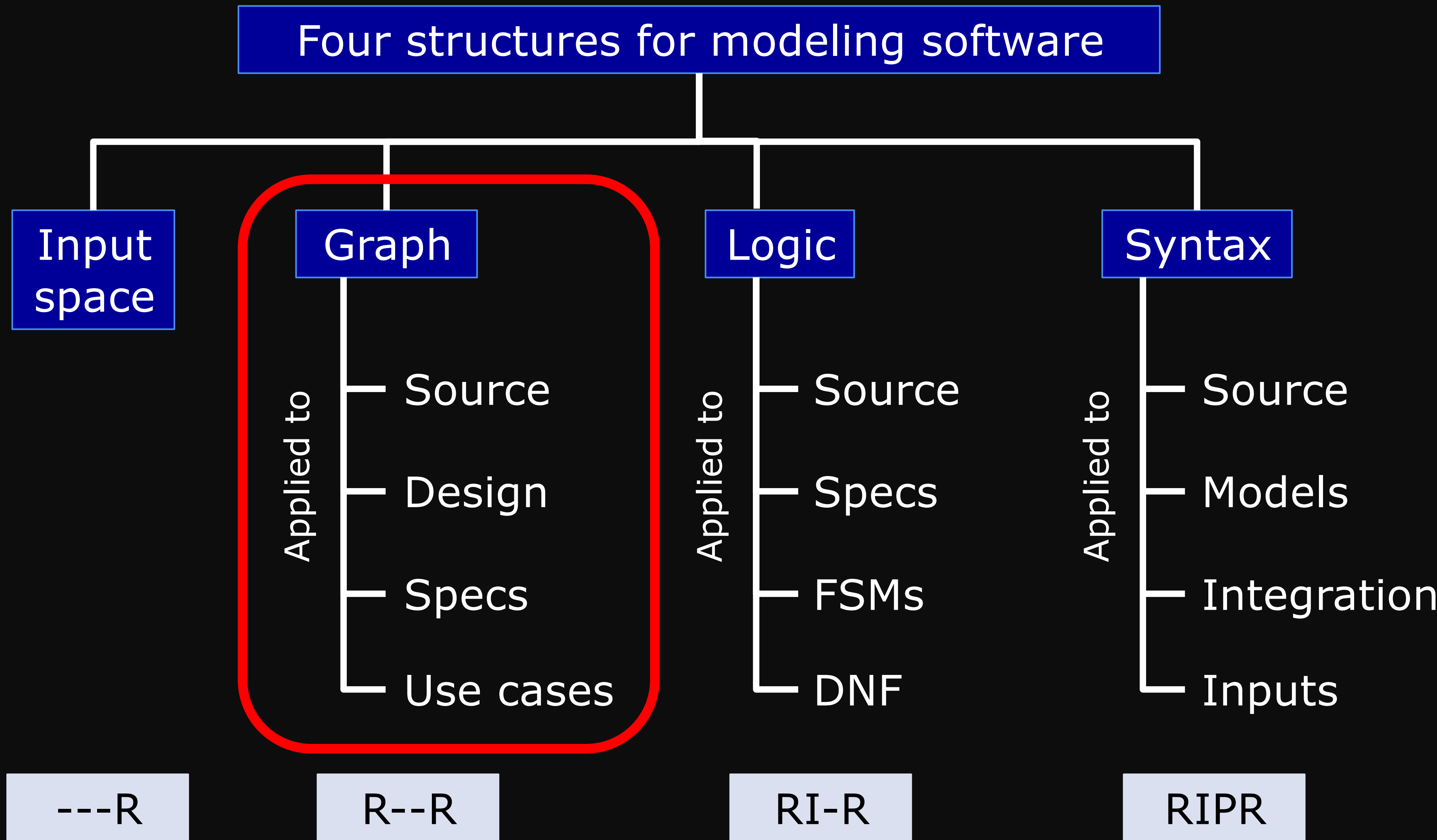
Graph: Structural Coverage Criteria

CS 3250 Software Testing

[Ammann and Offutt, "Introduction to Software Testing," Ch. 7]

Structures for Criteria-Based Testing



Today's Objectives

- Understand how to use graph to define criteria and design tests
  - Node coverage (NC)
  - Edge coverage (EC)
  - Edge-pair coverage (EPC)
  - Complete Path Coverage (CPC)
  - Prime Path Coverage (PPC)
    - Simple paths and prime paths
- Touring, sidetrips, and detours
- Dealing with infeasible test requirements
- Graph derived from various software artifacts (coming soon)

