Graph Algorithms

CptS 223
Getting from here to there...

What data structure would you use?

How do you even think about the problem?

Lets strip away the irrelevant details..
Getting from here to there...

Vertices
Getting from here to there...

Edges
Getting from here to there...

Weights
Getting from here to there...

looks a lot more simpler now...
Graphs

- Are a tool for modeling real world problems.
- Allow us to abstract details and focus on the problem.
- We can represent our domain as graphs apply algorithms to solve our problem.
Simple Graphs

\[ G = (V, E) \]

\[ (u, v) \in E \]
Directed Graphs

\[(u, v) \in E\]

\[(v, u) \not\in E\]
Weighted Graphs
Representation of graphs

Adjacency Matrix

\[
\begin{array}{ccc}
A & B & C \\
A & 0 & 1 & 1 \\
B & 1 & 0 & 1 \\
C & 1 & 1 & 0 \\
\end{array}
\]

| \( V^2 \) |

Adjacency List

\[
\begin{array}{c}
A \\
|V| + |E| \\
B \\
C \\
\end{array}
\]
Use Adjacency lists for Sparse Graphs
Topological Sort

A, B, D, C, E, F, G, H

No Cycles
Topological Sort

In degree = 3
Out degree = 1
void Graph::topsort( )
{
    for( int counter = 0; counter < NUM_VERTEXES; counter++ )
    {
        Vertex v = findNewVertexOfIndegreeZero( );
        if( v == NOT_A_VERTEX )
            throw CycleFoundException( );
        v.topNum = counter;
        for each Vertex w adjacent to v
            w.indegree--;
    }
}

$O(|V|^2)$
void Graph::topsort()
{
    Queue<Vertex> q;
    int counter = 0;

    q.makeEmpty();
    for each Vertex v
        if( v.indegree == 0 )
            q.enqueue( v );

    while( !q.isEmpty() )
    {
        Vertex v = q.dequeue();
        v.topNum = ++counter; // Assign next number

        for each Vertex w adjacent to v
            if( --w.indegree == 0 )
                q.enqueue( w );
    }

    if( counter != NUM_VERTICES )
        throw CycleFoundException();
}

\[ O(|V| + |E|) \]