Graph Coverage

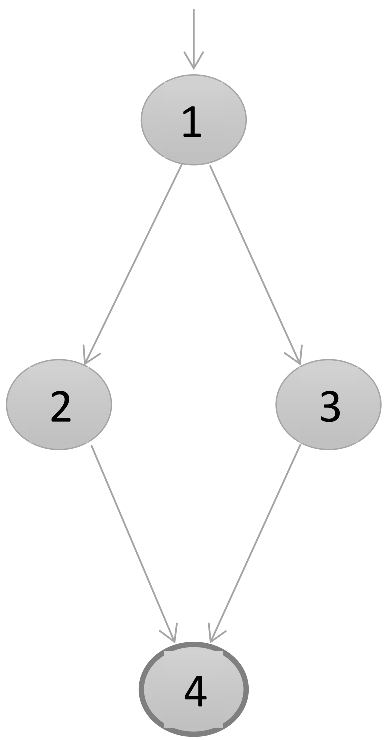
Software Testing
(3104313)

Amirkabir University of Technology
Spring 1399-1400

Definition of a Graph

- A set $\boldsymbol{N}$ of nodes, N is not empty
- A set $\boldsymbol{N_0}$ of initial nodes, N_0 is not empty
- A set $\boldsymbol{N_f}$ of final nodes, N_f is not empty
- A set $\boldsymbol{E}$ of **edges**, each edge from one node to another
  - (n_i , n_j), n_i is predecessor, n_j is successor

