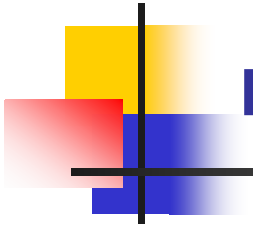
More On Paths

Supplement to Chapter 4, Graph Theory



Path definition

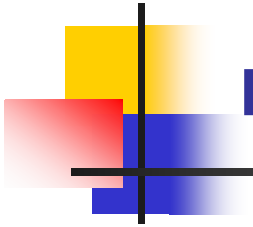
- **What is a path?**
  - We are in in a graph context

Path definition – 2

- **What is a path?**
  - **A path is a sequence of nodes $\langle n_1 , n_2 , \ldots , n_p \rangle$**
  - **Each adjacent pair of nodes is in the set of edges**

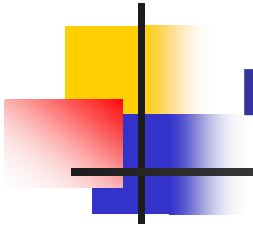
$$\forall j : 1..p-1 \bullet (n_j, n_{j+1}) \in E$$

- We say a path **visits** the nodes in the sequence



Path length definition

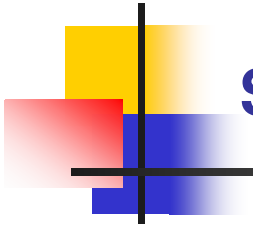
- What is the length of a path?



Path definition

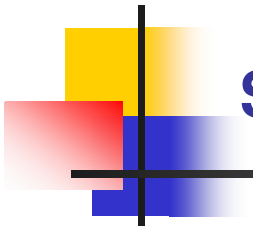
- What is the length of a path?
  - The number of edges it contains

    #path = #sequence – 1



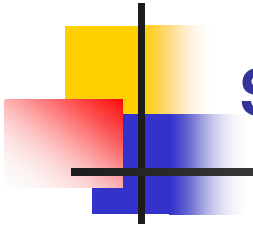
Simple path definition

- **What a simple path?**



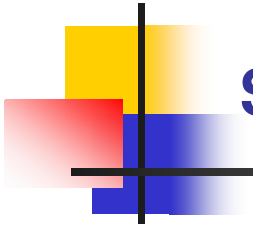
Simple path definition – 2

- **What a simple path?**
  - **A path from node n_j to node n_k is simple if**
    - **No node appears more than once**
      - There is no internal loop
    - **Exception the end points may be the same**
      - The entire path may form a loop



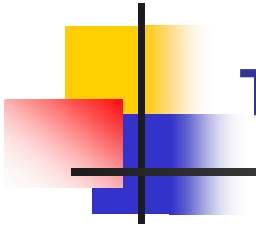
Simple path useful property

- **What useful property do simple paths have?**



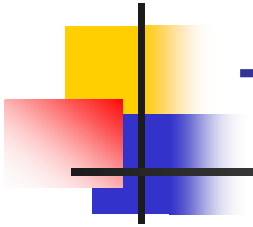
Simple path useful property – 2

- What useful property do simple paths have?
  - Any path is a composition of simple paths



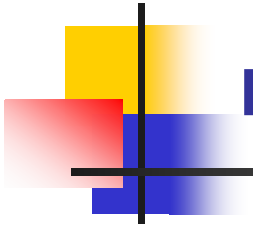
Test path definition

- **What is a test path?**



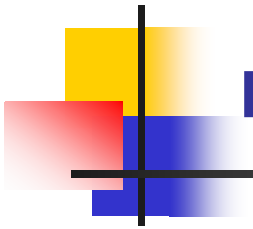
Test path definition – 2

- **What is a test path?**
  - **Starts at a starting node of a control flow graph**
  - **Terminates at an exit node of a control flow graph**
  - **Possibly of length zero**



Prime path definition

- **What is a prime path?**

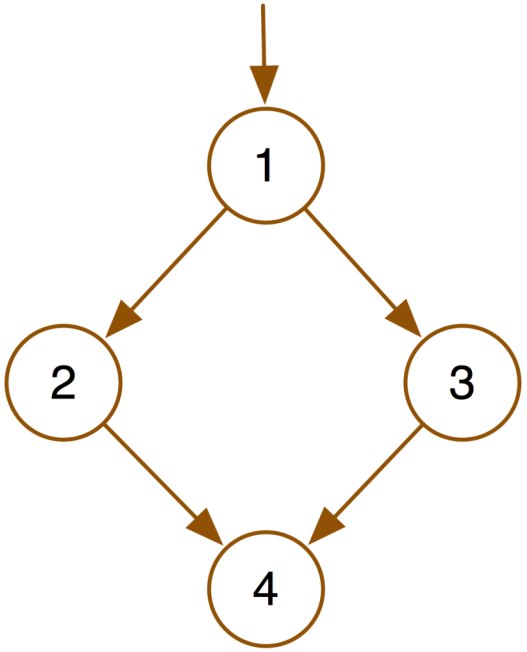


Prime path definition – 2

- What is a prime path?
  - A path from node n_j to node n_k is prime if
    - It is a simple path
    - It is not a proper sub-path of any other simple path
    - It is a maximal length simple path

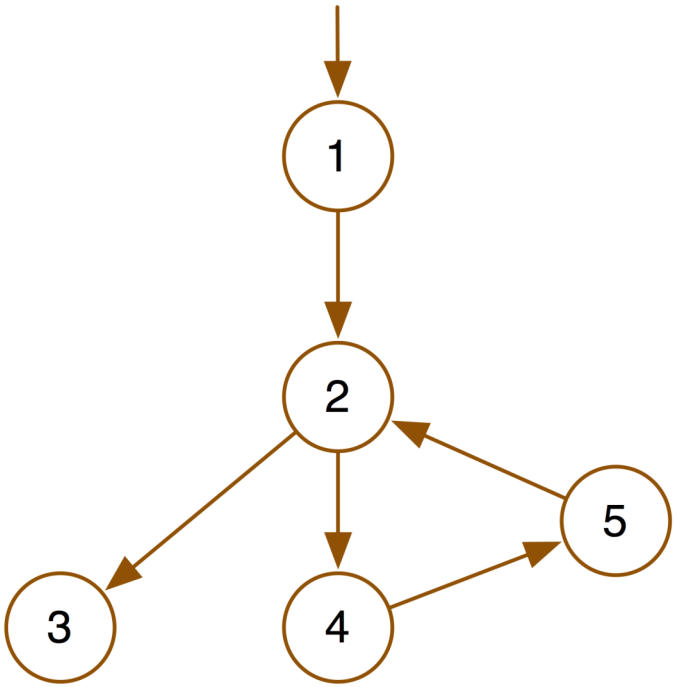
Prime path – Example 1

- Prime paths
  - $\langle n_1 , n_2 , n_4 \rangle$
  - $\langle n_1 , n_3 , n_4 \rangle$



