



Graph Algorithms

CptS 223 – Advanced Data Structures

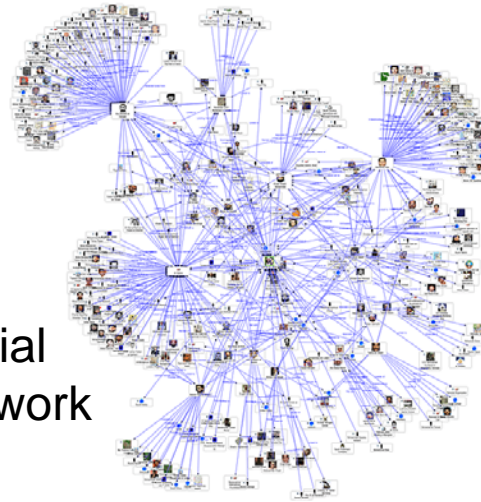
Larry Holder

School of Electrical Engineering and Computer Science
Washington State University

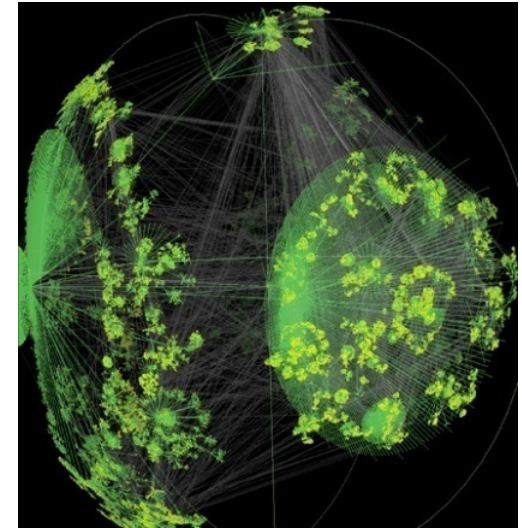
Graphs



Protein-protein Interaction

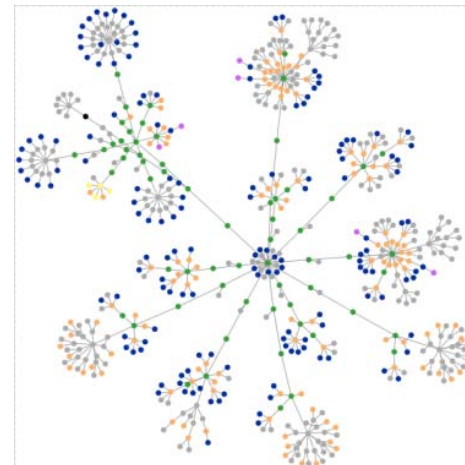
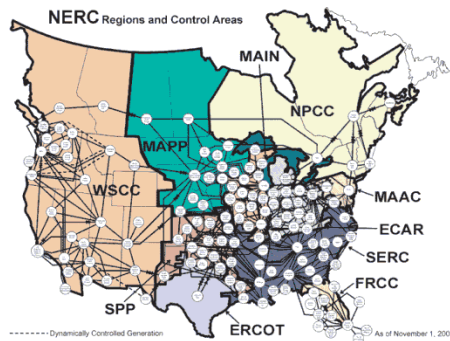


Social Network



Internet

Power Grid

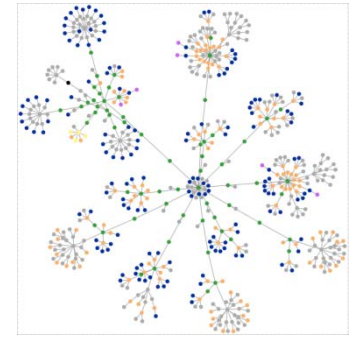


Web

Some Graph Statistics

■ Web

- 10B pages, 1T hyperlinks
- Topology storage: 10TB
- Google PageRank: Eigenvector on $10B \times 10B$ adjacency matrix (sparse)



■ MySpace

- 100M users, 10B friendship links
- Clique/community detection
- 300K new users per day





Graph Problems

- Degree
- Diameter
- Centrality
- Shortest path
- Cycles/tours
- Minimum spanning tree
- Traversals/search
- Connectivity
- Clustering
- Partitioning
- Cliques
- Motifs
- Subgraph isomorphism
- Frequent subgraphs
- Pattern learning
- Dynamics

