CS 5154: Software Testing

Dealing with Loops in Graph Coverage

Instructor: Owolabi Legunsen

Fall 2021

Handling Loops in Graph-Based Testing

- If a graph contains a loop, it has an infinite number of paths
- So, Complete Path Coverage is not feasible
- There have been very many attempts to “handle” loops:
  - 1970s : Execute cycles at least once
  - 1980s : Execute each loop, exactly once
  - 1990s : Execute loops 0 times, once, more than once
  - 2000s : Prime paths (touring, sidetrips, and detours)

A Prime Path is a Simple Path

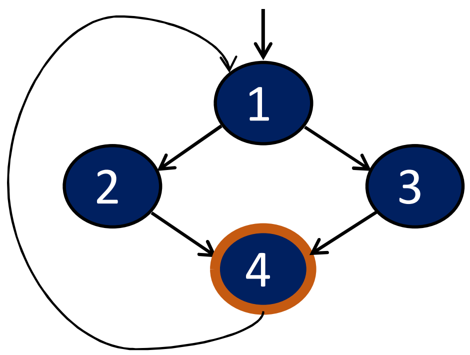
- Simple Path : *A path from node n_i to n_j is simple if no node appears more than once, except possibly the first and last nodes are the same*
    - A simple path has no internal loops
    - A loop is a simple path

Prime Paths defined

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

Example on Simple Paths and Prime Path

Write down three simple paths and three prime paths for this graph



Simple Paths:

[1,2,4,1], [1,3,4,1], [2,4,1,2], [2,4,1,3], [3,4,1,2], [3,4,1,3], [4,1,2,4], [4,1,3,4], [1,2,4], [1,3,4], [2,4,1], [3,4,1], [4,1,2], [4,1,3], [1,2], [1,3], [2,4], [3,4], [4,1], [1], [2], [3], [4]

Prime Paths:

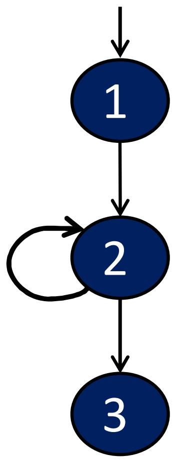
[2,4,1,2], [2,4,1,3], [1,3,4,1], [1,2,4,1], [3,4,1,2], [4,1,3,4], [4,1,2,4], [3,4,1,3]

Prime Path Coverage (PPC)

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

- PPC requires loops to be executed as well as skipped
- Tests that satisfy PPC will tour all paths of length 0, 1, …
  - That is, PPC subsumes node and edge coverage
- But does PPC subsume edge pair coverage?

Does PPC subsume Edge Pair Coverage?



EPC Test Requirements : ?

TR = { [1,2,3], [1,2,2], [2,2,3], [2,2,2] }

PPC Test Requirements : ?

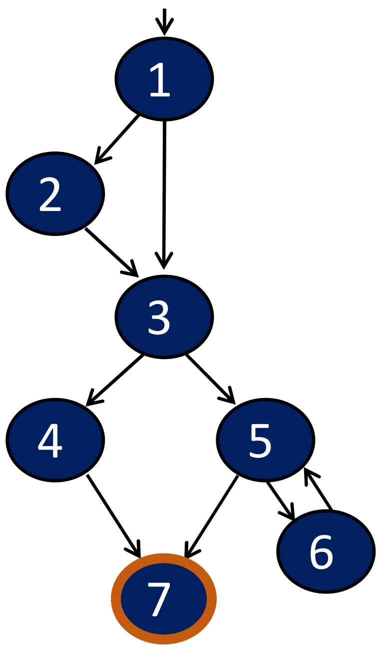
TR = { [1,2,3], [2,2] }

PPC does not subsume Edge Pair Coverage

- If a node *n* has a self-edge, EPC requires ([*m, n, n*] or [*n, n, m*]) and [*n, n, n*]

- [*n, n, m*], [*m, n, n*], and [*n, n, n*] are not simple paths (not prime)
  - Do you see why these are not simple paths?

Finding Prime Paths in a Graph

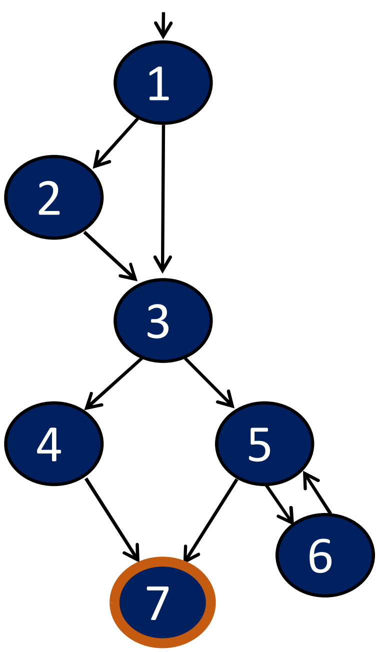


<u>Prime Paths</u>

[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]

How would you go about finding all the prime paths?

