Introduction to Software Testing
Graph Coverage Criteria (Ch. 7.1)
Covering Graphs (7.1)

Graphs are the most **commonly** used structure for testing

Graphs can come from many sources

- Control flow graphs
- Design structure
- FSMs and state charts
- Use cases

Tests usually are intended to “**cover**” the graph in some way
What is a graph?

A set $N$ of nodes, $N$ is not empty

A set $N_o$ of initial nodes, $N_o$ is not empty

A set $N_f$ of final nodes, $N_f$ is not empty

A set $E$ of edges, each edge from one node to another

- $(n_i, n_j)$, $i$ is predecessor, $j$ is successor

Is this a graph?

$N_o = \{ 1 \}$

$N_f = \{ 1 \}$

$E = \{ \}$

NO
Example graphs

\(N_0 = \{1\}; N_f = \{4\}\)
\(E = \{(1,2), (1,3), (2,4), (3,4)\}\)

\(N_0 = \{1, 2, 3\}; N_f = \{8, 9, 10\}\)
\(E = \{(1,4), (1,5), (2,5), (3,6), (3, 7), (4, 8), (5,8), (5,9), (6,2), (6,10), (7,10) (9,6)\}\)

\(N_0 = \{\}; N_f = \{4\}\)
\(E = \{(1,2), (1,3), (2,4), (3,4)\}\)

Not a valid graph
Paths in graphs

**Path**: A sequence of nodes – \([n_1, n_2, \ldots, n_M]\)
- 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]
Graph Definitions

Answer the following questions for the graph on the left

1. How many nodes are in the graph?
2. How many edges are in the graph?
3. What is the set of initial nodes?
4. What is the set of final nodes?
5. Write two paths in the graph.
6. Write a subpath of one of your paths.
Test paths and SESEs

**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

---

Double-diamond graph

Four test paths
- [1, 2, 4, 5, 7]
- [1, 2, 4, 6, 7]
- [1, 3, 4, 5, 7]
- [1, 3, 4, 6, 7]
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? \([1,2,4], [2,4,5], [4,5,7], [1,2,4,5], [2,4,5,7], [1,2,4,5,7]\)

(Technically, each edge is also a subpath)
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 a final node

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
  - This distinction becomes important in **section 7.3**
Tests and test paths

Deterministic software—test always executes the same test path

Non-deterministic software—the same test can execute different test paths
In-class Exercise

Test paths

Answer the following questions for the graph on the left:

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?