CSC212
Data Structure
- Section FG

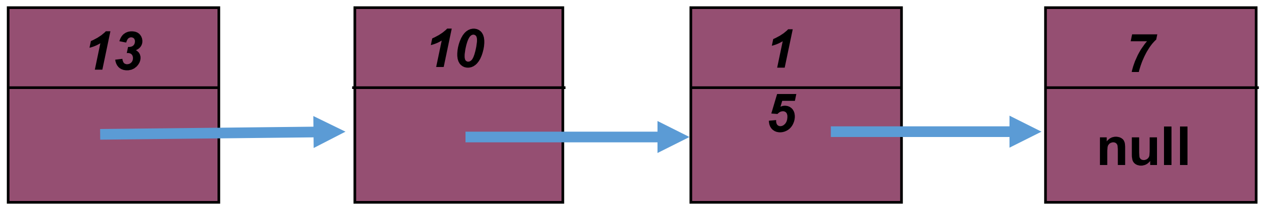
COMPUTER SCIENCE
CITY COLLEGE OF NEW YORK

Lecture 23
Introduction to Graphs

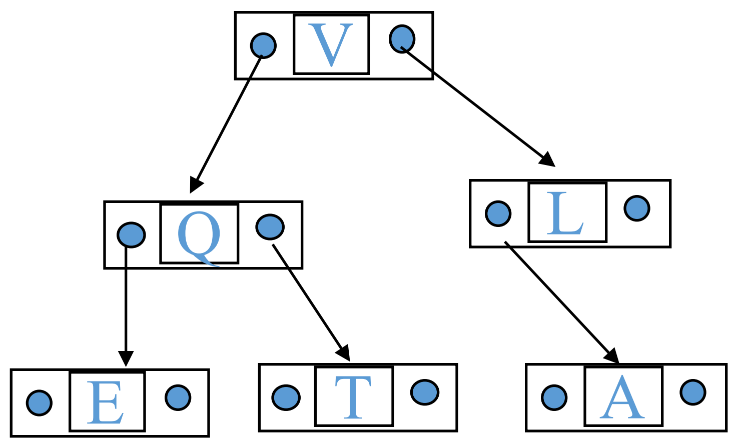
Instructor: Feng HU
Department of Computer Science
City College of New York