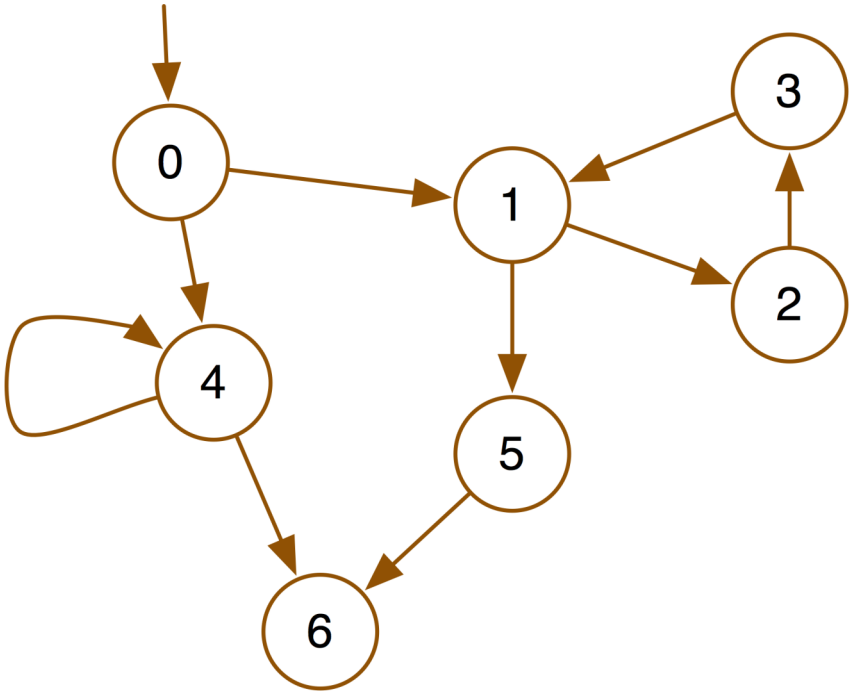
Prime path – Example 2

- Prime paths
  - $\langle n_1 , n_2 , n_3 \rangle$
  - $\langle n_1 , n_2 , n_4 , n_5 \rangle$
  - $\langle n_2 , n_4 , n_5 , n_2 \rangle$
  - $\langle n_4 , n_5 , n_2 , n_4 \rangle$
  - $\langle n_5 , n_2 , n_4 , n_5 \rangle$
  - $\langle n_4 , n_5 , n_2 , n_3 \rangle$



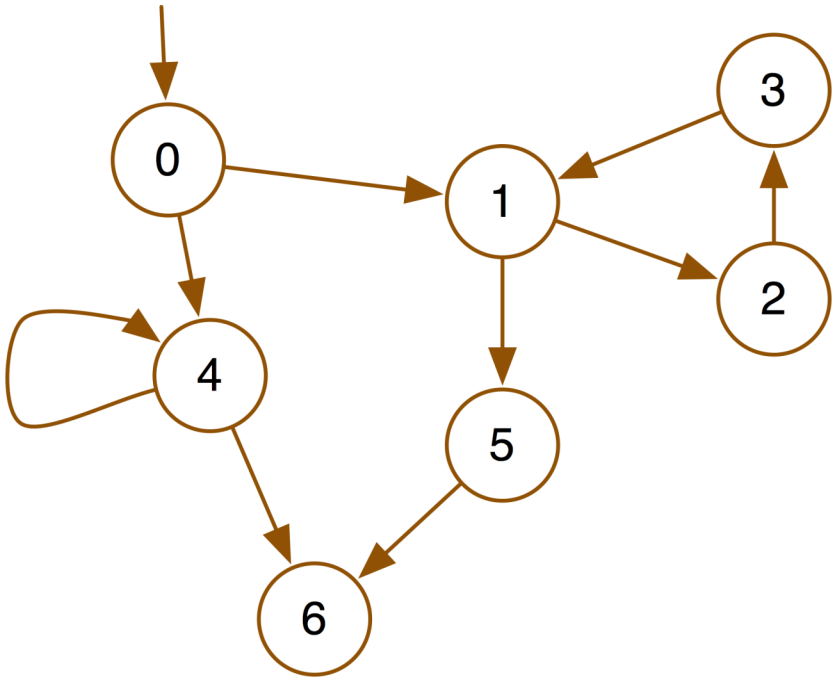
Finding prime paths

- Consider the following graph, what are its prime paths?



Finding prime paths – length 0 paths

- Start with a list of the nodes
  - **The ! Indicates that the path cannot be extended**

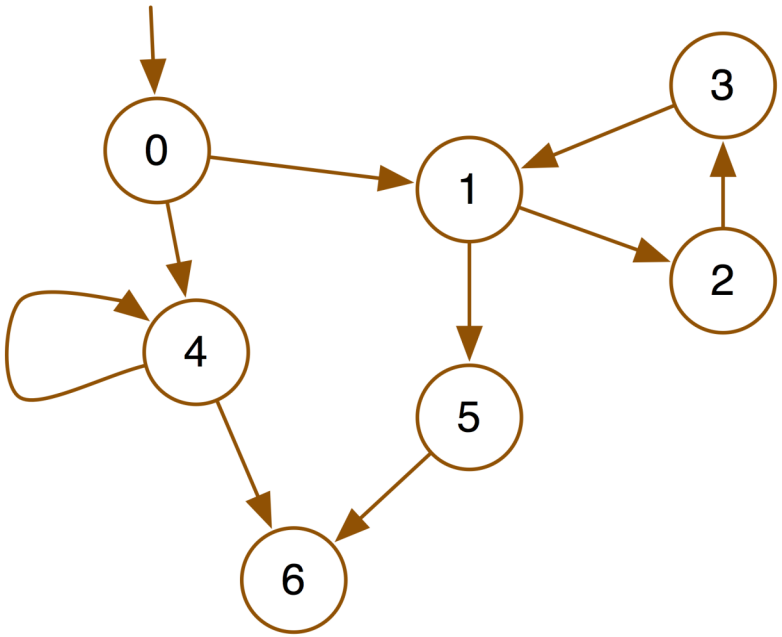


1 [0]
2 [1]
3 [2]
4 [3]
5 [4]
6 [5]
7 [6] !

Finding prime paths – length 1 paths

- Extend length 0 paths by one edge
  - **Path 7 cannot be extended**
  - **The * indicates a loop – cannot be extended**

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

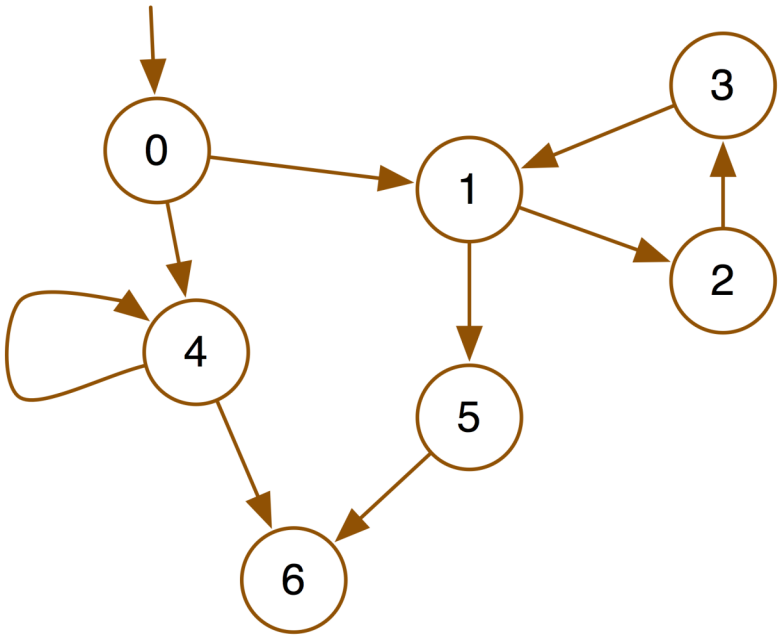


8 [0, 1]
9 [0, 4]
10 [1, 2]
11 [1, 5]
12 [2, 3]
13 [3, 1]
14 [4, 4] *
15 [4, 6] !
16 [5, 6] !

