The Control Plane: Route Calculation - Dijkstra's

In this lesson, we'll study Dijkstra's shortest path algorithm!

We'll cover the following

Phase II: Route Calculation

Each router then computes the spanning tree rooted at itself and calculates the entries in the routing table by using **Dijkstra's shortest path algorithm**. Dijkstra's is a common algorithm that is usually taught in *Algorithms* or *Data Structures* classes. Let's get a quick refresher of it.

Dijkstra's Algorithm

The goal is to find the shortest path from an **initial node** to all other nodes in the graph.

We first need to set up some data structures for us to use throughout the algorithm.

1. Create a set called the **unvisited set**. All the nodes are initially unvisited.
2. Create a set called the **visited set**. It's initially empty.
3. Create a list called the **parent** list. It will contain mappings of nodes to their parents.
4. Lastly, every node has a distance of it from the initial node. Initially, all the nodes besides the initial node itself have a starting distance of

infinity. We call this `d_node_n`,

5. Every link between two nodes in the graph has a certain weight. We call this `w_node_n_node_m`.

Algorithm

1. Start with the **initial node** in the graph. Mark it as the **current node**.
2. Consider each of its neighbor's that are NOT in the **visited** set.
3. If the sum of the distance of the current node and the distance to the neighbor from the current node is **lower** than the current distance of the neighbor, replace it with the new distance.
    - In other words, if `w_node_curr_node_n` + `d_node_curr` < `d_node_n`, set `d_node_n` to `w_node_curr_node_n` + `d_node_curr`.
    - Also, set the parent of this neighbor, `n`, to the current node.
4. Repeat step 3 for all unvisited neighbors. After that, add the current node to the visited set.
5. Repeat steps 2-4 for the neighbor with the lowest `d_node_n`. Continue until the entire graph is visited.

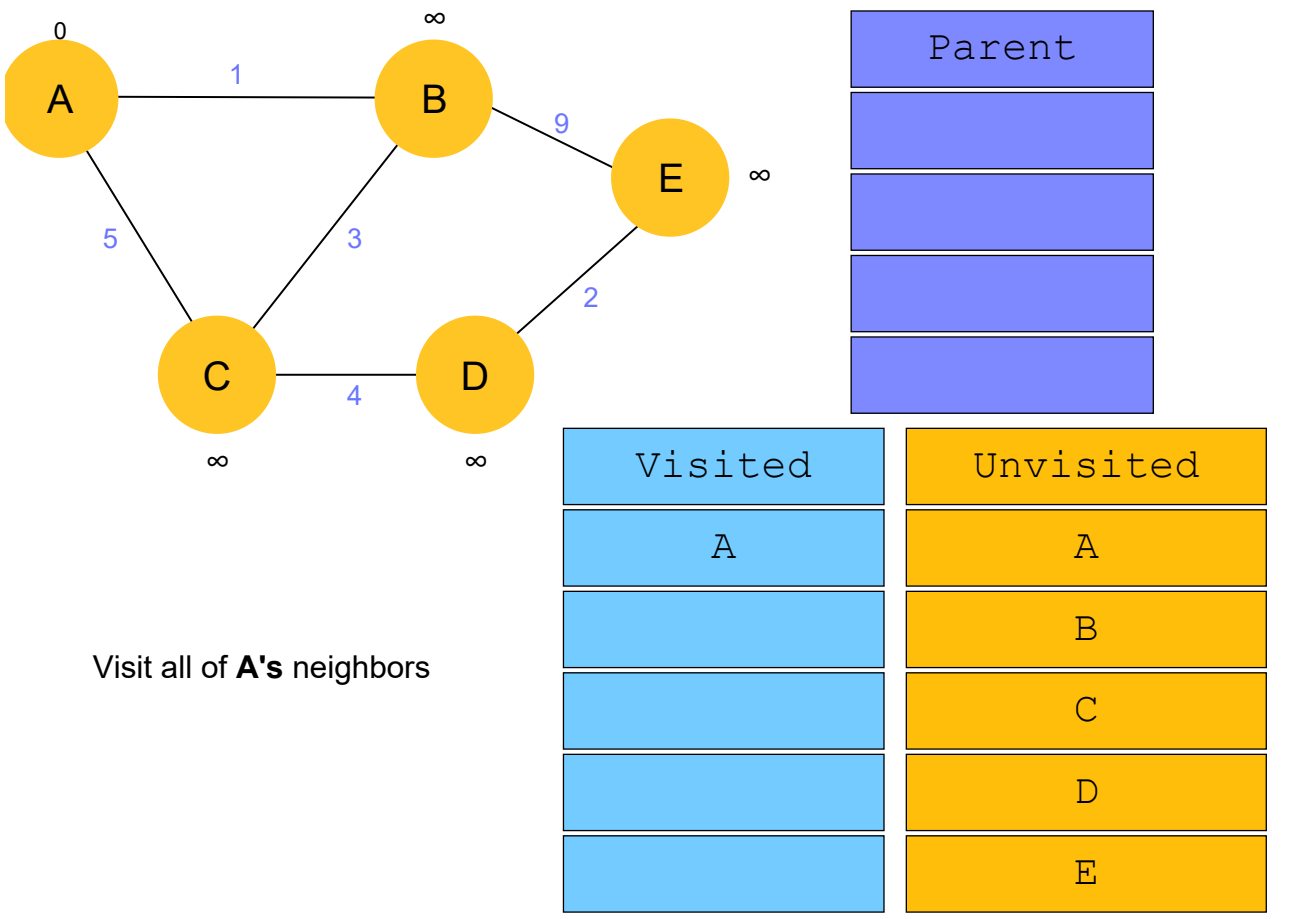
Finding the Shortest Path

To find the shortest path from a given node **n** to the initial node:

1. Find the parent of the current node. Initially the current node is **n**.
2. Set the current node to the new parent node.
3. Store each 'current node' in a stack.
4. Repeat steps 1-3 until the initial node is reached.
5. Pop and print the contents of the stack until it is empty.

Visual Example

Have a look at the following example to see how Dijkstra's would apply to a graph.



1 of 16

Quick Quiz!

1 What is the aim of Dijkstra's Algorithm?

- A) To find the shortest paths from every node to every other node in a given graph
- B) To find the most efficient paths from every node to every other node in a given graph
- C) To find the most efficient paths from a node in a given

graph to every other node in the graph

COMPLETED 0%

1 of 3

In the next lesson, you'll implement Dijkstra's Algorithm!