Discrete Math

Graph Theory

Hi

Lecture Objectives:

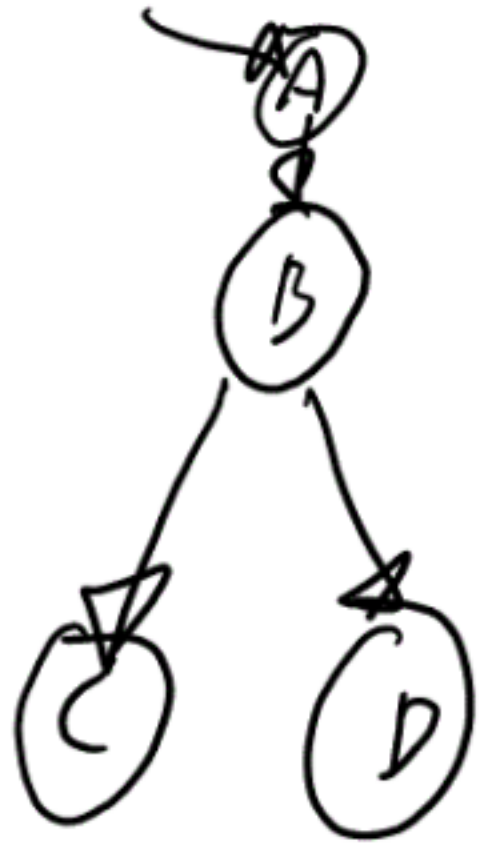
1) Define the term graph, node, initial node, final node, and edge.
2) Define reachability in a graph.
3) Define the term test path.
4) Explain the concept of a SESE graph.
5) Relate the concept of a SESE graph to good programming concepts.
6) Define the concept of visiting a node and explain the concept 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 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

