Coverage

Lecture Objectives:

1) Define node coverage
2) Define edge coverage
3) Define edge-pair coverage
4) Define prime path
5) Define prime path coverage
6) Construct a set of test paths which meet the criteria for node coverage
7) Construct a set of test paths which meet the criteria for edge coverage

Testing and Graphs

- We use graphs in testing as follows :
  - Developing a model of the software as a graph
  - Requiring 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

Code Formating

⇒ Colorization of
code keywords

⇒ Indentation

⇒ Method Abstraction

Visualized code:

UML class diagrams

↳ How classes are related

↳ static

↳ Sequence diagrams

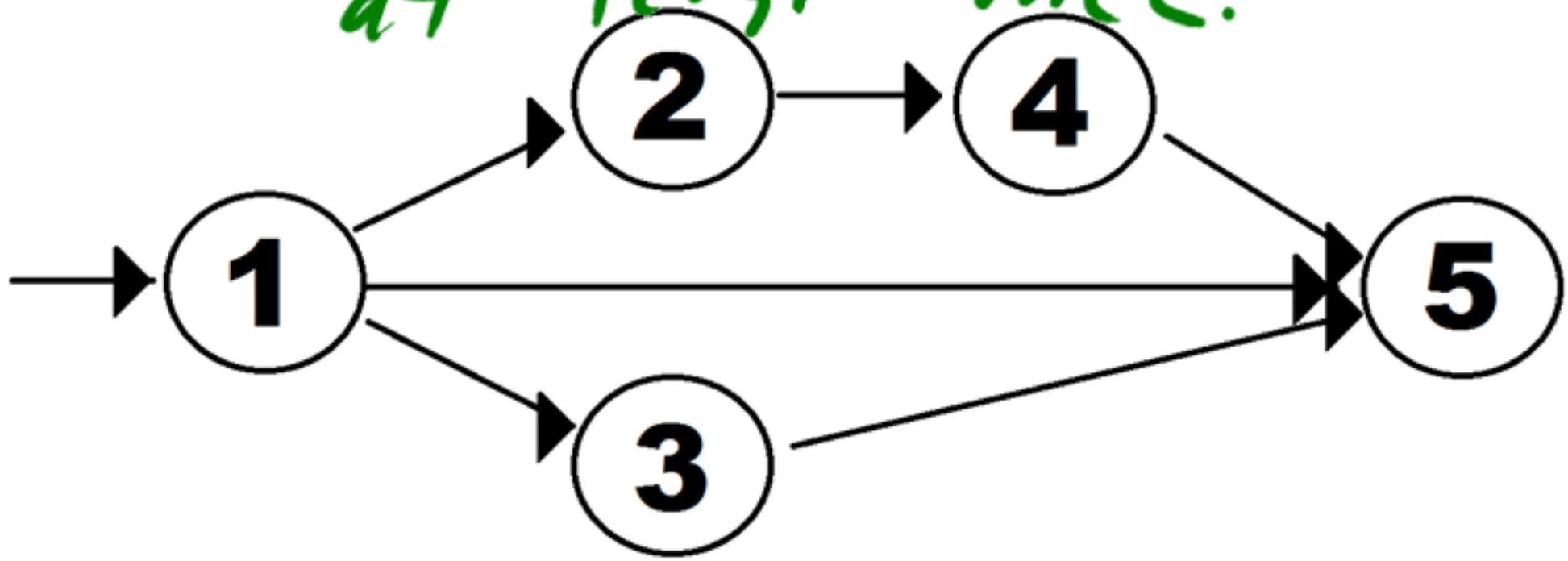
⇒ show execution over time.

Node Coverage

- 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.

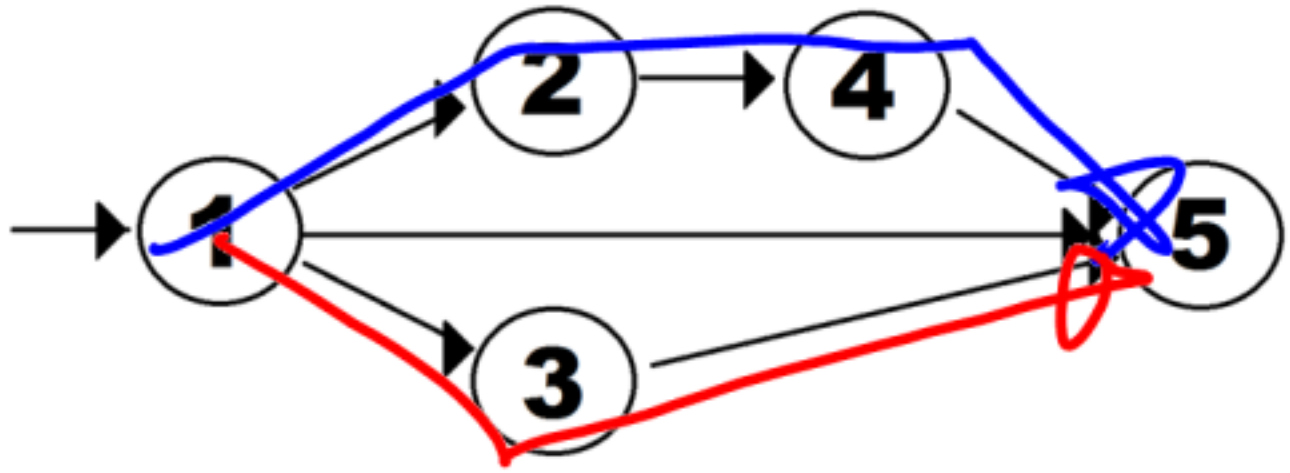
Minimal, test cases.

<=> Get to every node at least once.



Node Coverage

- 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.



How many test cases?

