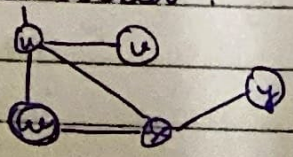
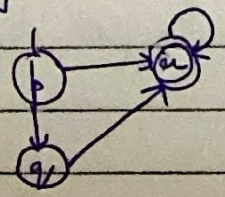
Week-2

L1 Basics of Graphs

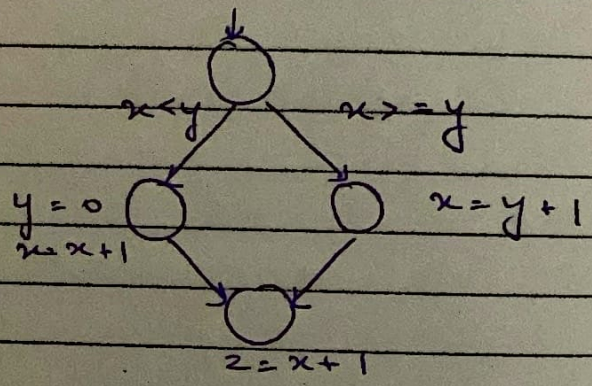
- A graph is a tuple $G=(V,E)$ where V is a set of nodes & $E \subseteq (V \times V)$ is a set of edges. It can be directed or undirected.



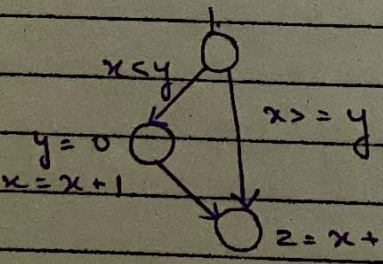


- Graphs can be finite or ∞. The degree of a vertex is the no. of edges that are connected to it. Edges connected to a vertex are said to be incident on the vertex.

-
```
if (x<y)
{
  y=0;
  x=x+1;
}
else
{
  x=y+1;
}
z=x+1;
```



-
```
if (x<y)
{
  y=0;
  x=x+1;
}
z=x+1;
```



- Path - p is a seq. of vertices $v_1, v_2 \ldots v_n$ such that $(v_i, v_{i+1}) \in E$ for $1 \leq i \leq n-1$
- Length - of a path is the no. of edges that occur in it. A single vertex path has length 0.
- Sub-path - Of a path is a sub-seq. of vertices that occur in the path.
- A vertex v is reachable in a graph G if there is a path from one of the initial vertices of the graph to v.

- An edge $e = (u, v)$ is reachable in a graph G if there is a path from one of the initial vertices to the vertex u & then to v thru the edge e.
- A sub-graph G' of a graph G is reachable if one of the vertices of G' is reachable from an initial node in G.
- DFS & BFS → can be used for reachability in graphs.
- A test path → is a path that starts in an initial vertex & ends in a final vertex.
- A test path visits a vertex v if v occurs in the path p. A test path p visits an edge e if e occurs in the path p.
- A test path p tours a path q if q is a sub-path of p
- TR → properties of test paths.
- Test Criterion → they are rules that define TR.
- Satisfac^{n} → Given a set TR of test requirements for a criterion C, a set of tests satisfies C on a graph iff. for every test req. in $t \in TR$, there is a test path in path(T) that meets the test req. t.
- Structural Coverage → defined on a graph just in terms of vertices & edges.
- Data Flow Coverage → Req. a graph to be annotated with ref. to vari. & defines criteria req. based on the annota^{n}.

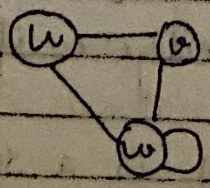
L2 Fundamentally Graph Algos.