Finding prime paths – length 2 paths

- Extend length 1 paths by one edge
  - **Paths 14, 15 and 16 cannot be extended**

8 [0, 1]
9 [0, 4]
10 [1, 2]
11 [1, 5]
12 [2, 3]
13 [3, 1]
14 [4, 4] *
15 [4, 6] !
16 [5, 6] !

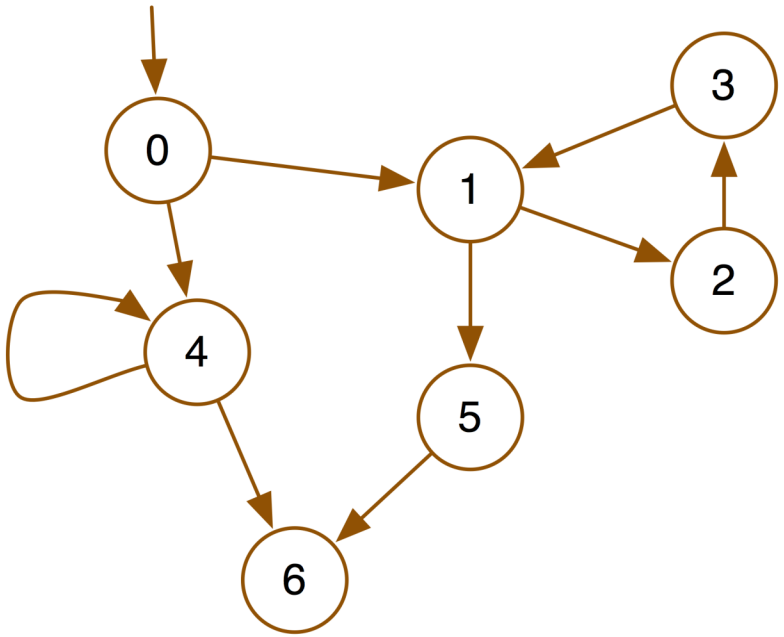


17 [0, 1, 2]
18 [0, 1, 5]
19 [0, 4, 6] !
20 [1, 2, 3]
21 [1, 5, 6] !
22 [2, 3, 1]
23 [3, 1, 2]
24 [3, 1, 5]

Finding prime paths – length 3 paths

- Extend length 2 paths by one edge
  - **Paths 19 and 21 cannot be extended**

17 [0, 1, 2]
18 [0, 1, 5]
19 [0, 4, 6] !
20 [1, 2, 3]
21 [1, 5, 6] !
22 [2, 3, 1]
23 [3, 1, 2]
24 [3, 1, 5]

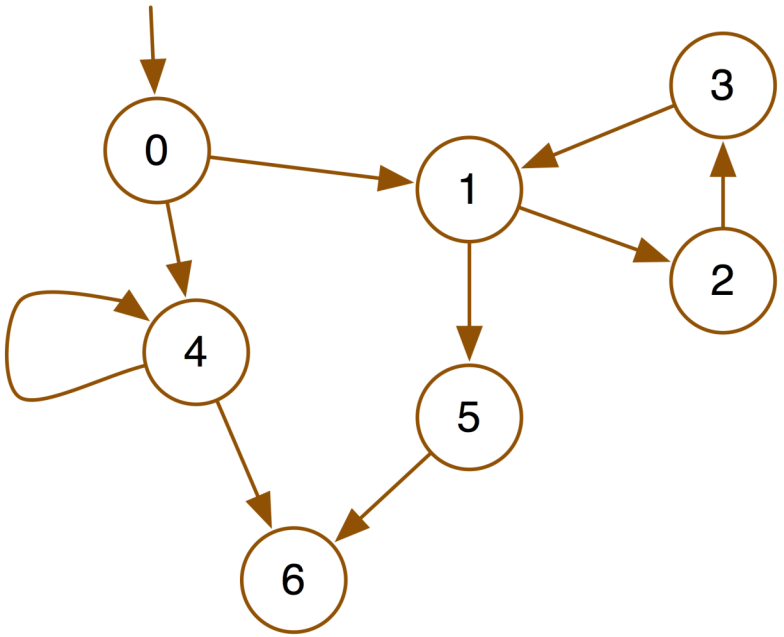


25 [0, 1, 2, 3] !
26 [0, 1, 5, 6] !
27 [1, 2, 3, 1] *
28 [2, 3, 1, 2] *
29 [2, 3, 1, 5]
30 [3, 1, 2, 3] *
31 [3, 1, 5, 6] !

Finding prime paths – length 4 paths

- Extend length 3 paths by one edge
  - **Only path 29 be extended and no further extensions are possible**

25 [0, 1, 2, 3] !
26 [0, 1, 5, 6] !
27 [1, 2, 3, 1] *
28 [2, 3, 1, 2] *
29 [2, 3, 1, 5]
30 [3, 1, 2, 3] *
31 [3, 1, 5, 6] !



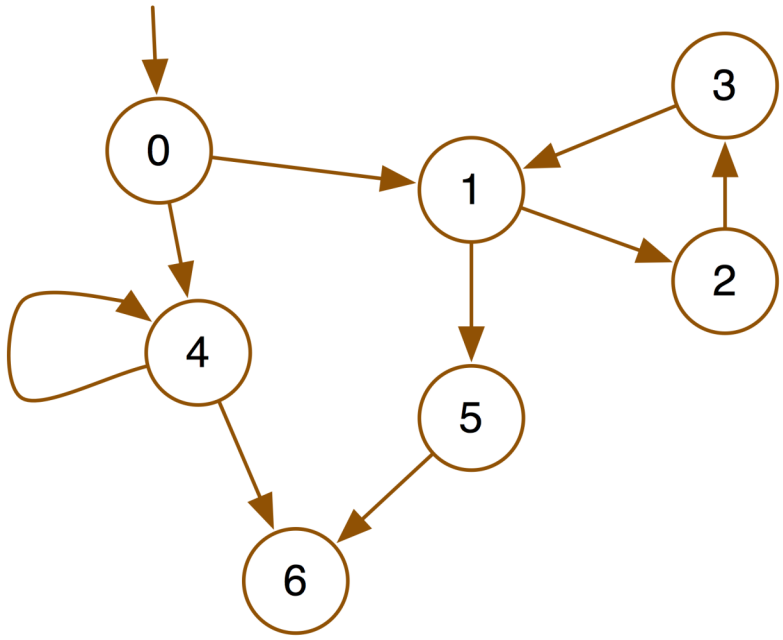
32 [2, 3, 1, 5, 6] !

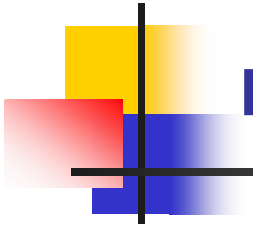
Finding prime paths – Collect paths

- No more paths can be extended.
- Collect all the paths that terminate with ! or *
- Eliminate any path that is a subset of another path in the list

These are the 8 prime paths
In the example graph

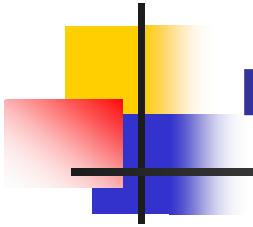
14 [4, 4] *
19 [0, 4, 6] !
25 [0, 1, 2, 3] !
26 [0, 1, 5, 6] !
27 [1, 2, 3, 1] *
28 [2, 3, 1, 2] *
30 [3, 1, 2, 3] *
32 [2, 3, 1, 5, 6] !