Graphs

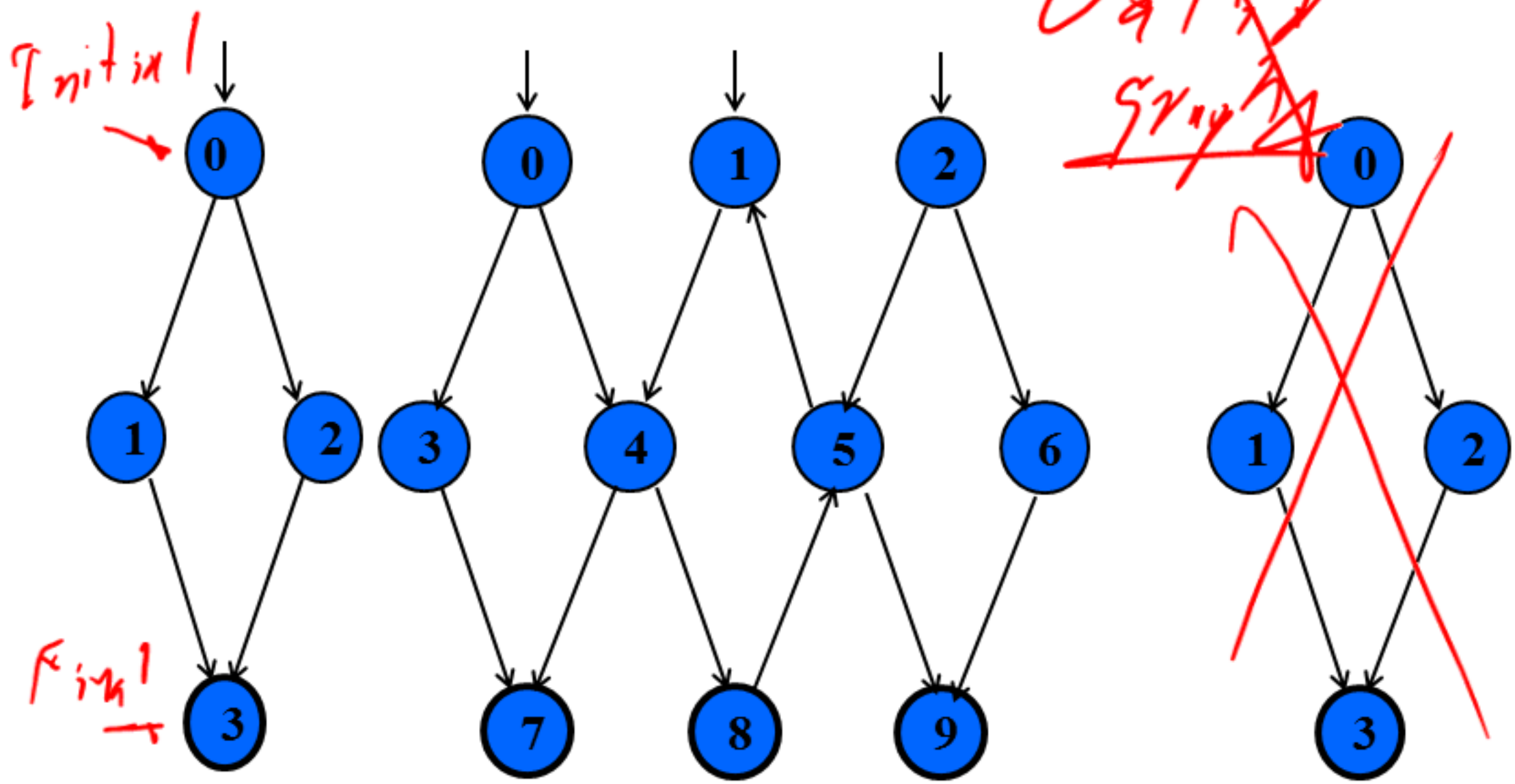
Example



Initial Node: A

Final Node: C, D

Three Example Graphs



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

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