Definitions

- Graph $G = (V, E)$ consists of vertices V and edges E

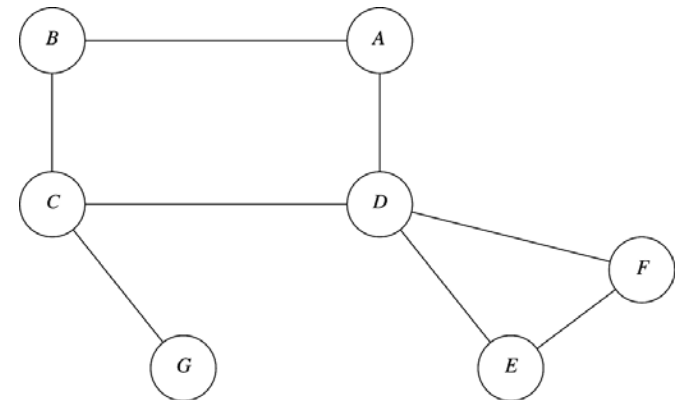
- $E = \{(u, v) \mid u, v \in V\}$

- v is adjacent to u

- E.g.,

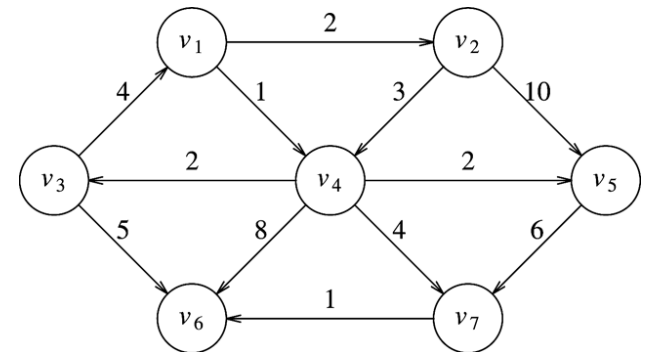
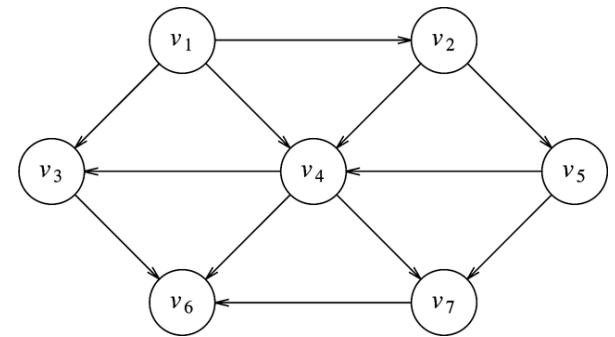
- $V = \{A, B, C, D, E, F, G\}$

- $E = \{(A, B), (A, D), (B, C), (C, D), (C, G), (D, E), (D, F), (E, F)\}$



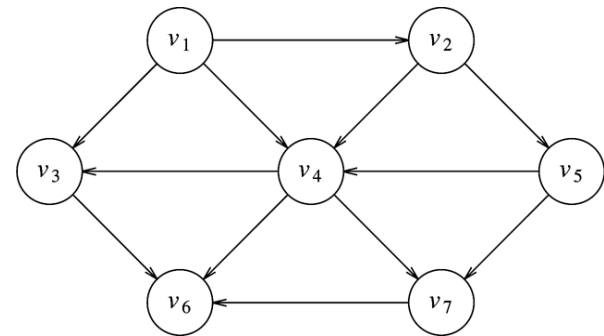
Definitions

- Undirected graph
 - Edges are unordered
- Directed graph
 - Edges are ordered
- Weighted graph
 - Edges have a weight $w(u,v)$ or cost $c(u,v)$



Definitions

- Degree of a vertex
 - Number of edges incident on a vertex
- Indegree
 - Number of directed edges to vertex
- Outdegree
 - Number of directed edges from vertex



$\text{degree}(v_4) = 6$
 $\text{indegree}(v_4) = 3$
 $\text{outdegree}(v_4) = 3$

$\text{indegree}(v_1) = 0$
 $\text{outdegree}(v_1) = 3$

$\text{indegree}(v_6) = 3$
 $\text{outdegree}(v_6) = 0$

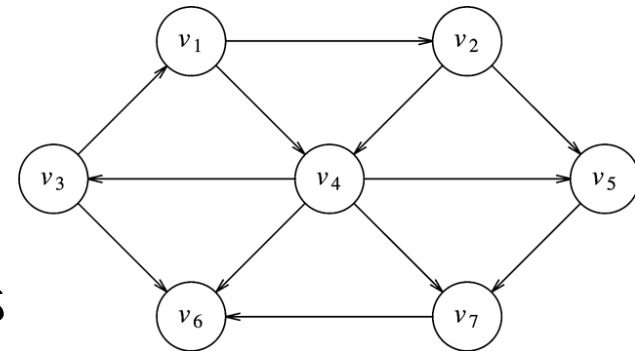
Definitions

■ Path

- Sequence of vertices v_1, v_2, \dots, v_N such that $(v_i, v_{i+1}) \in E$ for $1 \leq i < N$
- Path length is number of edges on path (N-1)
- Simple path has unique intermediate vertices