Observations about Prime Paths



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]

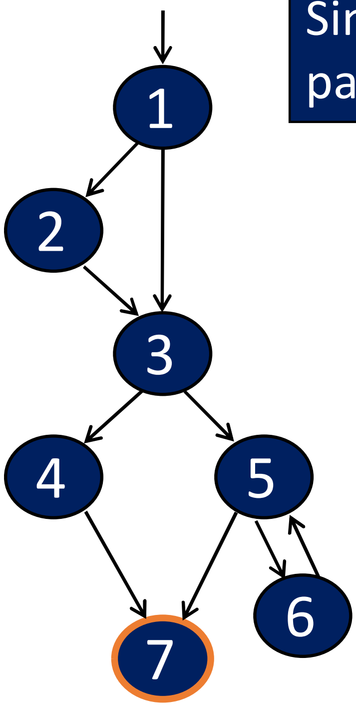
Execute loop 0 times

Execute loop once

Execute loop more than once

Illustrating an algorithm for finding Prime Paths

"!" Means "cannot be extended to a simple path"



Simple paths

Len 0
[1]
[2]
[3]
[4]
[5]
[6]
[7] !

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

Len 2
[1, 2, 3]
[1, 3, 4]
[1, 3, 5]
[2, 3, 4]
[2, 3, 5]
[3, 4, 7] !
[3, 5, 7] !
[3, 5, 6] !
[5, 6, 5] *
[6, 5, 7] !
[6, 5, 6] *

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

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

'*' means path cycles

Prime Paths ?

Any questions

?

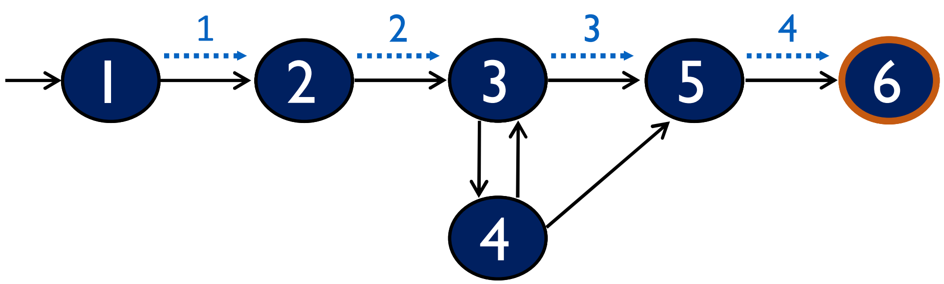
More on prime path algorithm

- There is another example in the textbook (reading 3)

- Implementing this algorithm used to be a homework question

Tension: test paths vs prime paths

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

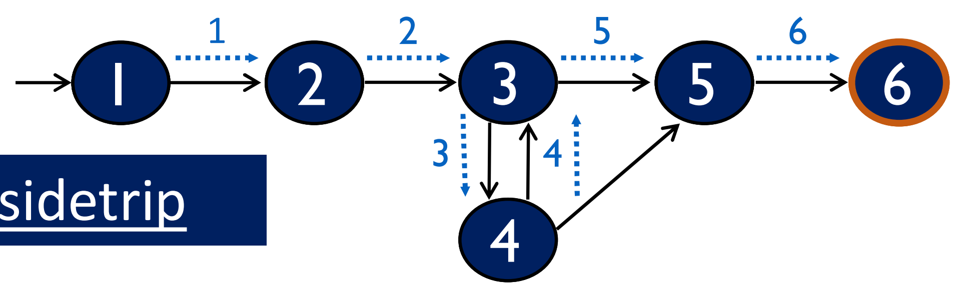


Does the test path [1, 2, 3, 4, 3, 5, 6] tour the prime path [1, 2, 3, 5, 6] ?