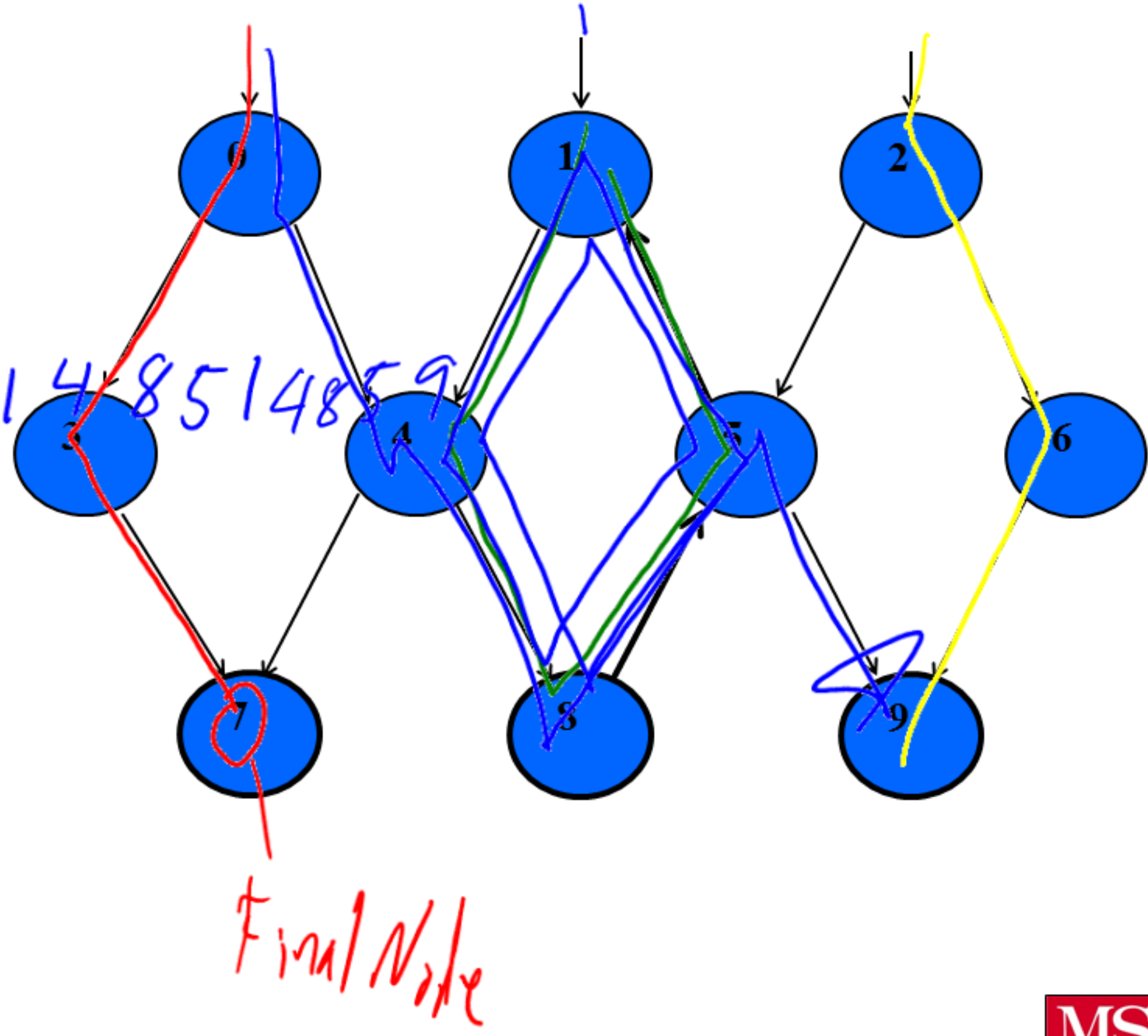
$N_0 = \{ \}$ $N_f = \{ 3 \}$

Paths in Graphs

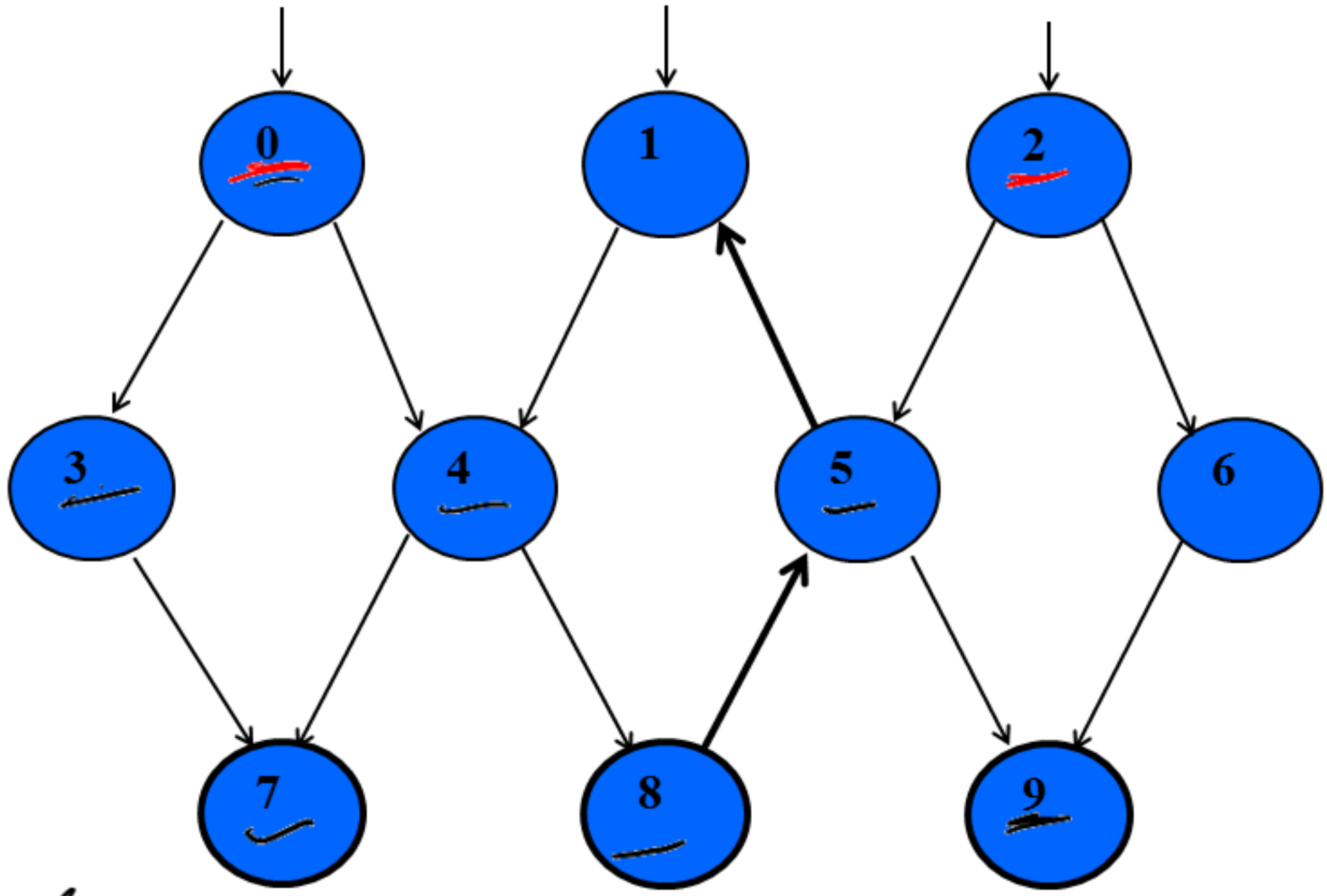
- Path : A sequence of nodes – $[n_1, n_2, ..., n_M]$
  - Each pair of nodes is an edge
- Length : The number of edges
  - A single node is a path of length 0
- Subpath : A subsequence of nodes in *p* is a subpath of *p*
- Reach (*n*) : Subgraph that can be reached from *n*
- *Reachable*
  - *Indicates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