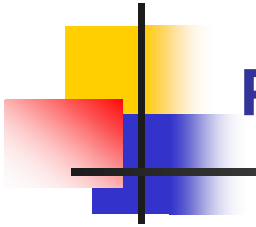
Prime path usefulness

- **Of what use is a prime path?**



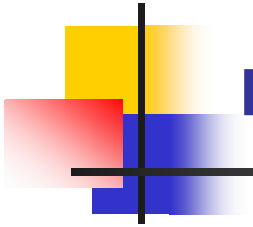
Prime path usefulness – 2

- Of what use is a prime path?
  - Reduces the number of test cases for path coverage



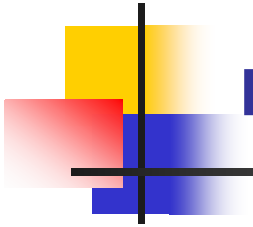
Prime path problem

- What is the problem with prime paths?



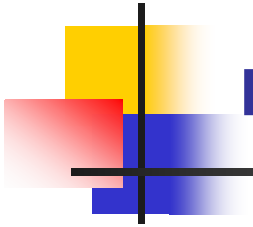
Prime path problem – 2

- What is the problem with prime paths?
  - A prime path may be infeasible but contain feasible simple paths
    - In such cases, the prime path is factored into the simple paths in order that it may be covered by testing



Round trip path definition

- **What is a round trip path?**

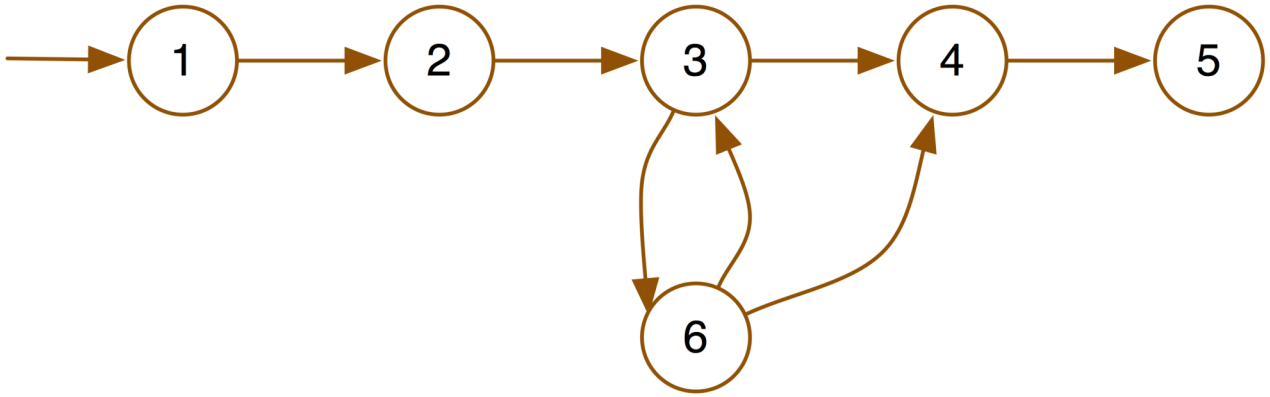


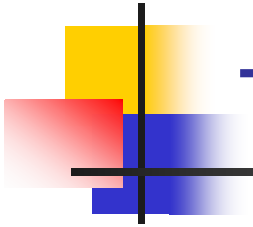
Round trip path definition – 2

- What is a round trip path?
  - It is a prime path, P
  - #P > 0
  - head P = last P
    - Recall that P is a sequence of nodes

Tour definition

- In the context of test paths and graph paths
- What is a tour?



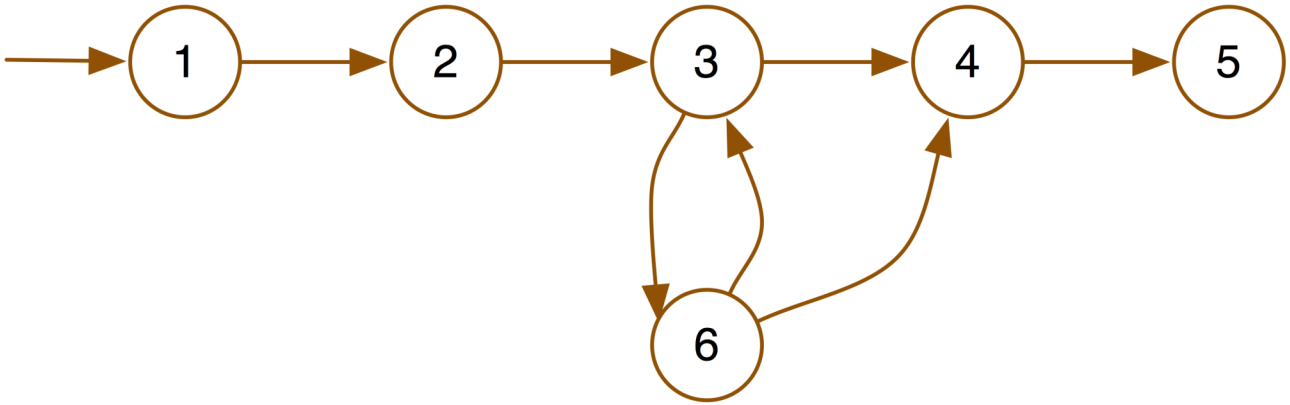


Tour definition – 2

- What is a tour?
  - A test path is said to tour a graph path if
    - graph-path $\subseteq$ test-path
    - $\subseteq$ in this context means sub-path – not subset
    - The test-path must visit the graph-path nodes in exactly the specified sequence with no intervening nodes

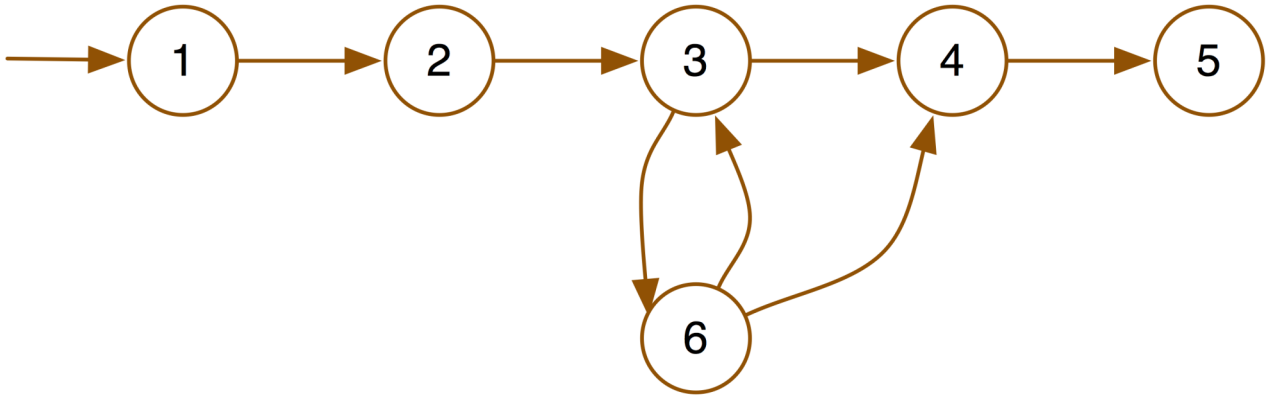
Tour definition – 3

- The following paths are tours
  - $\langle n_2 , n_3 , n_4 \rangle$
  - $\langle n_2 , n_3 , n_6 , n_4 \rangle$
  - $\langle n_2 , n_3 , n_6 , n_3 , n_4 \rangle$



