Graph Theory

Lecture Objectives:

1) Define the term graph, node, initial node, final node, and edge.
2) Define reachability in a graph.
3) Define the term test path.
4) Explain the concept of a SESE graph.
5) Relate the concept of a SESE graph to good programming concepts.
6) Define the concept of visiting a node and explain the concept of a graph tour.
7) Given a graph, construct a set of test paths through the graph
8) Construct a graph from a segment of source code.

Graph.

Lines and

Nodes, Edges

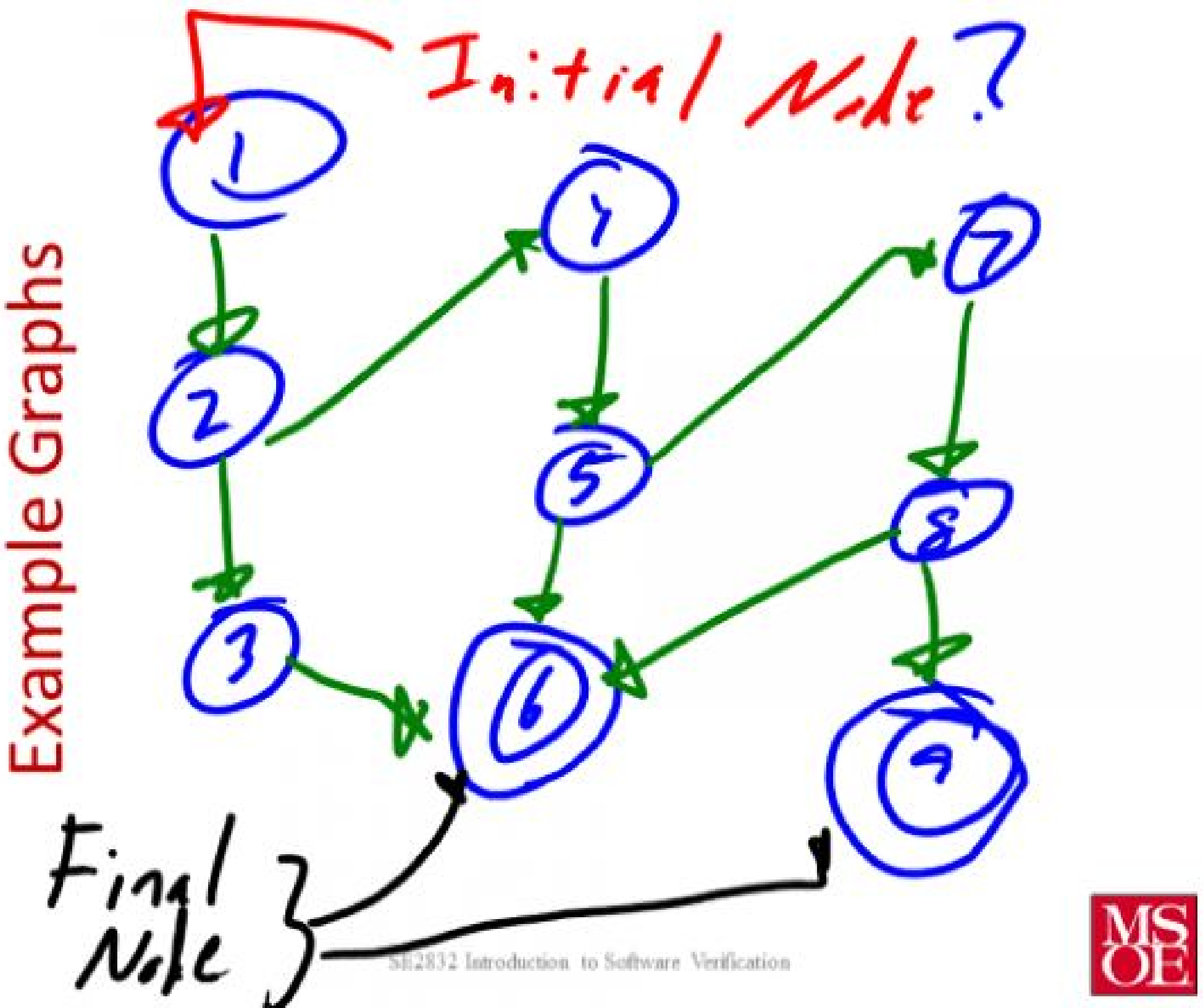
&

vertices...