Review

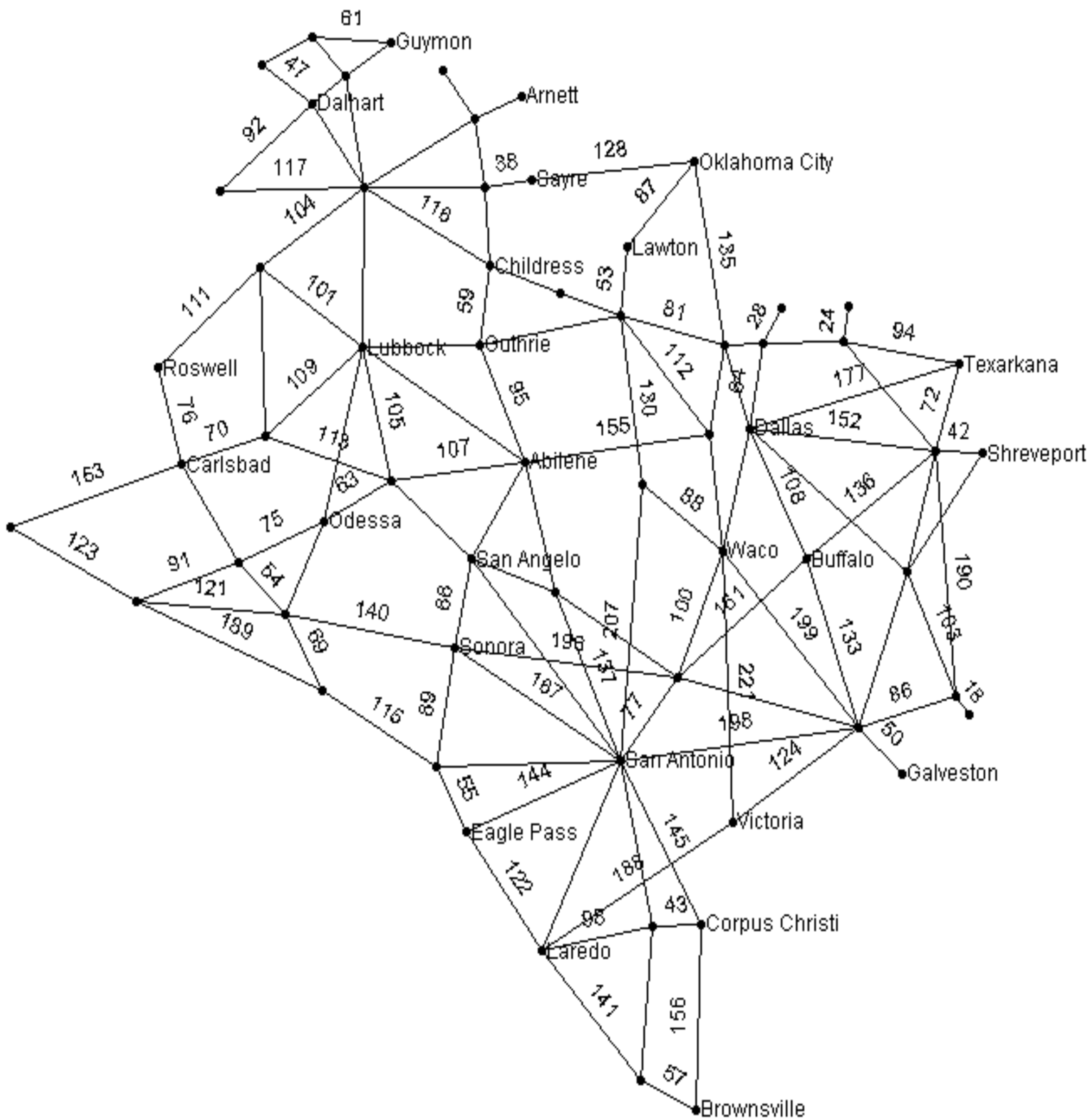
- Linked Lists



- Binary Trees



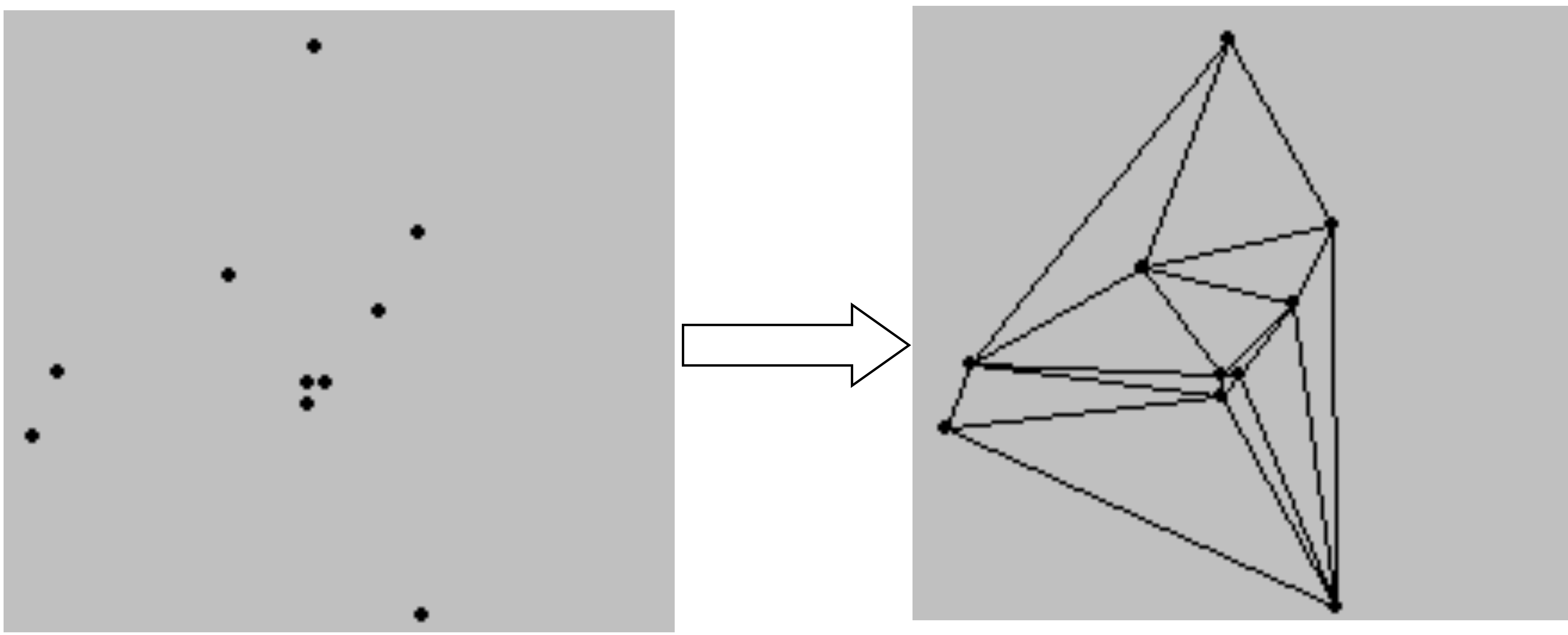
Examples: Texas Road Network



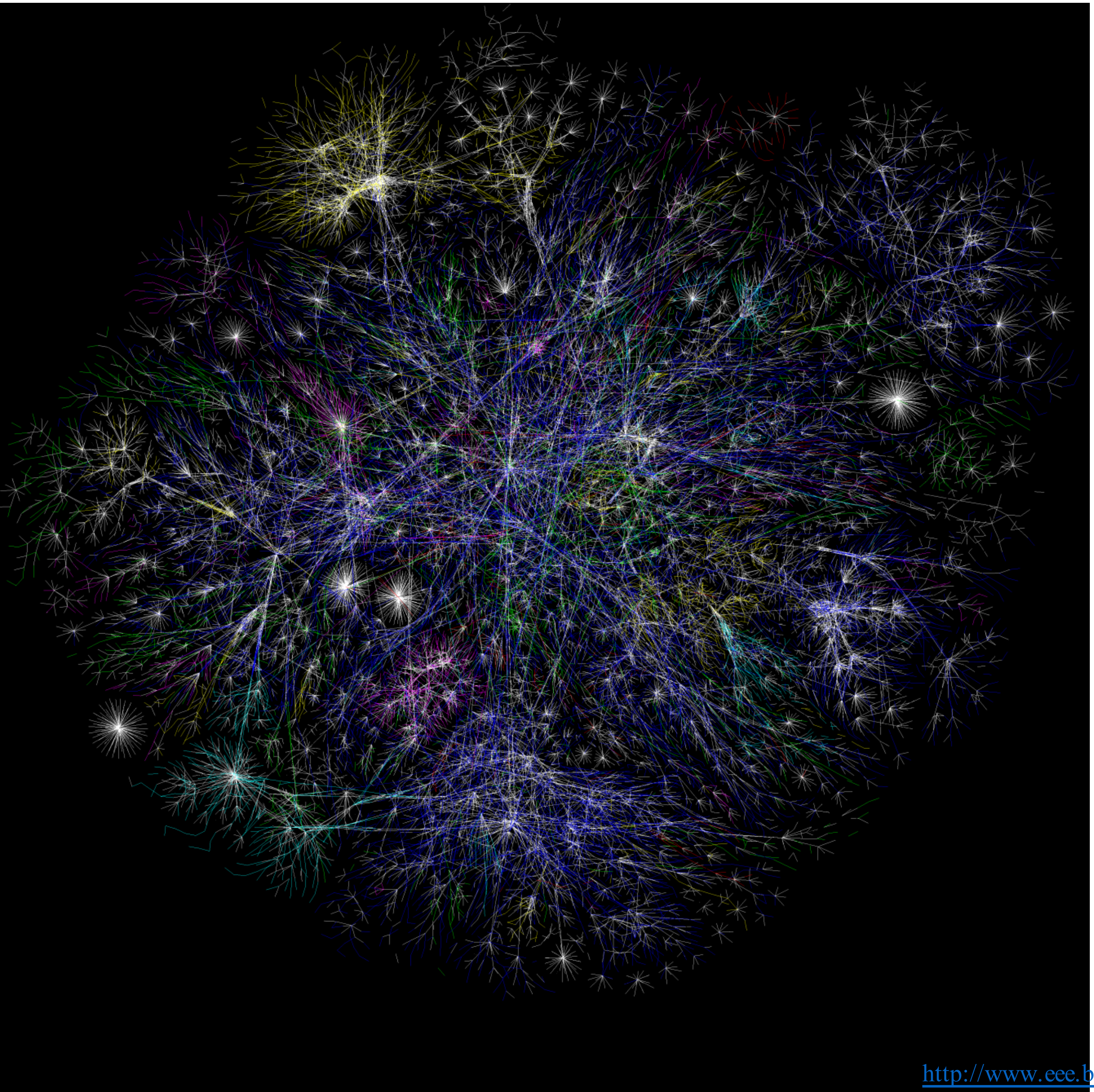
- 62 cities (nodes)
