Finding Shortest Paths in Unweighted Directed Graph
Single source shortest path - find path to all other vertices

$d=0$

$d=1$

$d=2$

Keep going until the desired destination is reached
How do we represent the solution?

Represent implicitly by storing the edge from parent.

shortest path

source is root

shortest path

tree
Shortest path tree

- STL
  - Flight from STL to BOS
  - Parent
    - BOS
      - Flight from BOS to PVD
      - Parent
        - PVD
Queue:  

Source: a

Parent:  

Extra:  

Tree:  

1. @
2. b
3. e
4. c
5. f
6. d
7. h
8. g
9. i
10. j

a → b → c → d → i → o

Red arrows indicate dependencies.
Time complexity

\( O(nm) \) (every vertex is placed into and removed from queue \( \leq 1 \) time)

\( O(n^2) \) (adj matrix)

\( O(nm) \) (adj list)

iterate over outedges of each vertex at most once each
Dijkstra's Shortest Path Algorithm

First let's review breadth first search (BFS) that solves the single-source shortest path algorithm for an unweighted graph.
bfs(V source)

For each u ∈ V

u.discovered = False
u.dist = ∞

u.parentEdge = null

source.discovered = true
source.dist = 0

queue.enqueue(source)

while (!queue.isEmpty())

u = q.dequeue()

for each edge e in outgoing edges from u

v = e.dest

if (!v.discovered)

v.discovered = true
v.dist = u.dist + 1
v.parentEdge = e

queue.enqueue(v)
Upon completion of DFS

any vertices not discovered are not reachable from source

For all discovered vertices \( v \)

\( v\text{.dist} \) is \# of edges in a shortest path from source to \( v \)

Following parent references from \( v \)
to \( s \) gives a shortest path from \( s \) to \( v \) (in reverse order)

Time complexity \( O(N+m) \) with adj list
Example from last class

\[ e_1, e_3, e_5 \]

Shortest path from a to h

Shortest path tree

X means it was discovered

Vdist shown in purple