2

1 2 4 5

1 3 5

MSOE

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

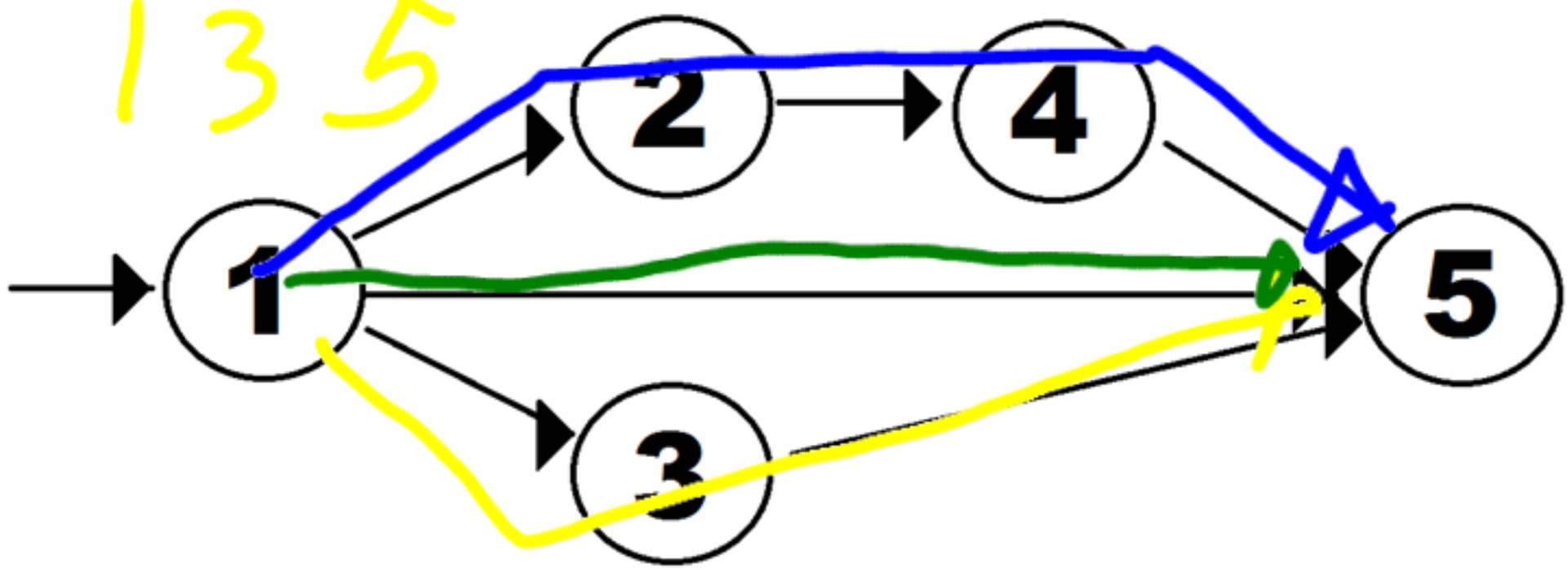
3 test case

Slightly stronger, better form of testing.

Edge Coverage

1 2 4 5

1 3 5



15

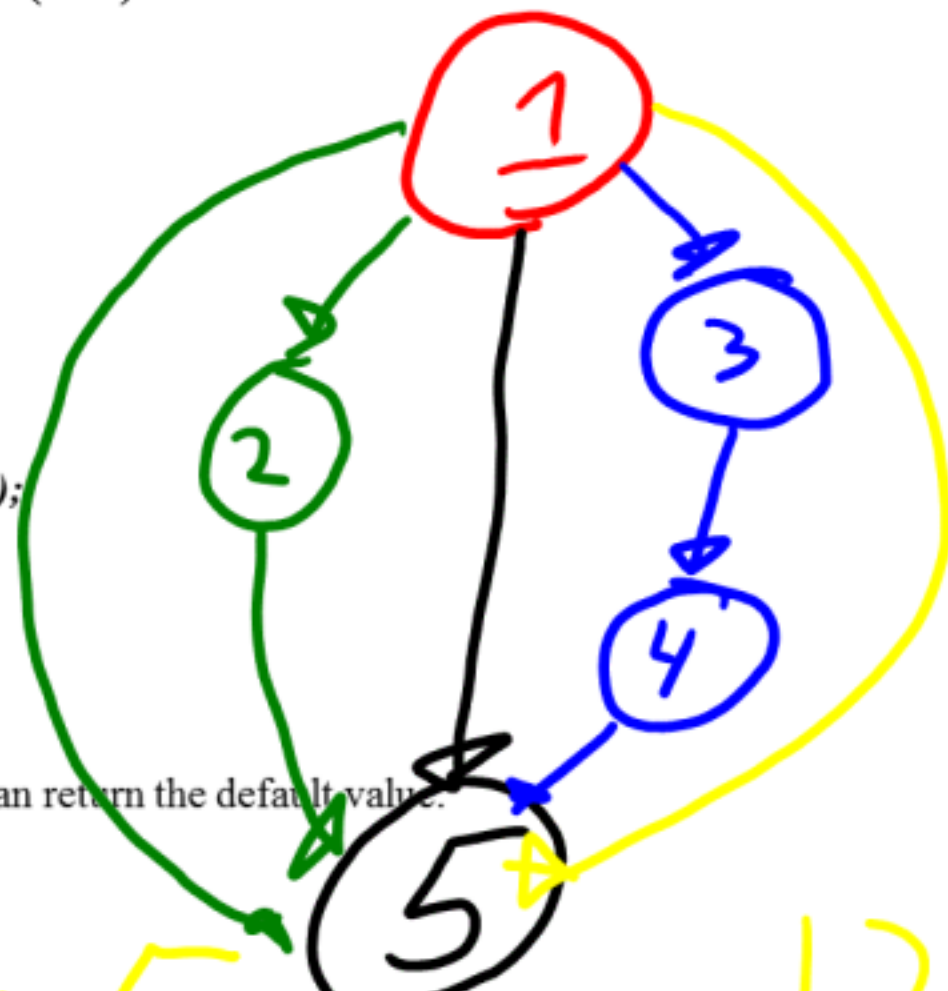
Example Code representing this graph

```
public static int factorial(int x)
{
  int returnValue = 1;

  if (x < 0)
  {
    returnValue = 0;
  }
  else if (x > 0)
  {
    int p1 = factorial(x-1);

    returnValue = p1 * x;
  }
  else
  {
    // Do nothing other than return the default value.
  }

  return returnValue;
}
```



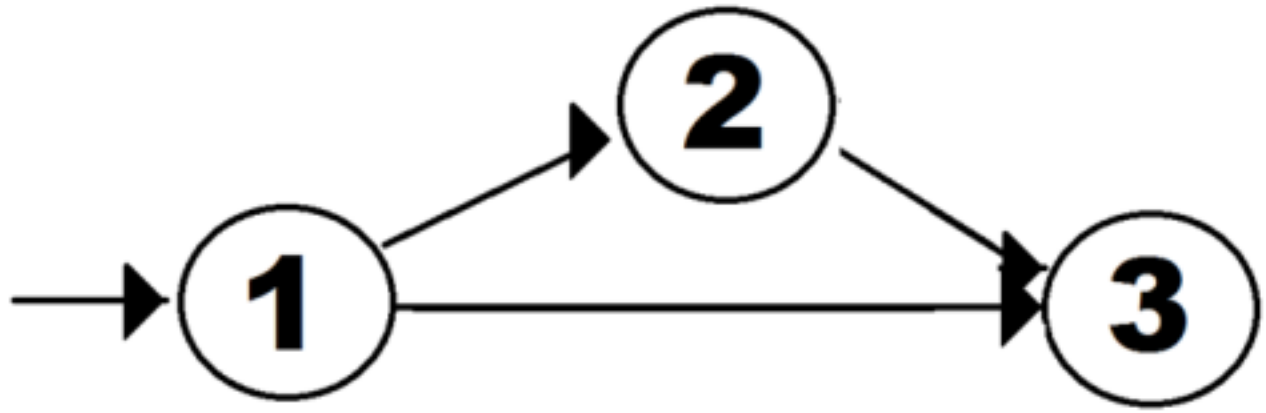
1 3 4 5 ⇒ 5 ⇒ 1 2 0

MSOE

1 2 5 ⇒ -1 ⇒ Expected 0

1 5 ⇒ 0 ⇒ 1

Why is edge coverage better than node coverage?



