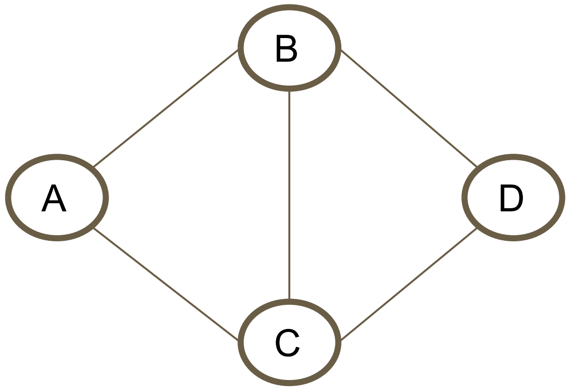
Dijkstra’s algorithm

Vishal Saini(203020048)
Prasad Kedar(203020019)
Digvijay Mishra(203020005)

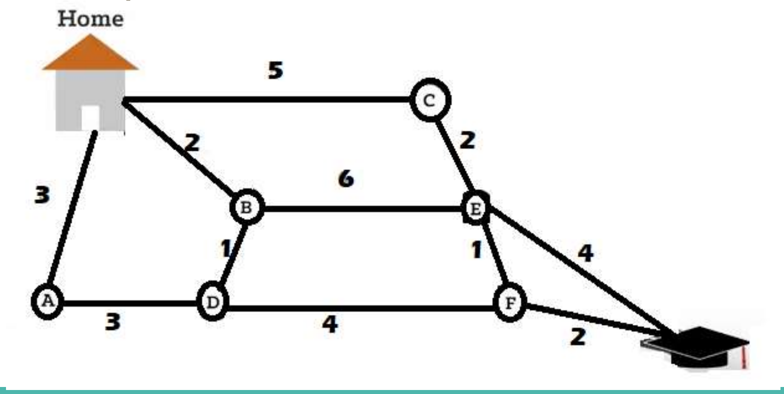
Introduction

- Dijkstra’s algorithm is popularly known as Single Source shortest path algorithm.
- Inspired by Bellman’s Principle of Optimality.
- Given a graph with a source vertex and if we need to find the shortest distance from the source vertex to any other vertex, we use this algorithm.



Short introduction to algorithm

- Suppose a student wants to go to school from home and there is some heavy construction going on in the path
- The Dijkstra's algorithm will assign more weight to such paths as choosing route with the higher route will result in taking more time to reach school.
- Dijkstra's algorithm will choose the shortest path with the least time required to reach the school.



In this case, the Dijkstra's algorithm will choose path:
Home → B → D → F → School

Introduction

- This algorithm is used in GPS devices to find the shortest path between the current location and the destination.
- Least-cost paths are calculated for instance to establish tracks of electricity lines or oil pipelines.
- Also useful when we have to distribute material at different locations from source location, so we find shortest route which connects all the different locations.

Literature survey

- The first application of Dijkstra’s algorithm was demonstrated by E. Dijkstra in 1956 during the official inauguration of ARMAC computer.
- Original algorithm was used to find the shortest path between two cities when the network of roads was given.
- Second problem that Dijkstra's dealt with was to find the shortest length of the wire needed to connect given set of points on a wiring panel, known as Minimum Spanning tree Algorithm
- These two algorithms were published in a paper titled “A Note on Two Problems in Connexion with Graphs” by E. W. Dijkstra in 1959.

Literature survey

- Other similar methods as Dijkstra’s algorithm are :

(1) A* Algorithm:

Consider a square grid having many obstacles and we are given a starting cell and a target cell. We want to reach the target cell from the starting cell as quickly as possible. Here A* Search Algorithm comes to the rescue.

What A* Search Algorithm does is that at each step it picks the node according to a value-‘**f**’ which is a parameter equal to the sum of two other parameters – ‘**g**’ and ‘**h**’. At each step it picks the node/cell having the lowest ‘**f**’, and process that node/cell.

Literature survey

We define ‘**g**’ and ‘**h**’ as simply as possible below

g = the movement cost to move from the starting point to a given square on the grid, following the path generated to get there.

h = the estimated movement cost to move from that given square on the grid to the final destination. This is often referred to as the heuristic, which is nothing but a kind of smart guess

(2) Prim’s Minimum Spanning Tree (MST) :

Prim’s algorithm is also a greedy algorithm. It starts with an empty spanning tree.