- 120 major roads (edges)
- Minimum distance between Dallas and Corpus Christi?

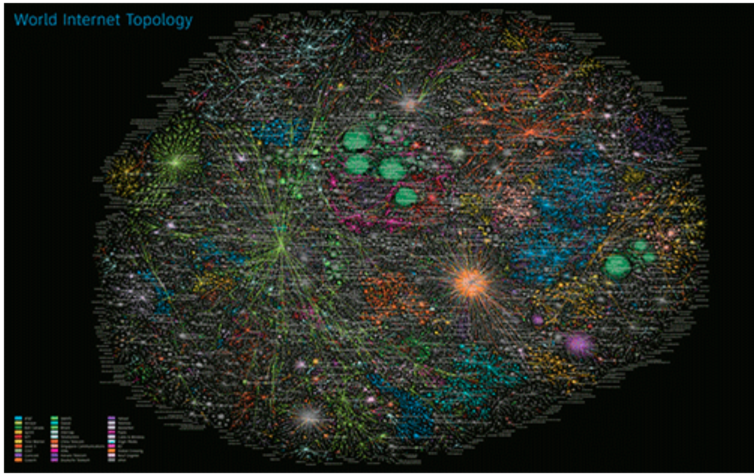
Examples:

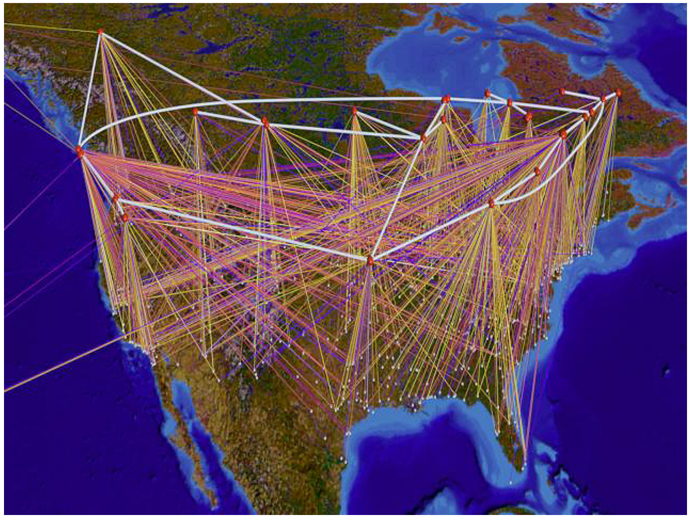
Delaunay Triangulation Network From a Point Data Set