Graph Coverage Criteria

Graph coverage criteria define test requirements TR in terms of properties of test paths in a graph G

Test criterion – rules that define test requirements

Test requirements (TR) – Describe properties of test paths

Steps:

1. Develop a model of the software as a graph
2. A test requirement is met by visiting a particular node or edge or by touring a particular path

Graph Coverage Criteria

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*

Two types

1. Structural coverage criteria
    - Define a graph just in terms of nodes and edges
2. Data flow coverage criteria
    - Requires a graph to be annotated with references to variables

Graph Coverage Criteria

Structural Coverage Criteria

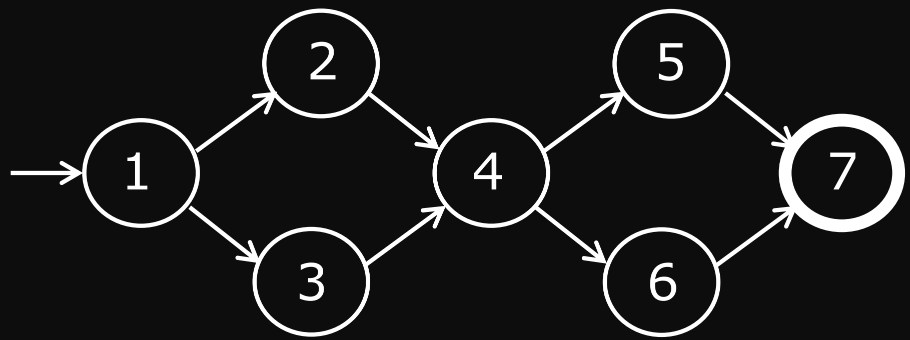
- Node Coverage (NC)
  - Statement coverage
- Edge Coverage (EC)
  - Branch coverage
- Edge-Pair Coverage (EPC)
- Complete Path Coverage (CPC)
- Prime Path Coverage (PPC)

Data Flow Coverage Criteria

- All-Defs Coverage (ADC)
- All-Uses Coverage (AUC)
- All-du-Paths Coverage (ADUPC)

Node Coverage (NC)

NC: TR contains each reachable node in G



Node *N* = {1, 2, 3, 4, 5, 6, 7}

Edge *E* = {(1,2), (1,3), (2,4), (3,4), (4,5), (4,6), (5,7), 6,7)}

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

Test path *p1* = [1, 2, 4, 5, 7]

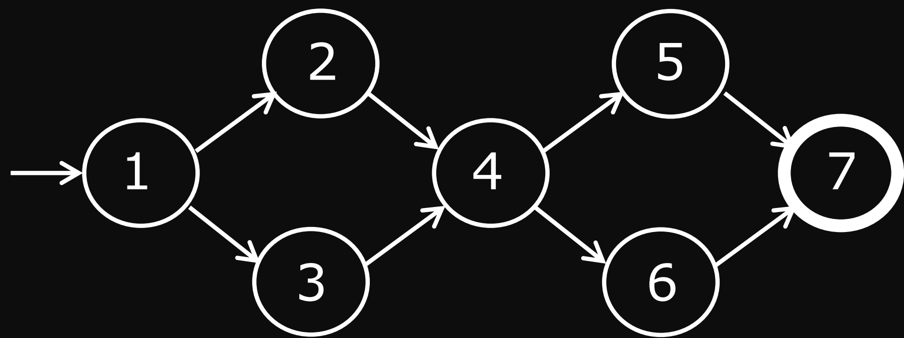
Test path *p2* = [1, 3, 4, 6, 7]

If a test set *T* = {*t1*, *t2*}, where path(*t1*) = *p1* and path(*t2)* = *p2*, then *T* satisfies Node Coverage on G

Edge Coverage (EC)