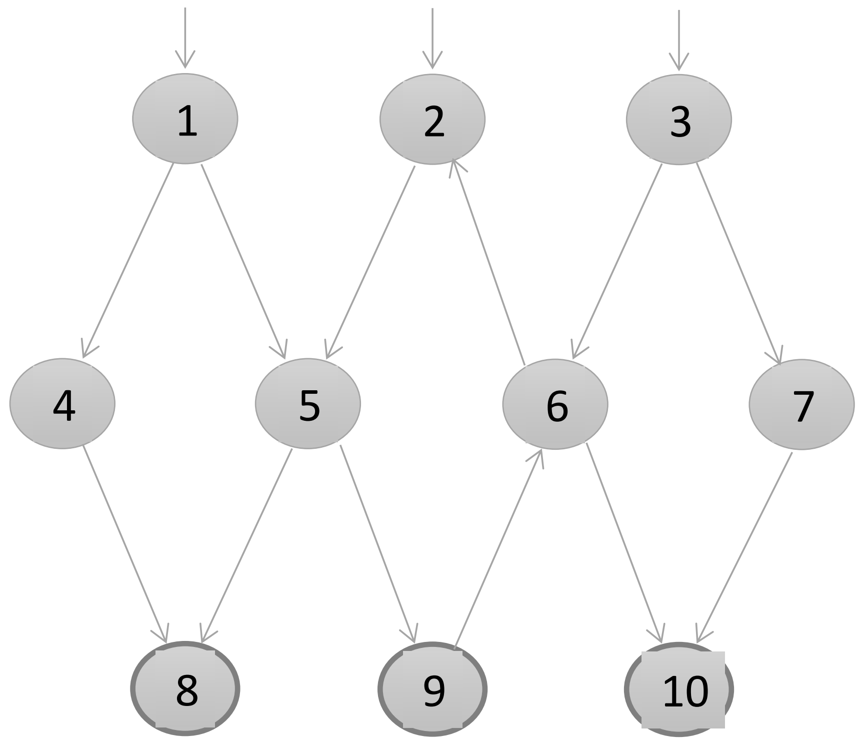
Example Graphs



N_0 = { 1 }

N_f = { 4 }

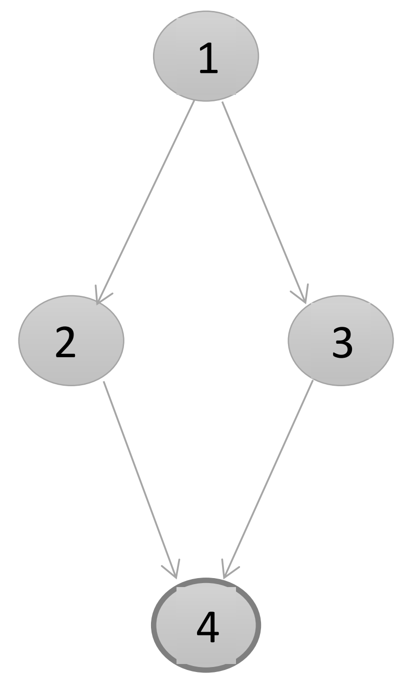
E = { (1,2), (1,3), (2,4), (3,4) }



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 Graph

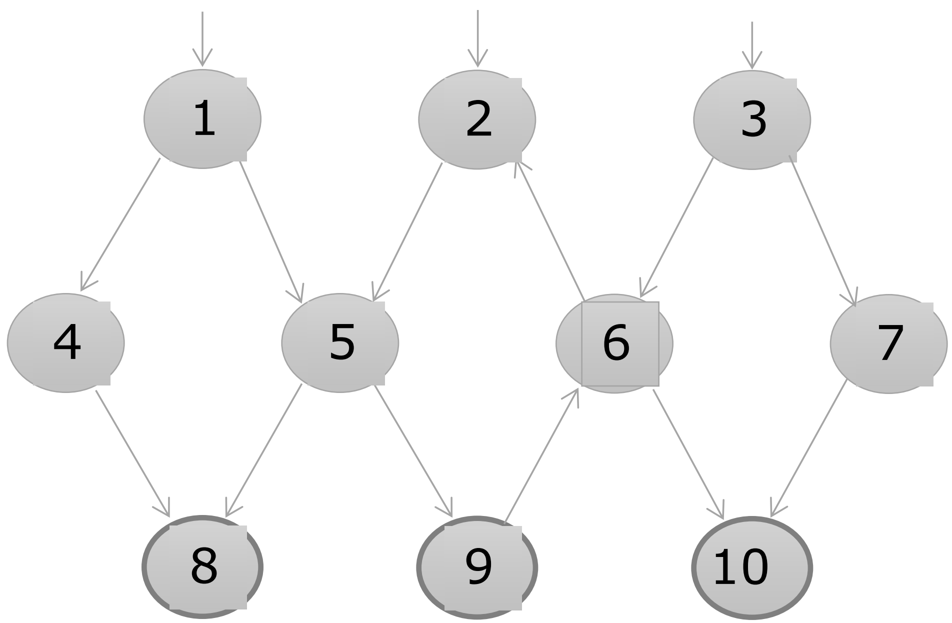
Path - A sequence of nodes

- [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*



A Few Paths

[1, 4, 8]

[2, 5, 9, 6, 2]

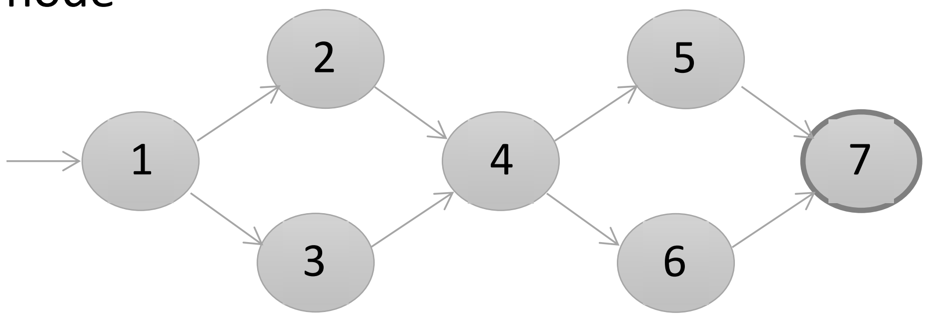
[3, 7, 10]