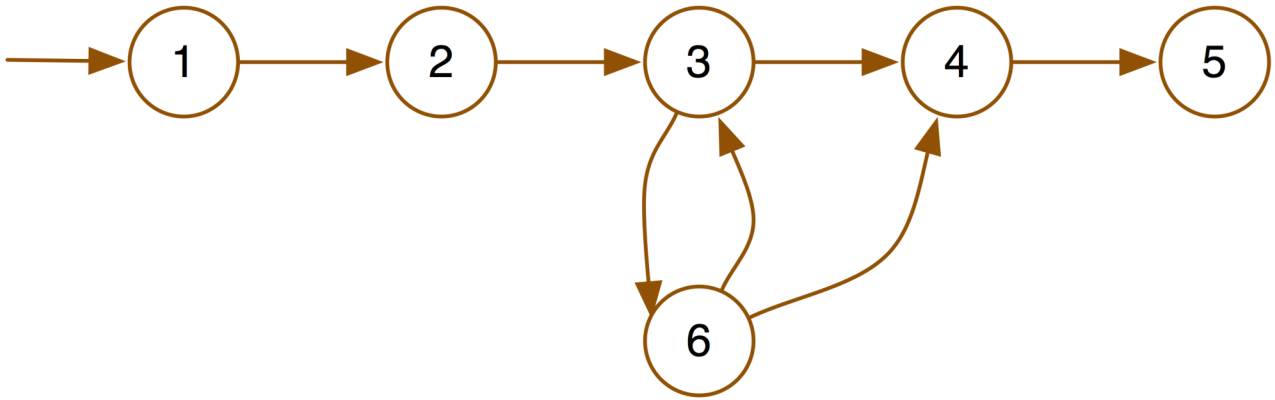
Tour with side trips definition

- **What is a tour with side trips?**



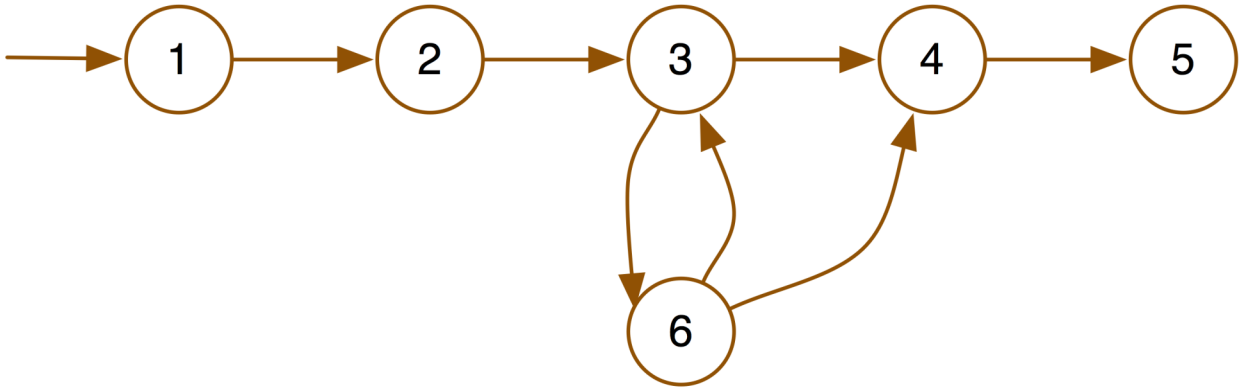
Tour with side trips definition – 2

- **What is a tour with side trips?**
  - A tour is restrictive in that many test paths would be infeasible
  - Occurs when loops are in the path
  - The path $\langle n_2 , n_3 , n_4 \rangle$ would be impossible to tour if the condition in n_3 is such that n_6 must be visited at least once



Tour with side trips definition – 3

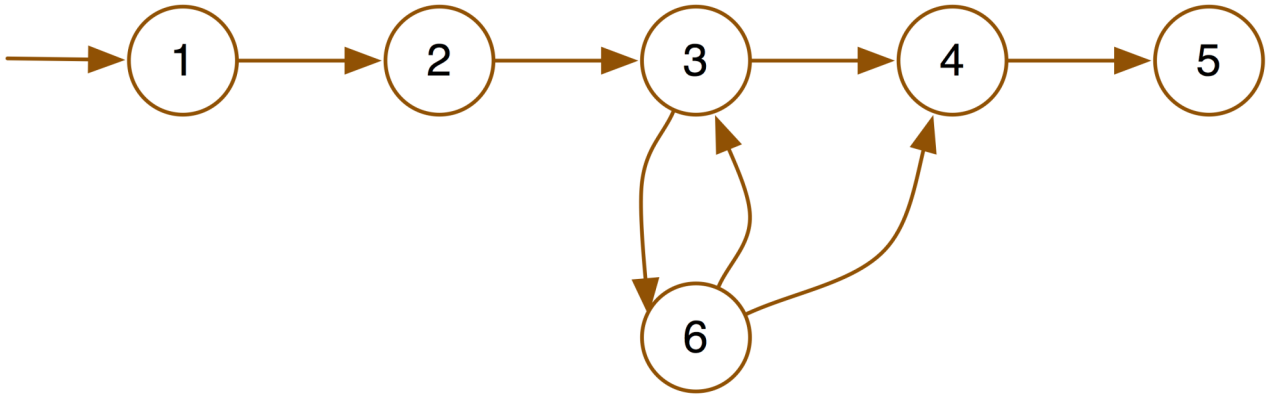
- We relax the definition of a tour to include side trips
  - **Leave the sub-path**
  - **But come back to the same node before continuing the sub-path – e.g. $\langle n_2 , n_3 , n_6 , n_3 , n_6 , n_3 , n_4 \rangle$**



- Test path tours the graph-path with side trips iff every **edge** of the graph-path is followed in the same order

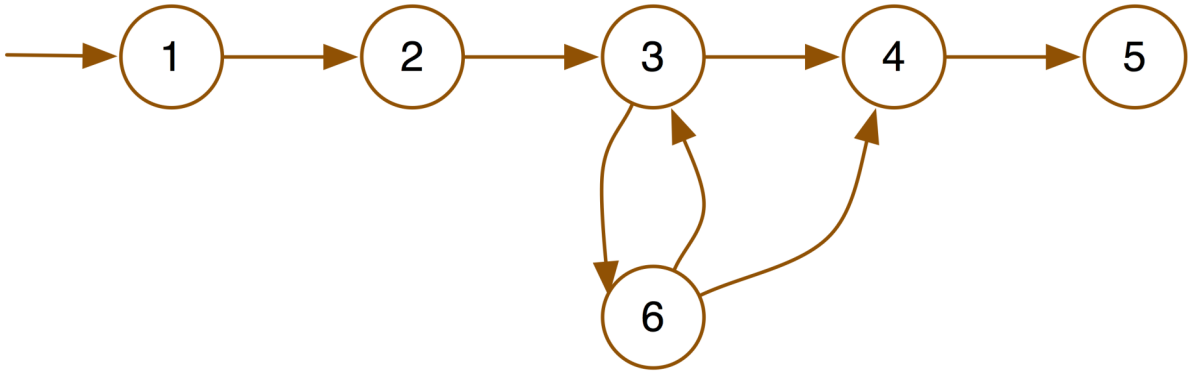
Tour with detours definition

- **What is a tour with detours?**

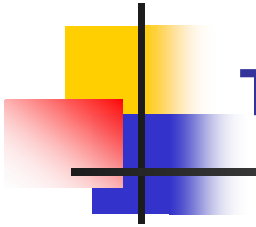


Tour with detours definition – 2

- We relax the definition of a tour to include detours
  - **Leave the sub-path**
  - **But come back to the node that follows the node where the sub-path was left**
    – e.g. ⟨ n_2 , n_3 , n_6 , n_3 , n_6 , n_4 ⟩

