**Shortest Path Routing Algorithms**

**Amit Sengupta**

**Abstract**

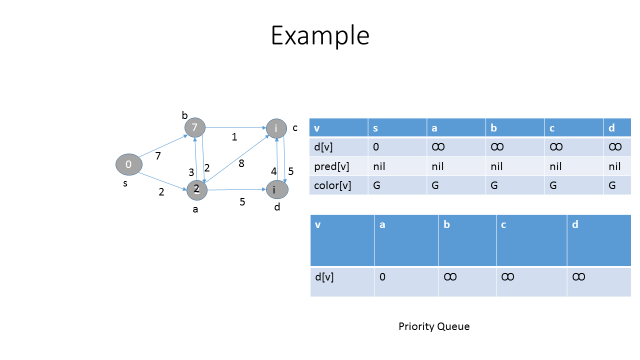
The shortest path algorithms have many important applications ranging from Google Map, network routing protocols, VLSI chip placement to prediction of infection risk and exploiting arbitrage opportunities in currency exchange. The purpose of this project is to study the single source shortest path algorithm (due to Dijkstra) and the all pair shortest path routing algorithm (due to Floyd and Warshall) in a graph. Both the algorithms assume that the graph is connected, the network topology and edge costs are known and nodes perform computation independent of each other. Dijkstra’s algorithm assumes that there is no negative edge cost while the Floyd Warshall’s algorithm assumes that there is no negative edge cycle. Dijkstra’s algorithm iteratively finds the shortest path by updating the estimated distance of the unvisited nodes that are connected to the newly discovered node. This is a greedy algorithm which uses two key observations: (a) the subpath of any shortest path is itself a shortest path; (b) the triangle inequality can lead to a better distance estimate. The running time of the algorithm is (V: the number of nodes) and can be improved to by using a Fibonacci heap. Floyd Warshall’s algorithm is a dynamic programming approach in which the intermediate results are stored in a table. This uses the following key observation: for a shortest path between and , there are two cases. In the first case, the intermediate node is not on the path leading to the shortest path of length whereas in the second case, the intermediate node is on the path resulting in the shortest path of length . The algorithm runs in . While Dijkstra’s algorithm can be invoked for every node to obtain the all pair shortest path, the algorithm will not accept a negative weight.

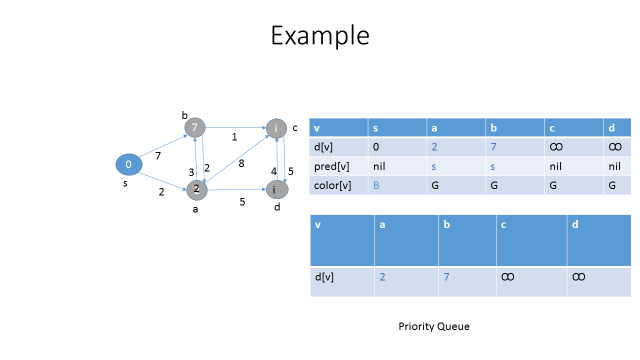
**Introduction**

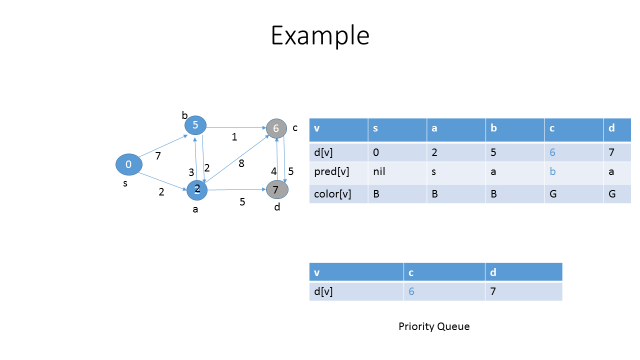
Finding the shortest path has many important applications ranging from Google Map, network routing protocols, VLSI chip placement to prediction of infection risk and exploiting arbitrage opportunities in currency exchange. However there are surprisingly efficient algorithms: (a) a single source shortest path routing (due to Dijkstra) and (b) the all pair shortest path routing (due to Floyd and Warshall). Both the algorithms assume that the graph is connected, the network topology and edge costs are known and nodes perform computation independent of each other. Dijkstra’s algorithm assumes that there is no negative edge cost while the Floyd Warshall’s algorithm assumes that there is no negative edge cycle. Dijkstra’s algorithm is a greedy approach which iteratively finds the shortest path by updating the estimated distance of the unvisited nodes that are connected to the newly discovered node. Floyd Warshall’s algorithm is a dynamic programming approach in which the intermediate results are stored in a table.

