Title: Graph Algorithms, Finding the Shortest Path behind Today's World

1033 words

By Ryu Gyeongbin Deputy Editor-in-Chief

Many will be familiar with the fairytale Cinderella, the story of a poor girl mistreated by her stepmother and sisters living happily ever after winning a prince’s heart under her fairy godmother’s guidance. In the story, Cinderella miscalculates her time and flees moments before midnight to hide leaving behind a glass slipper as the enchantment of her fairy godmother begins to wear off. Cinderella’s story is portrayed as ‘once upon a time’, but what if this story has a background of 2023? Cinderella wouldn’t have fled from the start. Before attending the ball, she could have checked her departure time by setting her arrival time as midnight and left the ball in time. This could have brought a different ending and the prince could have probably forgotten the beautiful girl at the ball and maybe Cinderella could have just cherished the beautiful memories with the prince after all. Like the example above, these days we can know the shortest path and time it takes to a certain destination. Behind the technology are various algorithms defining rules for finding the shortest distance.

An algorithm is defining a rule that finds out the solution for a certain problem. Algorithms used in calculating shortest distances are usually defined as a graph. A graph is a type of data structure, a way of organizing and storing data in a computer’s memory or storage system, which is composed of nodes and vertices. Nodes are units of a graph that represent entities and points in a graph while edges are connections or relations between nodes in the graph. In an algorithm finding the shortest paths, nodes mostly are locations or places while edges correspond to paths from one place to another. Weights are given to each edge which is the distance between two places. A common example of a graph would be the following picture. Nodes represent locations in Seoul and edges represent distances between them while weights represent the distance itself between places. The best-known algorithms for calculating the shortest distances using graphs are following the three methodologies.­ : The Dijkstra algorithm, the Bellmanford algorithm, and the Johnson algorithm. ­

The Dijkstra algorithm is an efficient algorithm used to find the shortest distance between two nodes in a graph, finding the shortest distance between two nodes in a weighted graph. The algorithm is “greedy” in nature making locally optimum solutions at each step to ensure globally shortest distances. The algorithm largely consists of five steps.

(1) Initialization: Assign distances as infinity for all nodes and mark all nodes as ‘unvisited’ except for the starting node, assigned a distance as zero.

(2) Select the node with the shortest distance as the current node.

(3) Explore neighbors of the current node by relaxing edges from the previously updated node. Update distances if a shorter path than the previous path is found.

(4) Repeat steps 2 and 3 until all nodes have been visited or until the algorithm reaches the destination node.

(5) Termination: If you are specifically searching for the shortest path from the starting node to a destination node, the algorithm terminates once the destination node is marked as visited and the shortest path has been determined. However, Dijkstra’s algorithm can be extended to find the shortest paths from the starting node to all other nodes in the graph.

Plugging into the graph provided above, when I intend to find the shortest distance from SSWU Station to Gangnam Station, the shortest distance would be 11.9, passing only Sindang station along the way. But this algorithm guarantees all edge weights are non-negative. To find the shortest distances with negative values, another algorithm should be used.

The Bellmanford algorithm finds the shortest path from a single source node to all nodes in a weighted graph. One of the key benefits of the algorithm compared to Dijkstra is that it can handle negative weight edges and can detect negative weight cycles. The process of the algorithm involves relaxation and distance updates, similar to the Dijkstra algorithm. However, the termination condition is quite different. The algorithm terminates after V-1 iterations, where V indicates the number of nodes of the graph. This guarantees that the shortest paths from the source node to all the other nodes have been computed without negative weight cycles.

(1) Initialization: Assign distances as infinity for all nodes and mark all nodes as ‘unvisited’ except for the starting node, assigned a distance as zero.

(2) Relax all edges: Repeat the process of relaxing all edges for the number of nodes in the graph minus one. Relaxing an edge means checking if the distance to the destination node through the current edge is shorter than the current known distance to the destination node. If it is, update the distance to the destination node.

(3) Check for negative cycles: After relaxing all edges for (number of nodes - 1) times, if there is still a shorter path possible, it indicates the presence of a negative cycle in the graph. The algorithm can then terminate, or you can continue to relax edges to detect which nodes are part of the negative cycle.

(4) Termination: the algorithm terminates once all nodes are visited

Adapting the algorithm in the example above, the shortest distance would be 11.9, the same result as the Dijkstra algorithm. However, unlike the Dijkstra algorithm, the Bellmanford algorithm visits all other nodes in a graph which causes high-time complexity. Lastly, the Johnson Algorithm is commonly used to find the shortest path between all pairs of nodes in a graph efficiently. It achieves this by combining the algorithm with a graph transformation step to handle negative edge weights and improve efficiency, the Bellman-ford algorithm step to handle negative edge weights and improve efficiency. By employing the Johnson algorithm, the calculations required to find the shortest distance between all pairs of nodes can be significantly reduced.

Depending on the nature of the problems, the best algorithm for a certain problem might vary. These algorithms are used in map applications that we use every day to arrive on time. Besides that, in airline routing, internet routing, GPS navigation, supply chain optimizations, and many other problems, the above algorithms are used to make our life easier.