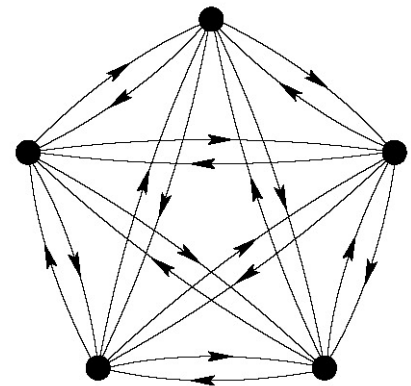
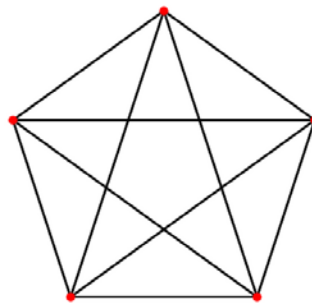
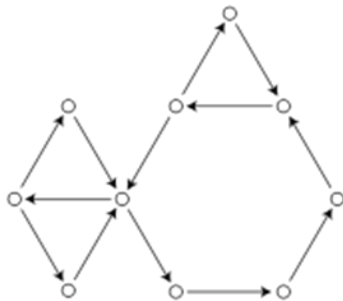
■ Cycle

- Path where $v_1 = v_N$
- Usually simple and directed
- Acyclic graphs have no cycles

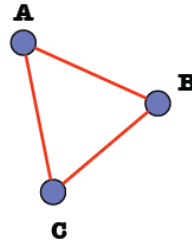


Definitions

- Undirected graph is connected if there is a path between every pair of vertices
- Connected, directed graph is called strongly connected
- Complete graph has an edge between every pair of vertices



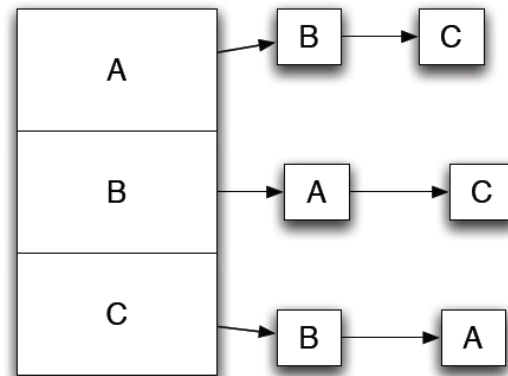
Representation



	A	B	C
A	0	1	1
B	1	0	1
C	1	1	0

Adjacency Matrix

$$|V^2|$$

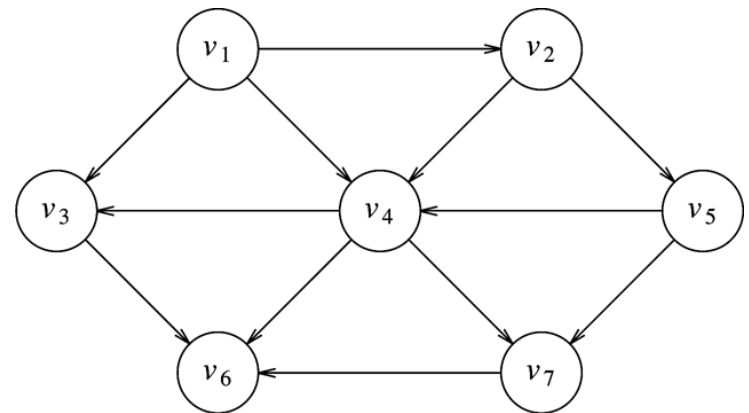
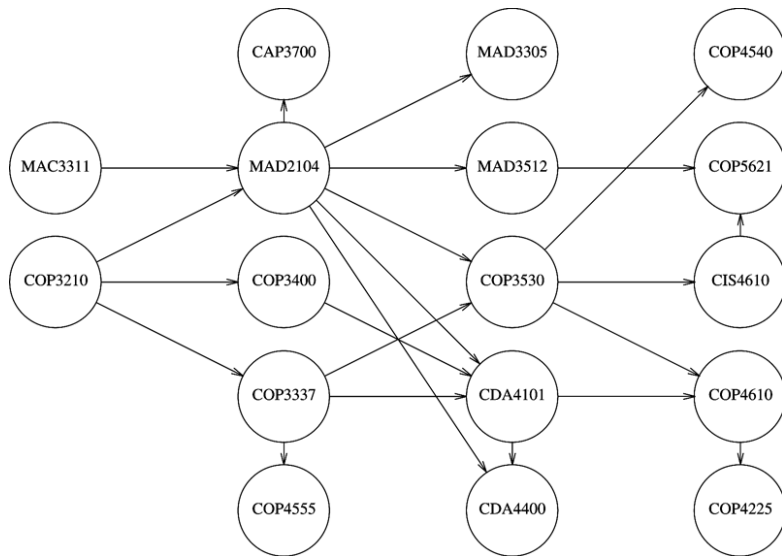