EC: TR contains each reachable path of length up to 1, inclusive, in G

"length up to 1" – allows for graphs with one node and no edges



Node N = {1, 2, 3, 4, 5, 6, 7}

Edge E = {(1,2), (1,3), (2,4), (3,4), (4,5), (4,6), (5,7), (6,7)}

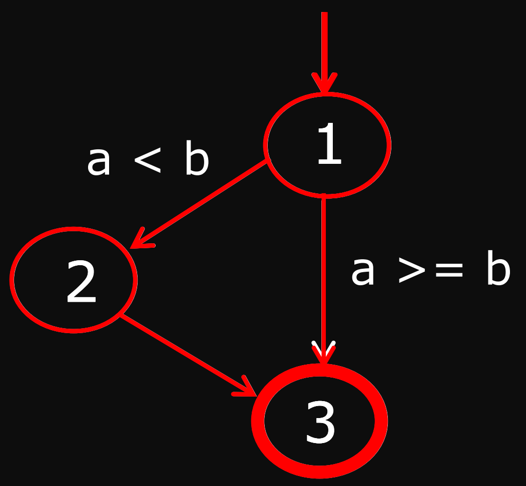
TR = {(1,2), (1,3), (2,4), (3,4), (4,5), (4,6), (5,7), (6,7)}

Test path *p1* = [1, 2, 4, 5, 7]

Test path *p2* = [1, 3, 4, 6, 7]

If a test set *T* = {*t1*, *t2*}, where path(*t1*) = *p1* and path(*t2*) = *p2*, then *T* satisfies Edge Coverage on G

Difference between NC and EC



Node *N* = {1, 2, 3}

Edge *E* = {(1,2), (1,3), (2,3)}

NC: TR = {1, 2, 3}

Test path = [1, 2, 3]

EC: TR = {(1,2), (1,3), (2,3)}

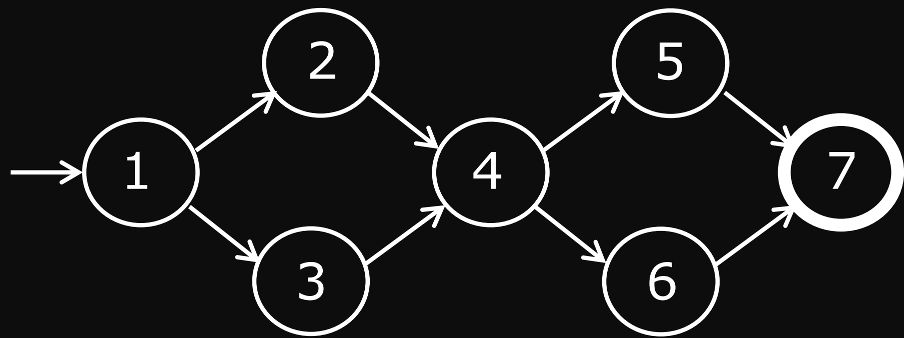
Test paths = [1, 2, 3], [1, 3]

NC and EC are only different when there is an edge and another subpath between a pair of nodes (as in an “if-else” statement)

Edge-Pair Coverage (EPC)

EPC: TR contains each reachable path of length up to 2, inclusive, in G

"length up to 2" – allows for graphs that have 0, 1, or 2 edges



Node N = {1, 2, 3, 4, 5, 6, 7}

Edge E = {(1,2), (1,3), (2,4), (3,4), (4,5), (4,6), (5,7), (6,7)}

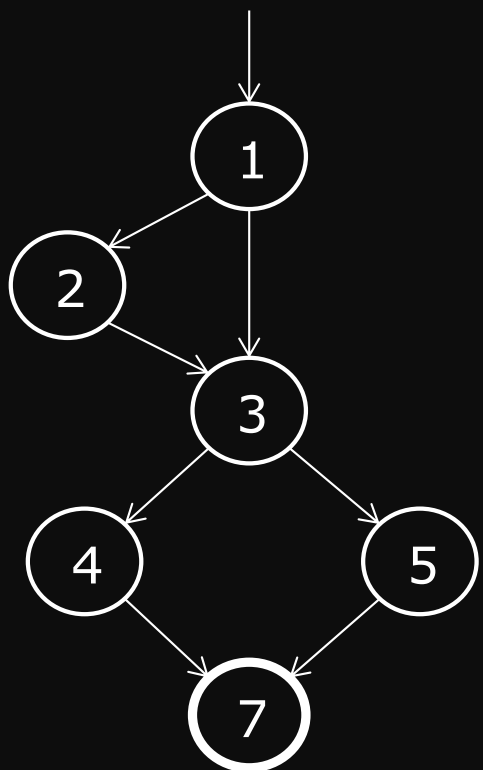
TR = {(1,2,4), (1,3,4),
(2,4,5), (2,4,6),
(3,4,5), (3,4,6),
(4,5,7), (4,6,7)}

