

Introduction to Software Testing

(2nd edition)

Chapter 7.1, 7.2

Overview Graph Coverage Criteria

(active class version)

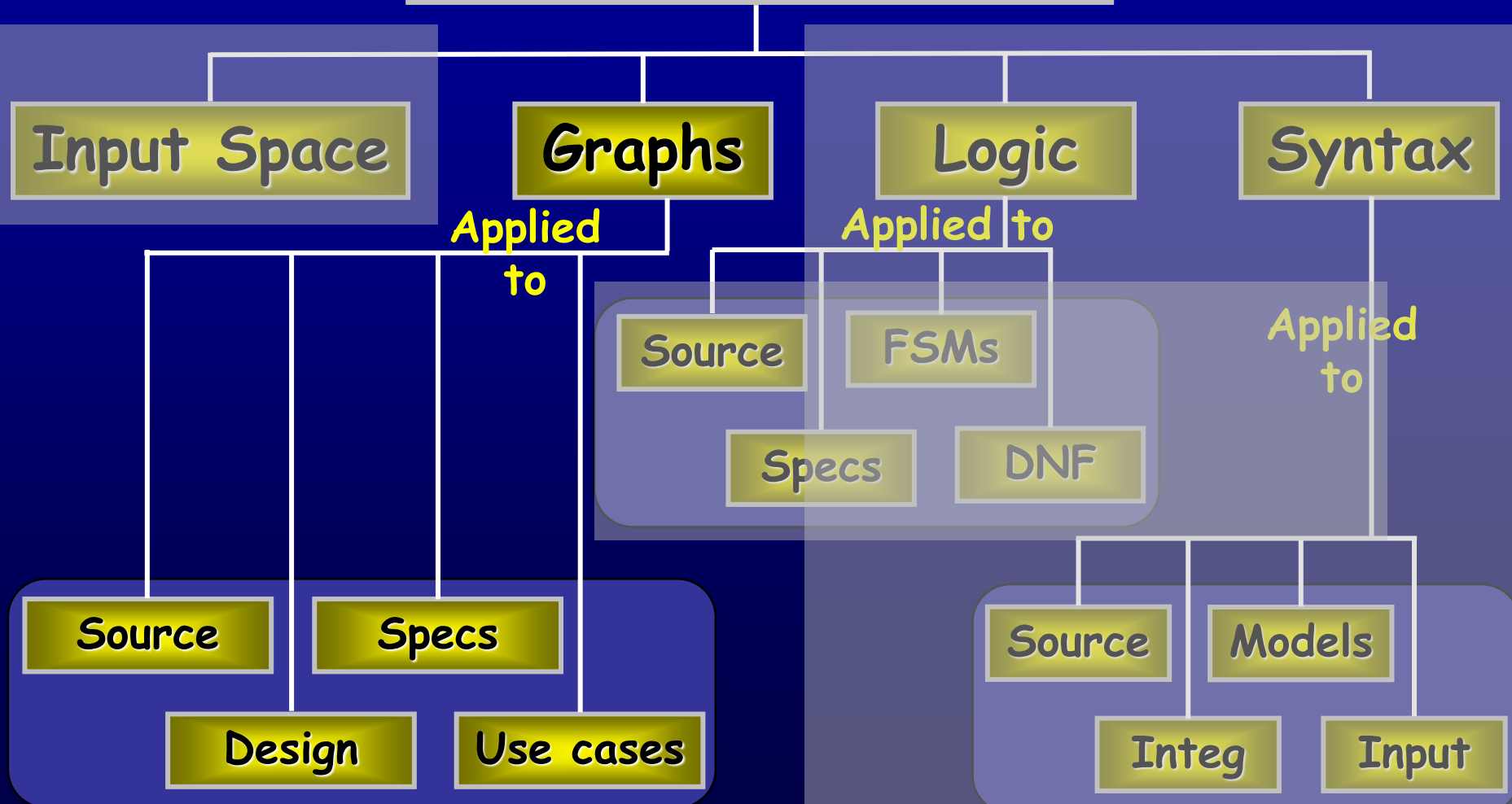
Paul Ammann & Jeff Offutt

<http://www.cs.gmu.edu/~offutt/softwaretest/>

Update, October 2016

Ch. 7 : Graph Coverage

Four Structures for Modeling Software



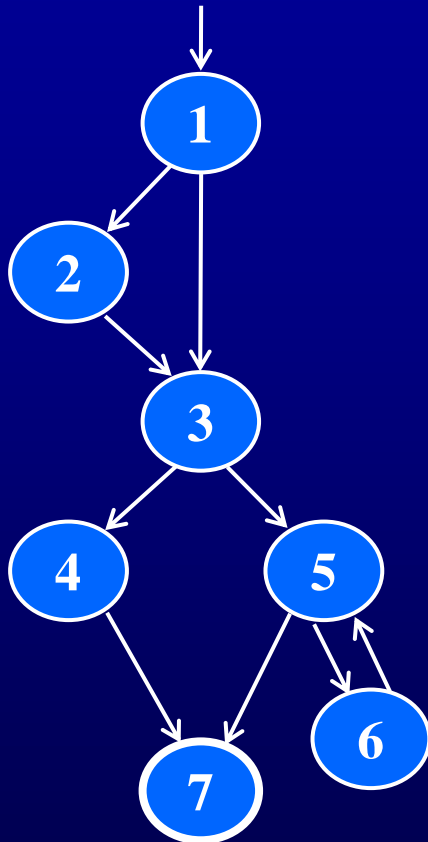
Covering Multiple Edges

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

Unfortunately, this is **impossible** if the graph has a loop, so a weak compromise makes the tester decide which paths:

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

Structural Coverage Example



Node Coverage

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

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

Write down
the TRs and
Test Paths for
these criteria

Edge Coverage

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

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

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

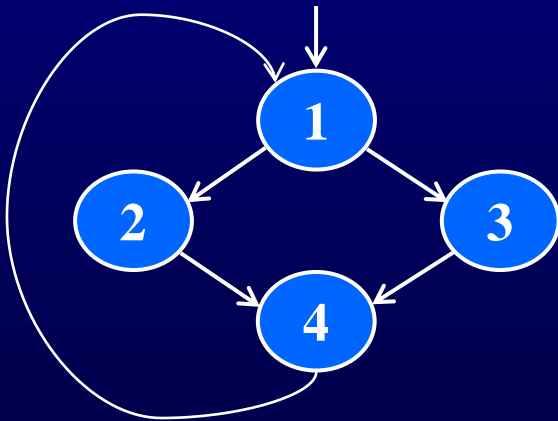
Test Paths: [1, 2, 3, 4, 7] [1, 2, 3, 5, 7] [1, 2, 3, 5, 6, 5, 7]