One Test case Node Coverage.

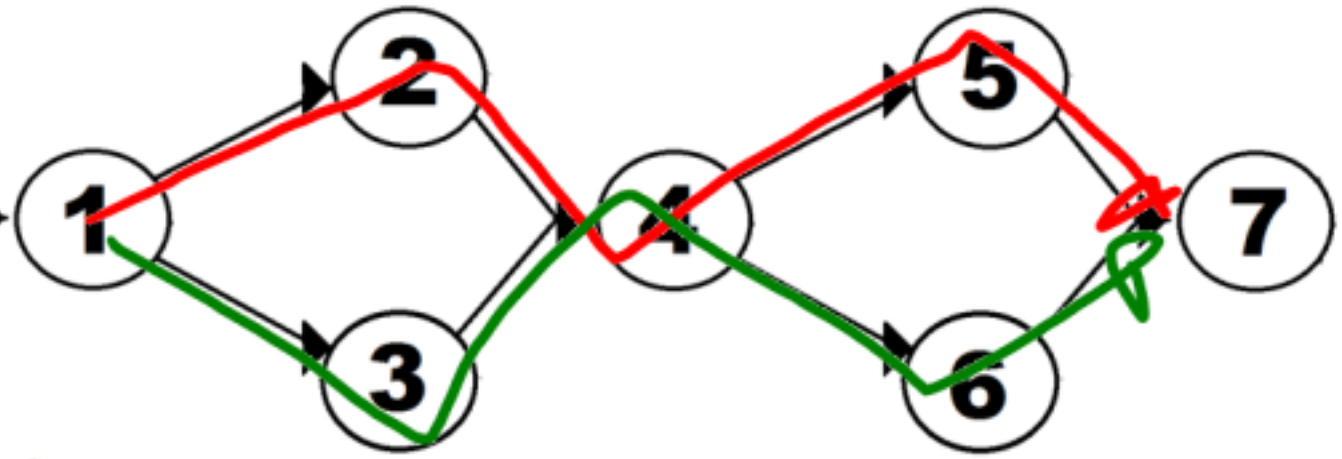
1 2 3

```
if( x<5)
{
  y = negative;
}
```


MSOE

- TR contains each reachable path of length up to 2, inclusive, in G.

Edge-Pair Coverage



Edge Coverage:

2 ⇒ 1 2 4 5 7

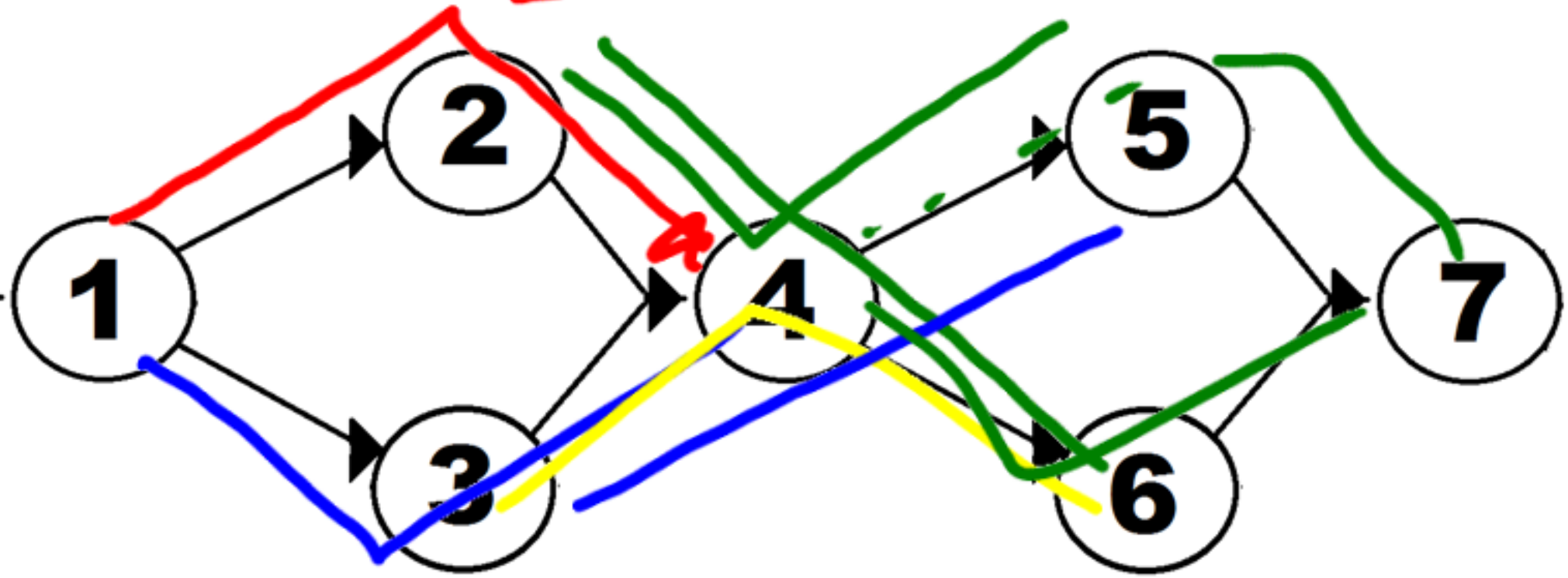
1 3 4 6 7

MSOE

Nodes 1 & 4 are if statements. one is y one is x based

- TR contains each reachable path of length up to 2, inclusive, in G.