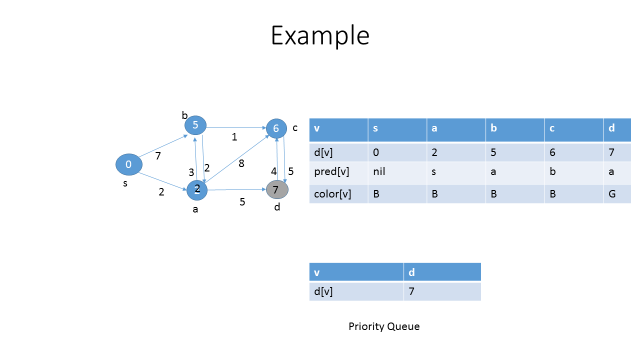
**Dijkstra’a Algorithm**

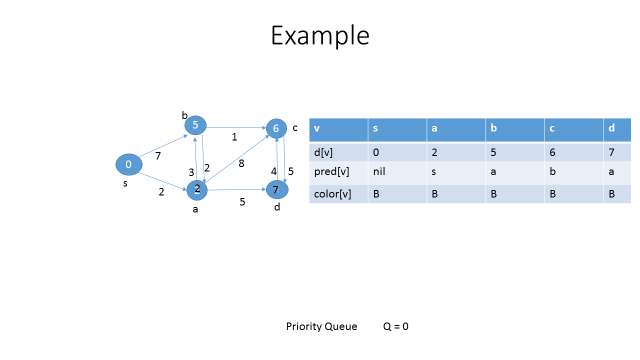
Dijkstra’a algorithm updates the distance from the current node to every unvisited adjacent node by checking if the estimated distance can be improved by going through an intermediate node. The distance is updated, the node is marked visited, and the process is continued until all nodes are visited. This is a greedy algorithm which uses two key observations: (a) the subpath of any shortest path is itself a shortest path; (b) the triangle inequality can lead to a better distance estimate. The running time of the algorithm is (V: the number of nodes) and can be improved to by using a Fibonacci heap. At the first iteration, the algorithm finds the nearest node from the source i.e the nearest neighbor of the source node. At the second iteration, the algorithm finds the second nearest node from the source i.e. the neighbor of the source node or the neighbor of the nearest node found in the first iteration. At the -th iteration, the algorithm finds the first nearest node from the source node. The working of the algorithm is illustrated in the following diagrams.











**Pseudocode**

d[s] = 0

For all y in V – {s}

set d[y] to ꝏ

Visited = {}

Q = V

While Q is not empty

x = least distance (Q,d)

Q = Q – {x}

Visited = Visited U {x}

for all y adjacent to x,

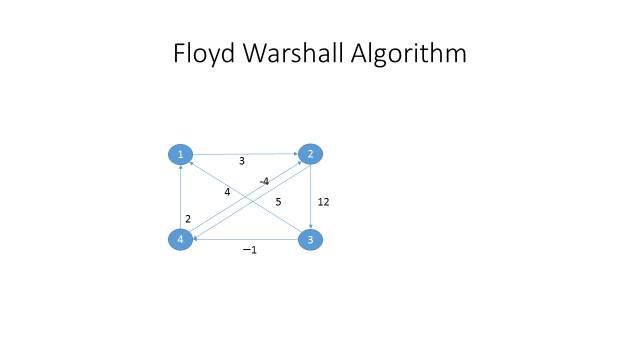
if (d[y] > d[x] + e[x,y]

