Introduction to Software Testing
(2nd edition)
Chapter 7.1, 7.2

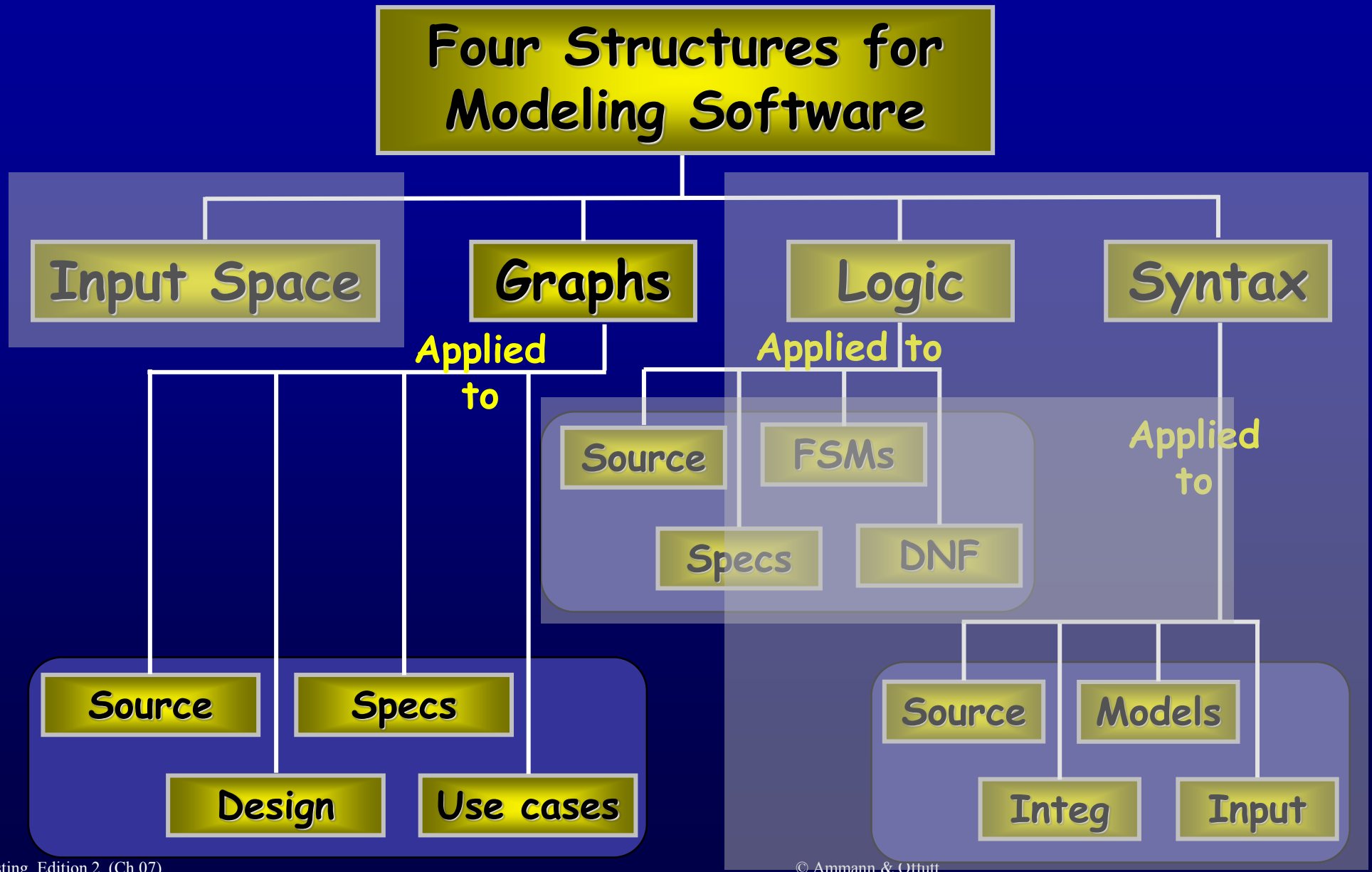
Overview Graph Coverage Criteria
(*active class version*)