The idea is to maintain two sets of vertices.

Literature survey

The first set contains the vertices already included in the MST, the other set contains the vertices not yet included. At every step, it considers all the edges that connect the two sets, and picks the minimum weight edge from these edges. After picking the edge, it moves the other endpoint of the edge to the set containing MST.

So, at every step of Prim's algorithm, we find a cut (of two sets, one contains the vertices already included in MST and other contains rest of the vertices), pick the minimum weight edge from the cut and include this vertex to MST Set (the set that contains already included vertices).

How does Prim's Algorithm Work? The idea behind Prim's algorithm is simple, a spanning tree means all vertices must be connected. So the two disjoint subsets (discussed above) of vertices must be connected to make a Spanning Tree. And they must be connected with the minimum weight edge to make it a Minimum Spanning Tree.

Literature survey

(3) Bellman-Ford Method:

The Bellman-Ford algorithm is an algorithm that computes shortest path from a single source vertex to all of the other vertices in a weighted graph. It is slower than Dijkstra’s algorithm but for the same problem, but more versatile, as it is capable of handling graphs in which some of the edge weights are negative numbers.

Negative edge weights are found in various applications of graphs, hence the usefulness of this algorithm. If a graph contains a "negative cycle" (i.e. a cycle whose edges sum to a negative value) that is reachable from the source, then there is no shortest path : any path that has a point on the negative cycle can be made cheaper by one more walk around the negative cycle. In such a case, the Bellman–Ford algorithm can detect and report the negative cycle

Literature survey (Comparison with Bellman-Ford)

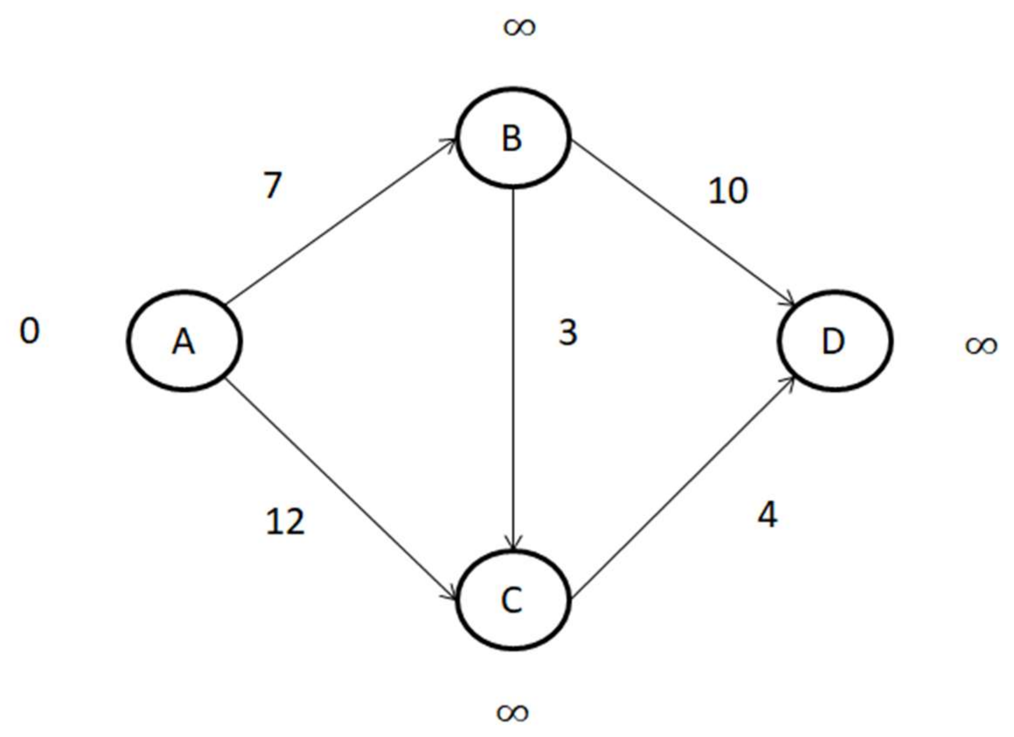
Dijkstra's Method.