Examples: Internet Topology



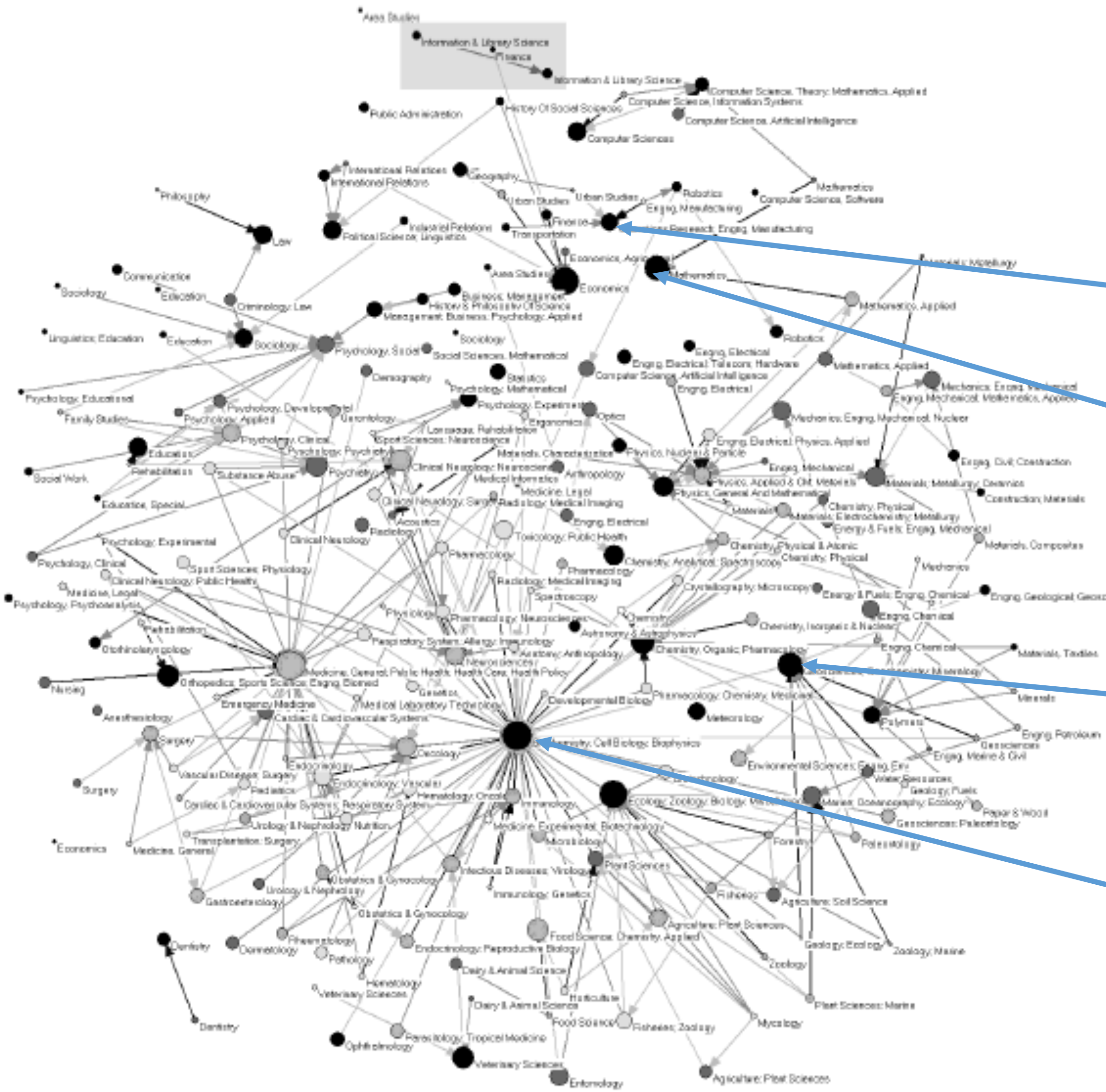




http://www.eee.bham.ac.uk/com_test/img%5Cdsnl%5Cinternet15jan06.png (01/16/200

Examples: Backbone of science

With 212 clusters comprising 7000 journals



Kevin W Boyack, Richard Klavans, Katy Börner, Mapping the backbone of science, Scientometrics, Vol. 64, No. 3. (2005), pp. 351-374.

Computer Science

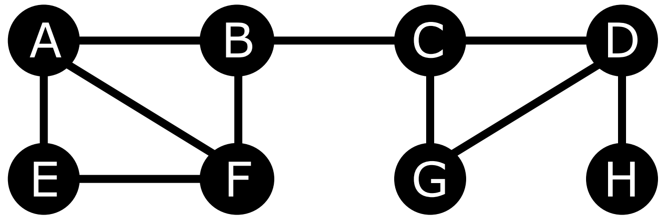
Mathematics

Geosciences, Geochemistry, Mineralogy

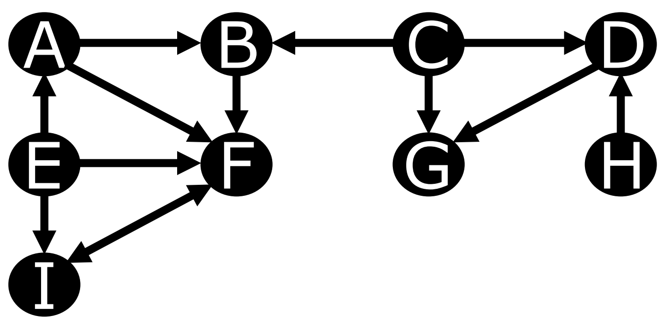
Biochemistry, Cell Biology, Biophyiscs

Terminologies

- A graph G = (V, E)
  - V: vertices
  - E : edges, pairs of vertices from $V \times V$
- Undirected Graph:
  - (u,v) is same as (v,u)