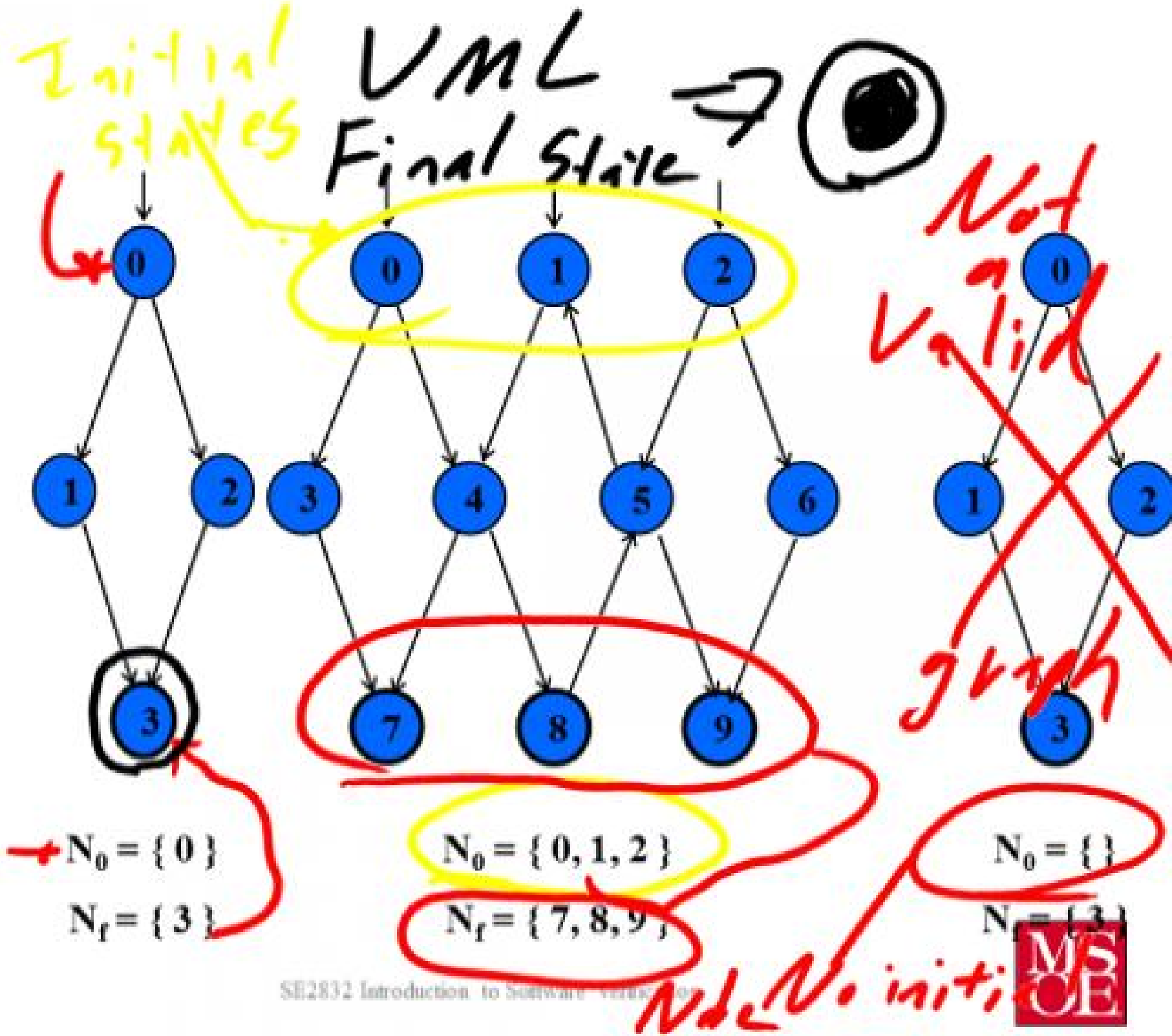
Graph Definitions

- 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

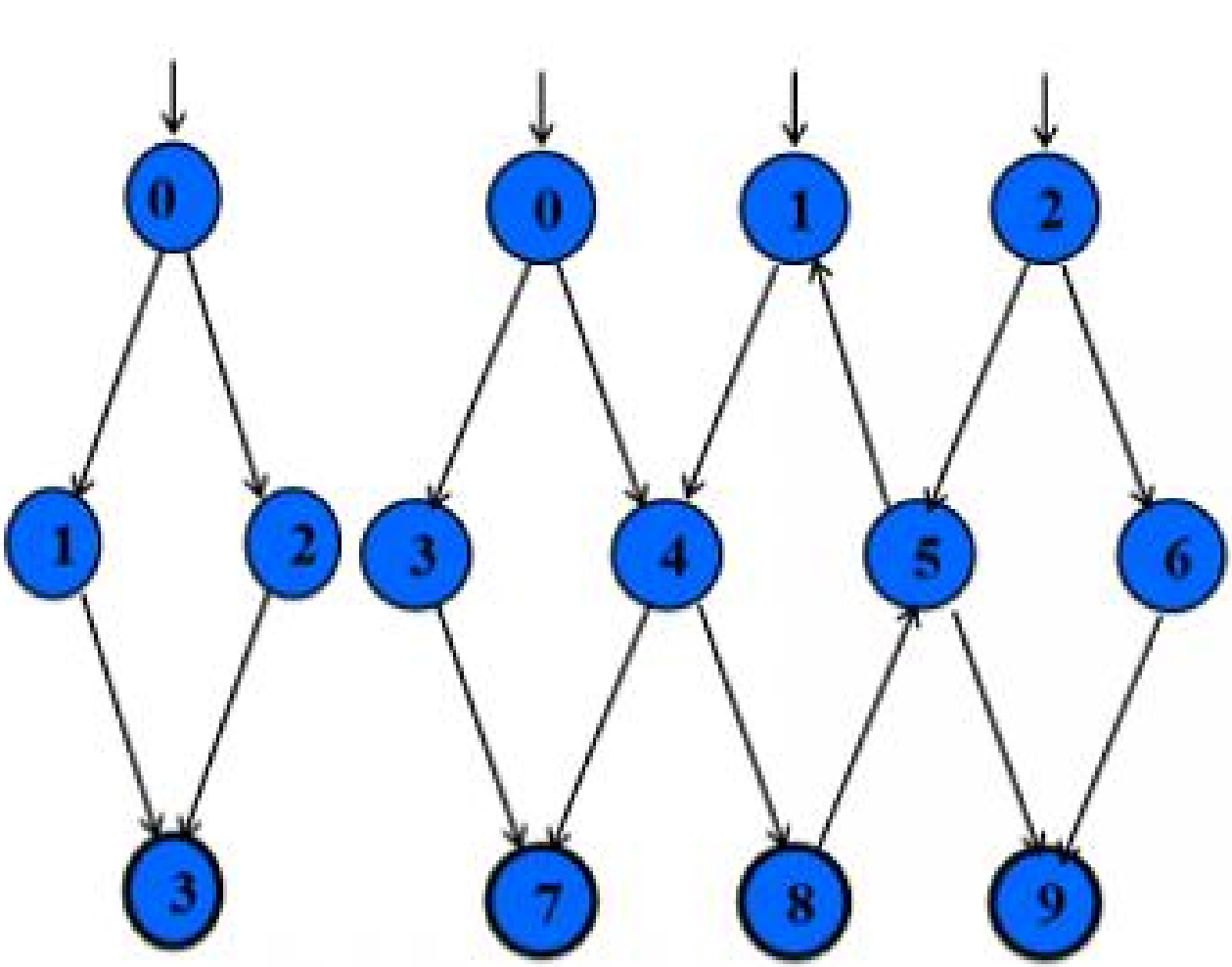
Example Graphs



Three Example Graphs



Three Example Graphs



$N_0 = \{ 0 \}$
$N_f = \{ 3 \}$

$N_0 = \{ 0, 1, 2 \}$
$N_f = \{ 7, 8, 9 \}$

$N_0 =$ 数字
$N_f = \{ 3 \}$

Paths in Graphs

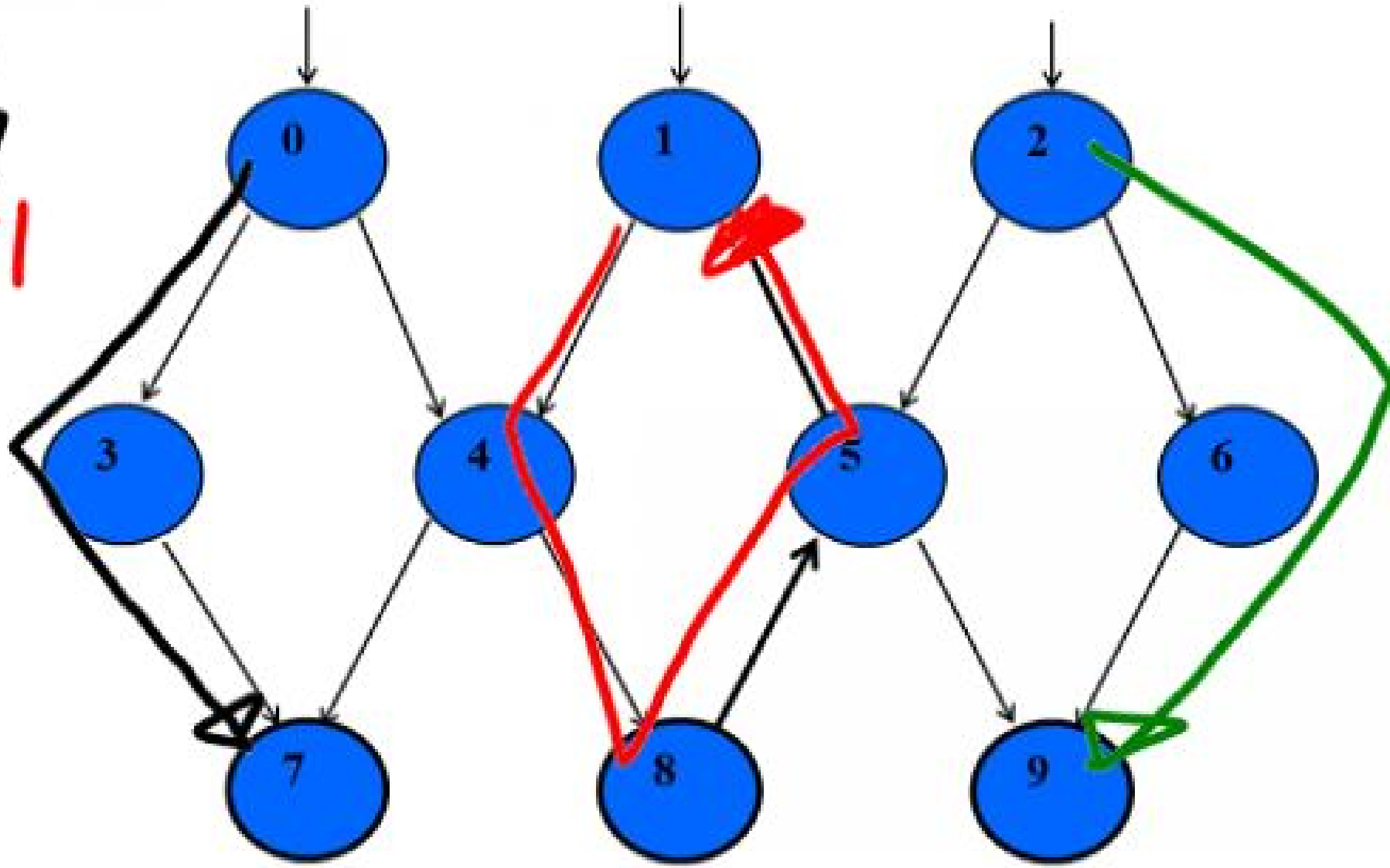
- Path : A sequence of nodes – [n_1, n_2, …, n_M]
  - Each pair of nodes is an edge
- Length : The number of edges ⇒ How long is a path?
  - A single node is a path of length 0
- Subpath : A subsequence of nodes in *p* is a subpath of *p* ⇒ Part of a path
- Reach (*n*) : Subgraph that can be reached from *n*
- *Reachable*
  - *Indicates that there exists a path from node I to node j.*

