CS 312 Traversals:

Depth First Paths:

Goal: to find a path from s to every other reachable vertex, visiting each vertex at most once.

- Mark v.
- For each unmarked adjacent vertex,
  - set edgeTo[w] = v.
  - dfs(w).

Demo @ 5:30 in lecture.

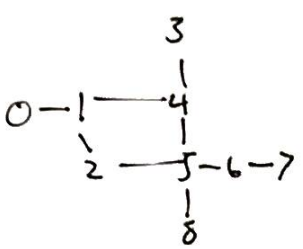
DFS = general term of any graph search that goes deep before backtracking.

* HasPathTo: if marked[v] == True!

Runtime = $\Theta(V+E)$ — Each vertex is visited once, each visit costs constant time. Each edge is used exactly once as well.

Space = $\Theta(V)$ — Call stack depth is at most V.

There are a lot of graph traversals too!

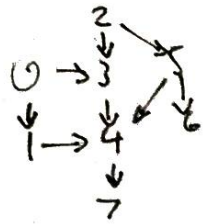


- DFS Preorder: order of DFS calls
  0 1 2 5 4 3 6 7 8
- DFS Postorder: order of returns from DFS
  3 4 7 6 8 5 2 1 0
- Level order: increasing distance from s.
  0 1 2 4 5 3 6 8 7

If we do level order and a vertex has 2 edges, use the shortest one.

Topological Sort

- DFS postorder from every vertex with indegree 0, not clearing markings in between traversals. * Demo @ 26:45
  - Record DFS post order in a list.
  - Topological ordering is the reverse of this list.



→ DFS post order:
[7, 4, 1, 3, 0, 6, 5, 2]

→ Topological ordering
[2, 5, 6, 0, 3, 1, 4, 7]

when you topological sort, all arrows point to the right.

topological sort
- for finding an ordering of vertices consistent w/ directed edges.
  Efficiency → $\Theta(V+E)$ time, $\Theta(V)$ space.

Breadth First Search
- Level-order → order visited by breadth first search.

1) Create a queue w/ starting vertex s. Mark it.
   - this is the fringe.
2) Repeat until queue is empty:
   a) Remove vertex v from queue.
   b) Add any unmarked vertices adjacent to v into the queue and mark them.

Demo @ 39:15

Breadth First Path
Goal: find a path from s to every reachable vertex.

1) Initialize the fringe → a queue w/ starting vertex s and mark it.
2) Repeat until queue is empty:
   a) Remove vertex v.
   b) For each unmarked neighbor of v:
      mark, add to queue, set edgeTo = v.

Invariant:
fringe queue always has:
- ≥ 0 vertices of distance k from s.
- ≥ 0 vertices of distance k+1 from s.

- Useful for finding shortest path

RT: $\Theta(V+E)$
Space: $\Theta(V)$

Breadth first is not good for GMaps because we want to minimize drive time.