Graphs II

Monday, March 26, 2018 7:28 PM

Shortest Path

- Path = sequence of vertices, cost = sum cost
- Single-source problem: given v, find path w minimum cost to every other vertex
- Unweighted: cost = 1, weight (u, v)

BFS for shortest path

- Use BFS(source node)
  - Distance = w.level, trace route with w.parent

Dijkstra's Algorithm

- Initialise all distances to infinity
- Start from node with smallest value, process all nodes
  - Cost for this is already optimised, use this to optimise other nodes
- Weighted graphs: keep track of shortest distance so far, update when better path is found
  - Relax(v, w)
- $O(V^2 + E)$ - V^2 to find minimum of V, V-1..., 1 and E to relax
- Optimisation: use priority queue to find minimum
  - Heap construction = O(V)
  - Remove top = O(log V)
  - For each neighbour, use indirect heap to find it in O(1) and update the value in O(log V)
    - O(adj(V)*O(log V)), total = O(E log V)
    - Without indirect heap: O(V) to find value
  - Total = O(V log V + E log V) = O((V+E) log V)

Minimum Spanning Trees (MST)

- Edges connect all vertices, without unnecessary loops
- Can be generated using DFS (stack)/ BFS
- Not unique, consider sum of all costs

Prim's Algorithm

- MST by adding node one at a time
- Start from any node, pick minimum edge
- From connected nodes, repeat and pick minimum edge (a greedy algorithm)
- O(V) initialisation + O(E log V)
  - $E = O(V^2)$, O(E log E) = O(E log V)