- Directed Graph
  - (u,v) is different from (v,u)
  - Source (u)
  - Target (v)



More Terminologies – P 728

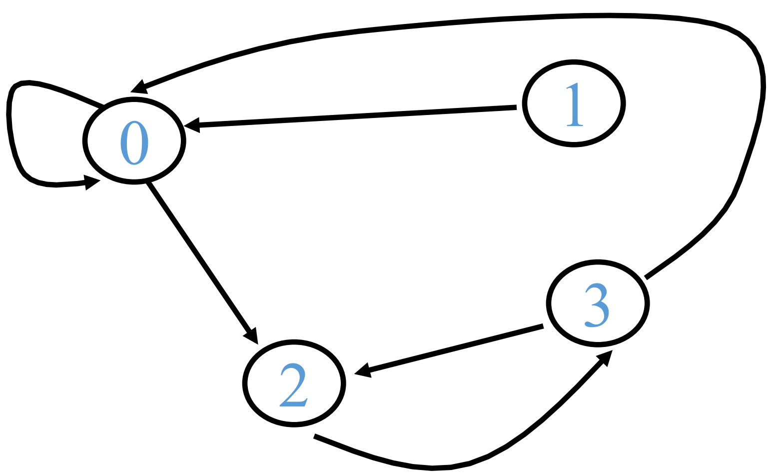
- Loop: an edge that connects a vertex to itself.
- Path: a sequence of vertices, $p_0, p_1, \ldots p_m$, such that each adjacent pair of vertices p_i and p_{i+1} are connected by an edge.
- Multiple Edges: two or more edges connecting the same two vertices in the same direction.
- Simple graph: have no loops and no multiple edges – required for many applications.
- Weighted graph and unweighted graph

Representations- Adjacency Matrix

An *adjacency matrix* represents the graph as a *n* x *n* matrix A:

$A[i, j] = 1$ if $(i, j) \in E$ (or weight of edge)

$= 0$ if $(i, j) \notin E$

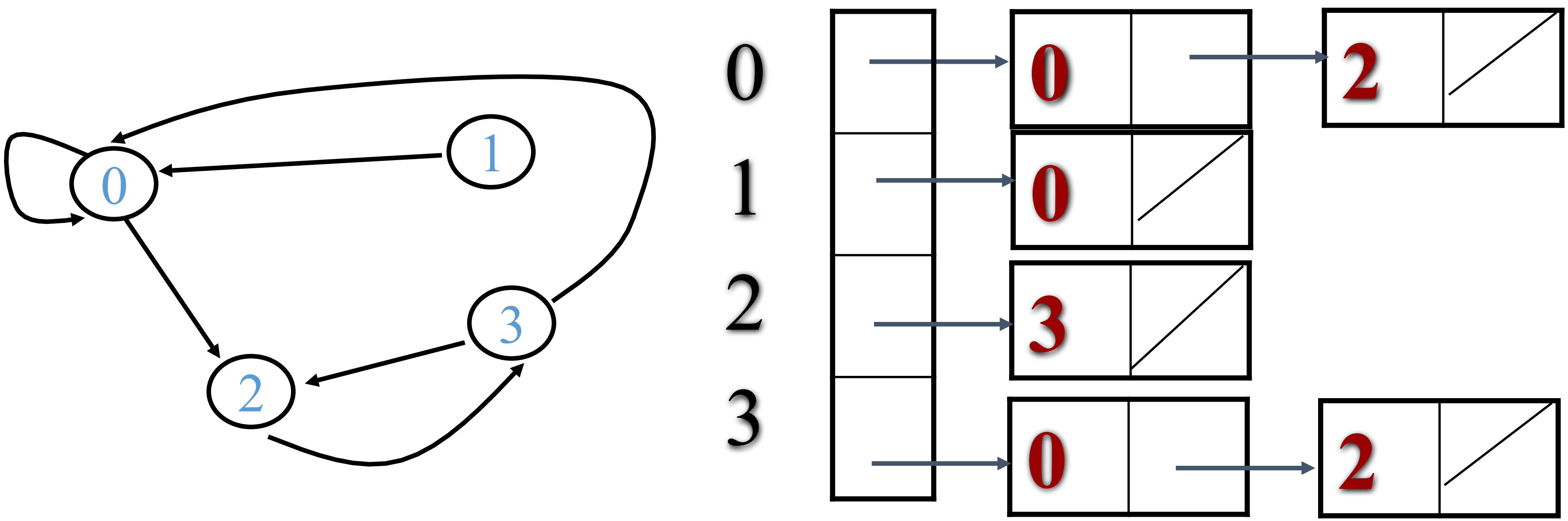


	0	1	2	3
0	1	0	1	0
1	1	0	0	0
2	0	0	0	1
3	1	0	1	0

Space Complexity with respect to |V| and/or |E|?

Representations-Linked List

A directed graph with n vertices can be represented by n different linked lists. List number i provides the connections for vertex i. To be specific: for each entry j in the list number i, there is an edge from i to j.