- 2 ways to rep. graphs – Adjacency Matrix & List → sparse graphs
  ↓ dense graphs

- Adj. List → It represents a graph $G = (V, E)$ is an array of Adj. of $|V|$ lists, one for each vertex in V. Directed graphs, sum of lengths of all adj. list is $|E|$ & undirected is $|2E|$. For both graphs, it req. $\Theta(|V| + |E|)$ memory.
- Adj. Matrix → It consists of a $|V| \times |V|$ matrix $A = (a_{ij})$ such that,

$$a_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

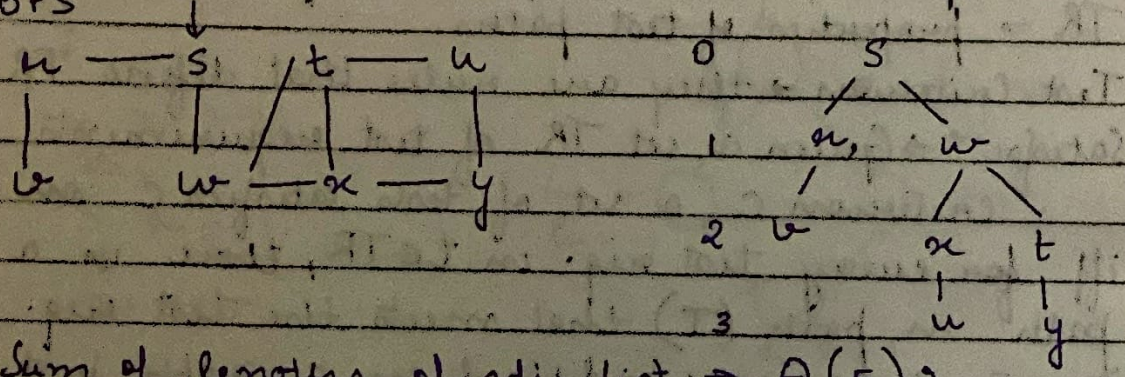
↳ $\Theta(V^2)$



$u \to v, w$
$v \to u, w$
$w \to u, v, w$

	u	v	w
u	0	1	1
v	1	0	1
w	1	1	1

- BFS

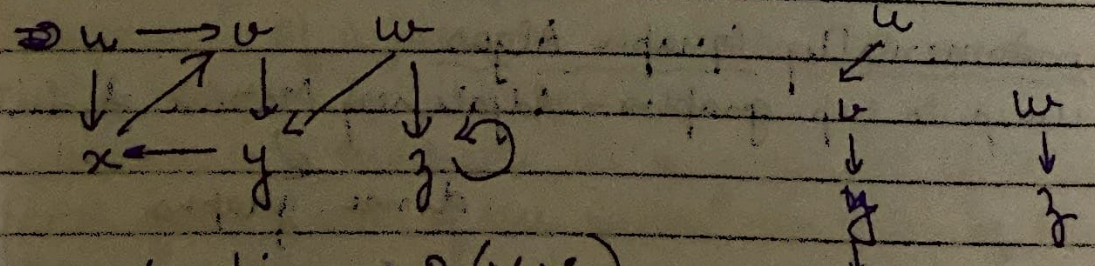


- Sum of lengths of adj. list ⇒ $\Theta(E)$
- Overhead for initializing is ⇒ $\Theta(V)$
- Total running time ⇒ $\Theta(V+E)$

} Finds shortest path.

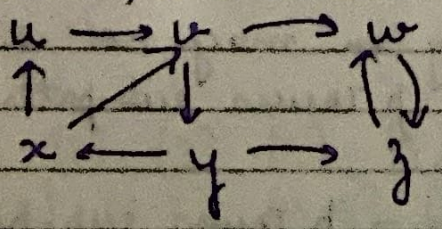
L3 Elementary Graph Algos:

- DFS



- running time ⇒ $\Theta(V+E)$

- Tree Edges – Edges in DFS. Edge (u,v) is a tree edge if v was first discovered by edge (u,v).
- Back Edges – Edge (u,v) connecting a vertex u to an ancestor v in a DFS. Eg. Self-loops
- Forward Edges – They are non-tree edges connecting a vertex u to a descendant v in a DFS.
- Others are cross edges
- A SCC (strongly connected comp.) of G_1 is a maximal set of vertices $C \subseteq V$ such that for every pair of vertices u & v in C, u & v are reachable from each other.

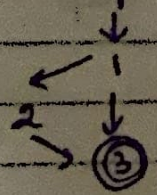


$C_1 = \{u, v, x, y\}$
$C_2 = \{w, z\}$

L4 Structural Graph Coverage Criteria

- Node Coverage – test case visits each node in graph once. TR contains each reachable node in G_1.
- Edge Coverage – TR contains each reachable path of length up to 1. Subsumes node coverage.

