Theorem: Top-Sort produces a topological sort of a dag G
Proof: We need to show that if edge (u,v) is in G, then f[v]<f[u].
Consider any edge (u,v) explored by DFS(G).
When (u,v) is explored, v cannot be gray (because graph is acyclic).
Therefore, v is either white or black.
If v is white, DFS moves to v and so f[v]<f[u].
If v is black, then f[v]<f[u].
**Topological Sort** of a directed acyclic graph (dag)

a linear ordering of all vertices such that if $G$ contains edge $(u,v)$, then $u$ appears before $v$ in the ordering

acyclic graph -- has no cycles

**Top-Sort($G$)**
call DFS($G$) to compute finishing times $f[v]$ for each vertex $v$ as each vertex is finished, insert it onto the front of a linked list
return the linked list of vertices

Running time: $\Theta(V+E)$
Running time: of DFS in terms of $|V|$ and $|E|$

DFS(G): $\Theta(V)$ time (not counting time spend on DFS-Visits)

DFS-Visit: is called only on a white vertex
- once it is called on vertex $v$, $v$ is colored non-white
- thus, DFS-Visit is run once on each vertex
- for each $u$, $\text{Adj}[u]$ is scanned once
- lengths of all adjacency lists is $E$

total time spent in DFS-Visits is $\Theta(E)$

$$T(n) = \Theta(V+E)$$

Predecessor subgraph: $G_\pi=(V,E_\pi)$,

$$E_\pi = \{ (\pi[v],v) : v \in V \text{ and } \pi[v] \neq \text{NIL} \}$$

Predecessor subgraph forms depth-first forest, made up of depth-first trees

Classification of edges:
1. tree edges (t)
2. back edges (b)
3. forward edges (f)
4. cross edges (c)
Depth-first search (DFS):
like BFS, but searches as deeply as possible first

DFS(G)
for each u in V
    color[u]=white   \pi[u]=NIL
    time=0
for each u in V
    if color[u]=white then DFS-Visit(u)

DFS-Visit(u)
    color[u]=gray
    time=time+1
    d[u]=time
    for each v ∈ Adj [u]
        if color[v]=white then DFS-Visit(v)
            \pi[v]=u
    color[u]=black
    time=time+1
    f[u]=time
Running time of BFS in terms of $|V|$ and $|E|$

loop 1: $\Theta(V)$
while loop:
  operations on Q: a vertex is whitened only once (in loop 1)
  thus, it is enqueued at most once, hence dequeued at most once
  enqueue or dequeue operations take $\Theta(1)$ time
  total time spent on queue operations $O(V)$
operations of the inner for loop:
  since Adj[u] is scanned only when u is dequeued,
    Adj[u] is scanned at most once
  lengths of all adjacency lists is E
  for each vertex scanned on adjacency list, we spend $\Theta(1)$ time
  total time spent in the inner for loop is $O(E)$

$T(n) = O(V+E)$

What does $d[u]$ mean?

shortest-path distance $\delta(s, v)$ is
the minimum number of edges on any path from s to v

Theorem: $d[v]$ at the end of BFS is $\delta(s, v)$
Breadth-first search (BFS)
given a graph and a source vertex s, BFS searches systematically through the graph to “discover” every vertex

color: white=undiscovered, gray=on the frontier, black=done

BFS(G,s)
for each vertex u in V-{s}
  color[u]=white   d[u]=∞
color[s]=gray    d[s]=0  Q={s}
while Q ≠ ∅
  u=head[Q]
  for each v ∈ Adj [u]
    if color[v]=white then
      color[v]=gray
      d[v]=d[u]+1
      Enqueue(Q,v)
  Dequeue(Q)
color[u]=black
TOPIC 16: Breadth-First Search and Depth-First Search

Graph Representations for G=(V,E)

Adjacency list: array Adj of |V| lists, one for each vertex in V
Adj[u] is a linked list of all the vertices v such that edge (u,v) in E

Array Adj takes O(E) space

Adjacency matrix: |V|x|V| matrix A, where $a[i,j] = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$

Matrix A takes O(V^2) space

Variations: weighted graphs, directed graphs, sparse vs. dense graphs