A Graph

Paths:
0,3,7
1 4 8 5 1
2 6 9
0 4 8 5 1 4 8 5 1 4 8 5 9



A Graph



Reach

Reach(0) = { 0, 3, 4, 7, 8, 5, 9, 1 }

Reach({0,2}) = G

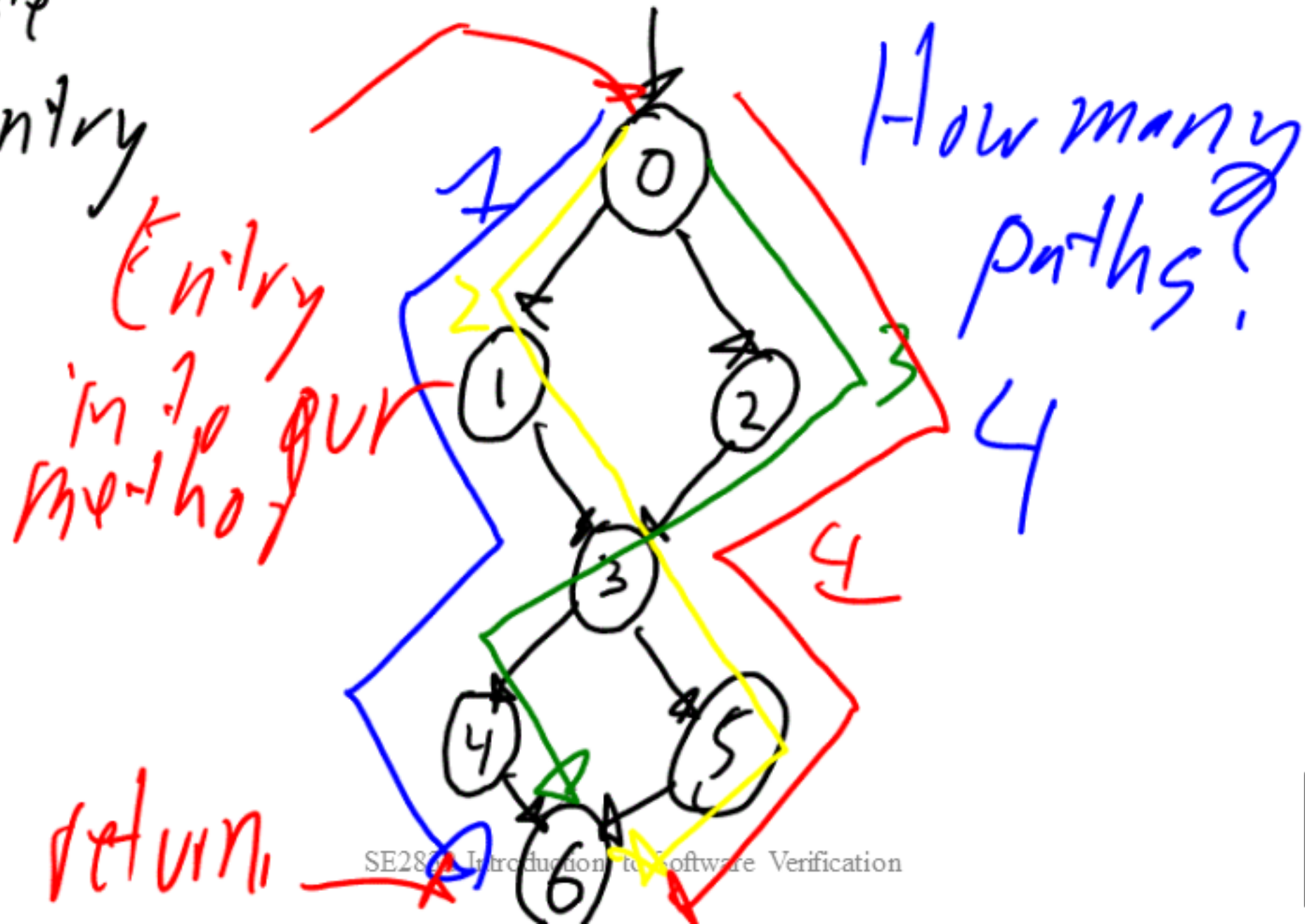
MSOE

