Graphs

Tuesday, March 20, 2018 12:19 PM

- Vertices + edges (weighted?)
- Directed vs. undirected (bidirectional)
- Complete graph
  - Clique: O(n^2) edges
- Path: v -> v
  - Length = #edges
  - Simple = no repeated vertices
  - Cycle = circular path (simple cycle)
- Connected: no discrete components

Problems

- Shortest path, Travelling salesman, Topological sort

Graphs

- G = (V, E, w) // vertices, edges, weight function

Adjacency

- adj(v) = neighbours/ successors of v
- Adjacency matrix - O(N^2) memory, not suitable for sparse graph
  - Undirected = symmetrical matrix
- Adjacency list - O(V + E), store edges of each vertex

BFS Traversal

- Source -> i -> i + 1
- Not unique
- Gives a BFS tree (root = source)
  - Vertex label = distance from source
- Connected & undirected graph: BFS visits all vertices
  - Disconnected: call BFS k times
- Recursively put neighbours into queue
- O(V + E)

DFS Traversal

- Use stack to remember where to back-track to
- O(V + E)

Topological Sort

- Directed Acyclic Graph (DAG)
- In-degree/ out-degree: #edges in/ out
- Recursively put vertices with in-degree 0 into queue
  - Dequeue: remove vertex + edges from graph
- Use adjacency list
  - When queued, update in-degree of neighbours
- O(V + E)