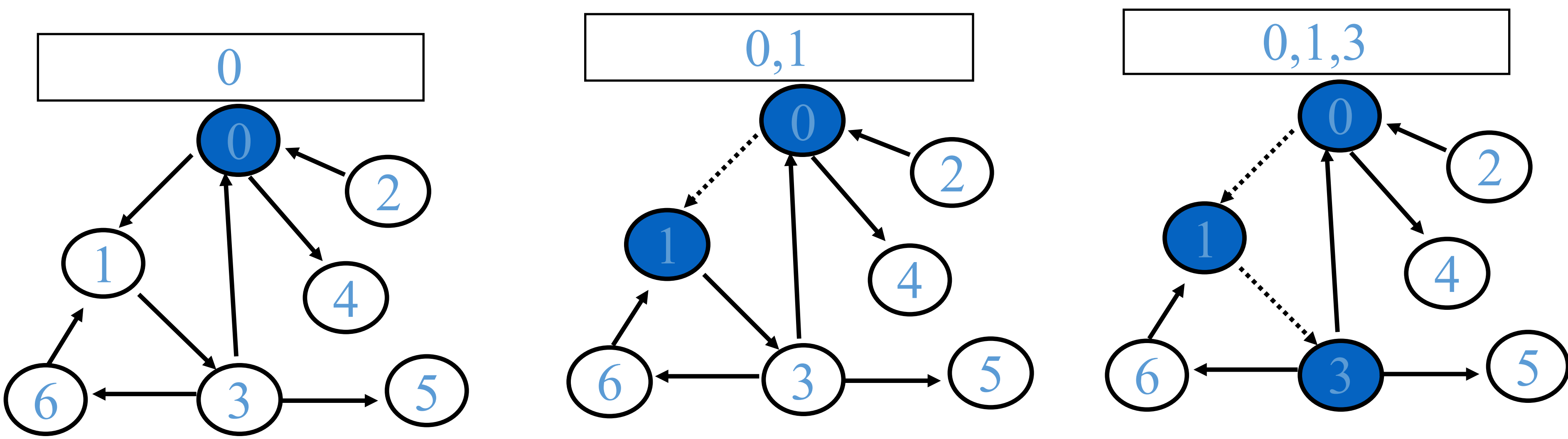
Space Complexity with respect to |V| and/or |E|?

Graph Traversals

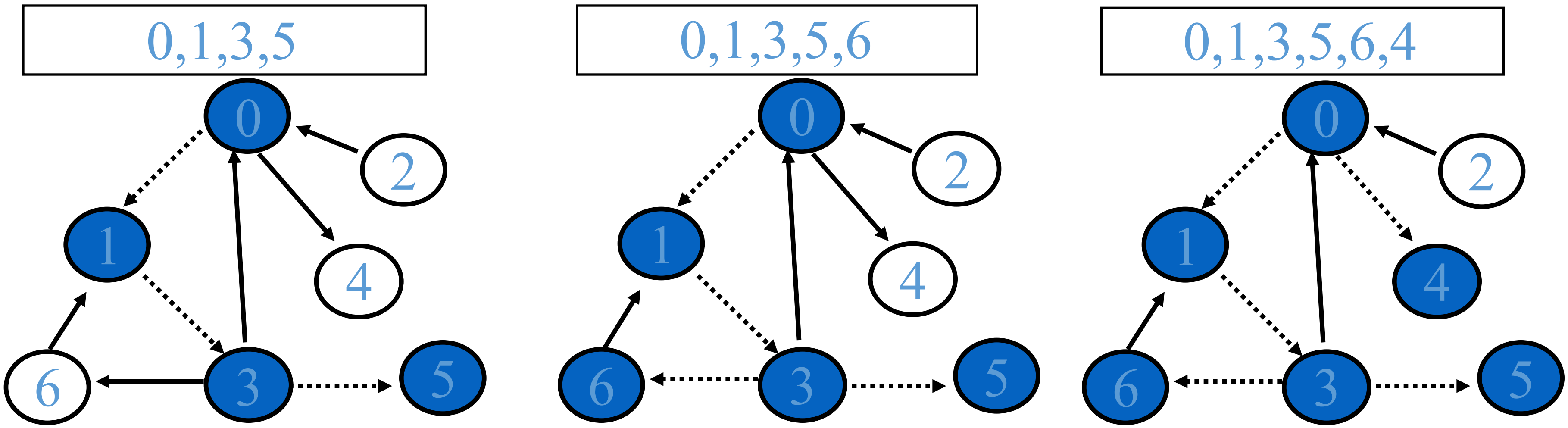
- Traversal:
  - Tree traversals (ch 10): visit all of a tree's nodes and do some processing at each node
- Types of Graph traversals
  - Depth First Search (DFS)
  - Breadth First Search (BFS)
- Issues to consider
  - There is no root – need a start vertex
  - Be careful and do not enter a repetitive cycle – mark each vertex as it is processed

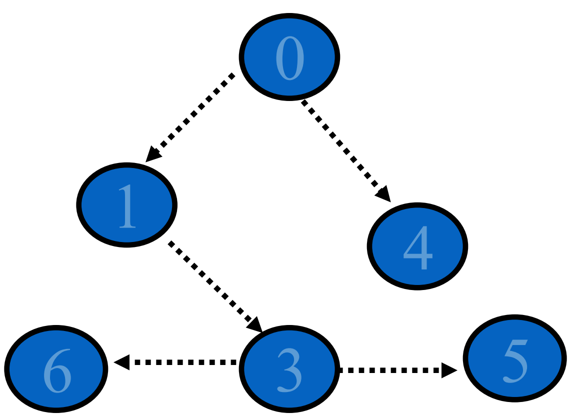
Graph Traversal-Recursive DFS

- DFS(*start*)
  - Initialize the boolean *visited* array
  - Rec_DFS(*start*)
- Rec_DFS(*start*)
  - For each of unvisited neighbor *next* of *start*
    - Rec_DFS(*next*)