[1, 2, 3, 5, 6, 5, 6, 5, 7] [1, 2, 3, 5, 6, 5, 6, 5, 6, 5, 7] ...

Handling Loops in Graphs

- If a graph contains a loop, it has an **infinite** number of paths
- Thus, CPC is **not feasible**
- SPC is not satisfactory because the results are **subjective** and vary with the tester
- Attempts to “deal with” **loops**:
 - **1970s** : Execute cycles once ([4, 5, 4] in previous example, informal)
 - **1980s** : Execute each loop, exactly once (formalized)
 - **1990s** : Execute loops 0 times, once, more than once (informal description)
 - **2000s** : Prime paths (touring, sidetrips, and detours)

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



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

Write down the simple and prime paths for this graph

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

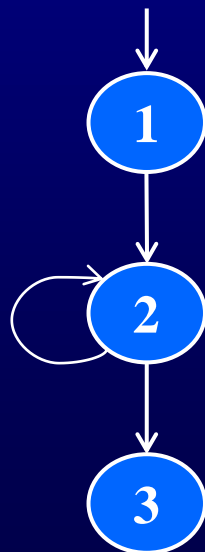
- A simple, elegant and finite criterion that requires **loops** to be executed as well as skipped

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

- Will tour all paths of length 0, 1, ...
- That is, it **subsumes** node and edge coverage
- PPC almost, but **not quite**, subsumes **EPC** ...