Edge-Pair Coverage



124

245

134

346

246

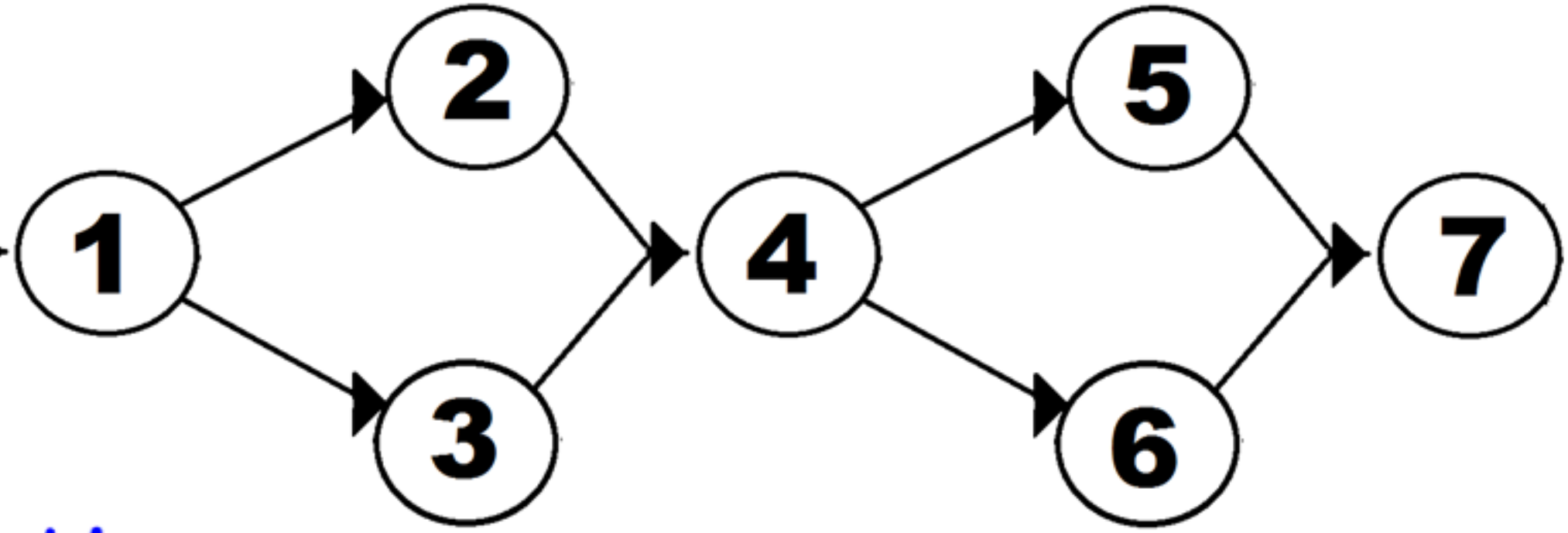
345

457

467

- TR contains each reachable path of length up to 2, inclusive, in G.

Edge-Pair Coverage



How many test cases?

4 => 12457 13457

12467 13467

How do we deal with loops in testing?

- If a graph contains a loop, it has an infinite number of paths
- Thus, CPC is not feasible

Complete Path Coverage Not Feasible

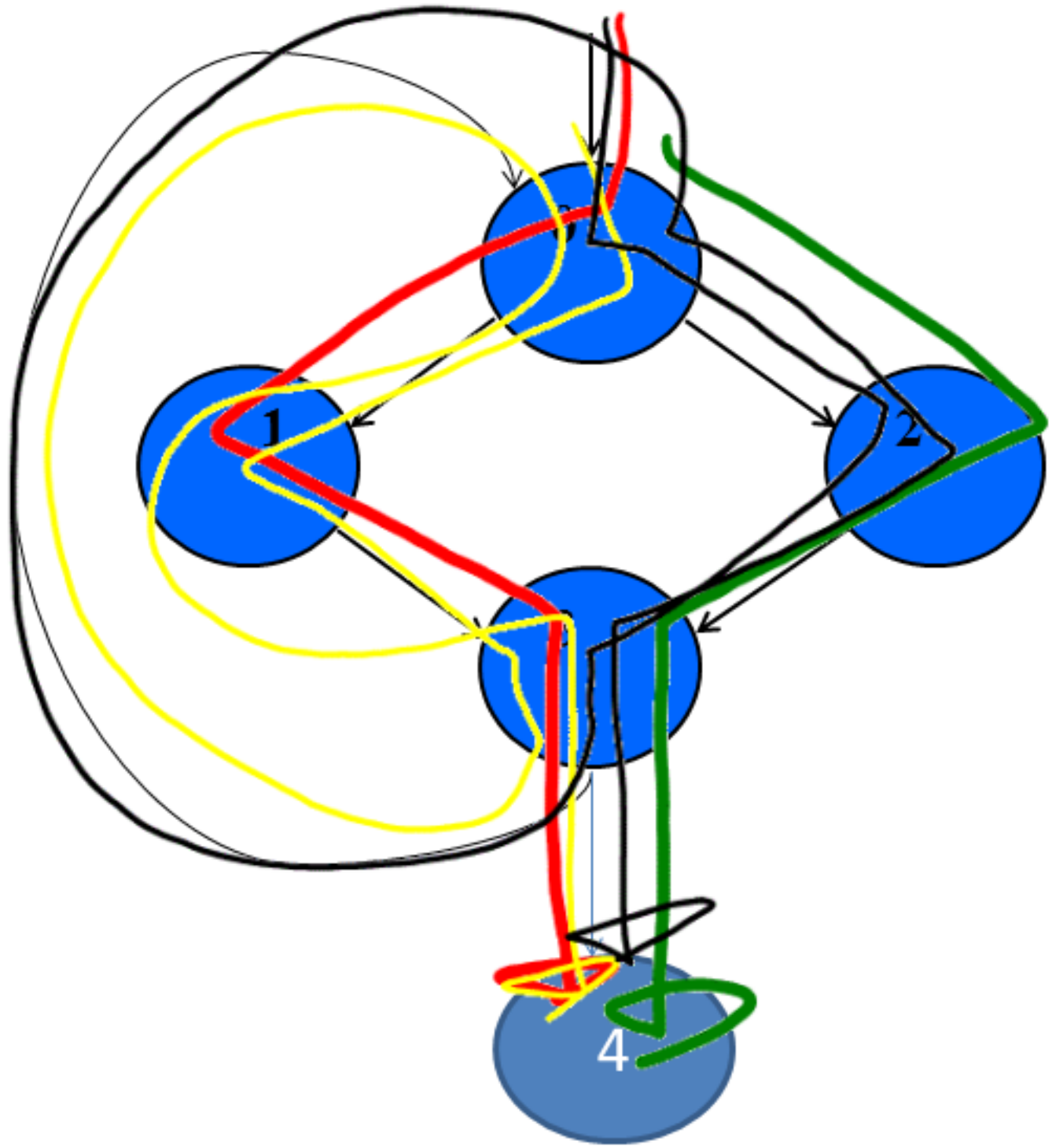
How do we deal with loops

- 1970s :
  - Execute cycles once ([4, 5, 4] in previous example, informal)
- 1980s :
  - Execute each loop, exactly once (formalized)
- 1990s : → Smarter
  - Execute loops 0 times, once, "more than once" (informal description)
- 2000s :
  - Prime paths ⇒ helps us a bit

Simple and prime paths

- Simple Path
  - A path from node ni to nj is simple if no node appears more than once, except possibly the first and last nodes are the same
    - No internal loops
    - Includes all other subpaths
    - A loop is a simple path
- Prime Path
  - A simple path that does not appear as a proper subpath of any other simple path