Paul Ammann & Jeff Offutt

http://www.cs.gmu.edu/~offutt/softwaretest/

Update, October 2016

Ch. 7 : Graph Coverage



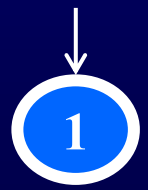
Covering Graphs (7.1)

- Graphs are the most commonly used structure for testing
- Graphs can come from many sources
  - Control flow graphs
  - Design structure
  - FSMs and statecharts
  - Use cases
- Tests usually are intended to "cover" the graph in some way

Definition of a Graph

- A set N of nodes, N is not empty
- A set N_0 of initial nodes, N_0 is not empty
- A set N_f of final nodes, N_f is not empty
- A set E of edges, each edge from one node to another
  - (n_i , n_j), i is predecessor, j is successor

Is this a graph?



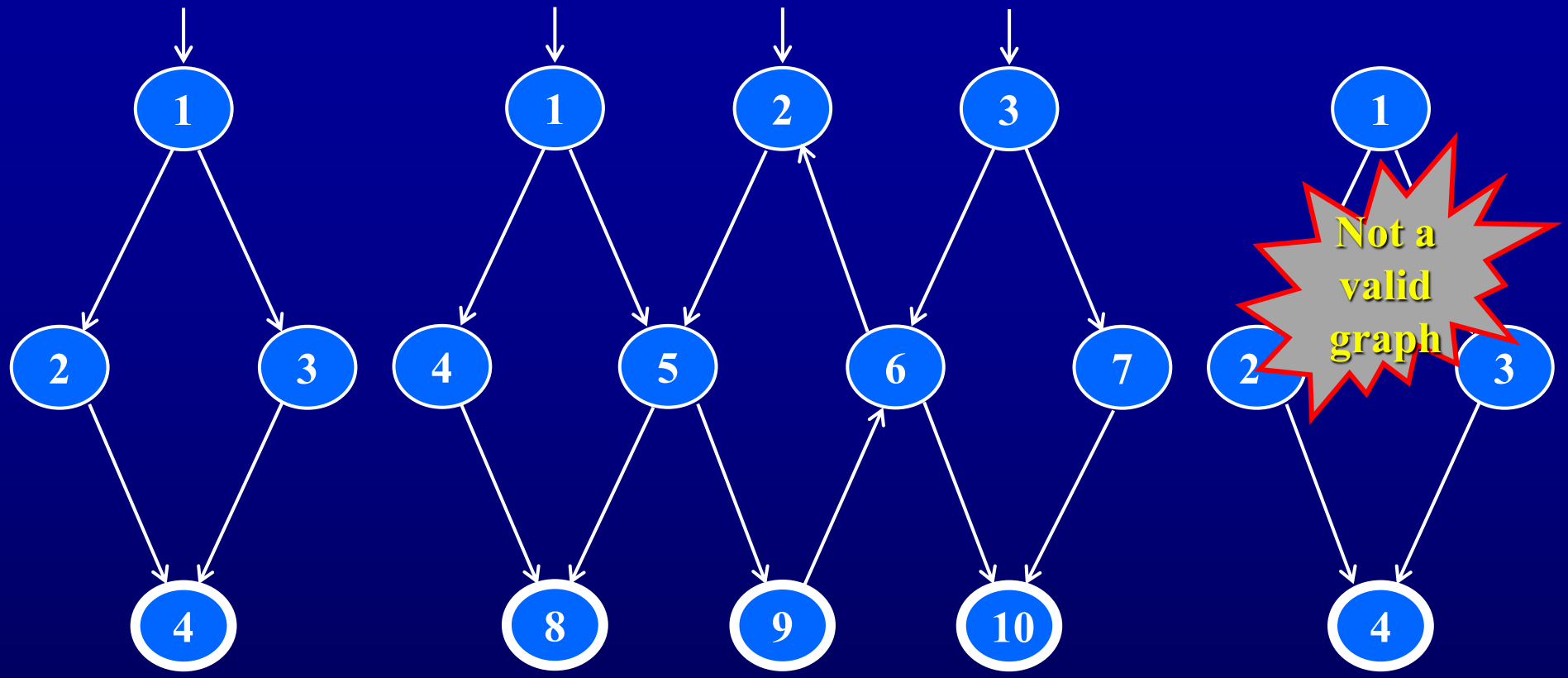
$N_0 = \{ 1 \}$

$N_f = \{ 1 \}$

$E = \{ \}$

Yes

Example Graphs



$N_0 = \{ 1 \}$

$N_f = \{ 4 \}$

$E = \{ (1, 2), (1,3), (2,4), (3,4) \}$

$N_0 = \{ 1, 2, 3 \}$

$N_f = \{ 8, 9, 10 \}$

$E = \{ (1,4), (1,5), (2,5), (3,6), (3, 7), (4, 8), (5,8), (5,9), (6,2), (6,10), (7,10) (9,6) \}$

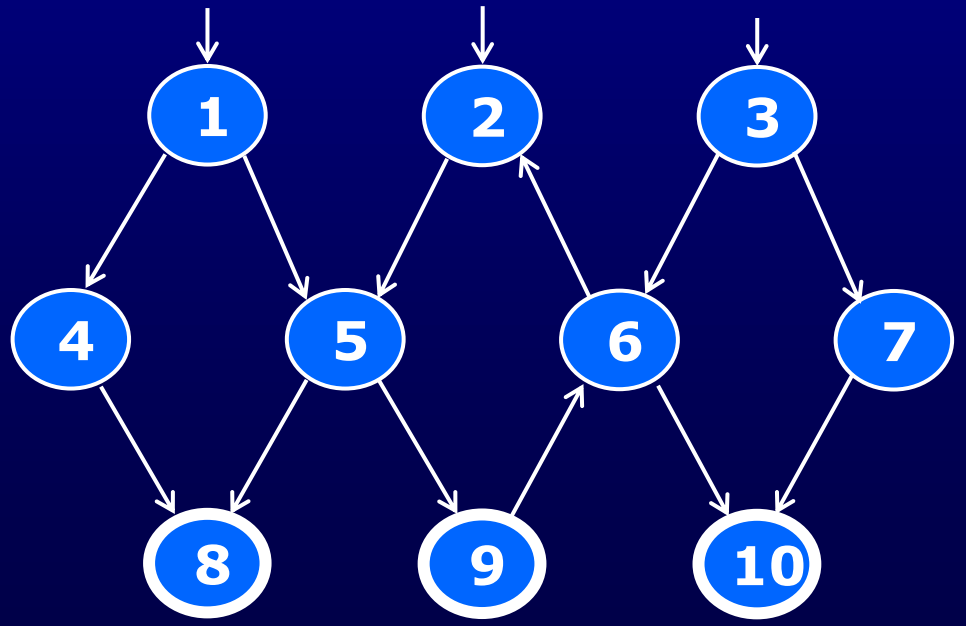
$N_0 = \{ \}$

$N_f = \{ 4 \}$

$E = \{ (1,2), (1,3), (2,4), (3,4) \}$

Paths in Graphs

- Path : A sequence of nodes – $[n_1, n_2, \ldots, n_M]$
  - Each pair of nodes is an edge
- Length : The number of edges
  - A single node is a path of length 0
- Subpath : A subsequence of nodes in p is a subpath of p



