Graphs IV

Lecture 29

Prof. Eric Polizzi
Weighted graphs: two 'interesting' algorithms

**Minimum spanning tree (MST):**
- Find cheapest set of flights that connect all airports; Cheapest installation for cable TV that connects a set of cities; Etc.
- **Two schemes:** Prim and Kruskal

**Shortest Path Problem:**
- What is the shortest (or cheapest) distance from one vertex to another, Etc.
- Etc.
MST - Prim

A in Tree
AB6, AD4
Dequeue → AD4

D in Tree
DE12, DC8, DB7, AB6
Dequeue → AB6

B in Tree
DE12, BC10, DC8, DB7, BE7
Dequeue → BE7

E in Tree
DE12, BC10, DC8, DB7, EF7, EC5
Dequeue → EC5

C in Tree
DE12, BC10, DC8, DB7, EF7, CF6
Dequeue → CF6

F in Tree
Done Tree{A, D, B, E, C, F}
The Shortest Path Problem

- Very common problem associated with weighted graphs
- Finding the 'shortest' path between two given vertices
- 'Shortest' stands for: shortest, cheapest, fastest, or best path by some other measure
- To make our approach even more general we can consider a directed weighted graph
- Example:

Question:
'shortest' path from A to E ?
Solution:
$140 path: A-D-C-E
Dijkstra's algorithm

- E. Dijkstra (1959)
- Find not only the shortest path from one specified vertex to another but also the shortest paths from one specified vertex to all other vertices.
- Find shortest path 'greedily' by updating distance to all other nodes
- We need to practice with examples to describe how the algorithm works
- Variables needed to maintain information
  - Set of vertices for which shortest path is set
  - Distance to start node
  - Predecessor node for current distance (needed to reconstruct the path tree)
Dijkstra's algorithm - Example I

Remark: distance to D through B is 140. However, 80<140, 80A is then retained.

PATHS: B (AB), D (AD), C (ADC), E (ADCE)
Dijkstra's algorithm - Example II

- From A to B C D E
- A 10A 5A oo oo
- C 8C 14C 7C
- E 8C 13E
- B 9B
- D Done

PATHS: C (AC), B (ACB), D (ACBD), E (ACE)
Dijkstra's algorithm - Example III + applet

https://www.youtube.com/watch?v=8Ls1RqHCOPw

- Todo:
  - Java applet
  - GraphDW.html
Dijkstra's algorithm - pseudocode

Let us consider:
A = source node, T = termination set
D(v) = current shortest distance to v; w(i,j) = weight of edge connecting i and j

T = {A}
Initialization
For all vertices
  if v is adjacent to A
    D(v) = w(A,v)
  else
    D(v) = oo

Loop
  Find u not in T such that D(u) is a minimum
  Add u to T
  Update D(v) for all v adjacent to u and not in T
    D(v) = min[D(v), D(u) + w(u,v)]
Until all nodes in T

Complete code
Textbook p703
For Large Graphs (N, and E are large), we need to use a sparse storage for the matrix, if not algorithm would run in $O(N^2)$.

For unweighted graphs:
  - DFS requires $O(N+E)$

For weighted graphs:
  - MST and and Dijkstra run in $O((N+E)\log N)$

Other famous algorithm for shortest path: Floyd–Warshall algorithm but it runs in $O(N^3)$ for dense graphs.

Note on other intractable problems- Traveling Salesman Problem

- Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?
- Examples 6 cities $\rightarrow 6!=720$ combinations
- This is a NP problem $O(2^N)$, $O(N!)$
- Famous CS problem, 'NP=P'?