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

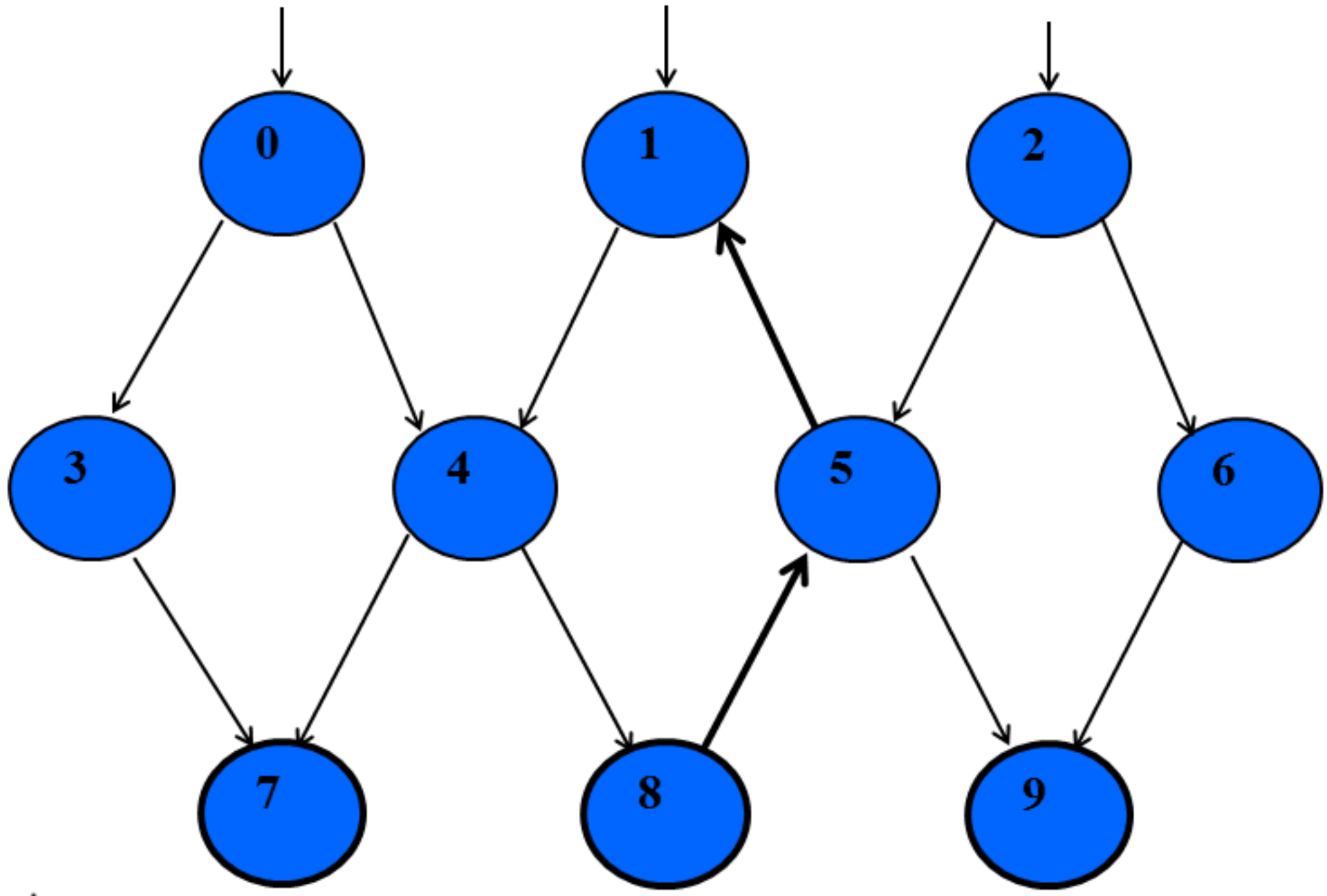
SESE Graph



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*

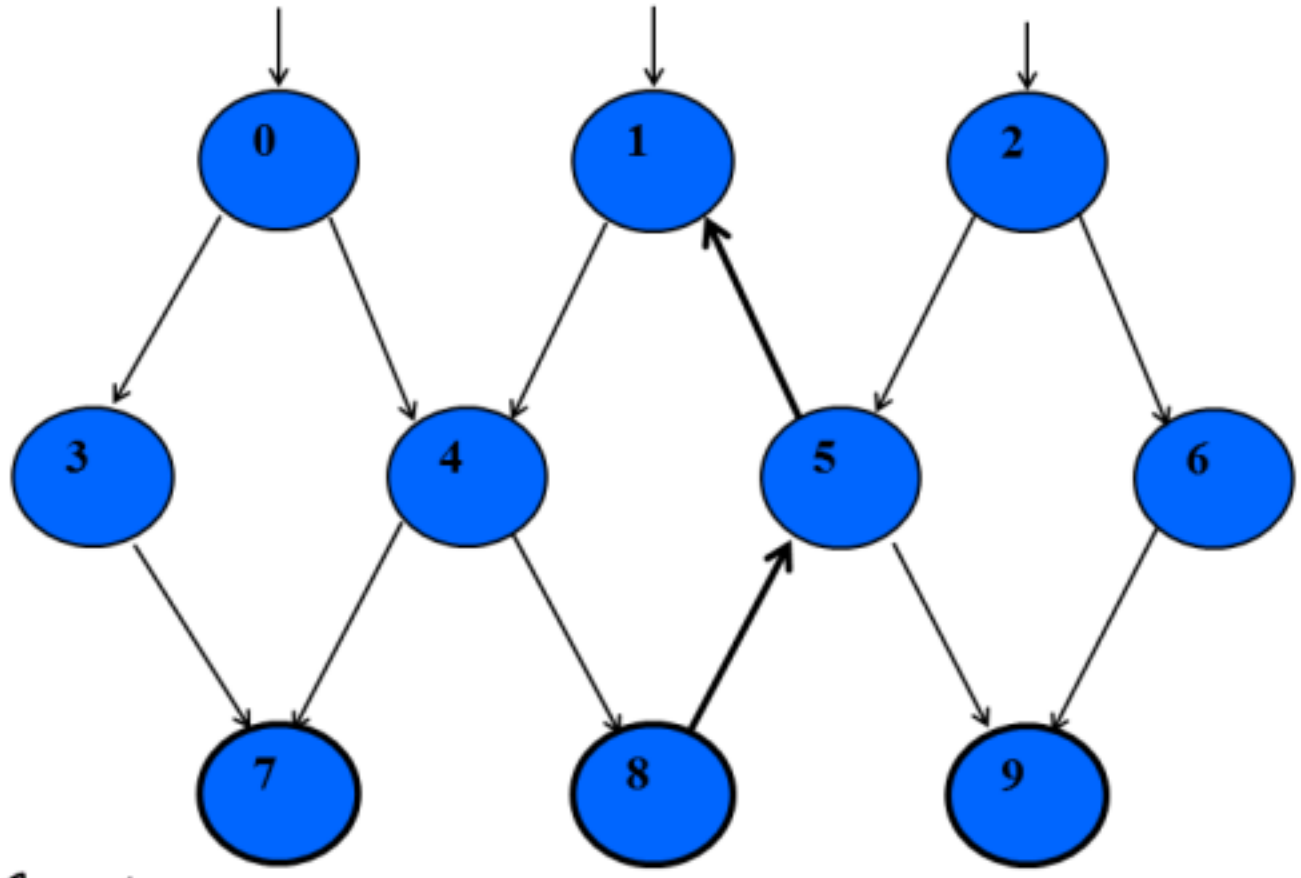