Test Paths

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
  - N_0 and N_f have exactly one node



Tests and Test Paths

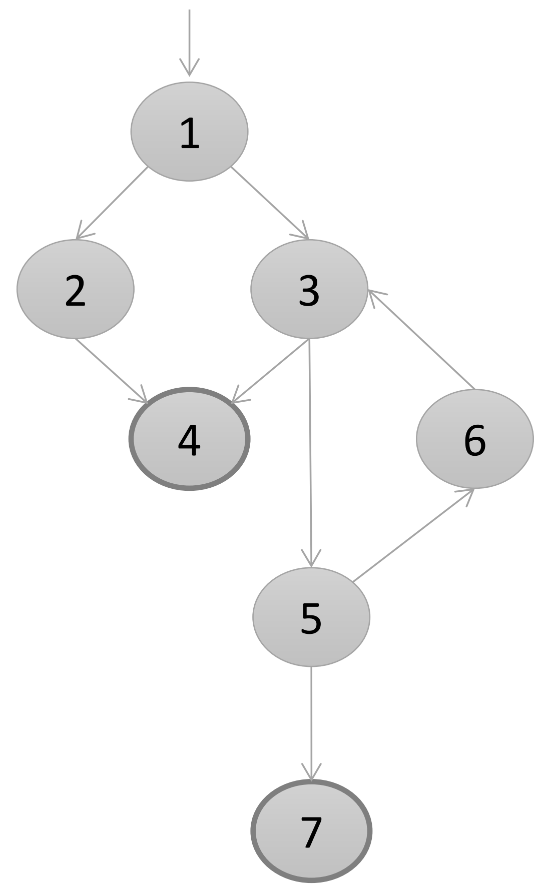
- path (t): The test path executed by test t
- path (T): The set of test paths executed by the set of tests T

- Each test executes one and only one test path
  - Complete execution from a start node to an final node

In/After-Class Exercise #13

Answer the following questions for the given graph

1. Identify the cycle in the graph
2. Write all test paths that go through the cycle no more than once
3. Write one path in the graph that is not a test path
4. Write one test path in the graph
5. How many test paths are in the graph?



- You have ∞ minutes ☺, but now think about 5 minutes!
- Do the exercise individually/in groups (of 3)
- Upload your answer in Moodle.

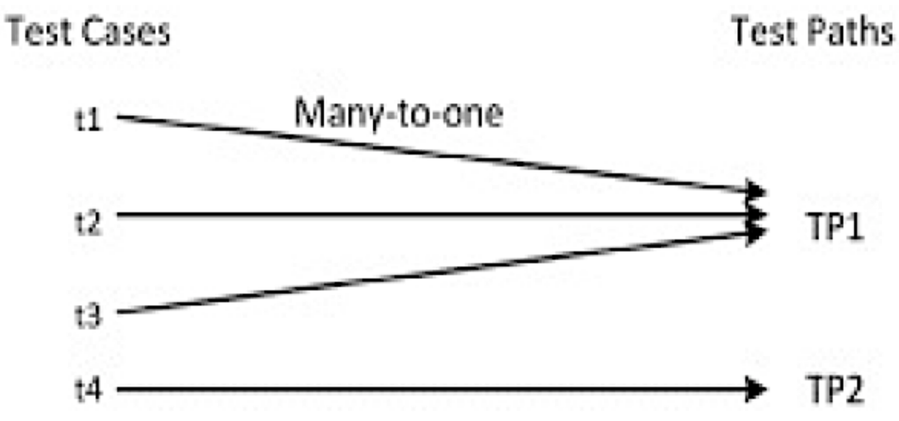
Tests and Test Paths

- path (t): The test path executed by test t
- path (T): The set of test paths executed by the set of tests T

