Algorithms in Systems Engineering
IE170

Lecture 16

Dr. Ted Ralphs

References for Today's Lecture

- Required reading
  - CLRS Chapter 24
- References
  - R. Sedgewick, *Algorithms in C++* (Third Edition), 1998.

Breadth-first Search

- Recall from last time that processing the vertices in first-in, first-out (FIFO) order results in an algorithm called *breadth-first search* (BFS).
- This corresponds to the policy of choosing a vertex at minimum depth in the search tree as the next to be processed.
- The implementation is identical to DFS, except that the neighbors of the vertex being processed are inserted into a queue, instead of a stack.
- This creates a very shallow search tree, unlike DFS.

BFS and Shortest Paths

- Consider the problem of finding the *shortest path* from a vertex u to a vertex v.
- A shortest path from u to v is a path containing the fewest intermediate vertices.
- A *shortest paths tree* (SPT) is a rooted tree in which the path from the root vertex to each other vertex in the graph is a shortest such path in the original graph.
- Question: Does such a tree always exist?
- Answer: Yes.
- Question: How do we find it?
- Answer: The search tree created by performing a BFS is an SPT.
- Why is this the case?

Weighted Graphs

- For most practical applications, we will need to consider *weighted graphs*.
- In a weighted graph, each edge has a real number, called its *weight*, associated with it.
- Usually, the weights are specified using a separate weight vector $w \in \mathbb{R}^E$.
- Depending on the application, edge weights can be interpreted in a number of different ways.
  - They are interpreted as lengths or distances in cases where the graph models a physical network, such as a transportation network.
  - They are also frequently interpreted as costs associated with building or operating a network.

Weighted Shortest Paths

- In a weighted graph, the length of a path is the sum of the weights of the edges encountered on the path.
- A shortest path between two vertices in a weighted graph is a path connecting the two vertices that is of minimum length.
- In a transportation network, the edge weights may represent distances between physical locations, such as specific intersections.
- In such a weighted graph, the *shortest path* between two vertices has a natural physical interpretation.
- We are interested in being able to find such a path.
- Actually, we will consider the problem of finding an entire shortest paths tree rooted at a given source vertex.

Weighted Shortest Paths Tree

- An SPT is exactly the same in the weighted case as in the unweighted case.
- Is such a tree still guaranteed to exist?
- Question: Is there always a shortest path between any two vertices in a weighted, undirected graph?
- Answer: Not always.
- If the graph has edges of negative length, there may not exist a shortest path.
- A shortest path exists if and only if there are no *negative length cycles* reachable from either vertex.
- If there are no cycles of negative length, then there is always a shortest path with no cycles.
- This is essentially all we need to show the existence of an SPT.

Properties of Shortest Paths Trees

- Let $G = (V, E)$ be an undirected graph with an associated weight vector $w \in \mathbb{R}^E$.
- Suppose T is an SPT rooted at r and define $\delta(v)$ to be the path length from r to v in the tree.
- By definition, we must have that $\delta(v)$ is the length of a shortest path from r to v.
- For any node u on the path from r to v, the length of a shortest path u to v must be $\delta(v) - \delta(u)$.
- For any edge $e = \{u, v\} \in E$ that is not part of T, we must have that $\delta(v) \leq \delta(u) + w_e$.
- To show that T is an SPT, we need to show that the inequalities above hold for all edges not in T.

Finding the Shortest Paths Tree

- Assuming that all the edge lengths are positive integers, one approach is to subdivide each weight edge into unweighted edges of unit length.
- In other words, we replace an edge of length l with a path consisting of l edges of unit length.
- This essentially converts a weighted graph into an unweighted graph.
- After converting to an unweighted graph, we could simply use breadth-first search to find the (unweighted) SPT.
- This could be converted back to the weighted SPT by contracting the paths that were added back into single edges.
- This is a potentially disastrous algorithm since the running time depends on the number of edges.
- Fortunately, we can modify the algorithm to eliminate this dependence.