PPC Does Not Subsume EPC

- If a node n has an edge to itself (self edge), **EPC** requires $[n, n, m]$ and $[m, n, n]$
- $[n, n, m]$ is not prime
- Neither $[n, n, m]$ nor $[m, n, n]$ are simple paths (not prime)



EPC Requirements : ?

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

PPC Requirements : ?

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

PPC Does Not Subsume EPC

- If a node n has an edge to itself (self edge), **EPC** requires $[n, n, m]$ and $[m, n, n]$
- $[n, n, m]$ is not prime
- Neither $[n, n, m]$ nor $[m, n, n]$ are simple paths (not prime)

1,2,3
1,2,2
2,2,3
2,2,2



EPC Requirements : ?

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

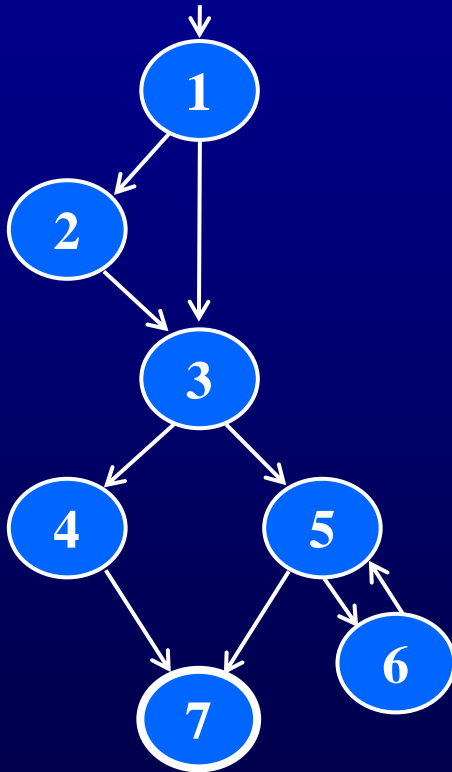
PPC Requirements : ?

1,2,3

2,2

Prime Path Example

- The previous example has 38 **simple** paths
- Only **nine** *prime paths*



Prime Paths

[1, 2, 3, 4, 7]

Write down all
9 prime paths

[1, 3, 4, 7]

[1, 3, 5, 7]

[1, 3, 5, 6]

[6, 5, 7]

[6, 5, 6]

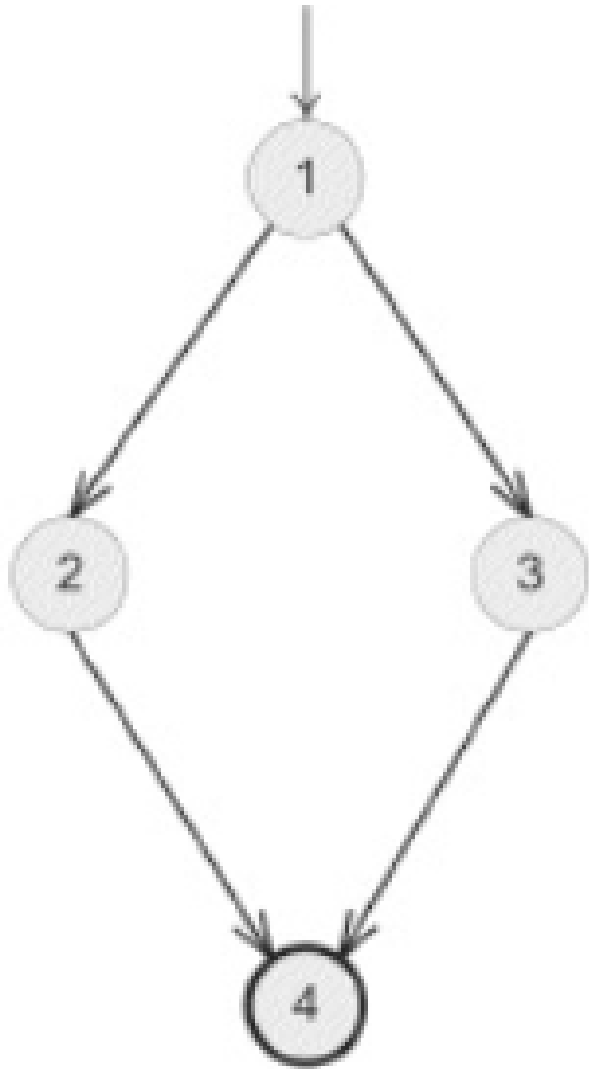
[5, 6, 5]