Path

0 3 7

1 4 8 5 1

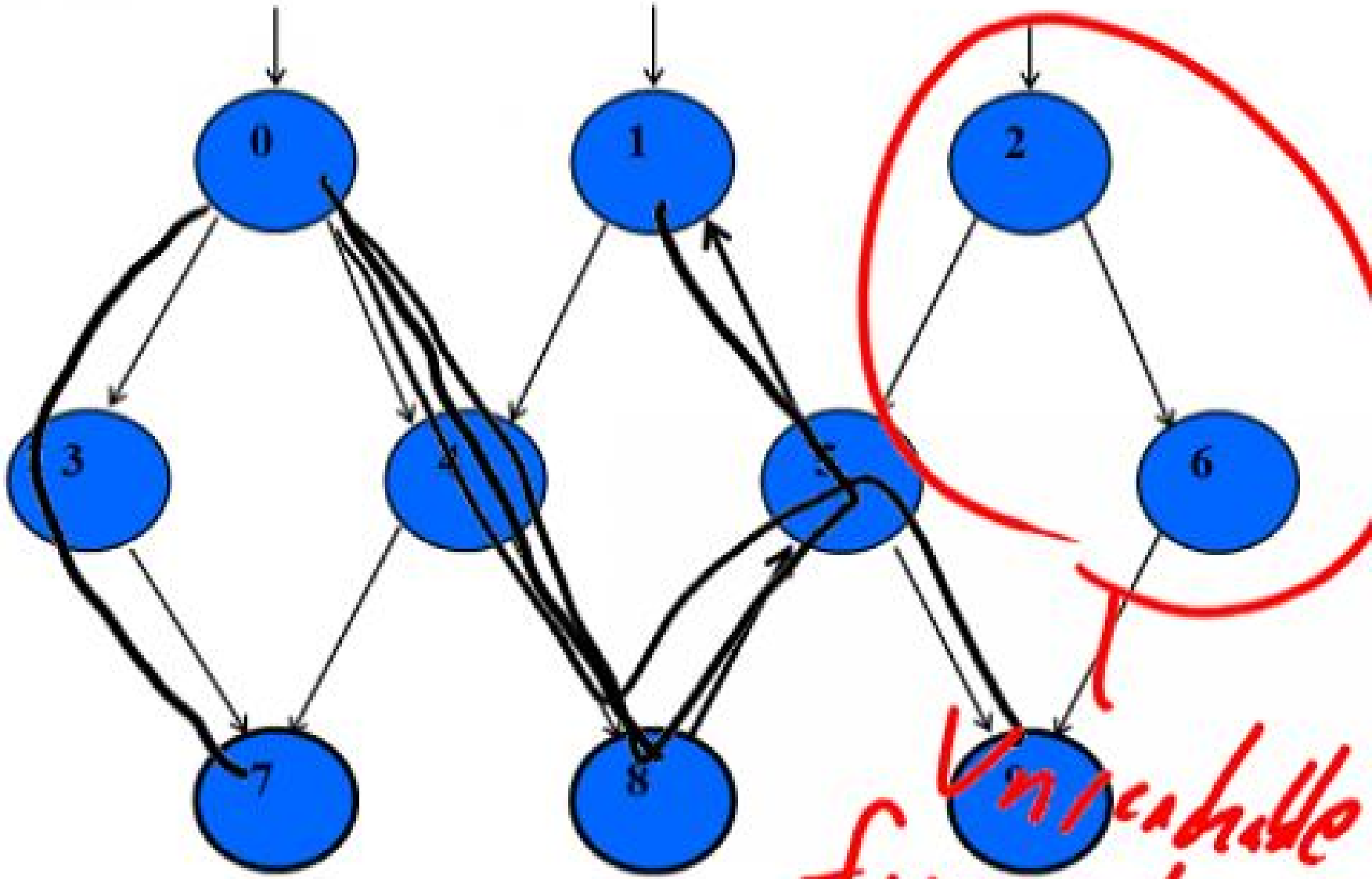
2 6 9



MSOE

Reach

A Graph

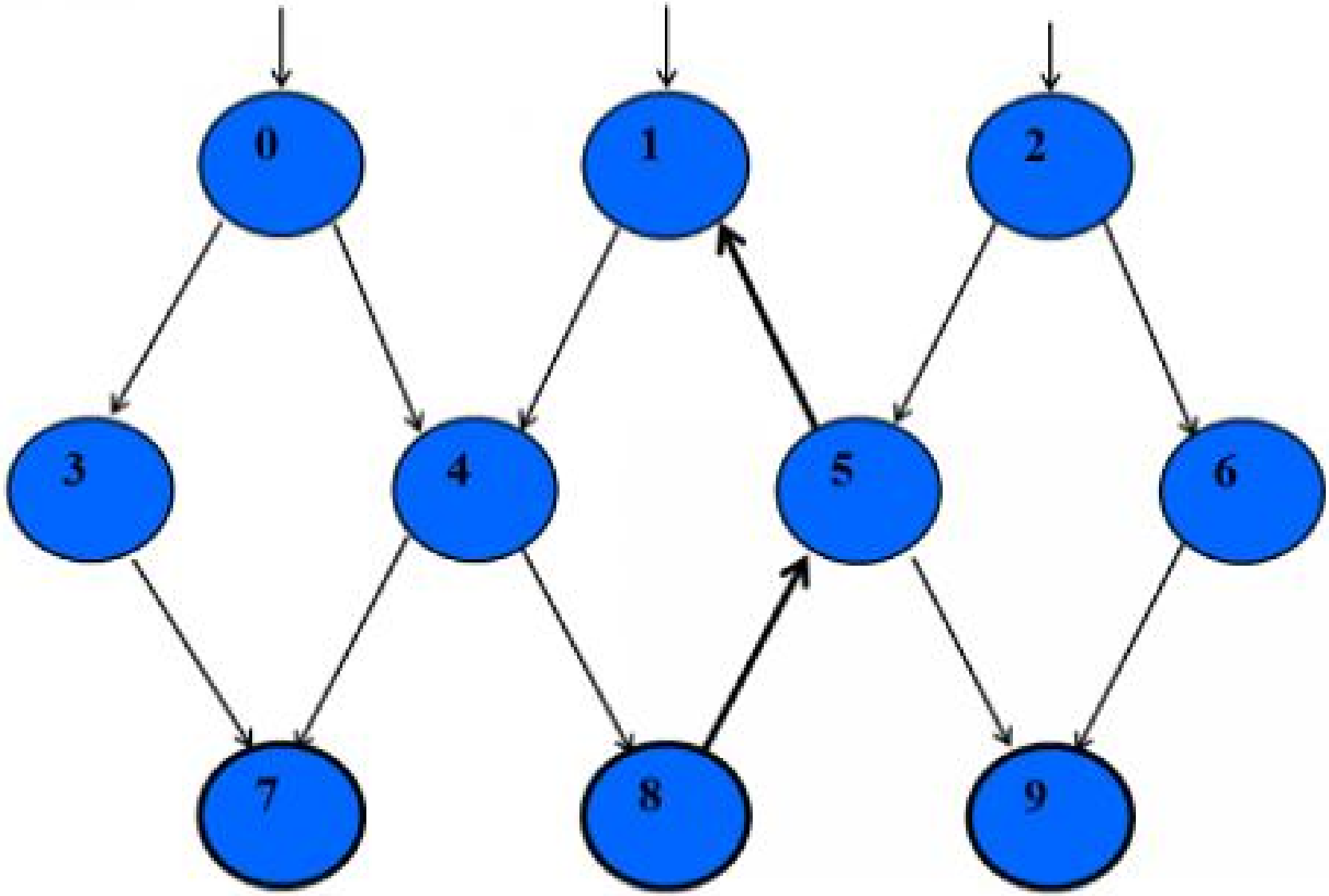


Vertices reachable from node 0.