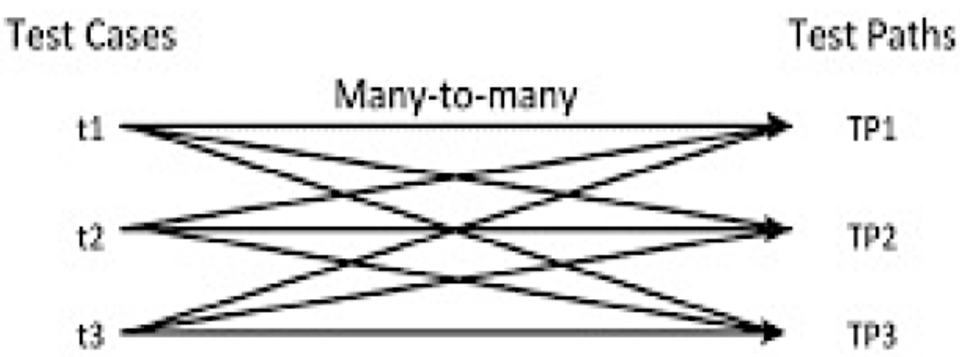
- Each test executes one and only one test path
  - Complete execution from a start node to an final node

Test Cases and Test Paths

Deterministic software–test always executes the same test path.



Non-deterministic software–the same test can execute different test paths



Reachability

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

break

GRAPH COVERAGE CRITERIA

Graph Coverage: Given a set ***TR*** of test requirements for a graph criterion ***C***, a test set ***T*** satisfies ***C*** on graph ***G*** if and only if for every test requirement ***tr*** in ***TR***, there is at least one test path ***p*** in ***path(T)*** such that ***p*** meets ***tr***.

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	**?**
Visits edges	**?**
Tours subpaths	**?**

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]

Node & Edge 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*.

Node & Edge 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*.

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

Node & Edge 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*.

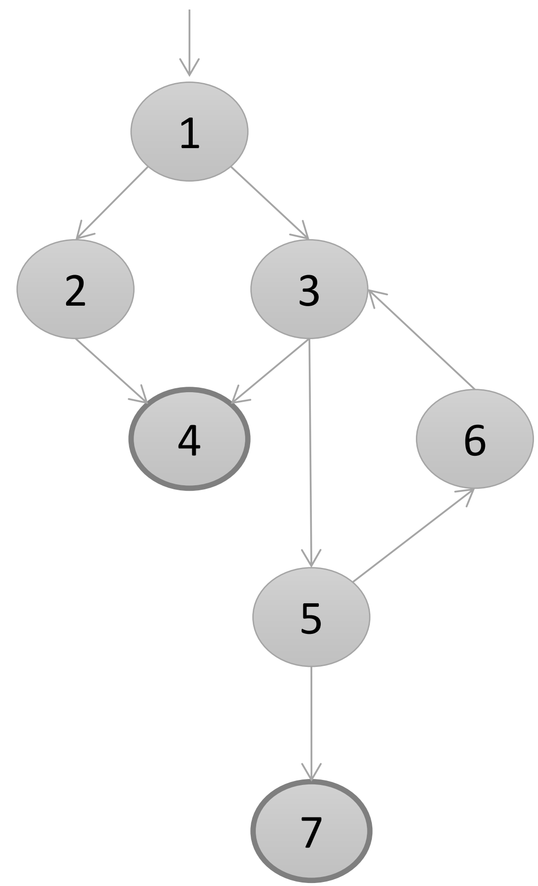
Node Coverage (NC): *TR* contains each reachable node in *G*.

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

In/After-Class Exercise #14

Answer the following questions for the given graph

1. List test paths that satisfy NC
2. List test paths that satisfy EC



- You have ∞ minutes ☺, but now think about 5 minutes!
- Do the exercise individually/in groups (of 3)
- Upload your answer in Moodle.

Node & Edge 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*.