A Few Paths

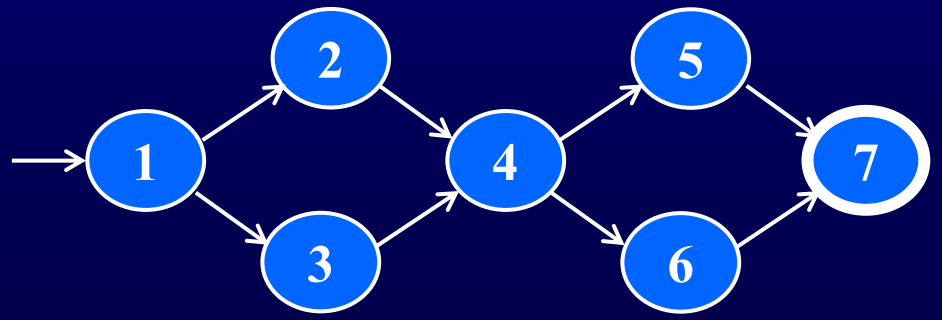
[1, 4, 8]

[2, 5, 9, 6, 2]

[3, 7, 10]

Test Paths and SESEs

- Test Path : A path that starts at an initial node and ends at a final node
- Test paths represent execution of test cases
  - Some test paths can be executed by many tests
  - Some test paths cannot be executed by any tests
- SESE graphs : All test paths start at a single node and end at another node
  - Single-entry, single-exit
  - N0 and Nf have exactly one node



Double-diamond graph

Four test paths

[1, 2, 4, 5, 7]
[1, 2, 4, 6, 7]
[1, 3, 4, 5, 7]
[1, 3, 4, 6, 7]

Visiting and Touring

- Visit : A test path *p* *visits* node *n* if *n* is in *p*

  A test path *p* *visits* edge e if e is in *p*
- Tour : A test path *p* *tours* subpath *q* if *q* is a subpath of *p*

Test path [1, 2, 4, 5, 7]

Visits nodes ? 1, 2, 4, 5, 7

Visits edges ? (1,2), (2,4), (4, 5), (5, 7)

Tours subpaths ? [1,2,4], [2,4,5], [4,5,7], [1,2,4,5], [2,4,5,7], [1,2,4,5,7]

(Also, each edge is technically a subpath)

Tests and Test Paths

- path (*t*) : The test path executed by test *t*
- path (*T*) : The set of test paths executed by the set of tests *T*
- Each test executes one and only one test path
  - Complete execution from a start node to an final node
- A location in a graph (node or edge) can be reached from another location if there is a sequence of edges from the first location to the second
  - *Syntactic reach* : A subpath exists in the graph
  - *Semantic reach* : A test exists that can execute that subpath
  - This distinction becomes important in section 7.3

```
if (x >7 and y> 5)
      {
            if (x <0)
                  print “Hi there”;
            else
                  print “Bye there”;
      }
```

Testing and Covering Graphs (7.2)

- We use graphs in testing as follows :
  - Develop a model of the software as a graph
  - Require tests to visit or tour specific sets of nodes, edges or subpaths
- Test Requirements (TR) : Describe properties of test paths
- Test Criterion : Rules that define test requirements
- Satisfaction : *Given a set TR of test requirements for a criterion C, a set of tests T satisfies C on a graph if and only if for every test requirement in TR, there is a test path in path(T) that meets the test requirement tr*
- Structural Coverage Criteria : Defined on a graph just in terms of nodes and edges
- Data Flow Coverage Criteria : Requires a graph to be annotated with references to variables

Node and Edge Coverage

- The first (and simplest) two criteria require that each node and edge in a graph be executed

Node Coverage (NC) : Test set *T* satisfies node coverage on graph *G* iff for every syntactically reachable node *n* in *N*, there is some path *p* in *path(T)* such that *p* visits *n*.

- This statement is a bit cumbersome, so we abbreviate it in terms of the set of test requirements

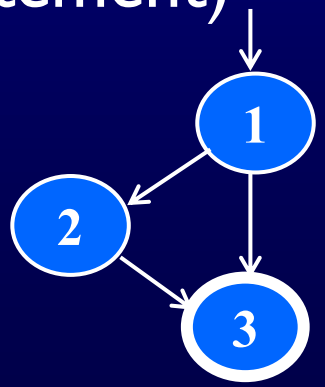
Node Coverage (NC) : TR contains each reachable node in G.

