Comparative Study, Implementation and Analysis of Shortest Path Algorithms

Ajay Sagar Parwani
Department of Computer Science
Graduate Student
SZABIST, Karachi
Email: ajay.sagar92@gmail.com

***Abstract*—The research intended for comparative understanding, implementation and analysis of shortest path algorithms specifically focused on Dijkstra and Bellman-Ford for weighted directed graphs.**

1. Introduction

Shortest path algorithms have gained huge trend and area of interest in our daily life due to its multiple applications available in computer science, networking, biological and social to model process and computations. The concept of graph theory comes from mathematics that are used to model relations between different objects. The concept and first problem of graph was solved by Swiss mathematician Leonhard Euler in 1736 when he wrote the 1st paper on Seven bridges of Konigsberg and the branch of mathematics was named as Topology. The actual terms graph was coined in 1878 by Slyvester in Algebra and molecular diagrams.

2. Literature Survey

The problems of shortest path are used to find path between two points working from directed, undirected or mixed graphs. The algorithms in shortest path are known as greedy algorithms works for both positive and negative weighted graphs depending on the type of applications and algorithms being used. The shortest path problems are sometimes called single-pair shortest path problems.

The shortest path for positive weighted directed or undirected graph is Dijkstra which was designed by Edsger W. Dijkstra in 1956 which produces shortest-path tree. Initially the algorithm was not using the minimum priority queue. The concept of minimum priority queue known as heap was coined by Leyzorek et al in 1957. The implementation was based on Fibonacci heap where running time decreases to O(—E * V— log —V—). The time decreases to extent where:

- Insertion = O(1)
- Finding Minimum = O(1)
- Delete Minimum = O(Log V)

The graph with negative weight can be computed for shortest path using Bellman Ford Algorithm which was designed by Richard Bellman and Lester Ford in 1958. The algorithm is bit slower but more versatile than Dijkstra because along with managing negative weights, it can also handle to compute if the shortest path exists if there is no negative cycle in graph.

3. Methods Explored

There are multiple methods for implementing graph structure. Some of the most widely and common methods used are:

1) Array based Implementation:
   The implementation based on 2D array where each row representing vertex and columns representing their neighbors.
2) Linked List Based Implementation:
   Graph structure is implemented using linked list. Different implementations are available in linked list. The 2 most common implementations are:
   a) Array of vertices having reference of each neighbor
   b) Array of vertices having reference of neighbor and that neighbor having reference of another neighbor of that parent vertex.

4. Implementation

Algorithm and basic graph structure is implemented in java language using array of linked list where each vertex has reference of its neighbour in a linear way.

The files created are:

1) Vertex.java
2) Graph.java
3) MinimumPriorityQueue.java
4) ShortestPath.java

4.1. Dijkstra Pseudo code

Here is the pseudo code of Dijkstra Algorithm:

Let the Node at which we are beginning being known as the original Node. Dijkstra's calculation will allocate some

default separation esteems and will endeavour to enhance them well ordered.

1) Algorithm initially assigns zero to Source Node and infinity to all other Nodes.
2) Insert neighbours of unvisited Node in minimum priority Queue.
3) Calculate the distanceN which is distance of parent or predecessor node + weight of visited neighbour edge and update the distanceN if it is less than distanceP (distance previous)
4) Update the hash map with new distance along with its predecessor
5) Insert the new minimum distance into priority queue and repeat the step 3 until priority queue size becomes zero.
6) Return the updated hash map with new distances.

4.2. Bellman Ford Pseduo Code

Here is the pseudo code of Bellman Ford Algorithm:

1) The algorithms start and follows the same steps as Dijkstra Algorithm.
2) Iterate the loop to identify if the negative cycle exists, then no shortest path solution exists.

5. Comparitive Analysis

As both algorithms are used for calculating shortest path but comparitively Dijkstra is bit faster than Bellman Ford algorithm because Bellman Ford algorithm computes other than shortest path whether the negative cycle exists or not. If its exists then the shortest path can not be computable.

The time complexity is computed using best case and worst case scenario.

- **Best Case Scenario**
  The best-case scenario of any graph is when every vertex has exactly one neighbour.

- **Worst Case Scenario**
  The worst-case scenario of any graph is when every vertex having edges outward to each other vertex

The analysis of these algorithms depends on the basic structure used for graph and scenario ignoring some other computations being done in Bellman Ford for negative cycles.

- **Best Case Analysis**
  $|V|log|V|$: Where V are the total verticies and log V is for using minimum priority queue

- **Worst Case Analysis**
  $|V * E|log|V|$: Where V are the total verticies and log V is for using minimum priority queue

6. Costing

The worst-case costing of Dijkastra is attached on seperate page for each line. Moreover the costing of Bellman Ford is almost same except the extra loop iterated for finding negative cycle, if exists or not.

7. Applications

There are multiple applications for shortest path. Some of them are as follows:

- Maps
- Robots Navigation
- Urban Traffic Planning
- Sub routine in Advance Algorithms
- Routing of Telecommunication messages
- Network Routing Protocols

8. Conclusion

Both algorithms are used to find shortest path in graph but Bellman ford works for negative weight edges and find if the shortest path avaiable in graph when there is no negative cycle.