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

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

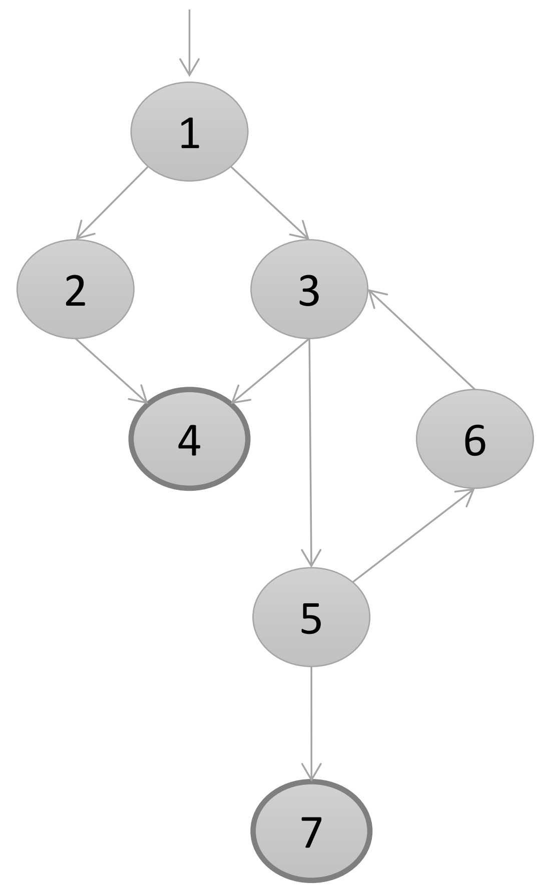
Covering Multiple Edges

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

In/After-Class Exercise #14

Answer the following questions for the given graph

3. Write the set of test requirements for EPC
4. List test paths that satisfy EPC



- You have ∞ minutes ☺, but now think about 5 minutes!
- Do the exercise individually/in groups (of 3)
- Upload your answer in Moodle.

Covering Multiple Edges

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

Complete Path Coverage (CPC): *TR* contains all paths in *G.*

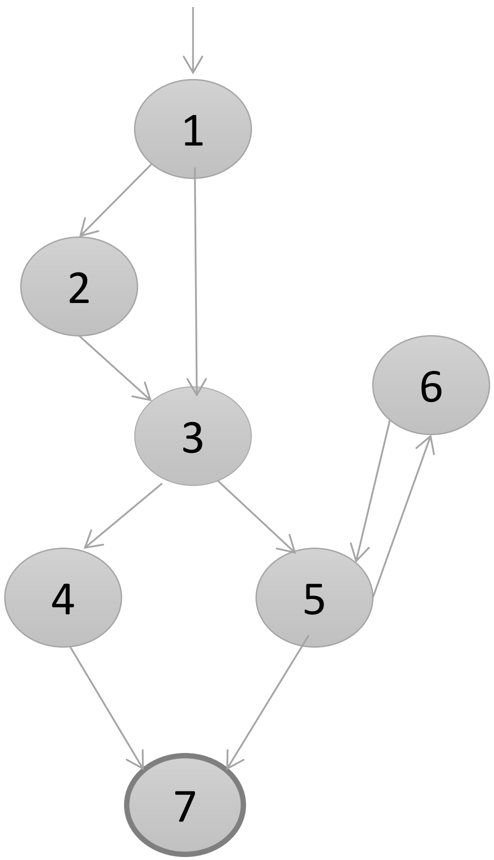
Covering Multiple Edges

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

Complete Path Coverage (CPC): *TR* contains all paths in *G*.

Specified Path Coverage (SPC): *TR* contains a set of *S* test paths, where *S* is supplied as parameter.