Node and Edge Coverage

- Edge coverage is slightly stronger than node coverage

Edge Coverage (EC) : TR contains each reachable path of length up to 1, inclusive, in G.

- The phrase "*length up to 1*" allows for graphs with one node and no edges
- NC and EC are only different when there is an edge and another subpath between a pair of nodes (as in an "if-else" statement)



Node Coverage : ? TR = { 1, 2, 3 }
Test Path = [1, 2, 3]

Edge Coverage : ? TR = { (1, 2), (1, 3), (2, 3) }
Test Paths = [1, 2, 3]
[1, 3]

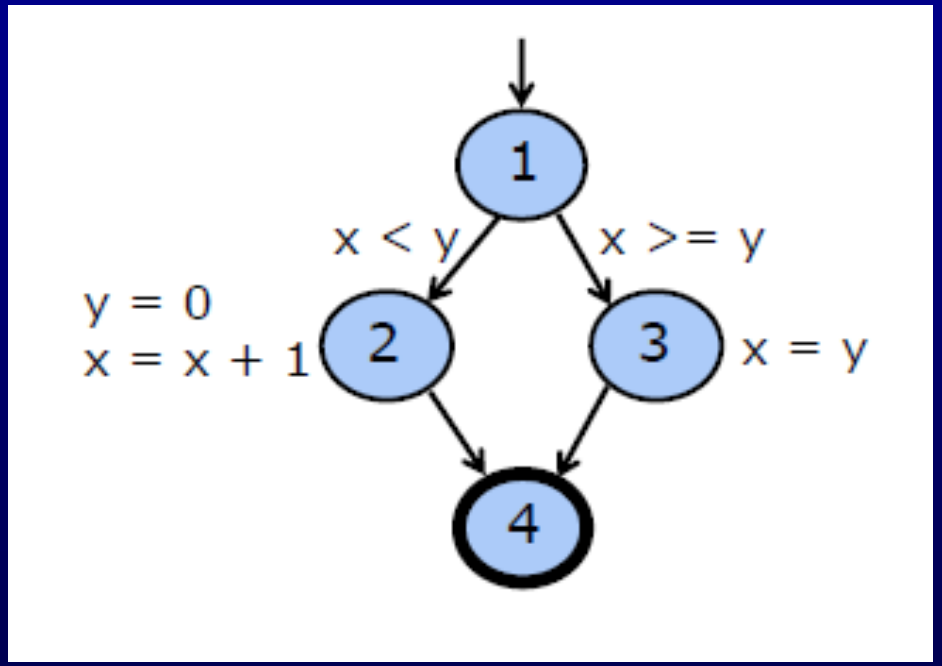
Example 1

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

- Basic blocks (nodes)
  - 1: if (x < y) → entry node
  - 2: y=0; x = x+1;
  - 3: x = y;
  - 4: implicit *return* after else block→ exit node