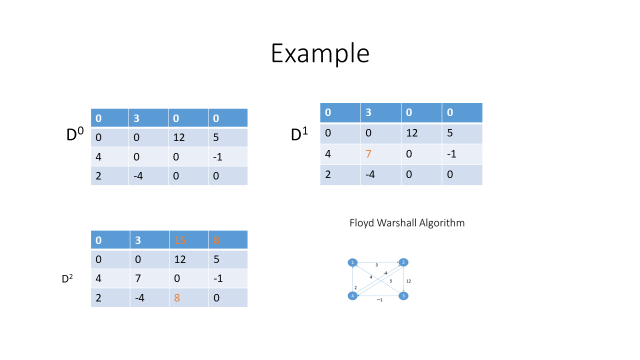
update d[y] = d[x] + e[x,y]

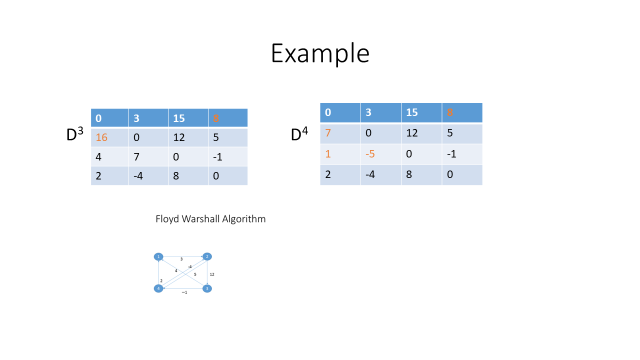
return d

**Floyd Warshall’s Algorithm**

Floyd Warshall’s algorithm is a dynamic programming approach in which the intermediate results are stored in a table. This uses the following key observation: for a shortest path between and , there are two cases. In the first case, the intermediate node is not on the path leading to the shortest path of length whereas in the second case, the intermediate node is on the path resulting in the shortest path of length . The running time of the algorithm is O() where is the number of nodes. The working of the algorithm is illustrated in the following diagrams.







**Pseudocode**

Floyd\_Warshall (D) {

V = D.rows

initialize pred

for m = 1 to V

for i= 1 to V

for j = 1 to V

if (dij > dim + dmj) {

dij = dim + dmj

predij = predmj

}

}

}

}

return D

}

**Conclusion**

The shortest path problem appears in many important applications ranging from Google Map, network routing protocols, VLSI chip placement to prediction of infection risk and exploiting arbitrage opportunities in currency exchange. There are surprisingly efficient algorithms such as the single source shortest path algorithm (due to Dijkstra) and the all pair shortest path routing algorithm (due to Floyd and Warshall) in a graph. Both the algorithms assume that the graph is connected, the network topology and edge costs are known and nodes perform computation independent of each other. Dijkstra’s algorithm assumes that there is no negative edge cost while the Floyd Warshall’s algorithm assumes that there is no negative edge cycle. Dijkstra’s algorithm iteratively finds the shortest path by updating the estimated distance of the unvisited nodes that are connected to the newly discovered node. This is a greedy algorithm which uses two key observations: (a) the subpath of any shortest path is itself a shortest path; (b) the triangle inequality can lead to a better distance estimate. The running time of the algorithm is (V: the number of nodes) and can be improved to by using a Fibonacci heap. Floyd Warshall’s algorithm is a dynamic programming approach in which the intermediate results are stored in a table. This uses the following key observation: for a shortest path between and , there are two cases. In the first case, the intermediate node is not on the path leading to the shortest path of length whereas in the second case, the intermediate node is on the path resulting in the shortest path of length . The algorithm runs in . While Dijkstra’s algorithm can be invoked for every node to obtain the all pair shortest path, the algorithm will not accept a negative weight.