Execute
loop 0 times

Execute
loop once

Execute loop
more than once

Prime Path Coverage vs Complete Path Coverage



Prime Paths = $\{ [1, 2, 4], [1, 3, 4] \}$

$path(t_1) = [1, 2, 4]$

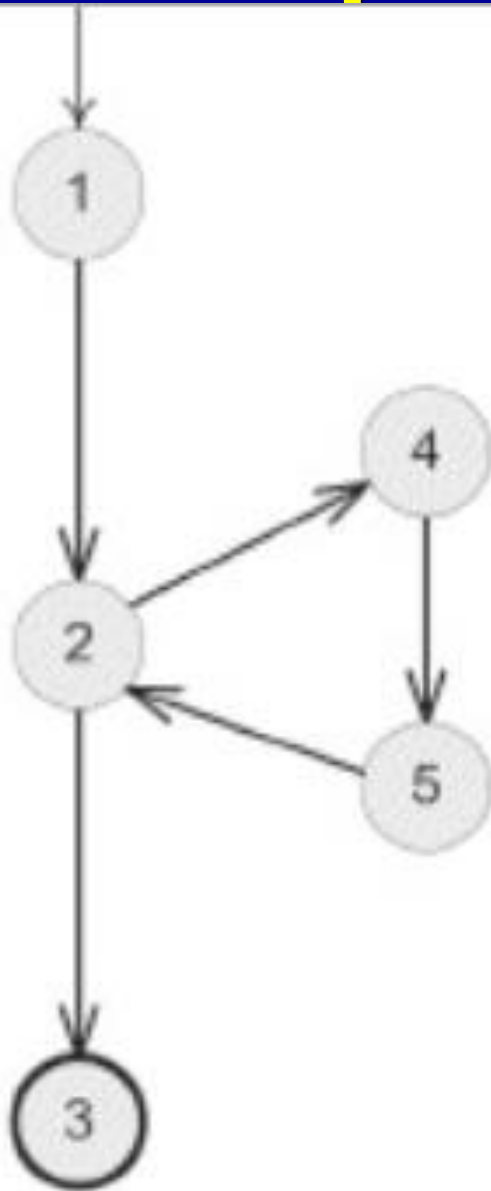
$path(t_2) = [1, 3, 4]$

$T_1 = \{t_1, t_2\}$

T_1 satisfies prime path coverage on the graph

(a) Prime Path Coverage on a Graph
With No Loops

Prime Path Coverage vs Complete Path Coverage



Prime Paths = { [1, 2, 3], [1, 2, 4, 5], [2, 4, 5, 2],
[4, 5, 2, 4], [5, 2, 4, 5], [4, 5, 2, 3] }