EPC and CPC Example



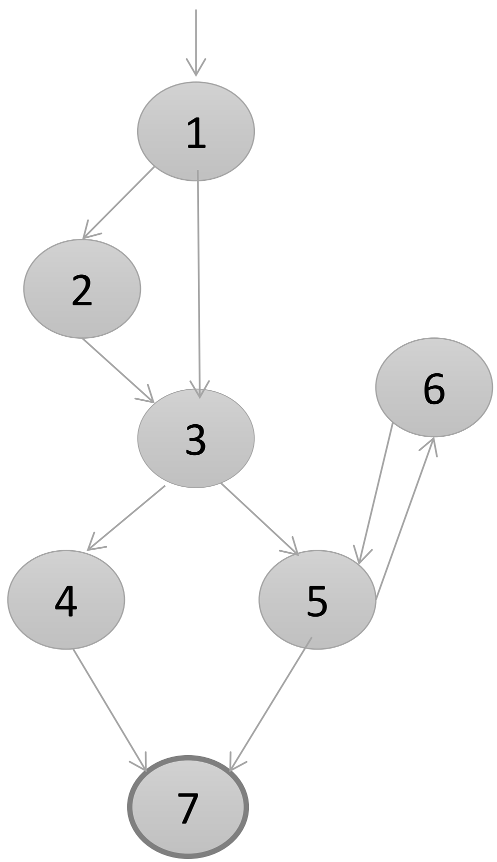
Edge-Pair Coverage

TR = { [1,2,3], [1,3,4], [1,3,5], [2,3,4], [2,3,5], [3,4,7],
[3,5,6], [3,5,7], [5,6,5], [6,5,6], [6,5,7]}

Test Paths: [1, 2, 3, 4, 7] [1, 2, 3, 5, 7] [1, 3, 4, 7]
[1, 3, 5, 6, 5, 6, 5, 7]

Complete Path Coverage

EPC and CPC Example



Edge-Pair Coverage

TR = { [1,2,3], [1,3,4], [1,3,5], [2,3,4], [2,3,5], [3,4,7],
[3,5,6], [3,5,7], [5,6,5], [6,5,6], [6,5,7]}

Test Paths: [1, 2, 3, 4, 7] [1, 2, 3, 5, 7] [1, 3, 4, 7]
[1, 3, 5, 6, 5, 6, 5, 7]

Complete Path Coverage

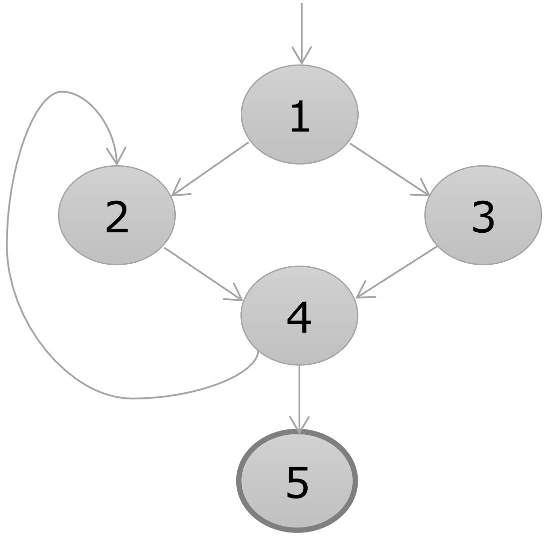
Test Paths: [1, 2, 3, 4, 7] [1, 2, 3, 5, 7] [1, 2, 3, 5, 6, 5, 7] [1, 2, 3, 5, 6, 5, 6, 5, 7] [1, 2, 3, 5, 6, 5, 6, 5, 6, 5, 7] …

How can we handle loops in graphs?

Simple Paths

Simple Path: A *path* from node n_i to n_j is simple if no node appears more than once, except possibly the first and last nodes are the same