A Graph



Path 0 1 3 4 6

Visited Nodes: 0 1 3 4 6

A Graph



Path: 0 4 8 5 9

Nodes visited: 0, 4, 8, 5, 9

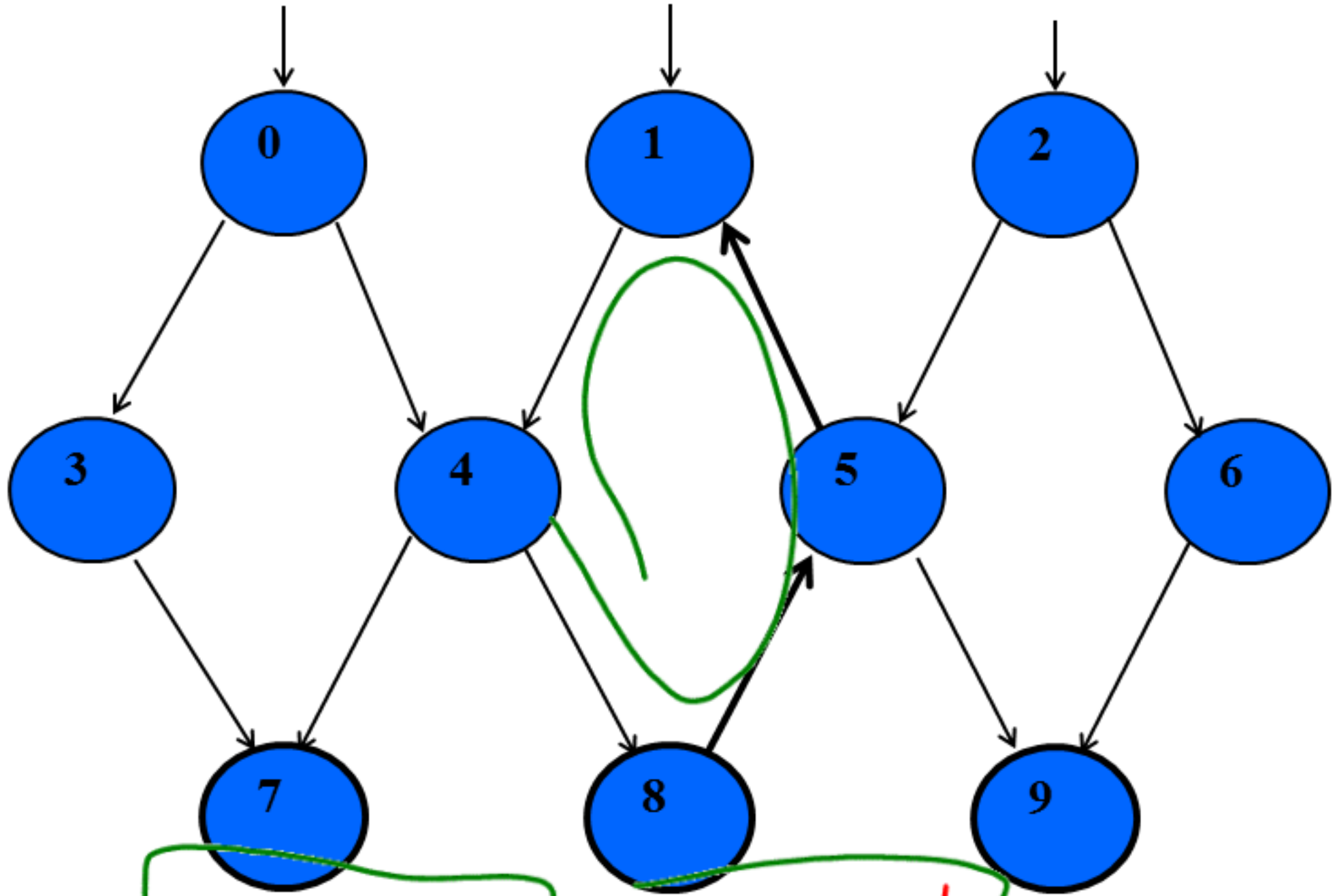
Edges visited: (04) (48) (85) (59)