Test path *p1* = [1, 2, 4, 5, 7]
Test path *p2* = [1, 3, 4, 5, 7]
Test path *p3* = [1, 2, 4, 6, 7]
Test path *p4* = [1, 3, 4, 6, 7]

EPC requires pairs of edges, or subpaths of length 2
– covering multiple edges

Complete Path Coverage (CPC)

CPC: TR contains all paths in G



Node N = {1, 2, 3, 4, 5, 6, 7}

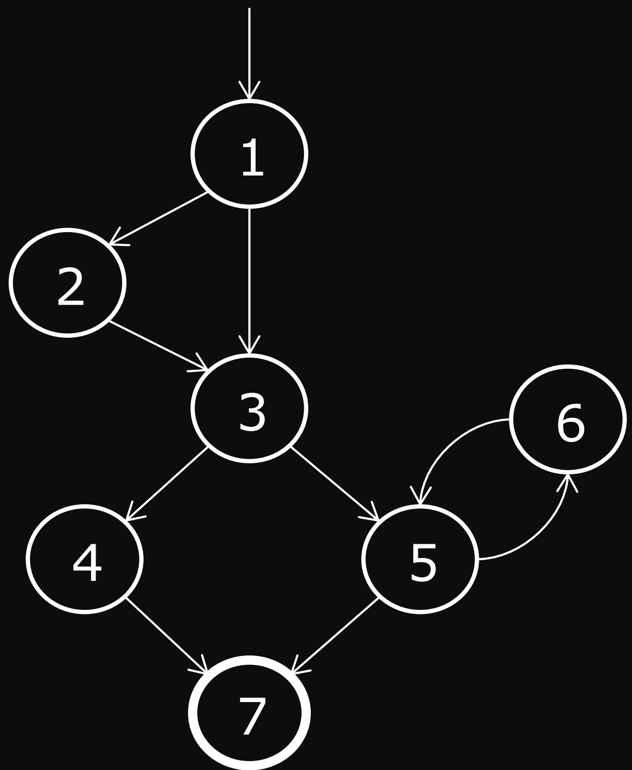
Edge E = {(1,2), (1,3), (2,3), (3,4), (3,5), (4,7), (5,7)}

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

List all test paths:

- Test path *p1* = [1, 2, 3, 4, 7]
- Test path *p2* = [1, 2, 3, 5, 7]
- Test path *p3* = [1, 3, 4, 7]
- Test path *p4* = [1, 3, 5, 7]

CPC: Graph with Loop



Node *N* = {1, 2, 3, 4, 5, 6, 7}

Edge *E* = {(1,2), (1,3), (2,3), (3,4), (3,5), (4,7), (5,7), (5,6), (6,5)}

List all test paths:

[1, 2, 3, 4, 7], [1, 2, 3, 5, 7],
[1, 3, 4, 7], [1, 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],
...

Impossible if a graph has a loop
≈ infinite number of paths
≈ infinite number of test requirements

Handling Loops in Graphs

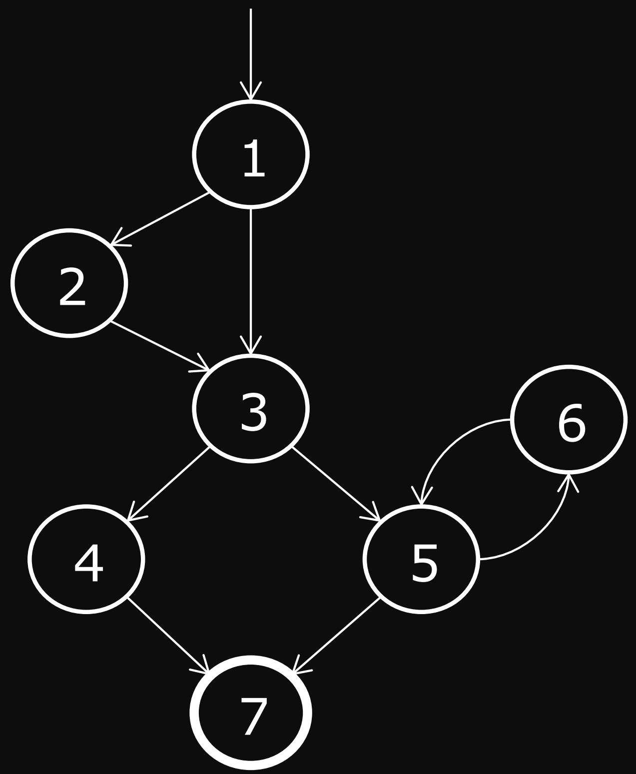
Attempts to deal with loops:

- 1970s: Execute cycles once ([5, 6, 5] in previous example)
- 1980s: Execute each loop, exactly once
- 1990s: Execute loops 0 times, once, more than once
- 2000s: Prime paths (touring, sidetrips, and detours)

Simple Paths

Path from node n_i to n_j that has no internal loops

- A loop is a simple path



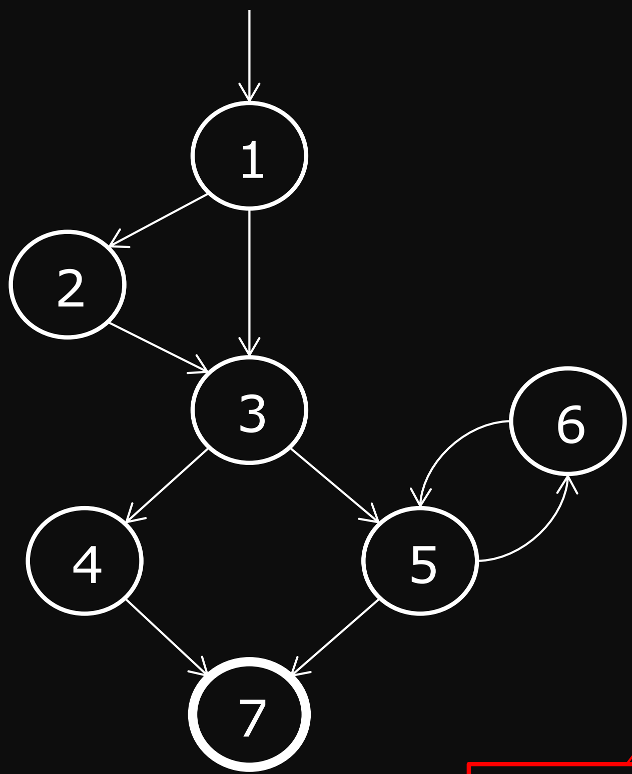
List simple paths: 31 simple paths

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

Subpaths of other simple paths → avoid these

Prime Paths

Simple path that is not subpath of any other simple path



List prime paths: 9 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],

[5,6,5],

[6,5,6], [6,5,7]

Execute loop 0 time

Execute loop once

Execute loop more than once

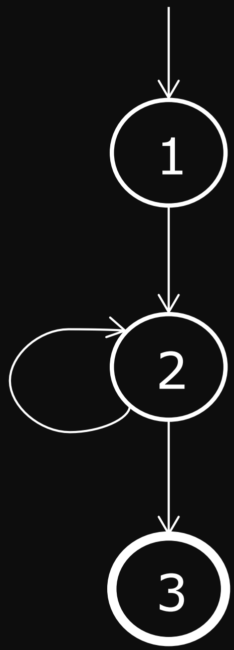
Prime Path Coverage (PPC)

PPC: TR contains each prime path in graph G

- Keep the number of test requirements down
- For a given infeasible prime path that consists of some feasible simple paths, replace the infeasible prime path with relevant feasible subpaths

Note on PPC

- PPC does not subsume EPC
- If a node *n* has an edge to itself ("self edge"), EPC requires [*n*, *n*, *m*] and [*m*, *n*, *n*]
- [*n*, *n*, *m*] and [*m*, *n*, *n*] are not simple paths (prime paths)



List EPC requirements:

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

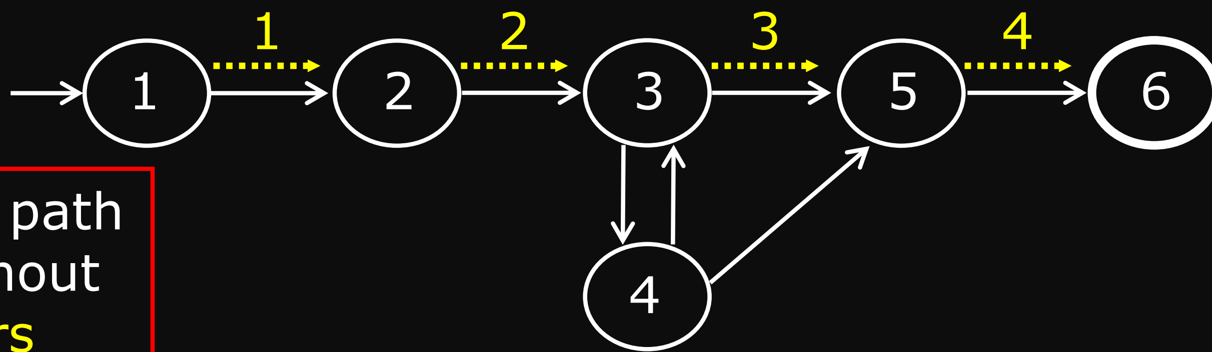
List PPC requirements:

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

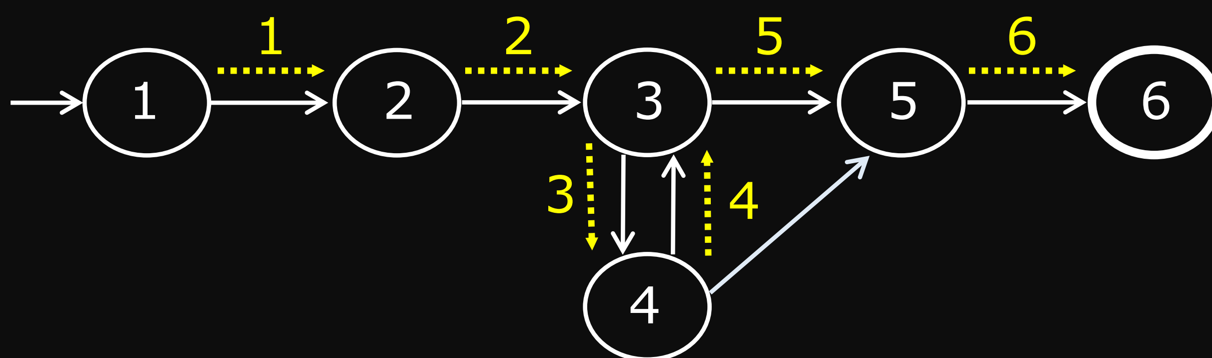
Touring, Sidetrips, and Detours

"direct tour"

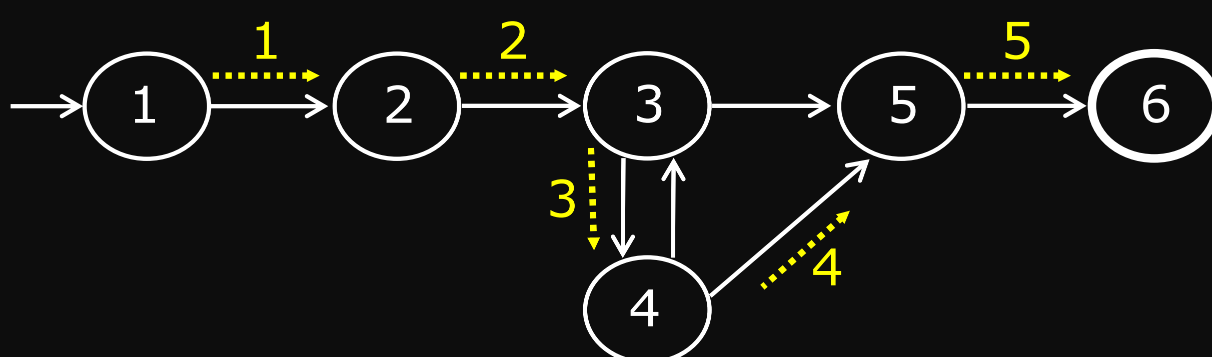
Touring the prime path [1, 2, 3, 5, 6] without sidetrips or detours



Touring with a sidetrip



Touring with a detour



[AO, Figures 7.8, 7.9]

Infeasible Test Requirements

- An infeasible test requirement cannot be satisfied
  - Unreachable statement (dead code)
  - Subpath that can only be executed with a contradiction ($X > 0$ and $X < 0$)

- Most test criteria have some infeasible test requirements

- When sidetrips are not allowed, many structural criteria have more infeasible test requirements

- Always allowing sidetrips weakens the test criteria

Practical recommendation—Best Effort Touring

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

Graph Coverage Criteria Subsumption