- Uncertainties with negative weight edge
- Less time consuming as compared to Bellman Ford
- Greedy approach
- Dijkstra's algorithm is one of the SSSP (Single Source Shortest Path) algorithms
- Low complexity algorithm

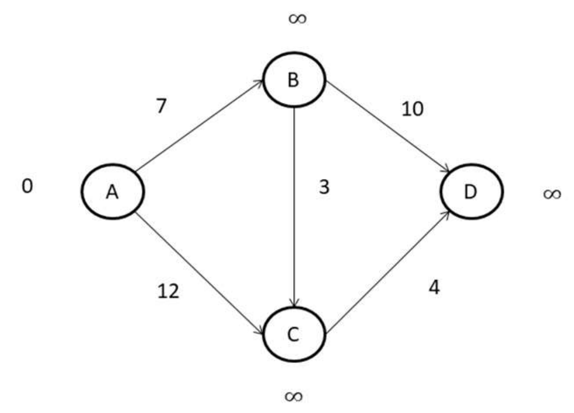
Bellman-Ford Method

- works when there is negative weight edge, it also detects the negative weight cycle.
- Usually More time consuming
- Dynamic Programming Approach
- Bellman-Ford algorithm is one of the SSSP (Single Source Shortest Path) algorithms
- Larger complexity than Dijkstra's algorithm

Problems where the algo has been implemented



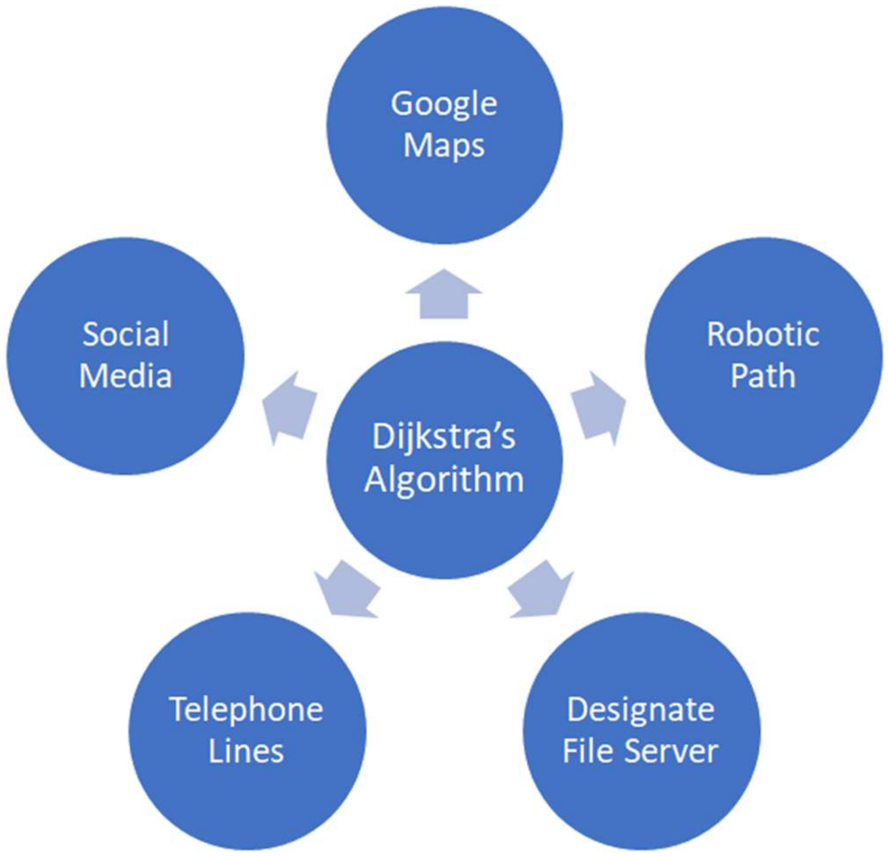
Dijkstra’s algo implementation



Conclusion

- Dijkstra’s algorithm was successful in finding shortest path for given non negative weights.
- For the defined problem shortest distance
  - From point A to B was 7 .
  - From point A to C was 10 .
  - From point A to D was 14 .
- Above problem was also solved with Bellman ford method and achieved same results.
- Dijkstra’s have taken **952.8** micro seconds while Bellman-Ford have taken **4.2** micro seconds to find the shortest path between specified two locations

Real World Use Cases



Remarks/comments

- Whenever there is a negative weight , the given algo may/may not work.