Vtc 0 ⇒ 3 7 4 8 5 1 7

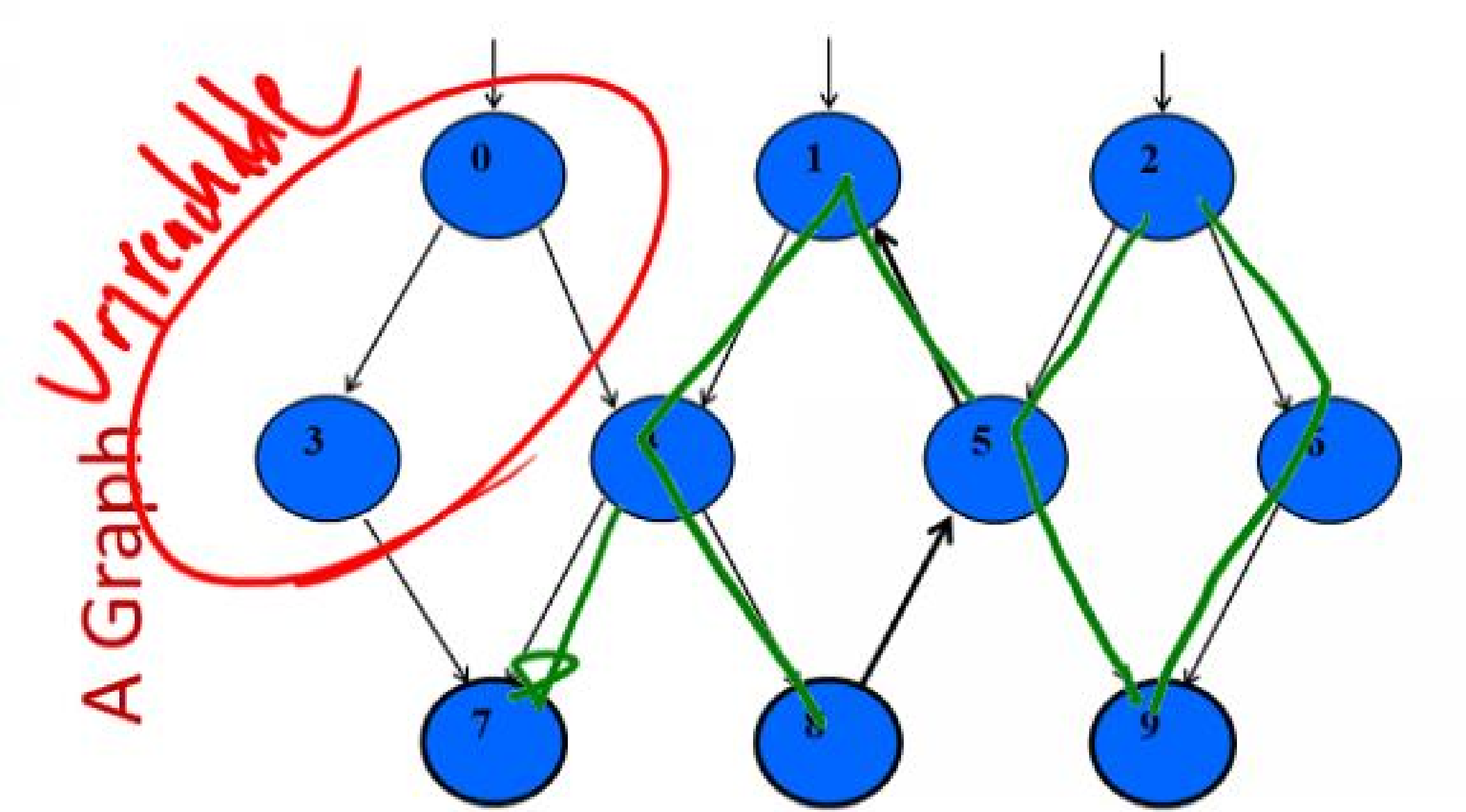
MSOE

A Graph



Reach(0,2) ⇒ All nodes (G)

MSOE



Reach(2,6) => 2, 6, 9

1 4 8

MSOE

Test Path

- 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
  - Ideally what we want to see in software