Graph Traversal-DFS





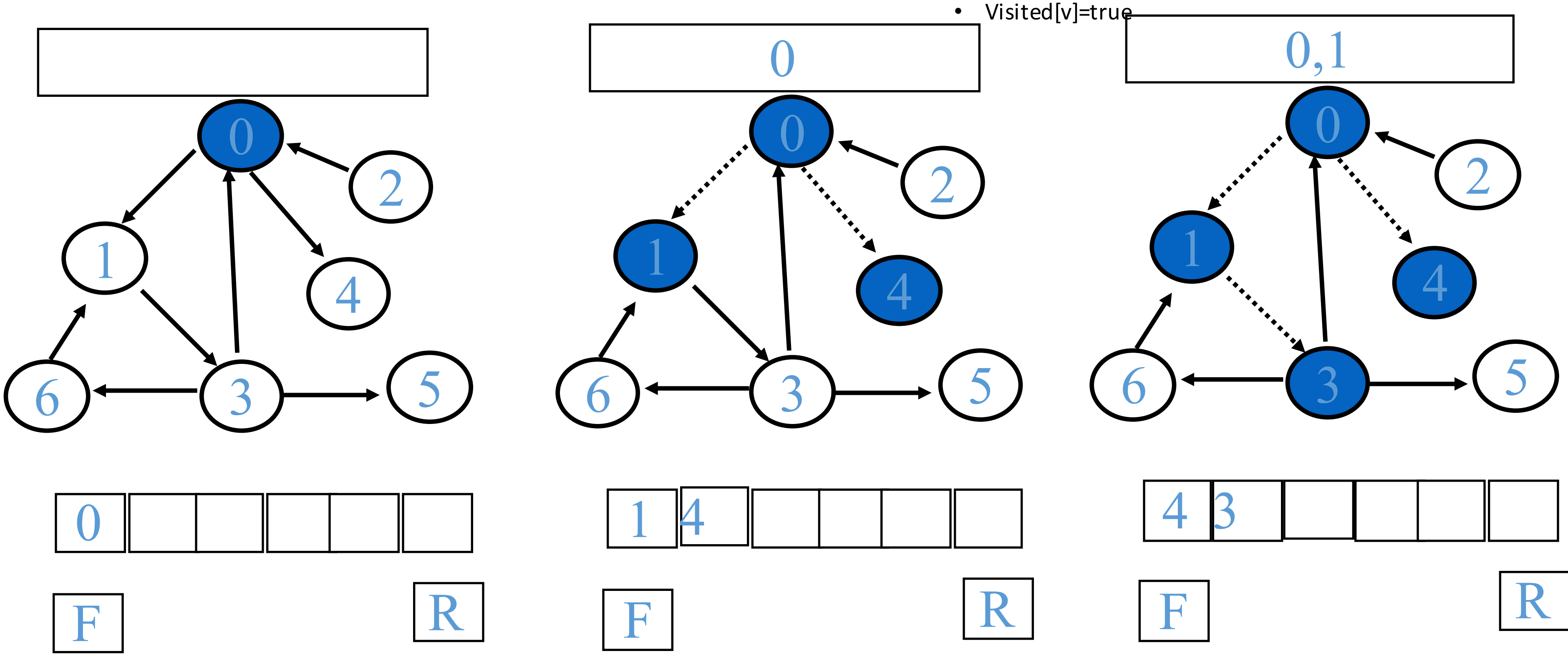
- 2 is never visited
- DFS search/spanning tree
- Non-recursive version? (using a stack)

Graph Traversal-BFS

- BFS(*start*)
  - Initialize the boolean visited array
  - Add *start* to an empty queue Q
  - Visited[*start*]=true;
  - While(!Q.empty())
    - u=Q.dequeue () //top+pop/get_front
    - For each unvisited neighbor v of u
      - Q.enqueue(v) //push
      - Visited[v]=true

