
  - How many possible paths?
- State explosion
  - The loop cause infinite path
  - As the different loop provide different path
- The tester should define how many loop to use
  - Fix the number of loop to be tested

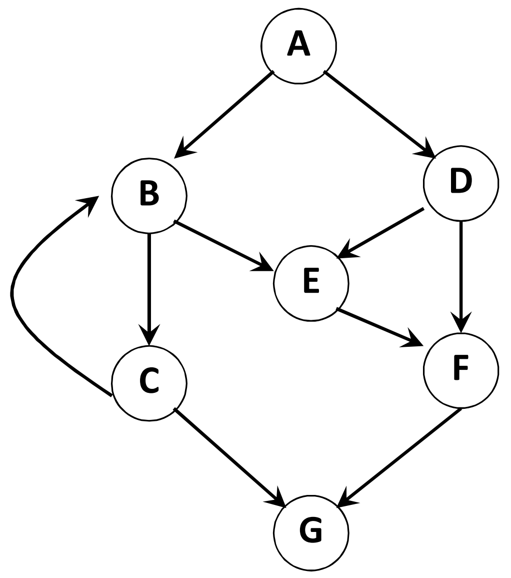
In the complex control flow chart

- How do we know that how many path we need in each coverage criteria?
- The McCabe Cyclomatic complexity

McCabe's Cyclomatic Complexity

- Measure a number of linear independent path through a program
  - V(G) = e – n + 2
    - e => number of edges
    - N => number of Nodes
  - V(G) = number of closed loop + 1
    - Or number of node that contain condition + 1
- Lower bound of number of possible paths
- Upper bound of number of test case that necessary to achieve a complete branch coverage

Cyclomatic Complexity



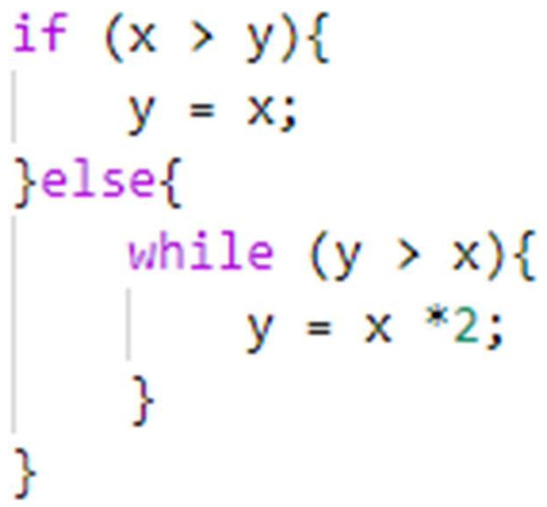
<u>Path Coverage</u>

P1: A-B-C-G
P2: A-B-C-B-C-G
P3: A-B-E-F-G
P4: A-D-E-F-G
P5: A-D-F-G

<u>Condition Coverage</u>

P1: A-B-C-G
P2: A-B-C-B-E-F-G
P3: A-D-E-F-G
P4: A-D-F-G

What about the given code?



```
if (x > y){
    y = x;
}else{
    while (y > x){
        y = x *2;
    }
}
```

To do the test

- Find the test data that drive the path
- Try with your case

Homework

```
If (a < b+c) and (b < a+c) and (c < a+c)
   then IsATriangle = true
   else IsATriangle = false
If IsATriangle
   Then If (a=b) and (b=c)
            Then output (“Equilateral”)
            Else If (a <> b) and (a<>c) and (b<>c)
             Then output(“Scalene”)
              Else output(“Isolene”)
                    End If
            End If
   Else output(“not A triangle”)
End If
```

- Create the control flow graph and find the cyclometric complexity value

Q&A