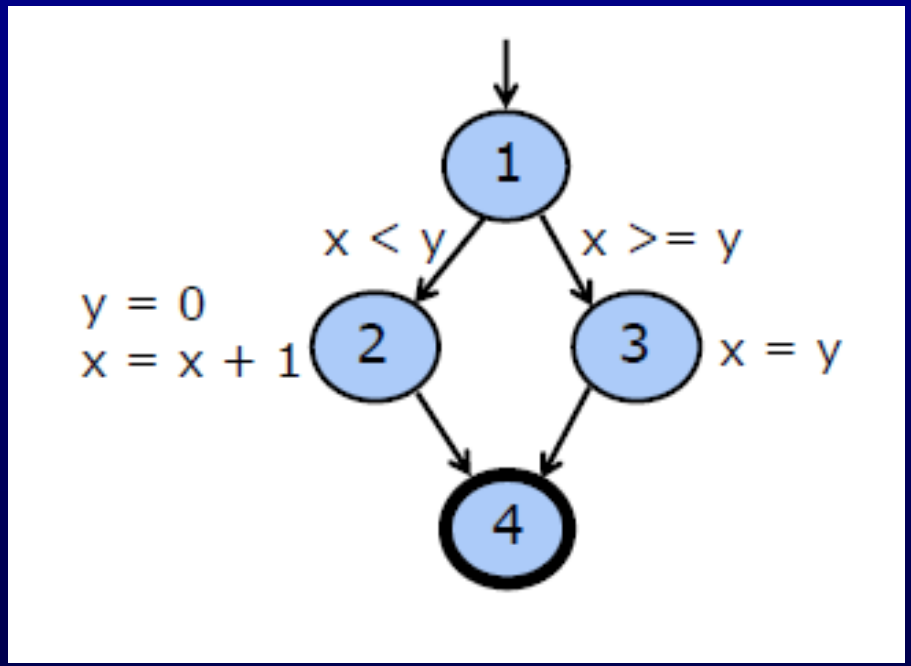
```
if (x < y)
{
     y = 0;
     x = x+1;
}
else
{
     x = y;
}
```

- Contrl Flows(edges)
  - 1 → 2
  - 1 → 3
  - 2 → 4
  - 3 → 4



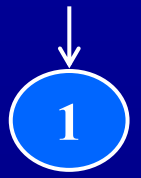
```
if (x < y)
{
    y = 0;
    x = x+1;
}
else
{
    x = y;
}
```

- Applying Node Coverage (NC)
  - Test Requirement: {1,2,3,4}
  - Test Paths:
    - T1 → [1,2,4]
    - T2 → [1,3,4]
  - Test case values:
    - T1 → x=1 , y=2
    - T2 → x=3 , y=2

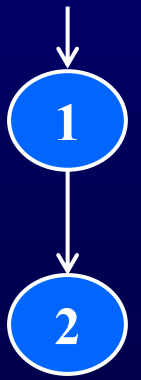


Paths of Length 1 and 0

- A graph with only one node will not have any edges



- It may seem trivial, but formally, Edge Coverage needs to require Node Coverage on this graph
- Otherwise, Edge Coverage will not subsume Node Coverage
  - So we define "length up to 1" instead of simply "length 1"
- We have the same issue with graphs that only have one edge – for Edge-Pair Coverage …

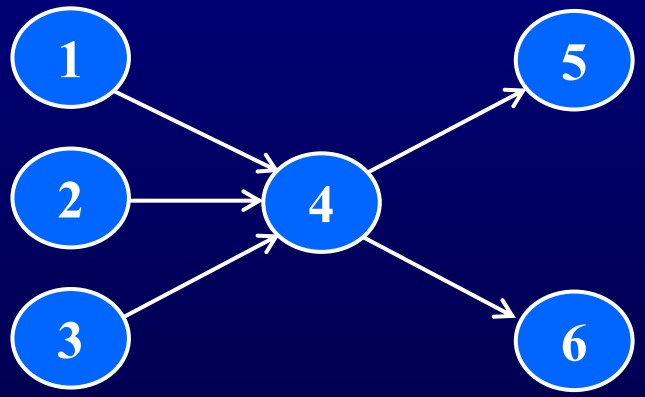


Covering Multiple Edges

- Edge-pair coverage requires pairs of edges, or subpaths of length 2

Edge-Pair Coverage (EPC) : TR contains each reachable path of length up to 2, inclusive, in G.

- The phrase "length up to 2" is used to include graphs that have less than 2 edges



Edge-Pair Coverage : ?

TR = { [1,4,5], [1,4,6], [2,4,5], [2,4,6], [3,4,5], [3,4,6] }

- The logical extension is to require all paths …