Single Source Shortest Path (SSSP)

Definition

Find the shortest path from the source vertex *s* to every other vertex.
Solved by finding a **shortest path** tree **rooted at *s*** that contains all the desired shortest paths.

Why do all shortest paths constitute a tree?

1. If all shortest paths are unique, then the union of shortest paths is a tree (recall: unique paths towards a root node)
2. If there are multiple shortest paths to t, we can pick and choose to make the union a tree, e.g.

Differences between Minimum Spanning Tree (MST) and Shortest Path Tree (SPT)

Clarifying Some Definitions:

A spanning tree of an undirected graph is a connected subgraph that covers all the graph nodes with the minimum possible number of edges.
A **minimum spanning tree** is a spanning tree whose weight is the smallest among all possible spanning trees.
Shortest Path Tree is a spanning tree such that the path from the **source** node *s* to any other node *v* is the **shortest** one in *G*.

Minimum Spanning Tree	Shortest Path Tree
Undirected	Directed
Doesn't have a root	Rooted in the source node
Can be unique	Distinct for different root

Negative Edges

For most shortest path problems, its natural to assume that all edge weights are non-negative. However, for many applications of shortest-path algorithms, it is also natural to consider edges with negative weights.
Negative edges in a SSSP is problematic and can cause problems.

- If a cycle is negative, then the shortest path may not be well defined.
  e.g.
  INSERT FIGURE 8.3 HERE
  Because we need to consider negative weights, this chapter **explicitly considers only directed graphs**. Nevertheless, all of the algorithms described also work for undirected graphs with modifications. BUT that is out of the scope of this course.

The Only SSSP Algorithm

Each vertex *v* in graph stores two values, which inductively describe a *tentative* shortest path from to *s* to *v*.

- *dist(v)* is the predecessor of v in the tentative shortest path or ∞ if no path exist.
- *pred(v)* the predecessor of *v* in the tentative shortest path or NUL if no path exist.

The predecessor *pred(v)* automatically defines a tree rooted at source *s*. At the beginning of the algorithm, we initialize the distances and predecessors as follows:

Pseudocode

```
function initSSP(s):
        distance = 0
        pred(s) = null
        for all vertices v != s:
                dist = ∞
                pred(v) = null
```

Algorithm Description

Repeatedly relax tense edges, until no more tense edges.