Topological sorting and shortest paths

1/30/15
Topological sorting

• Take a directed graph:

• Produce an ordering so all edges go forward (see red numbers)
Directed Acyclic Graph (DAG)

• Directed graph that has no directed cycles
Directed Acyclic Graph (DAG)

• Directed graph that has no directed cycles

• Claim: If G can be topologically sorted, it is a DAG
Directed Acyclic Graph (DAG)

- Directed graph that has no directed cycles

- Claim: If G can be topologically sorted, it is a DAG

- Claim: If G is a DAG, it can be topologically sorted
Algorithm based on DFS

Modified_DFS(G, u):
    mark u
    for each neighbor v of u
        if(v is unmarked)
            Modified_DFS(G,v)
    add u to front of the top-sorted list

Works because u is added after all its descendents.
Thus, it proceeds them in the final list
Graph practice problem

You’re helping a group of ethnographers analyze oral histories they’ve collected. They’ve learned about n people P₁, P₂, ..., Pₙ. The histories contain information relating their lives. The information comes in two forms:

• For some i and j, person Pᵢ died before person Pⱼ was born
• For some i and j, the lives of people Pᵢ and Pⱼ overlapped

Since the information is from people’s memories, it may be inaccurate. You are asked to give an efficient algorithm to determine whether the given information is internally consistent, i.e. whether it is possible for these people to have birth and death dates that make all the information correct.
Shortest Paths
Floyd-Warshall: All-pairs shortest paths

Creates matrix $D$; $D_{i,j}$ is min distance from $i$ to $j$
Floyd-Warshall: All-pairs shortest paths

Creates matrix D; $D_{i,j}$ is min distance from i to j

When i,j are connected, set $D_{i,j}$ with edge weight
Others get $\infty$
Floyd-Warshall: All-pairs shortest paths

Creates matrix $D$; $D_{i,j}$ is min distance from $i$ to $j$

When $i,j$ are connected, set $D_{i,j}$ with edge weight
Others get $\infty$

for $k = 1$ to $|V|$
    for $i = 1$ to $|V|$
        for $j = 1$ to $|V|$
            $D_{i,j} = \min\{D_{i,j}, D_{i,k} + D_{k,j}\}$
Floyd-Warshall: All-pairs shortest paths

Creates matrix $D$; $D_{i,j}$ is min distance from $i$ to $j$

When $i,j$ are connected, set $D_{i,j}$ with edge weight
Others get $\infty$

for $k = 1$ to $|V|$
  for $i = 1$ to $|V|$
    for $j = 1$ to $|V|$
      $D_{i,j} = \min\{D_{i,j}, D_{i,k} + D_{k,j}\}$

After iteration $k$, $D_{i,j}$ has min distance for paths using only
vertices 1..$k$ as intermediaries
Dijkstra’s: Single source shortest paths

Compute distance from vertex s to all others
Each vertex v has estimate $d_v$ (becomes distance)
Dijkstra’s: Single source shortest paths

Compute distance from vertex s to all others
Each vertex v has estimate $d_v$ (becomes distance)

$d_s = 0$, $d_{\text{others}} = \infty$
Q = priority queue of all vertices (based on d)
Dijkstra’s: Single source shortest paths

Compute distance from vertex s to all others
Each vertex v has estimate \( d_v \) (becomes distance)

\[ d_s = 0, \quad d_{\text{others}} = \infty \]

Q = priority queue of all vertices (based on \( d \))
while Q is non-empty

\[ u = \text{extract-min}(Q) \]

for each neighbor v of u

\[ d_v = \min\{d_v, d_u + \text{weight of (u,v) edge}\} \]
//updates to \( d_v \) affect state of Q