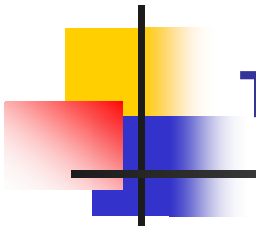


- Test path tours the graph-path with detours iff every **node** of the graph-path is followed in the same order



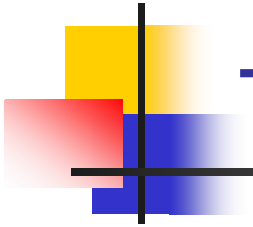
Test requirement

- **What is a test requirement?**



Test requirement – 2

- **What is a test requirement?**
  - **A specific element of a software artifact that a test case must satisfy or cover.**
  - **Usually come in sets**
    - Use the abbreviation ***TR*** to denote a set of test requirements

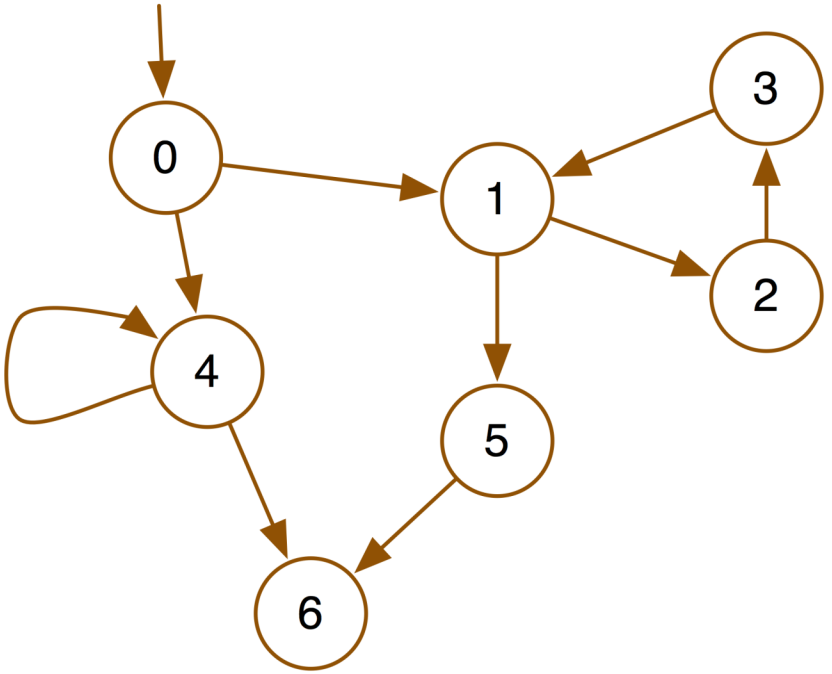


Test requirement – 3

- Test requirements
  - **Are described with respect to a variety of software artifacts, including**
    - **Program text**
    - **Design components**
    - **Specification modeling elements**
    - **Even descriptions of the input or output space**

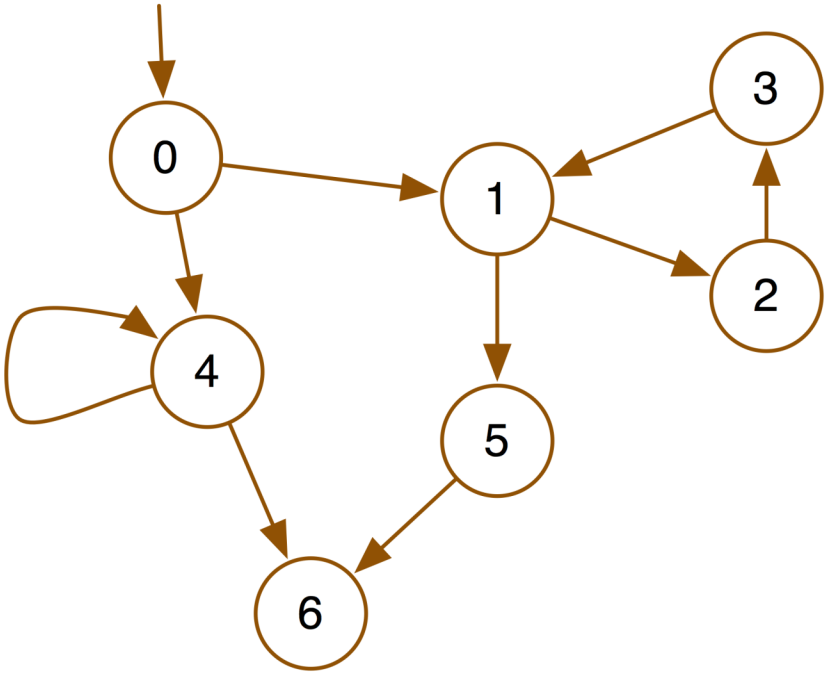
Test requirements – All nodes

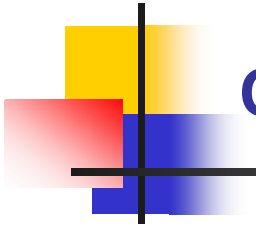
- Given the pictured graph and the coverage criterion **all nodes**
- **What would be the test requirements?**



Test requirements – All nodes

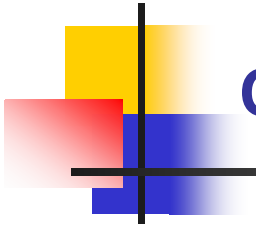
- Given the pictured graph and the coverage criterion **all nodes**
  - **The test requirements is a listing of all the nodes in the graph, with the implication testing should cover the requirements**
  - **{ 0, 1, 2, 3, 4, 5, 6 }**





Coverage criteria

- **What is a coverage criterion?**

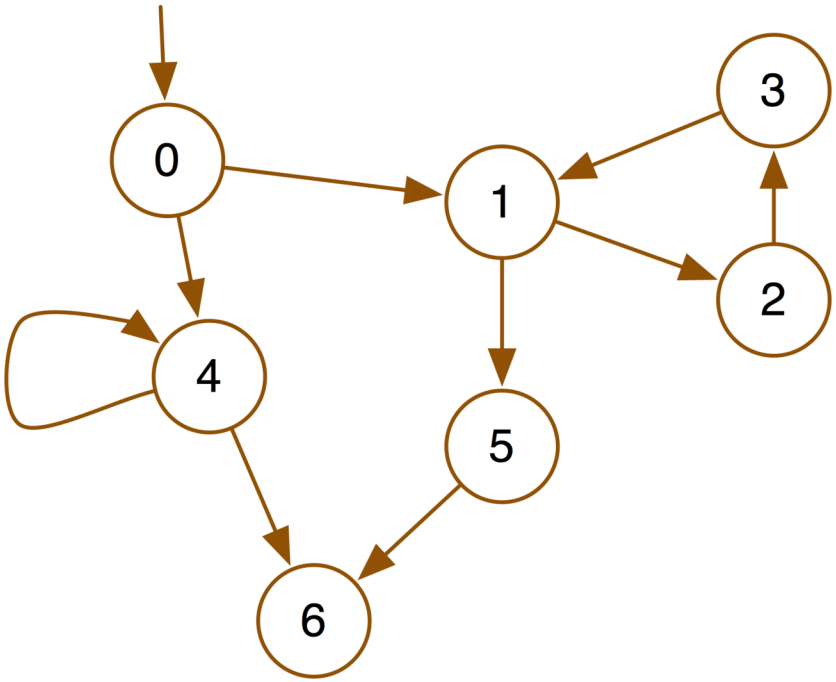


Coverage criteria

- What is a coverage criterion?
  - A coverage criterion is a rule or collection of rules that impose test requirements on a test set.
    - A recipe for generating test requirements in a systematic way