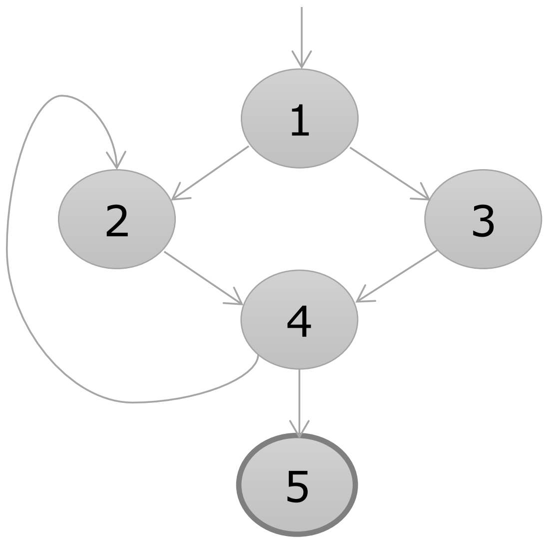
- No internal loops
- A loop is a simple path



Simple Paths: [1,2,4,5], [1,3,4,2], [1,3,4,5], [1,2,4], [1,3,4], [2,4,2], [2,4,5], [3,4,2], [3,4,5], [4,2,4], [1,2], [1,3], [2,4], [3,4], [4,2], [4,5], [1], [2], [3], [4], [5]

Prime Paths

Prime Path: A *simple path* that does not appear as a proper subpath of any other simple path.



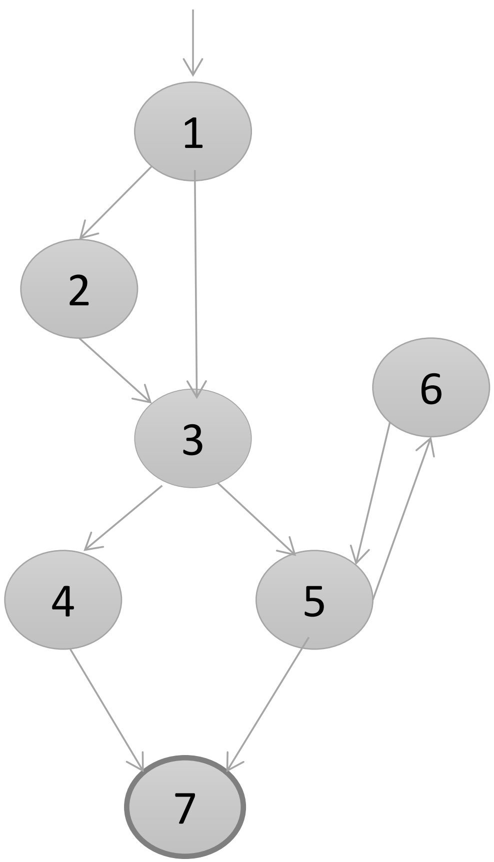
Simple Paths: [1,2,4,5], [1,3,4,2], [1,3,4,5], [1,2,4], [1,3,4], [2,4,2], [2,4,5], [3,4,2], [3,4,5], [4,2,4], [1,2], [1,3], [2,4], [3,4], [4,2], [4,5], [1], [2], [3], [4], [5]

Prime Paths: [1,2,4,5], [1,3,4,2], [1,3,4,5], [2,4,2], [4,2,4]

Prime Path Coverage

Prime Path Coverage (PPC): *TR* contains each prime path in *G.*

Prime Path Example



Prime Paths

[1, 2, 3, 4, 7]

[1, 2, 3, 5, 7]

[1, 2, 3, 5, 6]

[1, 3, 4, 7]

[1, 3, 5, 7]

[1, 3, 5, 6]

[6, 5, 7]

[6, 5, 6]

[5, 6, 5]

Execute loop 0 times

Execute loop once

Execute loop once

Prime Path Coverage

Prime Path Coverage (PPC): *TR* contains each prime path in *G.*

- Does PPC subsume NC and EC?
- Does PPC subsume EPC?

Prime Path Coverage

Prime Path Coverage (PPC): *TR* contains each prime path in *G.*

- Does PPC subsume NC and EC?
- Does PPC subsume EPC?
  - If a node *n* has an edge to itself (*self edge*), EPC requires [*n, n, m*] and [*m*, *n, n*]
  - Neither [*n, n, m*] nor [*m*, *n, n*] are simple paths (not prime)
  - ….

