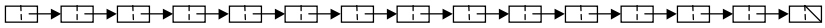
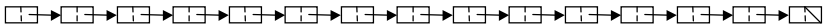


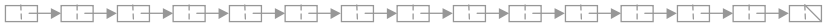
CSE 373: Graphs

(Traversals, Shortest Paths)

Chapter 9

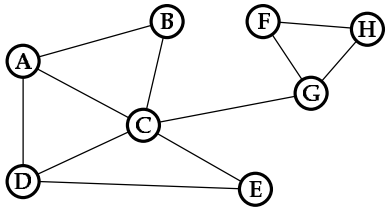


Depth-First Search (DFS)



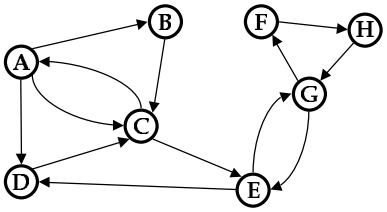
A fundamental method for traversing a graph:

- start from a particular node…
- walk along a single path visiting nodes
- go back and try a different path only when stuck



DFS on Directed Graphs

Also applicable to directed graphs…



- Similar to pre-order traversal on tree/DFS on maze
- Running time?
- edges walked by DFS form a *spanning tree*
- how would we implement this?

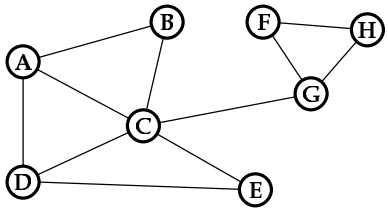
DFS Implementation

```
void DFS(vertex v) {




}
```

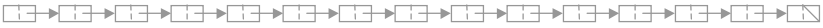
Breadth-First Search (BFS)

Another way of traversing a graph

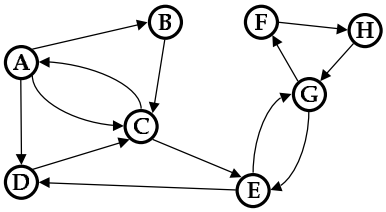
- start from a particular node…
- take one step along each of its edges
- then take one step per edge for each of those nodes



BFS on Directed Graphs



Also applicable to directed graphs…



- Running time?
- visited edges again form a spanning tree
- how would we implement this?

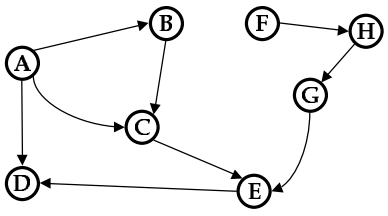
BFS Implementation

```
void BFS(vertex v) {



}
```


Topological Sort

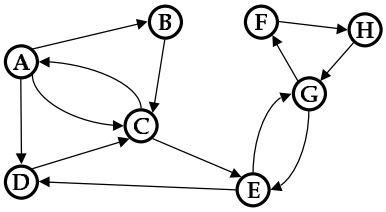
A notion of ordering the vertices in a DAG:

- If there is a path from *u* to *v*, *u* must appear before *v* in the ordering
- OR, a vertex v may not be printed out until all vertices with edges leading to it have been printed

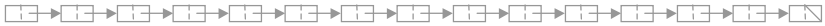


Topological Sort on non-DAGs

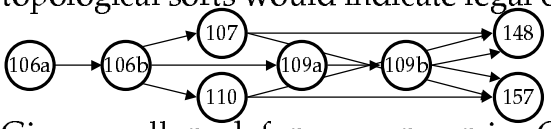
Why don't topological sorts make sense on graphs that aren't DAGs?



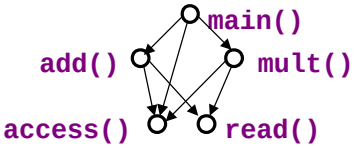
Topological Sort Applications



- Given a graph representing course prerequisites, topological sorts would indicate legal course schedules



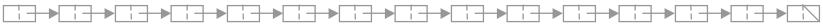
- Given a callgraph for a non-recursive C program, topological sorts would indicate function orderings for the source file such that no prototypes are needed



Using Searches for Topological Sort?

- Would a depth-first search visit nodes in topological order?
- Would a breadth-first search?
- What would?

Naive Topological Sort



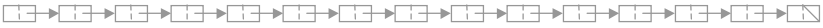
- Naive algorithm:
  - Keep track of each vertex's *in-degree:* the number of edges leading into it
  - Scan the vertex list looking for one whose in-degree is zero
  - Print that vertex out
  - Decrement the in-degrees of all adjacent vertices
- Running Time?
- How could this be improved?

Improved Topological Sort

```
void TopSort(Graph G) {



}
```

Running Time?

Shortest Path Problems



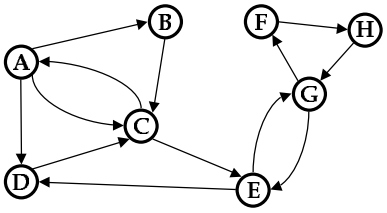
Given a graph $G = (V,E)$ and a vertex $s \in V$, find the shortest path from s to all other vertices

Many variations:

- unweighted graphs
- weighted graphs with no negative weights
- weighted graphs with negative weights
- weighted acyclic graphs

Unweighted Shortest Path Problem

Assume source vertex is **C**…

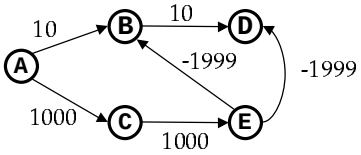


distance to: **A B C D E F G H**

What approach could we use to implement this?

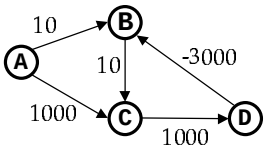
The Problem With Negative Weights

case 1:



A B C D E

case 2:



A B C D