directed, arc/edge, weighted, labeled. drawings
representation
relations → edges
tree, planar
traversal: graph $\rightarrow$ forest
1. foreach vertex, label it undiscovered and \( v.d \leftarrow \infty \)
2. \( \text{start}'s \) label \( \leftarrow \) discovered, \( d \leftarrow 0, \pi \leftarrow \text{nil} \)
3. \( Q \leftarrow \{\text{start}\} \)
4. while \( Q \) not empty
5. \( u \leftarrow \text{dequeue}(Q) \)
6. foreach neighbor \( v \) of \( u \)
7. if \( v \) is undiscovered
8. label \( v \) discovered, \( v.d \leftarrow u.d + 1, v.\pi \leftarrow u \)
9. enqueue \( v \) in \( Q \)
10. label \( u \) finished

Which vertices does \( Q \) hold (at line 4)?
Do we really need all the labels?
What’s the time complexity?
1. Distances we assign always stay the same or go down.

2. \( v.d \geq \delta(s, v) \)
   Proof: Show \( v.d \geq \delta(s, v) \) \( \forall v \) via induction over iterations:
   Holds at start.
   \( v.d \) is updated to \( u.d + 1 \geq \delta(s, u) + 1 \geq \delta(s, v) \).

3. \( d \) values in queue are nondecreasing and last in queue exceeds first by at most 1.
   Proof: By induction. True when queue is \( s \).
   Preserved by dequeue.
   Enqueue: \( \text{new.d} = \text{removed.d} + 1 \leq \text{first.d} + 1 \) and \( \text{last.d} \leq \text{removed.d} + 1 = \text{new.d} \).

4. At termination, \( v.d = \delta(s, v) = \) shortest path length
Proof

Claim: at termination, $v.d = \delta(s, v) =$ shortest path length

Consider $v$ with minimum incorrect distance, and $u$ that is before it on a shortest path. $v.d > \delta(u) + 1 = u.d + 1$. When $u$ is dequeued:

if $v$ is undiscovered, it would then be correct, contradiction.
if $v$ is already finished, then $v.d \leq d.u$, contradiction.
if $v$ is discovered, let $w$ be predecessor. $v.d = w.d + 1$ and $w.d \leq u.d$ so $v.d \leq u.d + 1$, contradiction.
Graph Traversal
- Graphs
- Breadth-first
- The Algorithm
- Factoids
- Proof
- Break
- Depth-first Search
- Edges
- EOLQs

- asst 9
- midterm
- wildcard
Depth-first Search

DFS
1. forall vertices, label ← undiscovered
2. DFS-visit(start)

DFS-visit(u)
3. label u discovered
4. foreach neighbor v of u
5. if v is undiscovered
6. v.π ← u
7. DFS-visit(v)
8. label u finished

What’s the time complexity?
Discovery and finish times are parenthesized
Vs breadth-first?
tree: in depth-first tree
back: connects to ancestor in tree
forward: non-tree edge connecting to descendant in tree
cross: others: non-ancestors/non-descendants or different DFS tree

when edge is explored, label of arc dest gives type
For example:

- What’s still confusing?
- What question didn’t you get to ask today?
- What would you like to hear more about?

Please write down your most pressing question about algorithms and put it in the box on your way out.

*Thanks!*