$path(t_3) = [1, 2, 3]$

$path(t_4) = [1, 2, 4, 5, 2, 4, 5, 2, 3]$

$T_2 = \{t_3, t_4\}$

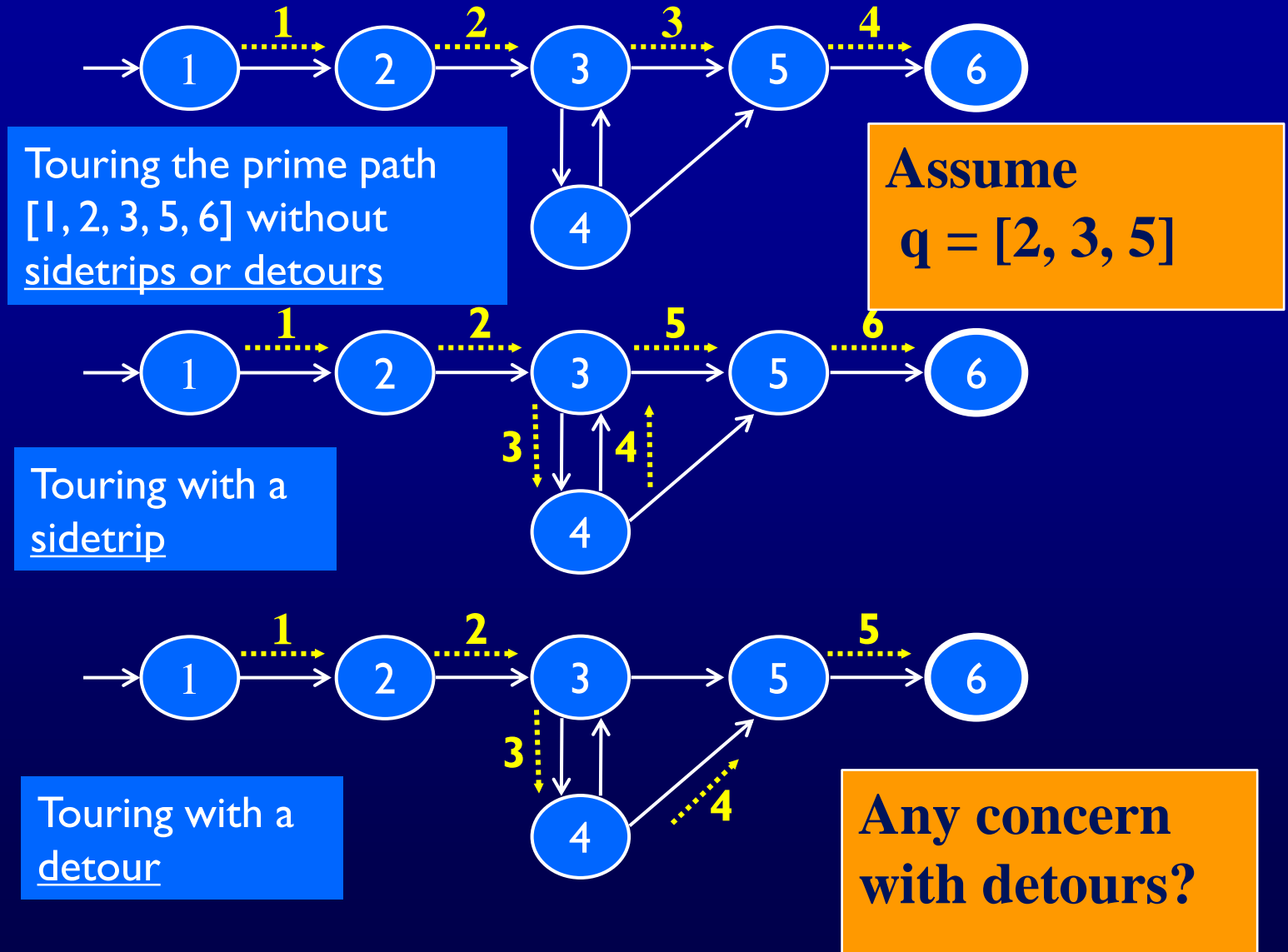
T_2 satisfies prime path coverage on the graph

(b) Prime Path Coverage on a Graph
With Loops

Touring, Sidetrips, and Detours

- Prime paths do not have **internal loops**
- Assume that q is a **simple path**. Test paths might
- **Tour (directly)** : 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
- **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

Sidetrips and Detours Example



Prime Path coverage : 0 or more iterations

```
print( Say hello! );  
for (i=0;i<5;i++)  
    print Hii;  
  
print Bye
```

```
print (Say Hello)  
  
int = 0  
  
do{  
    print Hi  
} while (i< 5)  
  
print Bye
```