Adjacency List

$$|V| + |E|$$

Topological Sort

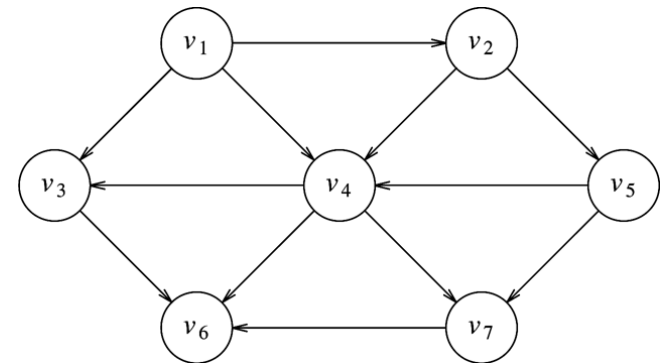
- Order vertices in a directed, acyclic graph such that if $(u, v) \in E$, then u before v in the ordering



Topological order:
 $v_1, v_2, v_5, v_4, v_3, v_7, v_6$

Topological Sort

- Solution #1
 - While vertices left in graph // $O(|V|)$
 - Find vertex v with indegree = 0 // $O(|V|)$
 - Output v
 - Remove edges to/from v
 - $T(V,E) = O(|V|^2)$

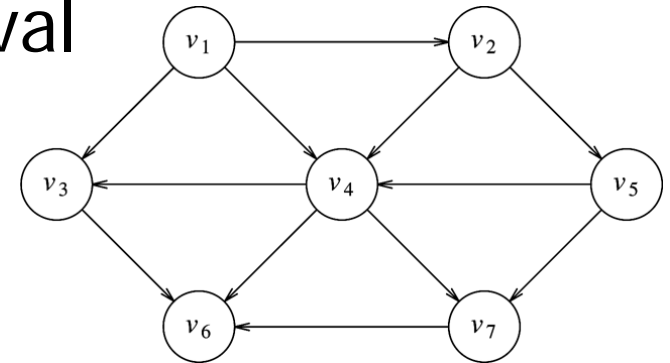


Topological order:
v1, v2, v5, v4, v3, v7, v6

Topological Sort

■ Solution #2

- Don't need to search all vertices for indegree = 0
- Only vertices who lost an edge from the previous vertex's removal
- $T(V,E) = O(|V| + |E|)$
- Note: $|E| = O(|V|^2)$



Topological order:

v1, v2, v5, v4, v3, v7, v6 13

```

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

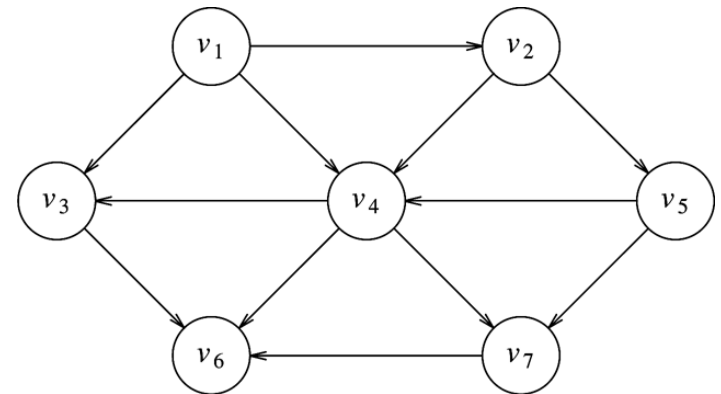
        for each Vertex w adjacent to v
            if( --w.indegree == 0 )
                q.enqueue( w );
    }

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

```

Topological Sort

Vertex	Indegree Before Dequeue #						
	1	2	3	4	5	6	7
v_1	0	0	0	0	0	0	0
v_2	1	0	0	0	0	0	0
v_3	2	1	1	1	0	0	0
v_4	3	2	1	0	0	0	0
v_5	1	1	0	0	0	0	0
v_6	3	3	3	3	2	1	0
v_7	2	2	2	1	0	0	0
<i>Enqueue</i>	v_1	v_2	v_5	v_4	v_3, v_7		v_6
<i>Dequeue</i>	v_1	v_2	v_5	v_4	v_3	v_7	v_6





Graph Algorithms

- Topological Sort
- Shortest paths
- Network flow
- Minimum spanning tree
- Applications
- NP-Completeness