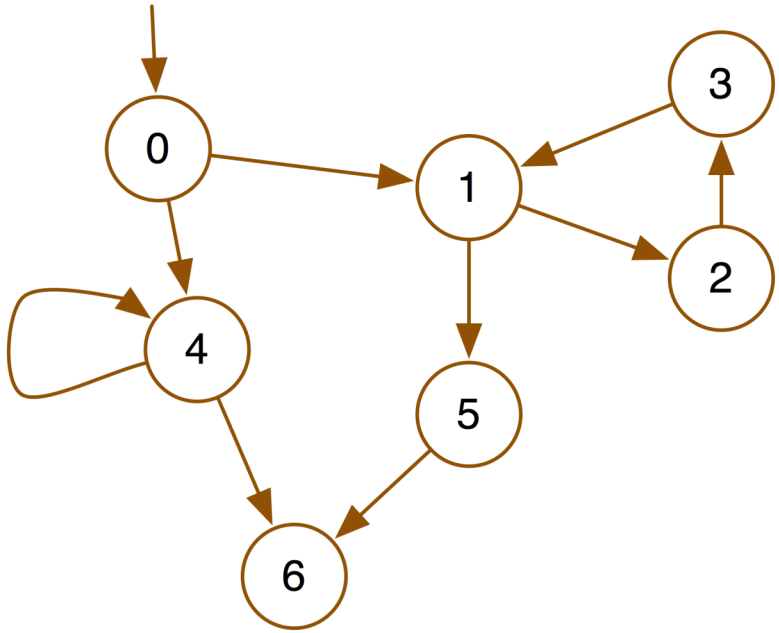
Coverage criteria – 2

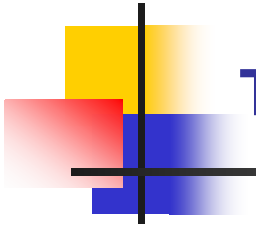
- Consider the following graph
  - **What test coverage criteria can we have?**



Coverage criteria – 3

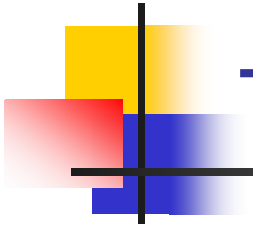
- Coverage can be the following
  - All nodes
  - All edges
  - All edge pairs
    - More edges not useful
  - All simple paths
  - All prime paths
  - All simple round trips
    - 1 trip each reachable node that begins & ends the path
  - All complete round trips
    - All trips each reachable node
  - All specified paths – as all paths is not feasible
  - All paths





Test set

- **What is a test set?**

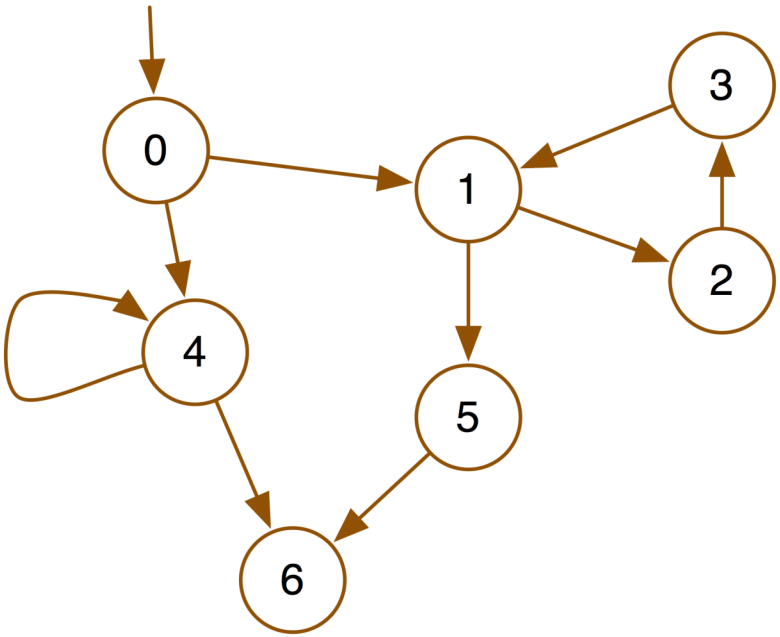


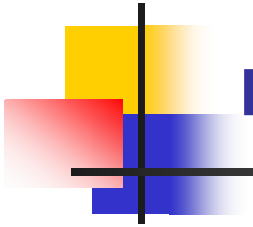
Test set – 2

- **What is a test set?**
  - **Satisfies test requirements by visiting every artifact in the test requirements**

Test set – 3

- Given the test requirements to visit **all nodes** in the following set for the pictured graph
  - **{ 0, 1, 2, 3, 4, 5, 6 }**
- The following test set satisfies the test requirements
  - **{ [0, 4, 4, 4, 6]**
    , [0, 1, 2, 3, 1, 5, 6] }



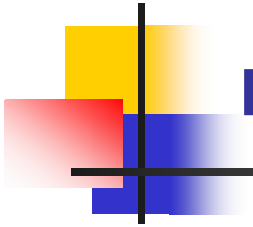


Best effort touring

- In the context of test requirements
- What is a best effort tour?

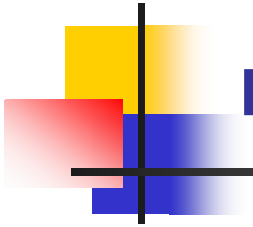
Best effort touring – 2

- **What is a best effort tour?**
  - **TR_{tour} is a set of test requirements such that**
    - **Paths in a graph that must be covered**
      - Can be directly toured
  - **$TR_{sidetrips}$ is a set of test requirements**
    - **Paths in a graph that must be covered**
      - Can be directly toured
      - Or toured with sidetrips



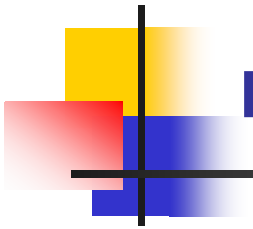
Best effort touring – 3

- Trips with detours are rarely considered
  - **They are less practical than sidetrips in dealing with infeasible paths**



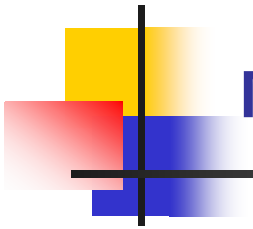
Best effort touring

- In the context of best effort touring
- What is a test set?