MSOE

Subpaths toured: 048 485

0485 859

A Graph



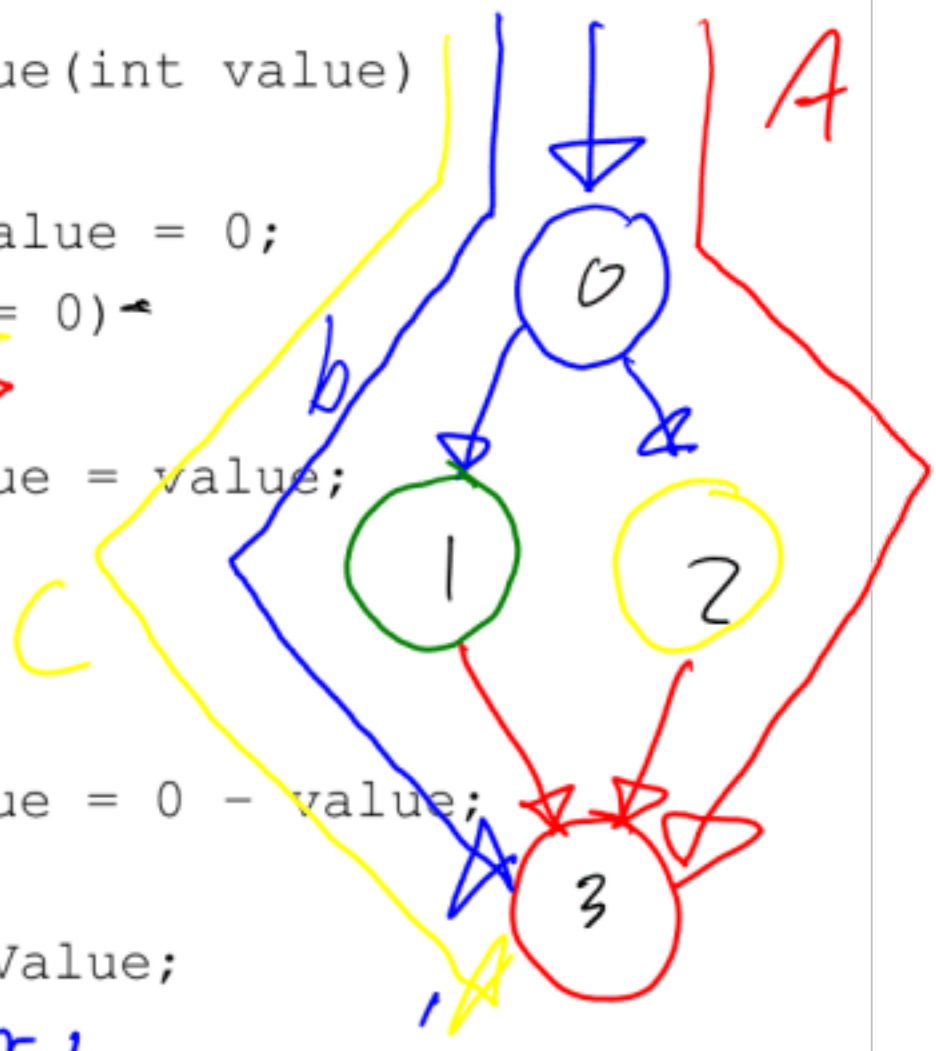
Path: 0 4 8 5 1 4 1 8 5 1 4 8 5 9

Subpath 8 5 1 4

MSOE

Converting source code to a graph

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



Control Flow map

MSOE

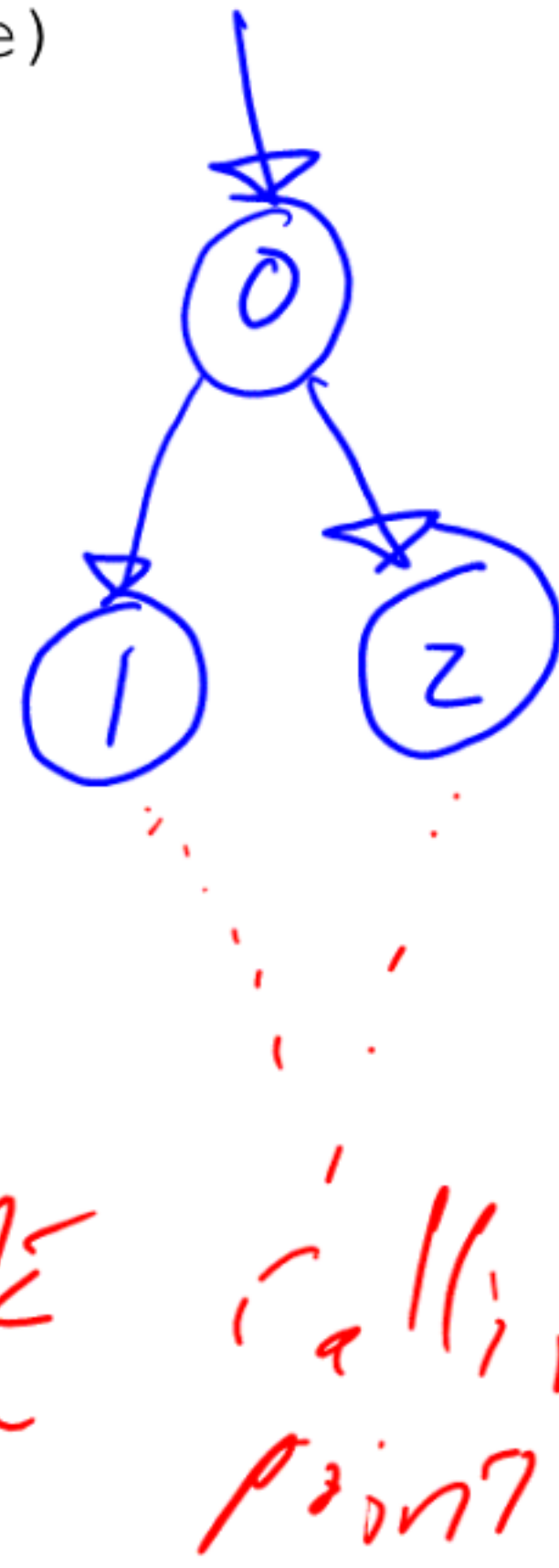
Test Cases: (-5, 5) - A

(5, 5) - B

(0, 0) - C

Converting source code to a graph

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



Not SESE calling point

```
int Mod 5 (int val)
{   if (val % 5 == 0)
    { return 0; }
    else if (val % 5 == 1)
    { return 1;
    else if (val % 5 == 2)
    { return 2; }
    else if (val % 5 == 3) 3),
else return 4;
```