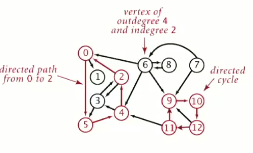
Directed Graphs (Digraphs)

*Edges now have directions*

Digraph: set of connected vertices pairwise by directed edges

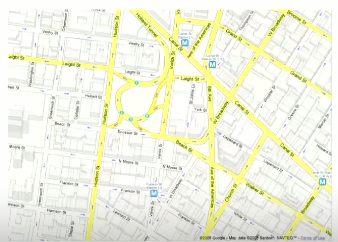


The order of the pair matters. The edge goes from one vertex to another (undirected graphs just have connections). If we travel graphs on edges, we must follow the directions of the edges.

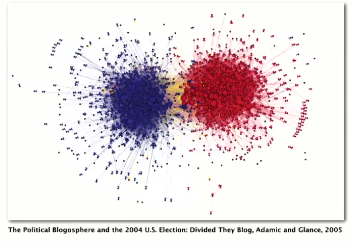
Digraphs have indegrees (number of variables coming in to the vertex) and outdegrees (number of variables leaving the vertex).

Digraph applications

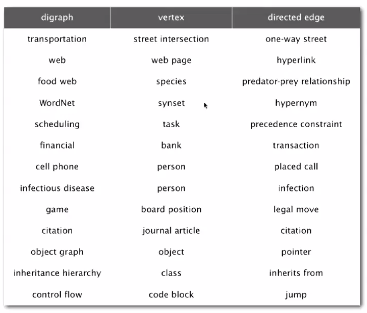
* Road networks (think 2-way and 1-way streets)



* Political blogosphere graph (vertex = political blog; edge = link)

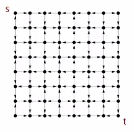


Other applications:



Digraph problems

**Path**: Is there a directed path from s to t?



**Shortest path**: what is the shortest directed path from s to t?

**Topological sort**: Can you draw a digraph so that all edges point upwards?

**Strong connectivity**: Is there a directed path between all pairs of vertices?

**Transitive closure**: For which vertices v and w is there a path from w to v?

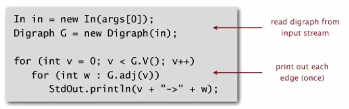
**PageRank:** What is the importance of a web page?

Digraph API

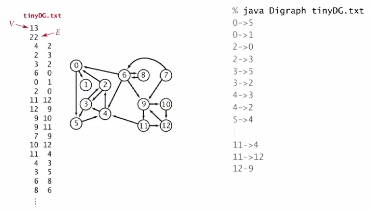
Public class Digraph  
Digraph(int V) : create empty digraph with V vertices  
Digraph(In in) : create a digraph from an input stream  
void addEdge(int v, int w) : add a directed edge v -> w  
Iterable<Integer> adj(int v) : vertices pointing from v  
int V() : number of vertices  
int E() : number of edges  
Digraph reverse() : reverse of this digraph  
String toString() : string representation

---

Implementation of sample client

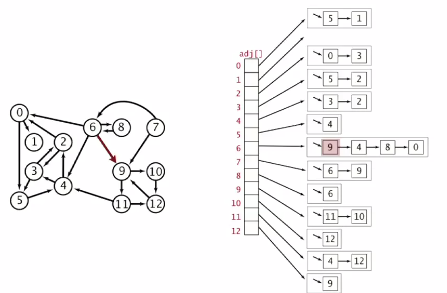


’



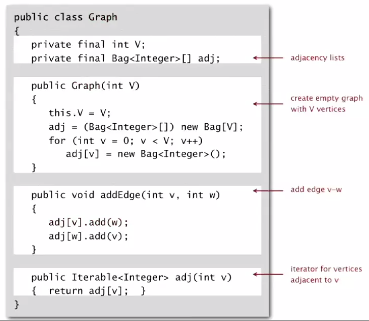
Adjacency-lists digraph representation

* Maintain vertex-indexed array of lists

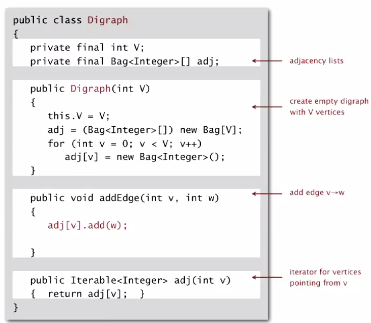


Adjacency-lists graph representation..

Compare Graph implementation:

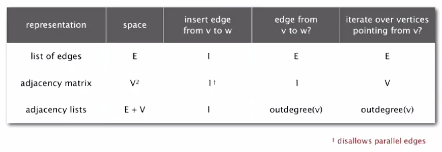


And the Digraph implement:



Only difference aside from name is that there is no ‘adj[w].add(v)’ in addEdge()

Performance



Graph processing algorithms summary