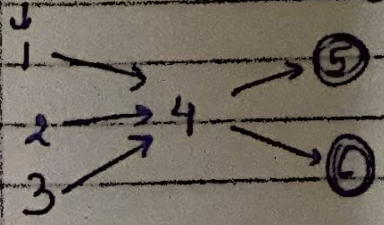
Eg.



Node coverage
TR = {1, 2, 3}
Path = [1, 2, 3]

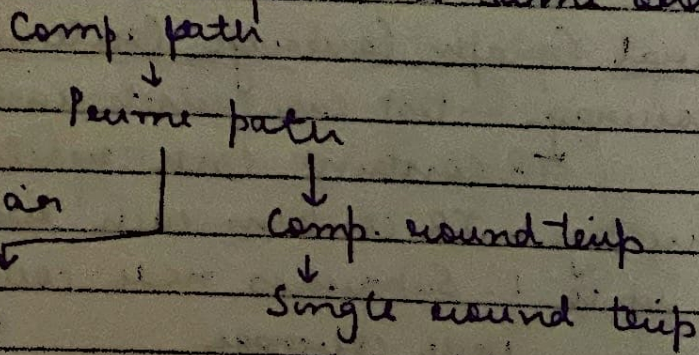
Edge coverage
TR = {(1,2), (2,3), (1,3)}
Path = {[1, 2, 3], [1, 3]}

- Edge Pair Coverage – TR contains each reachable path of length up to 2



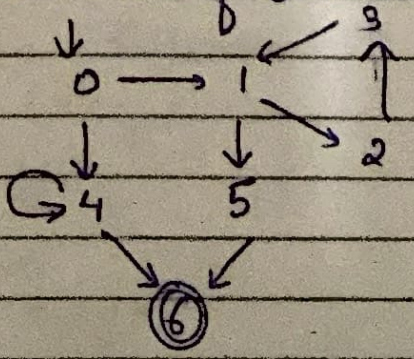
TR = Path = {[1,4,5], [2,4,5], [3,4,5], [1,4,6], [2,4,6], [3,4,6]}

- Comp. Path Coverage - TR contains all paths in G.
- Specified Path Coverage - TR contains a set S of paths, where S is specified by the user.
- Simple Path - A path from node n_i to n_j is simple if no node appears more than once, except 1st & last node. No internal loops
- Prime Path - Maximal ~~path~~ simple path.
- Prime path coverage - TR contains each prime path in G. Ensures that loops are skipped as well as executed. It subsumes node & edge coverage. May or may not subsume edge pairs.
- Tour - A test path p tours sub-path q if q is a sub-path of p.
- Tour with side-trip - p tours sub-path q, iff. every edge in q is also in p in same order.
- Tour with detour - If every node in q is also in p in the same order.



L5 Structural Graph Coverage Criteria

- TR for node coverage - The set of vertices / nodes.
- TR for edge coverage - The set of edges & nodes
- TR for edge-pair coverage - all paths of length 2, need to include self-loops too. At basic level, the TR itself constitute the test paths.



1. First numerate all simple paths
2. ! → show path can't extend
3. * → already simple cycle, can't extend.
4. find prime paths
5. write test paths

7 → [0], [1], [2], [3], [4], [5], [6]

9 → [0,1], [0,4], [1,2], [1,5], [2,3], [3,1], [4,4]*, [4,6]!, [5,6]!

8 → [0,1,2], [0,1,5], [1,2,3], [1,5,6]!, [2,3,1], [3,1,2], [3,1,5], [0,4,6]!

7 → [0,1,2,3]!, [0,1,5,6]!, [1,2,3,1]*, [2,3,1,2]*, [2,3,1,5], [3,1,2,3]*, [3,1,5,6]!

1 → [2,3,1,5,6]!

32 → simple paths

8 prime paths → [4,4]*, [0,4,6]!, [0,1,2,3]!, [0,1,5,6]!, [1,2,3,1]*, [2,3,1,2]*, [3,1,2,3]*, [2,3,1,5,6]!

Test paths → [0,4,6], [0,4,4,6], [0,1,5,6], [0,1,2,3,1,5,6], [0,1,2,3,1,2,3,1,5,6]

- Cs.gmu.edu