4 test paths

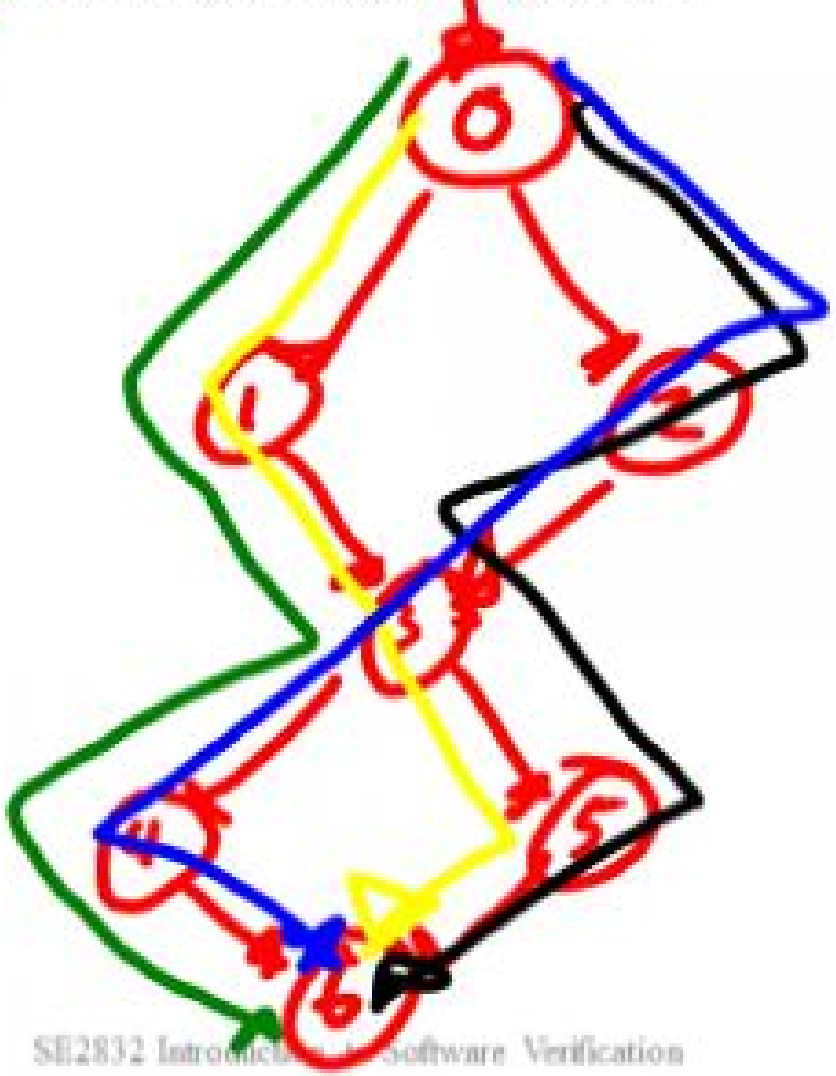
SESE Graph

0 1 3 4 6

0 1 3 5 6

0 2 3 5 6

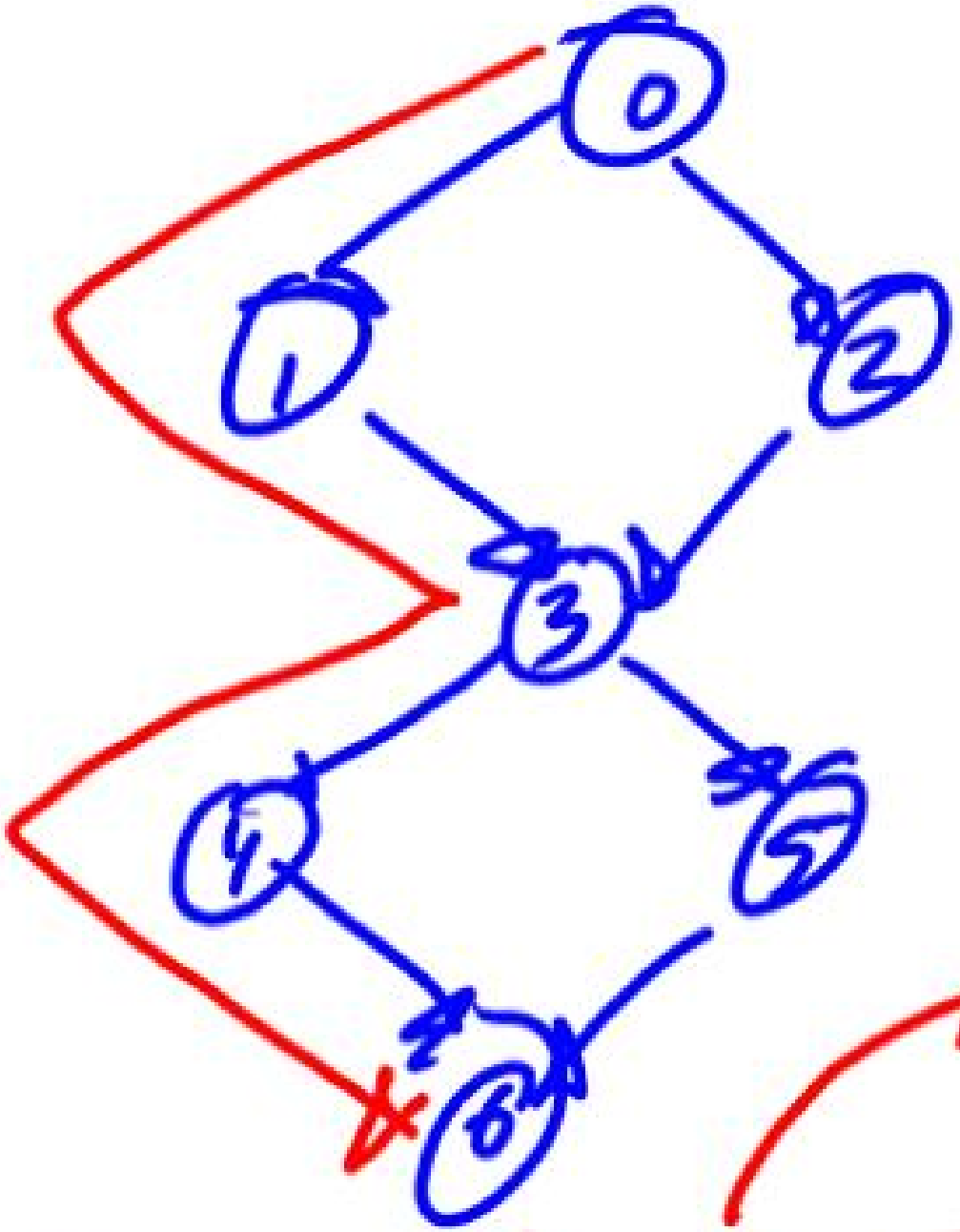
0 2 3 4 6



MSOE

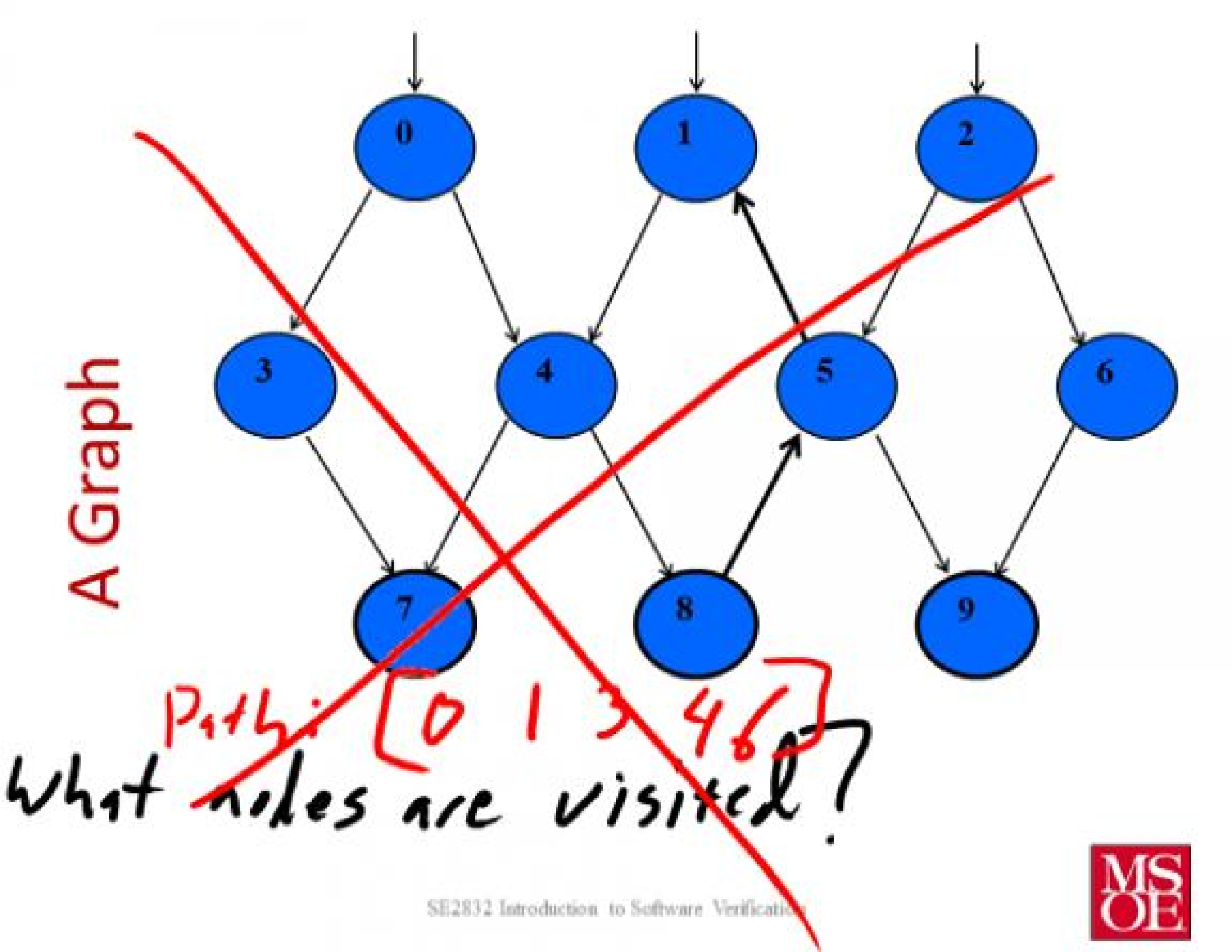
Visiting and Touring

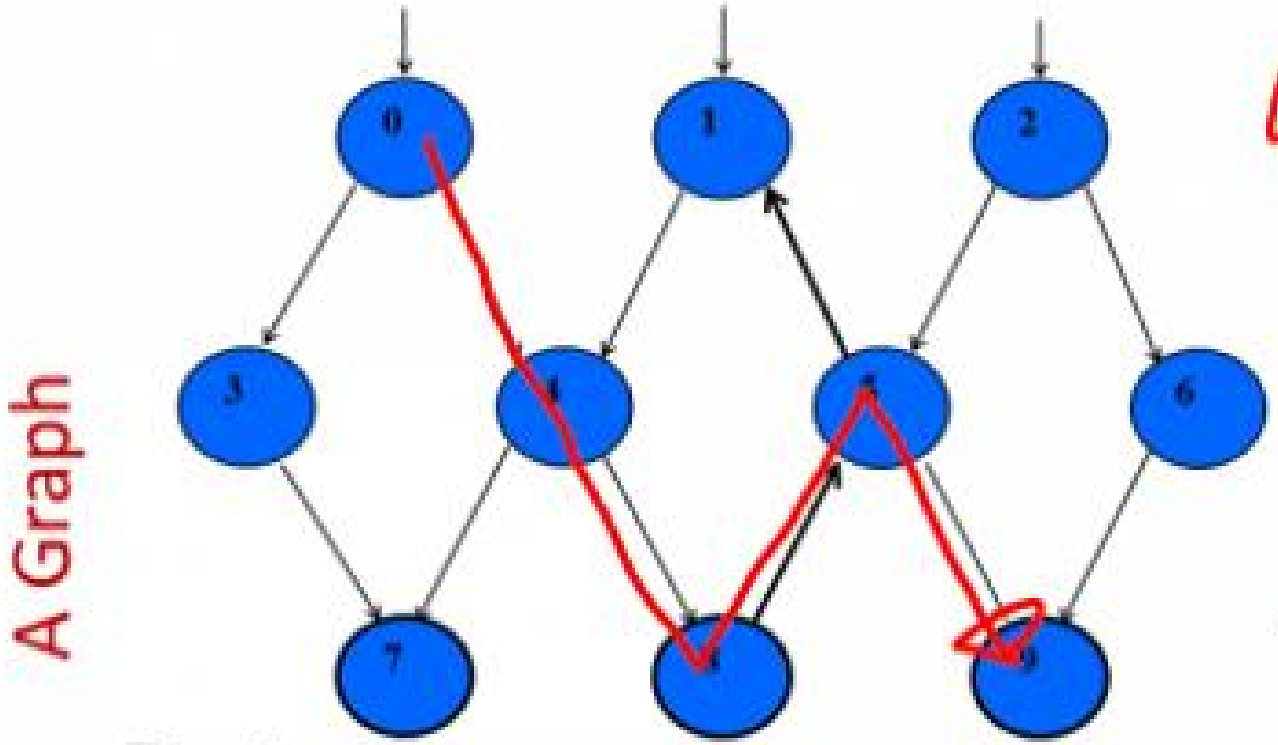
- Visit :
  - A test path *p* *visits* node *n* if *n* is in *p*
  - A test path *p* *visits* edge *e* if *e* is in *p*
- Tour :