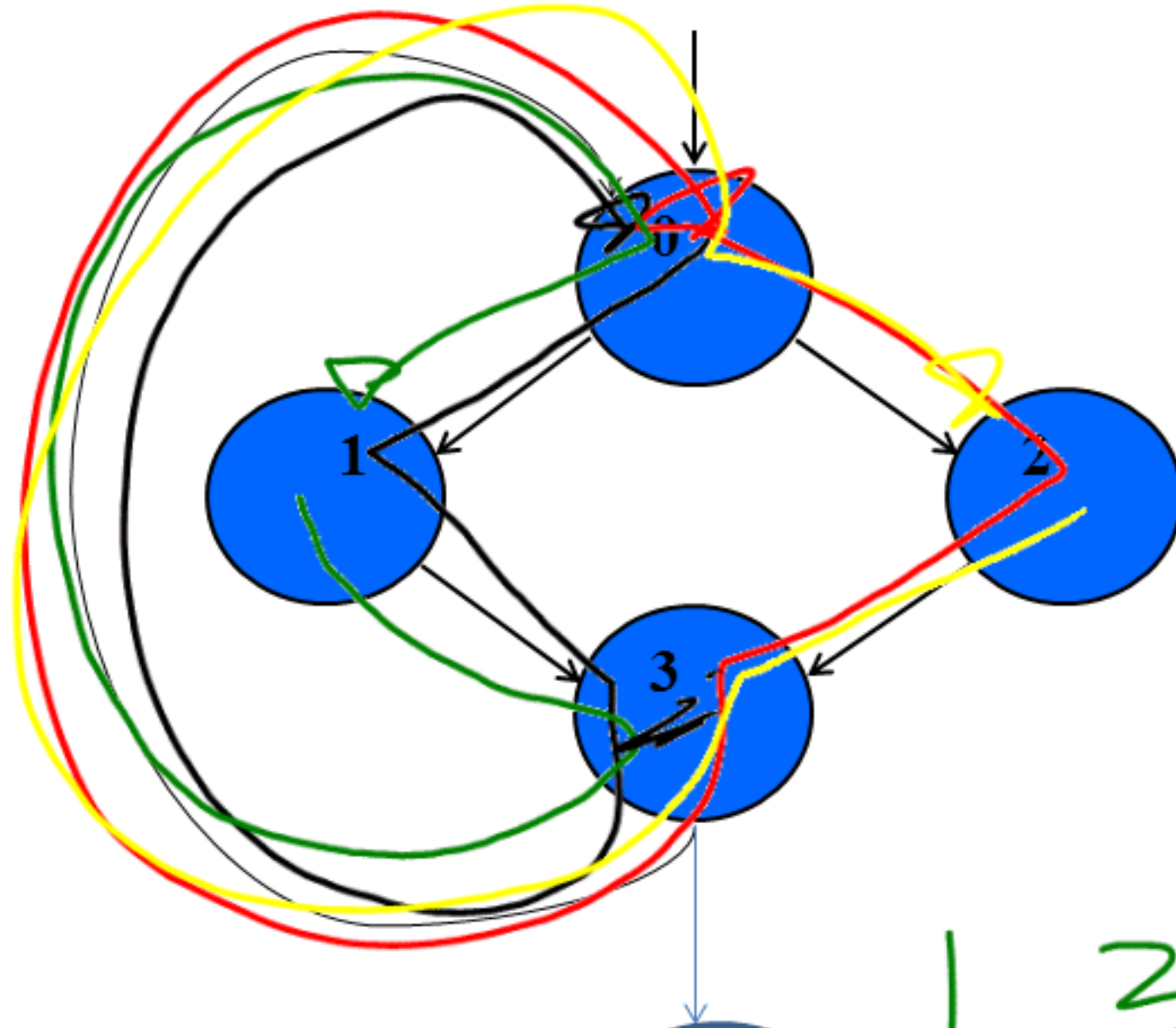
Simple and Prime Paths



0 1 3 4 0 2 3 4

0 1 3 0 1 3 4 0 2 3 0 2 3 4

Simple and Prime Paths



0 1 3 0

0 2 3 0

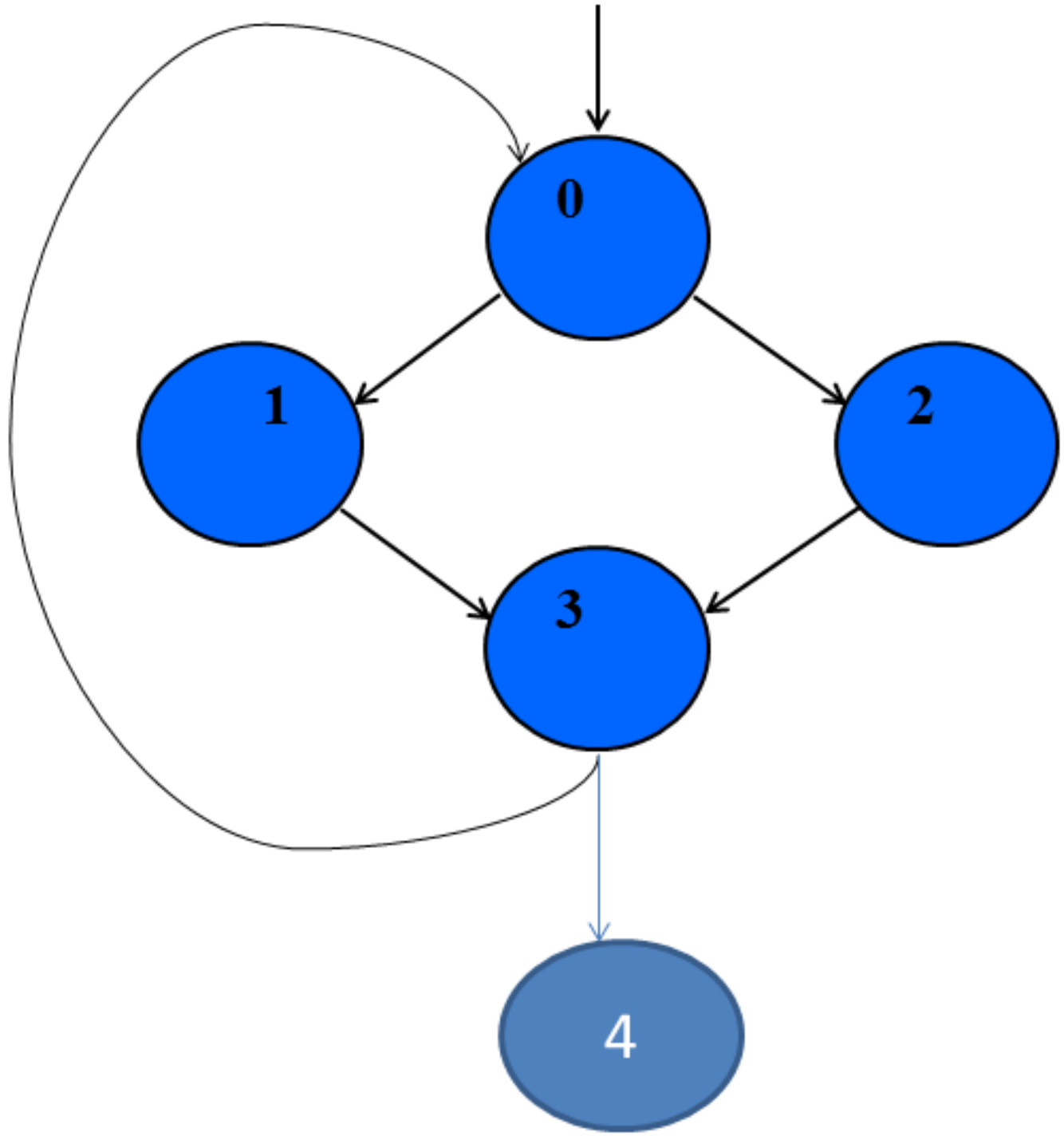
[3 0 1 3] [3 0 2 3]