Touring, sidetrips, and detours

Tour: *A test path p tours subpath q if q is a subpath of p.*

Tour With Sidetrips: A test path p tours subpath q with *sidetrips* iff every edge in q is also in p in the same order

- The tour can include a sidetrip, as long as it comes back to the same node

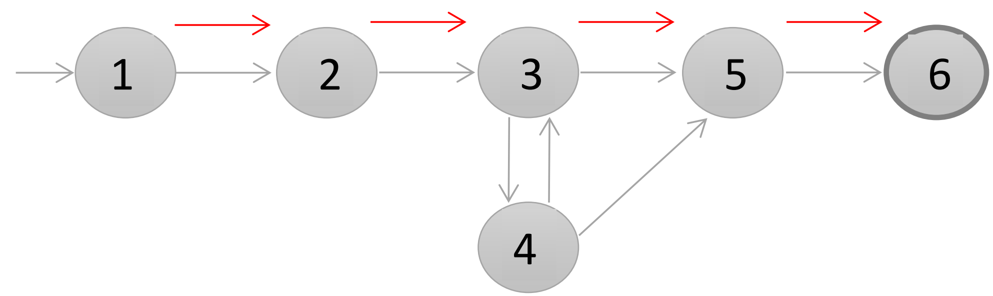
Tour With Detours: A test path p tours subpath q with *detours* iff every node in q is also in p in the same order

- The tour can include a detour from node *ni*, as long as it comes back to the prime path at a successor of *ni*

Touring, sidetrips, and detours

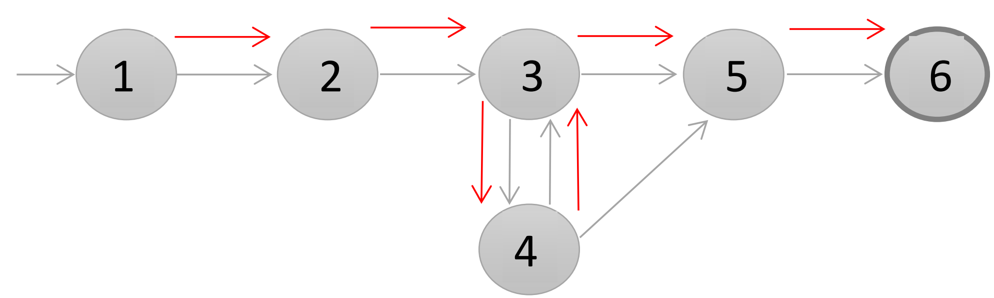
Prime Path: [1, 2, 3, 5, 6]

Touring



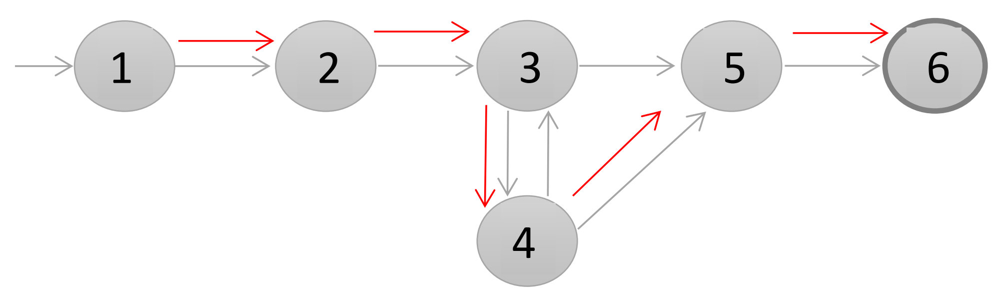
Touring

with sidetrip



Touring

with detour



Dealing with Infeasible Test Requirements

- Satisfy as many test requirements as possible **without sidetrips**
- Allow **sidetrips** to try to **satisfy remaining** test requirements

“Best Effort Touring”

Graph Coverage Criteria Subsumption