  - A test path *p* *tours* subpath *q* if *q* is a subpath of *p*



[3,4,6]

Tours subpath [0,1,3]


A Graph
0
1
2
3
4
5
6
7
8
9
Path: [0 1 3 4 6]
what nodes are visited?



Path: 0 4 8 5 7

Nodes Visited: 0 4 8 5 7

Which subpaths were taured?

(0 4 8) (4 8 5) (8 5 7)

(4 8 5)

(8 5 7)

(0 4 8 5)

Visited which edges?

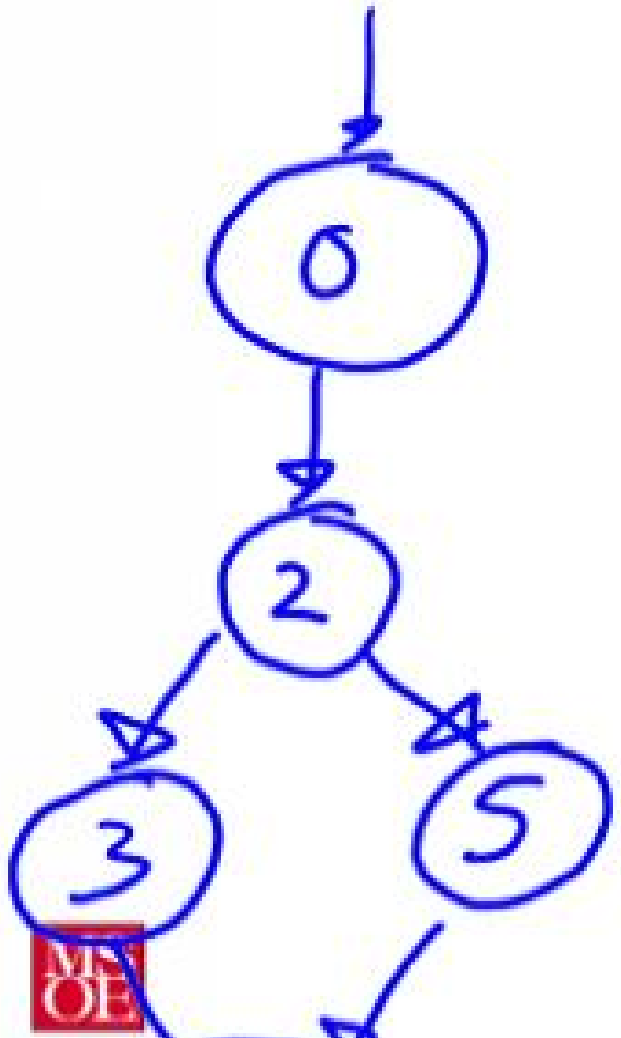
(0 4) (4 8) (8 5) (5 7)