1 3 0 1

2 3 0 2

1 3 0 2

Simple and Prime Paths



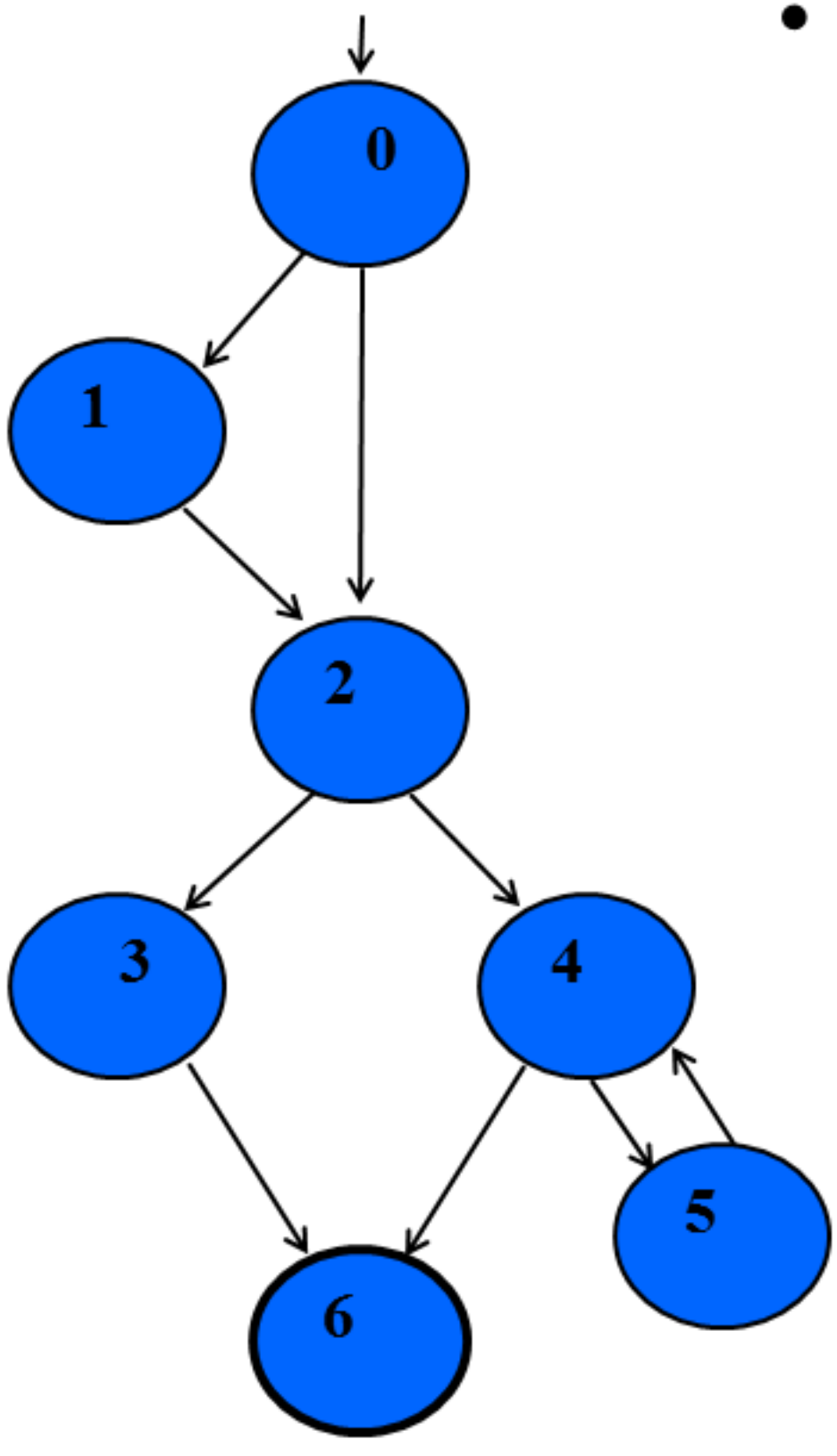
Prime Path Coverage / Complete Path Coverage

- Prime Path coverage
  - TR contains each prime path in G.

A path from start to finish w/out duplicating proper subpaths.

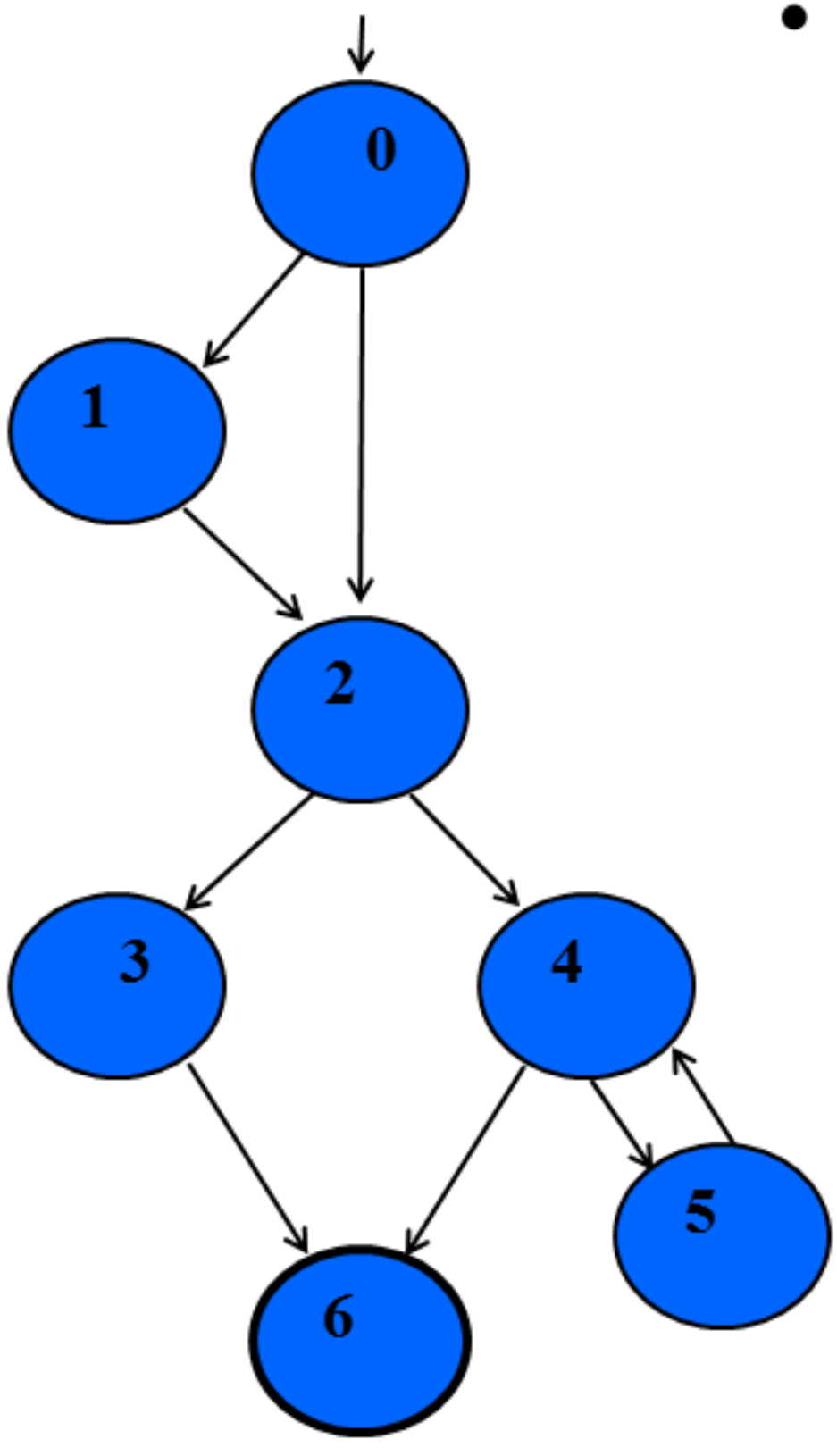
- Complete Path Coverage
  - TR contains all paths in G.