We relax the definition of “tour” in two ways

Touring with Sidetrips

- Tour With Sidetrips : *A test path p tours subpath q with sidetrips if and only if every edge in q is also in p in the same order*
- Tour can have a sidetrip if it comes back to the same node



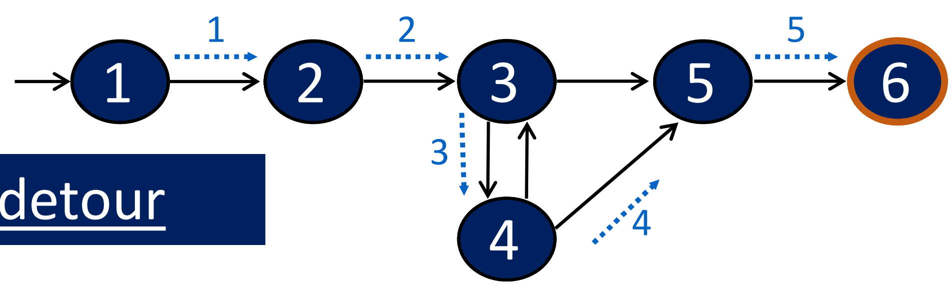
Touring with a <u>sidetrip</u>

Touring with Detours

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

- Tour can have a detour from node n_i, if it returns to the prime path at a successor of n_i

Touring with a detour



How to handle infeasible test requirements?

- Drop infeasible *tr* from TR

- Replace infeasible *tr* with less stringent TR

- Thoughts?

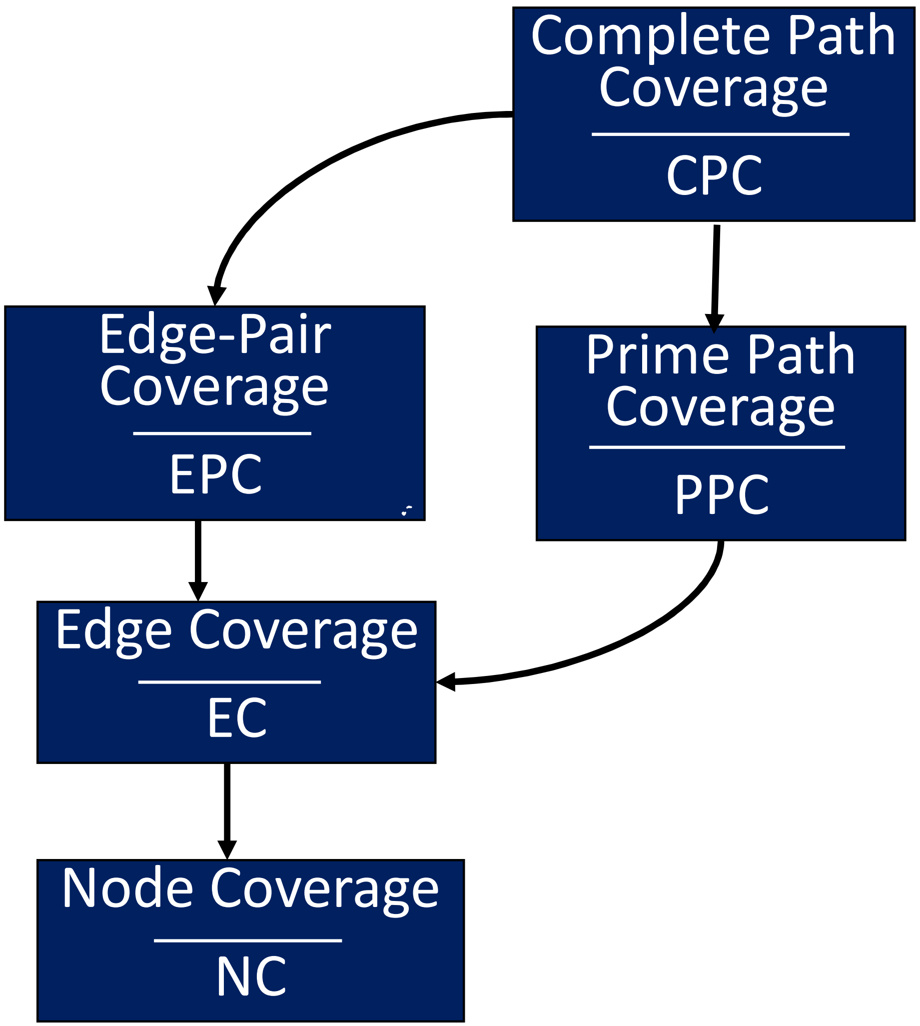
Are sidetrips and detours useful for testing?

- Without sidetrips, there are many more infeasible test requirements
- But allowing sidetrips “weakens” the criteria ☹
- So, what should be do?

Best Effort Touring

- First, satisfy as many test requirements as possible without sidetrips
- Then, allow sidetrips to try to satisfy remaining test requirements
- (It is not clear yet if detours help as much.)

Subsumption among Graph coverage criteria



What we have seen so far

- Prime Paths as one way to deal with loops in Graph-based MDTD
- An algorithm for computing the prime paths in a graph
- Best effort touring for dealing infeasible test requirements
- We worked entirely at the “design abstraction level”

Next

- Applying Graph-Based MDTD to source code