Infeasible Test Requirements

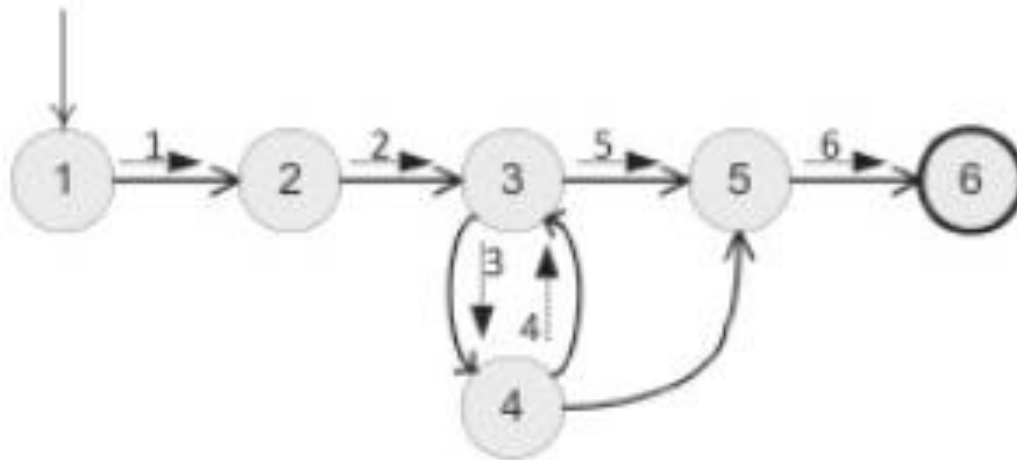
- An **infeasible** test requirement cannot be satisfied
 - Unreachable statement (dead code)
 - Subpath that can only be executed with a contradiction ($X > 0$ and $X < 0$)
- Most test **criteria** have some infeasible test requirements

```
If (false)
    unreachableCall();
```

```
If (x>0)
    if(x < 0)
        unreachableCall();
```


Infeasible Test Requirements

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



(a) Graph being toured with a sidetrip

- When do you need to tour this graph with side trips?
- When would side trips be a bad idea?

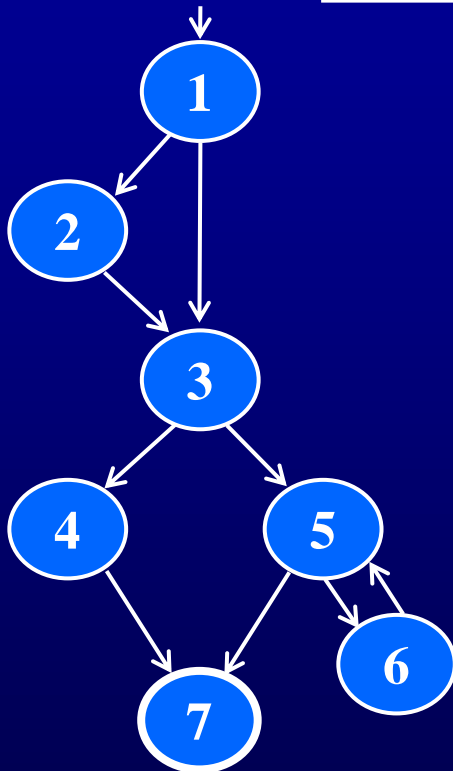
Refining Coverage Criteria

- We could define each graph coverage criterion and explicitly include the kinds of tours allowed, e.g.
 - prime paths, with direct tours;
 - prime paths, side-trips allowed;
 - prime paths, detours allowed.
- Detours seem less practical, so we do not include detours further.
- However, 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

Finding Prime Test Paths



Simple paths

Len 0
[1]

Write paths of length 0

[5]
[6]
[7]!

Len 1
[1, 2]
[1, 3]

Write paths of length 1

[4, 7]!
[5, 7]!
[5, 6]
[6, 5]

Len 2

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

Write paths of length 2

[2, 3, 5]!
[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]

Write paths of length 3

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

Len 4

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

Prime Paths?

Test paths?

Test paths need not be prime!

Required Reading

- Sections 7.1 and 7.2 from the text book: An Introduction to Software Testing, 2nd edition.