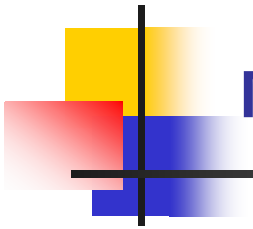
Best effort touring – 3

- A test set T is best effort touring if
  - For every path p in TR_{tour}
    - Some path in T tours p
      - Directly
  - For every path p in $TR_{sidetrips}$
    - Some path in T tours p either
      - Directly
      - Or with a sidetrip



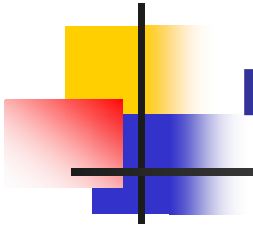
Meeting strict requirements

- Each test requirement is met in the strictest possible way
  - **Which means?**



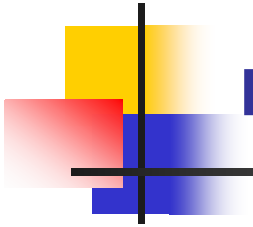
Meeting strict requirements – 2

- Each test requirement is met in the strictest possible way
  - **Edges and nodes must be visited in the same order as in the graph**



Path behaviours

- **When test requirements describe paths, we distinguish three types of path behaviours, what are they?**

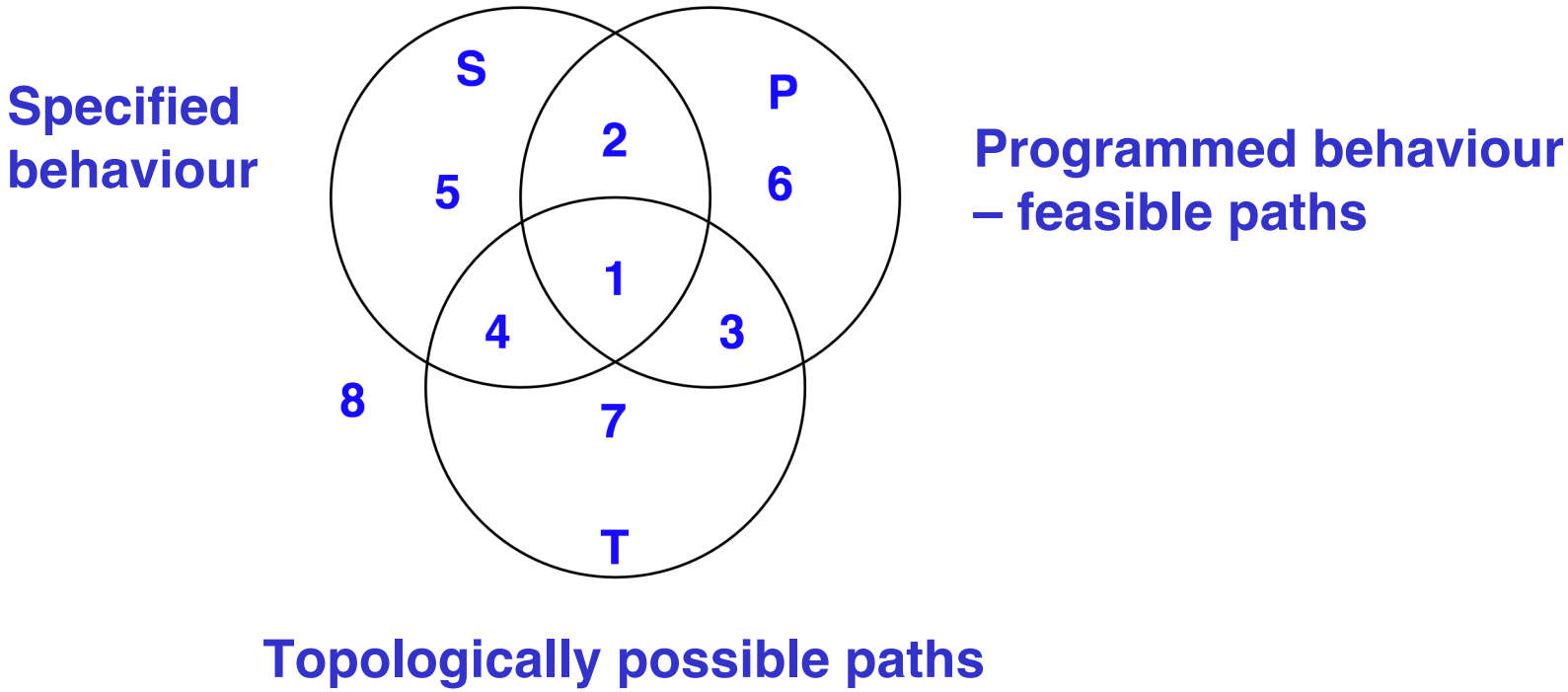


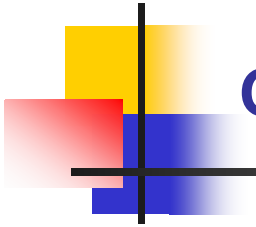
Path behaviours– 2

- When test requirements describe paths, we distinguish three types of path behaviours, what are they?
  - Feasible
  - Specified
  - Topologically possible
    - Just look at paths in the graph
      - Ignore conditions

Path behaviours – 3

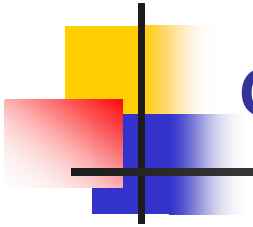
- Re-examine the Venn diagram in the context of path testing





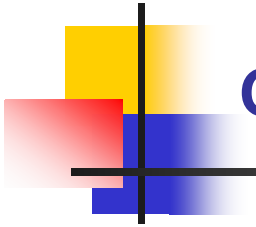
Guidelines

- **What is the relationship between program text and each of the following?**
  - **Functional testing**
  - **Path testing**



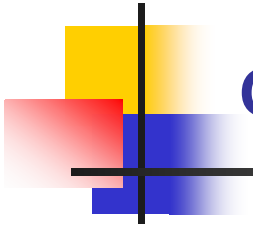
Guidelines – 2

- What is the relationship between program text and each of the following?
- Functional testing
  - Too far from the program text
- Path testing
  - Too close to the program text
  - Obscures feasible and infeasible paths



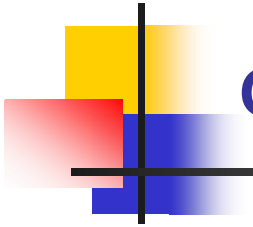
Guidelines – 3

- What are the benefits and drawbacks of using path testing?



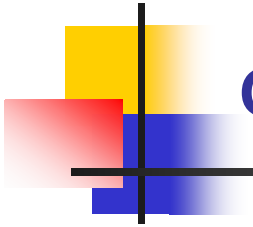
Guidelines – 4

- What are the benefits and drawbacks of using path testing?
  - Gives good measures of quality of testing through coverage analysis
  - Provides set of metrics that cross-check functional testing
  - Use to resolve gap and redundancy questions
    - Missing paths – have gaps
    - Repeated paths – have redundancy
  - Does not give good help in finding test cases



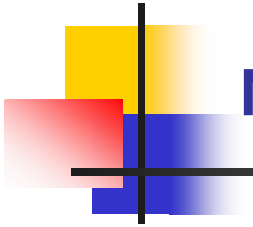
Guidelines – 5

- What is meant by
  - Does not give good help in finding test cases?



Guidelines – 6

- **What is meant by**
  - **Does not give good help in finding test cases?**
    - **Too many paths**
    - **Infeasible paths**
    - **Need selective coverage**
      - Loop coverage
      - Computation coverage
      - Interesting paths



Next step

- **Use dataflow testing to move out a bit**
  - **Move closer to functional testing**
  - **Make use of the functional aspects of a program**
  - **Less reliance on physical program structure**