Breadth-First Search

- find which verts are reachable from start
- finds shortest paths to reachable verts

1. Breadth-First Search:
   - $d=0$ for start
   - $d=v$ for each neighbor
   - $d[v] = d[u] + 1$
   - $d[v]$ = shortest distance from start
   - $pred[v]$ = vertex before $v$ on shortest path
   - $Q$ = queue of vertices to visit
   - $Q$ is empty
   - $d[v]$ = distance from start to $v$

2. Variables:
   - $d[v]$ = distance from start
   - $pred[v]$ = vertex before $v$ on shortest path
   - $Q$ = queue of vertices to visit

3. Steps:
   - Initialize $d[v]$ = unseen for all $v$
   - $Q$ = queue of vertices to visit
   - $pred[v]$ = vertex before $v$ on shortest path
   - While $Q$ not empty:
     - Dequeue $(u, v)$
     - For each outneighbor $v$ of $u$:
       - If $color[v] = \text{unseen}$:
         - $color[v] = \text{in queue}$
         - $pred[v] = u$
         - $d[v] = d[u] + 1$

4. Complexity:
   - $O(nm)$ for adj list
   - $O(n^2)$ for adj matrix
DFS-\text{VISIT}(u)
mark \( u \) as processing

- for each neighbors \( v \) of the current vertex \( u \)
- if \( v \) still unseen then DFS-\text{VISIT}(v)
- if \( v \) processing then cycle!

mark \( u \) as finished

\( O(n+m) \) for adj list
\( O(n^2) \) for adj matrix

\( \text{DFS VISIT}_{\text{t}}(1) \)
\( \text{DFS VISIT}_{\text{t}}(2) \)
\( \text{DFS VISIT}_{\text{t}}(3) \)
\( \text{DFS VISIT}_{\text{t}}(4) \)
\( \text{DFS VISIT}_{\text{t}}(5) \)
\( \text{DFS VISIT}_{\text{t}}(6) \)
\( \text{DFS VISIT}_{\text{t}}(7) \)
\( \text{DFS VISIT}_{\text{t}}(8) \)

Depth-First Search

finds all \( v \) with reachable from \( u \)
detects presence
Topological Sort

1. Topological sort
   (run DFS, record sets in reverse finish)

2. For each vertex in reverse order of topo sort
   long[v] ← max_{(v,u) is an edge} 1 + long[u]
   max = 0 if no edges

3. Longest path
   starting @ vertex

Dec 1\textsuperscript{i} 2022 Page 4