Converting source code to a graph

```
int absoluteValue(int value)
{
    int returnValue = 0;
    if (value >= 0)
    {
      returnValue = value;
    }
    else
    {
      returnValue = 0 - value;
    }
  return returnValue;
}
```

Converting source code to a graph

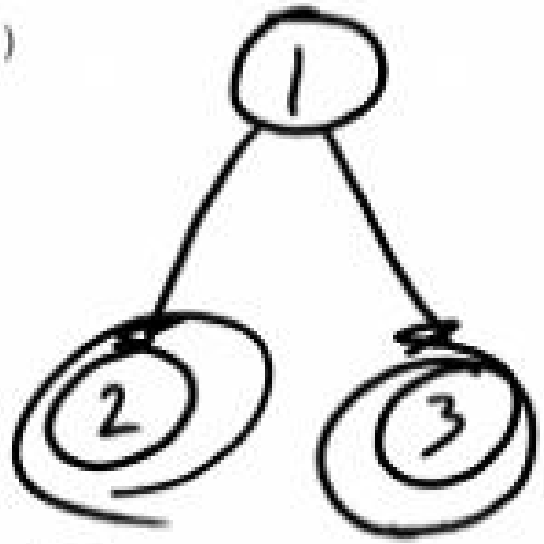
```
0 int absoluteValue(int value)
  {
1   int returnValue = 0;
2   if (value >= 0)
    {
3     returnValue = value;
    }
4   else
    {
5     returnValue = 0 - value;
    }
6 return returnValue;
  }
```



Paths: [0,2,3,6 → 1 ; 0,2,5,6 → -1]

Converting source code to a graph

```
boolean isOdd(int value)
{
1   if (value%2 != 0)
    {
2   return true;
    }
    else
    {
3   return false;
    }
}
```

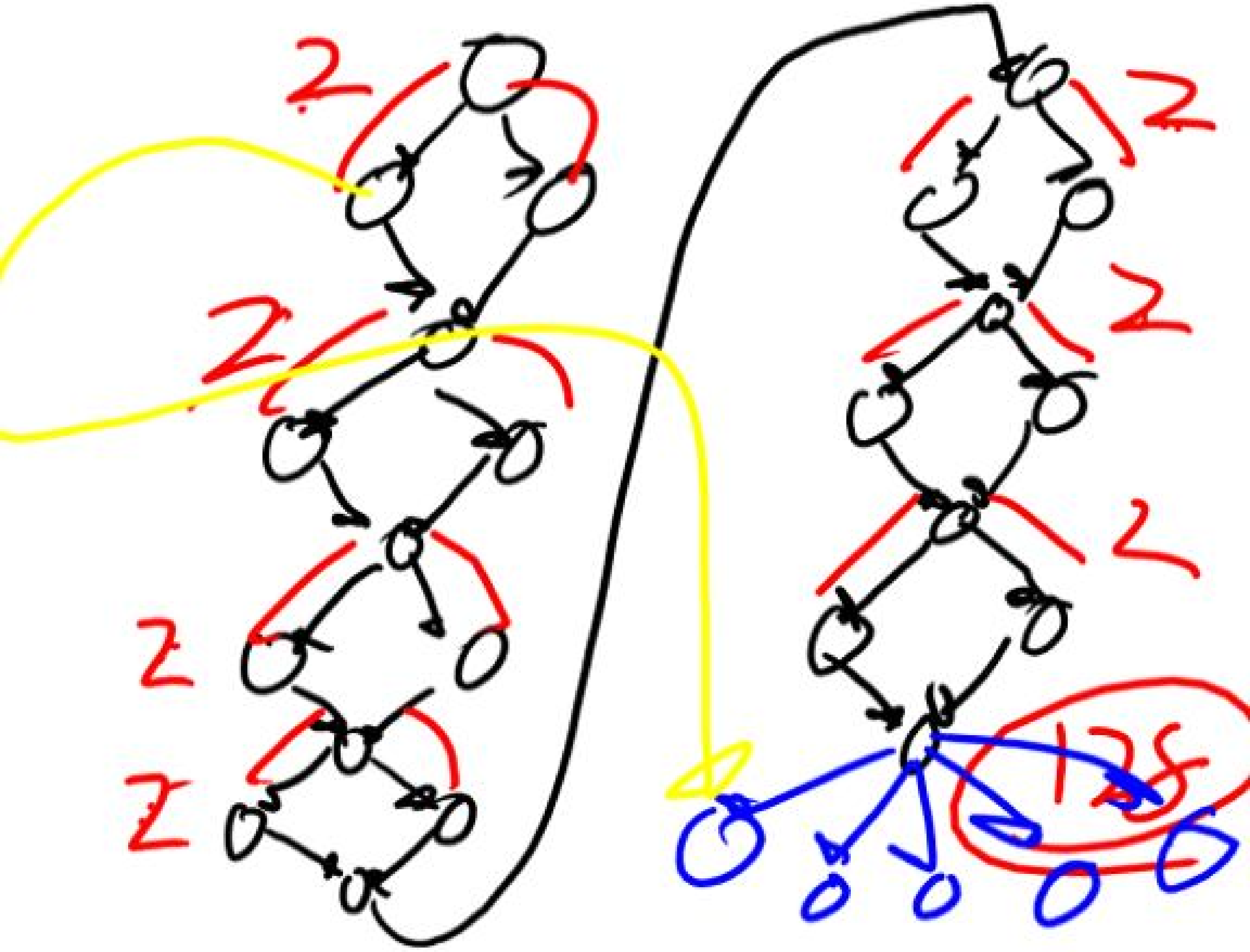


(1→2) ⇒ 3

(1→3) ⇒ 2

MS OE

NOT SESE


2
2
2
2
2
2
2
125