Exercise: Construct test cases



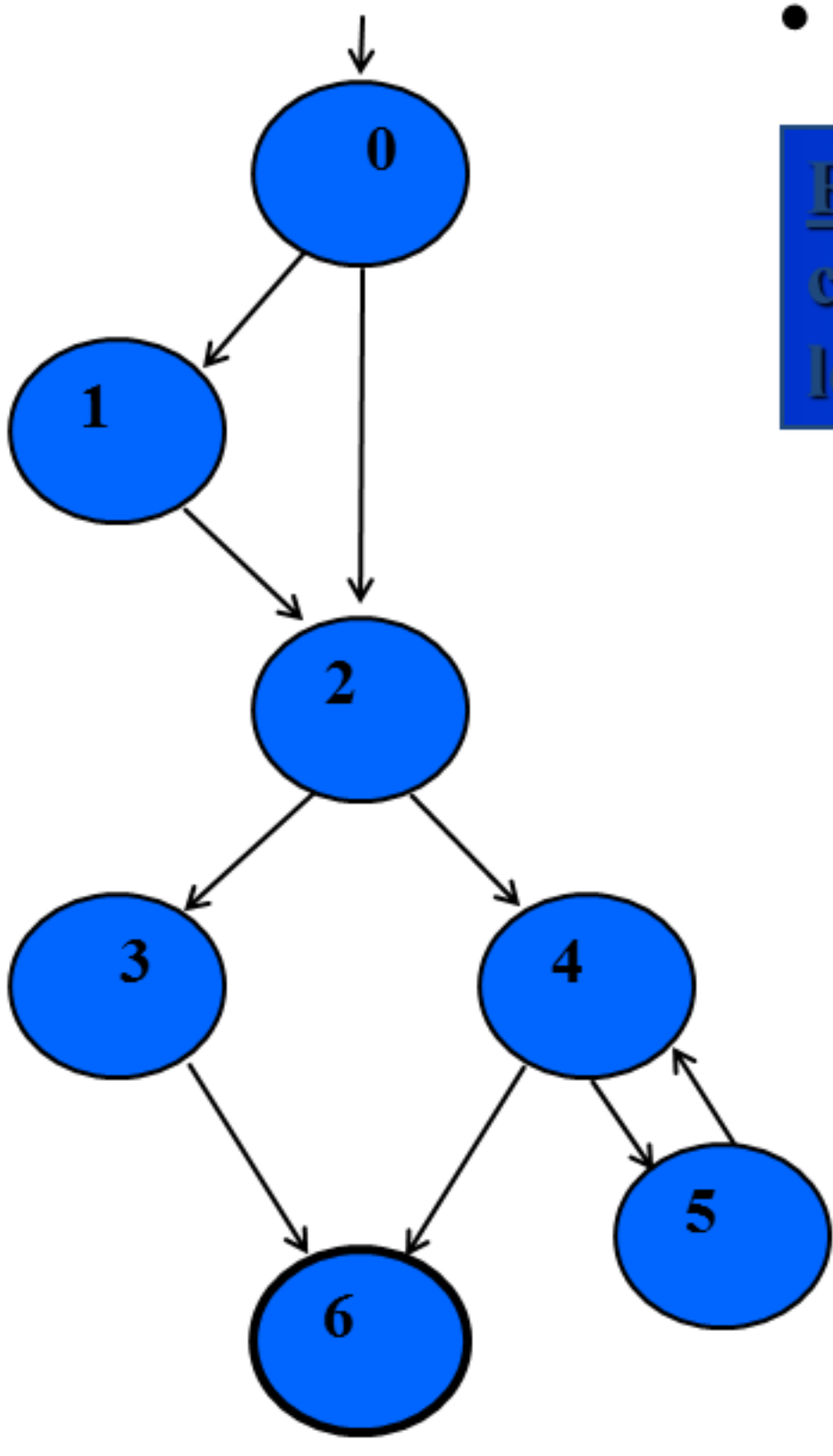
- Node Coverage

Exercise: Construct test cases



- Edge Coverage

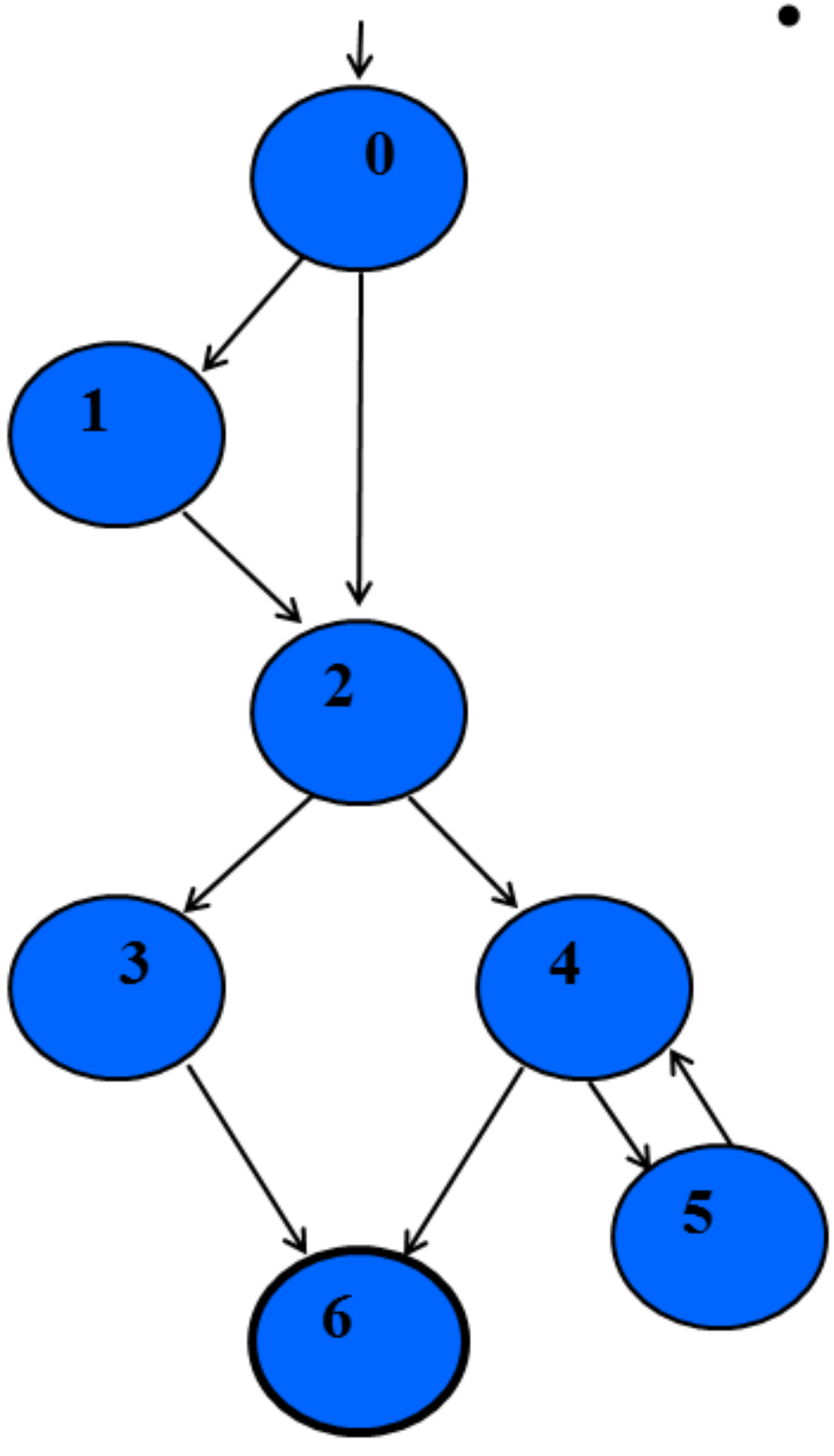
Exercise: Construct test cases



- Edge Pair Coverage

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

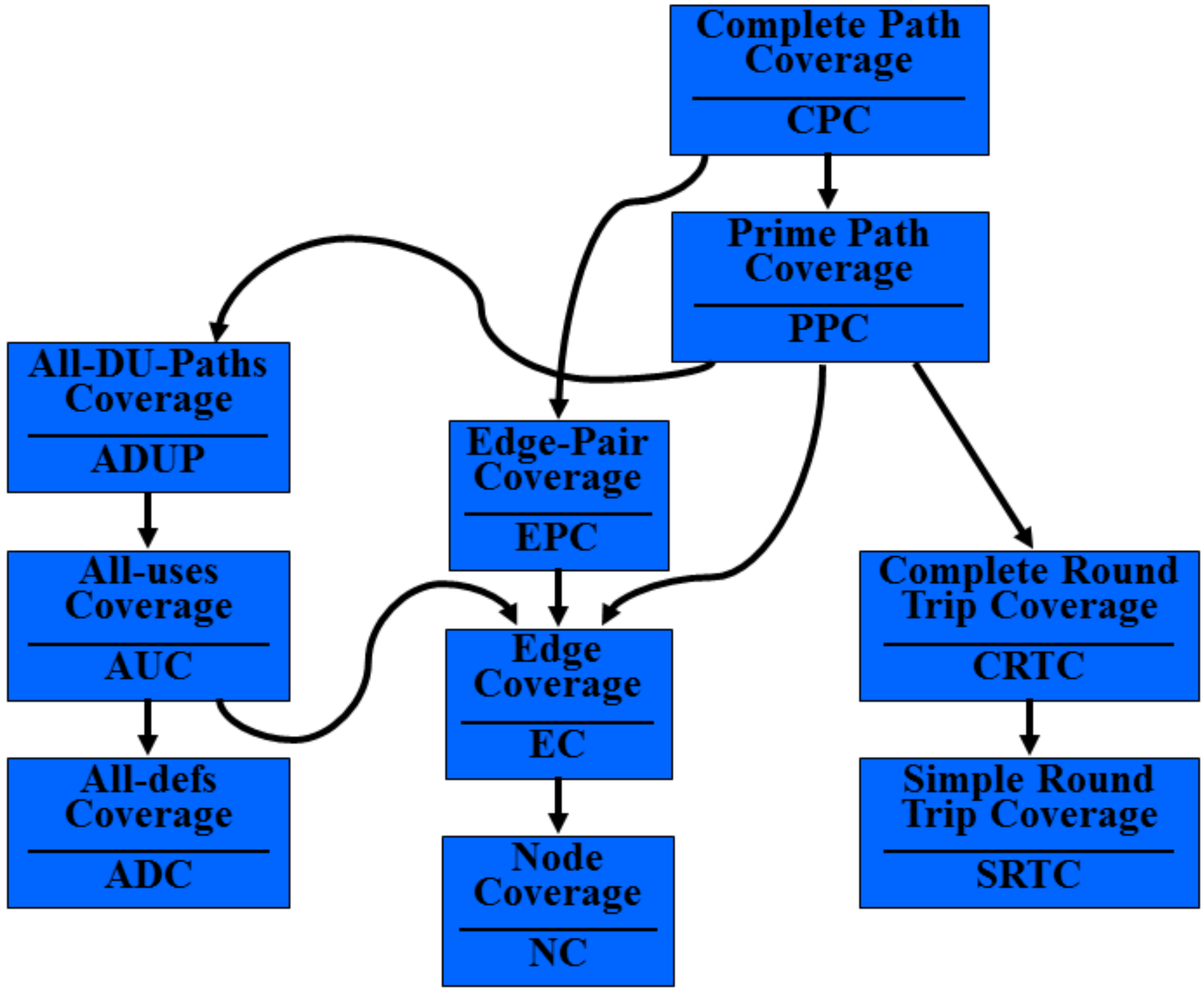
Exercise: Construct test cases



- Complete Path Coverage

Relationship between coverage criteria

(Note: We didn't cover all of these)



How does this relate to the real world?

- What does this mean from last week's lab?
- Which coverage types does EMMA provide?

Going Back to code

```
public static void printit(int value) {
    value = value % 5;
    switch (value) {
            case 1:
            System.out.println("1");
            break;
            System.out.println("Not good.");
            case 2:
            System.out.println("2");
            break;
            case 3:
            System.out.println("3");
            break;
            case 4:
            System.out.println("4");
            break;
            case 5:
            System.out.println("5");
            break;
            default:
            break;
     }
    return;
}
```

Dealing with loops

(Hint: There is a bug in this code…)

```
/**
 * The method will sum the numbers between 1 and the actual
number.
 * @param value This is the number that is to be counted up to
 * @return
 */
public static int sumNumbers(int value)
{
int retValue = 0;

  for (int index = -1; index < value; index++)
  {
     retValue += index;
  }
return retValue;
}
```