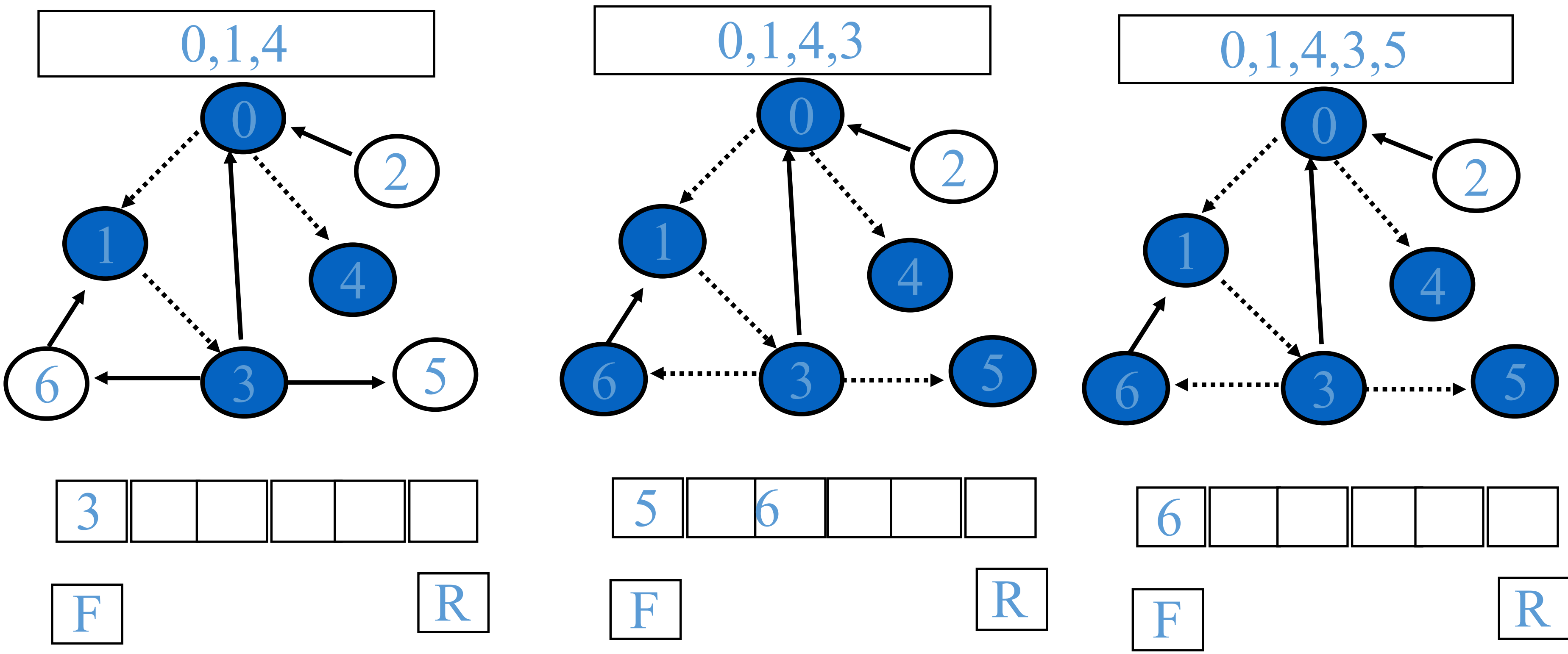
Graph Traversal-BFS

- BFS(*start*)
  - Initialize the boolean visited array
  - Add *start* to an empty queue Q
  - Visited[*start*]=true;
  - While(!Q.empty())
    - u=Q.dequeue () //top+pop/get_front
    - For each unvisited neighbor v of u
      - Q.enqueue(v) //push
      - Visited[v]=true

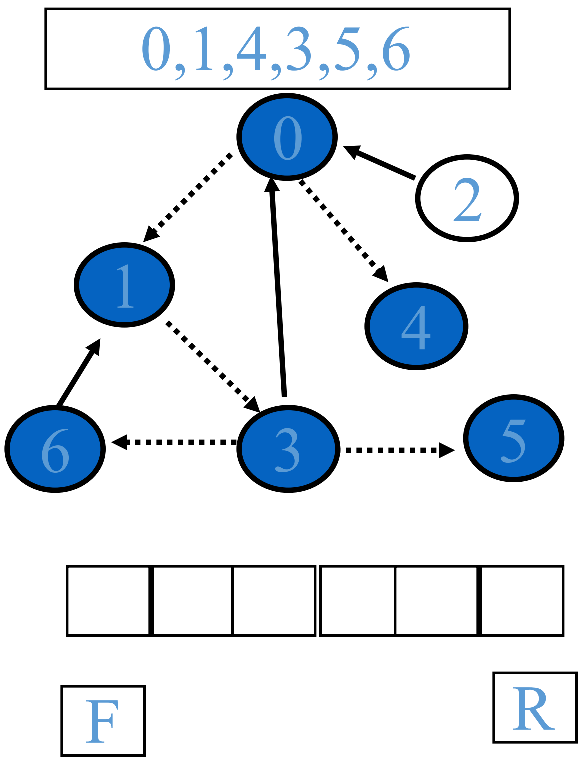


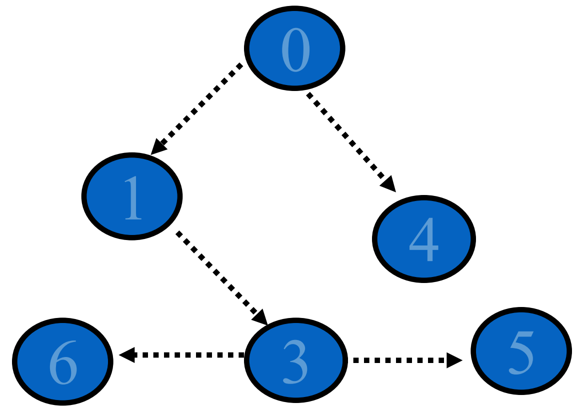
Graph Traversal-BFS

- BFS(*start*)
  - Initialize the boolean visited array
  - Add *start* to an empty queue Q
  - Visited[*start*]=true;
  - While(!Q.empty())
    - u=Q.dequeue () //top+pop/get_front
    - For each unvisited neighbor v of u
      - Q.enqueue(v) //push
      - Visited[v]=true



Graph Traversal-BFS





BFS Tree/Spanning Tree

Further Information

- Path Algorithms (Ch 15.4)
  - Dijkstra single source path
  - CSc 220 Algorithm
- Alternative Bellman-Ford algorithm for graphs with negative weights http://en.wikipedia.org/wiki/Bellman-Ford_algorithm
- All-pair shortest path algorithms
  - Floyd–Warshall algorithm http://en.wikipedia.org/wiki/Floyd-Warshall_algorithm
- Shortest path in dynamic networks (graphs)
- Graph Partition http://en.wikipedia.org/wiki/Graph_partitioning
- Graph Layout/Drawing