MATH 8650
Advanced Data Structures
Fall 2018

Term Project: Week 1 Report
Optimization of Bellman Ford Algorithm

Submitted by:
Vivek Koodli Udupa
Sai Harsha Nagabothu

November - 16, 2018

1 Introduction

This report addresses the implementation of Naive Bellman-Ford algorithm. The naive version serves as the basis of comparison for other variations of the Path Finding algorithm.

The Bellman Ford algorithm returns the shortest path from the source node to other nodes in the given graph. The graph may contain negative weight edges. The algorithm outputs shortest path only if there are no negative weight cycles. Negative weight cycles are reported.

2 Implementation

The implementation of the algorithm is as follows:

1. First step is to initialize distance from source node to all other nodes as infinity, and the distance to source node itself as zero.
   An array dist[] of size V is created and all its elements are set to "Inf" i.e infinity and dist[source] is set as zero. Here V is the number of vertices in the given graph.

2. The next step is to calculate the shortest distances. The following steps are repeated V-1 times.
   Note: x-y are two vertices and z is the weight between x-y

   (a) For each edge x-y:

   (b) If dist[y] is not infinity and dist[x] + z < dist[y], then update dist[y]

   (c) Updating y, dist[y] = dist[x] + weight of x-y(i.e z)

3. Now check for negative weight cycles. This is done by performing the distance check once more.

   (a) Do the following for each edge x-y:

   (b) if dist[y] > dist[x] + z then report "Negative Cycle".

   The idea here is that Step 2 ensures the shortest path if the graph does not contain negative weight cycle. If another iteration through all the edges finds a shorter path, then there exists a negative weight cycle. The algorithm reports this and exits.

3 Results

In order to test the Implementation, two graphs were selected. Graph 2 contains a negative cycle.

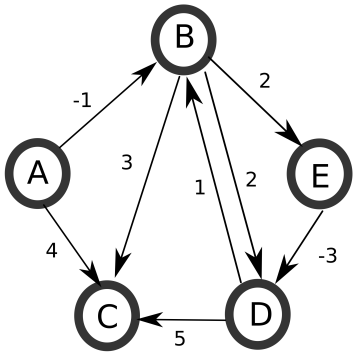

Figure 1: Graph used to test the BF Algorithm

Figure 1 shows the test set 1 used to find the shortest path. Figure 2 represents the test set 2 that contains a negative cycle. The algorithm must report a negative cycle for this graph.

Figure 3 represents the shortest path found by the implemented Algorithm.

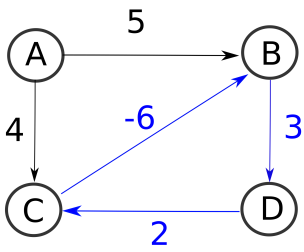

Figure 2: Graph 2 with highlighted negative weight cycle

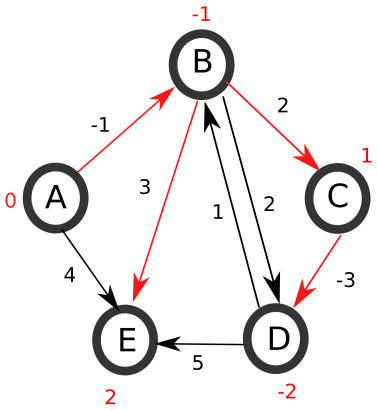

Figure 3: Highlighted Shortest Path for Graph1

Figure 4 shows the output of the python code for Graph1 and Figure 5 shows the "Negative Weighed Cycle Present" output for Graph2

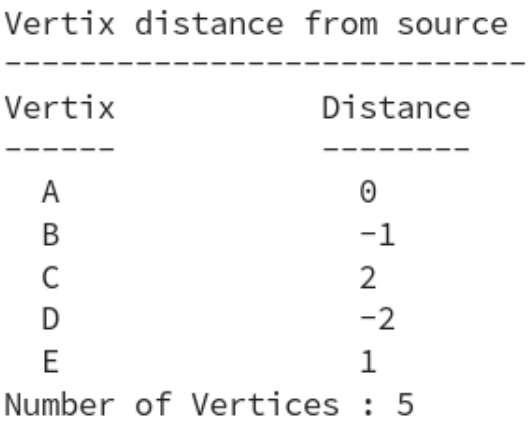

Figure 4: Output of the algorithm for Graph1

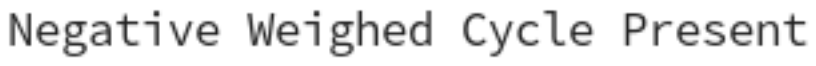

Figure 5: Output of the algorithm for Graph2

4 Next Steps

For week 2 the following things are planned:

1. Modify Bellman Ford to report shortest path even when a negative weight cycle is present
2. Implement the SPFA (Shortest Path Faster Algorithm)

5 Conclusion

In conclusion, the algorithm calculates the shortest path in bottom-up manner. At first it calculates the shortest distances that has maximum of one edge in the path. Then, it calculates shortest paths with at-most 2 edges and so on. After V-1 iterations the shortest path is found as there can be maximum of V-1 edges for any simple graph. Another shortest path check after V-1 iterations shows the presence of any negative weighed cycle in the graph.
Thus the algorithm takes an input that consists of a graph and the source node and outputs either the shortest distance from source to all other nodes or reports the presence of negative weighed cycles.

6 References

[1] https://www.geeksforgeeks.org/bellman-ford-algorithm-dp-23/
[2] https://en.wikipedia.org/wiki/Bellman%E2%80%93Ford_algorithm
[3] https://www.dyclassroom.com/graph/detecting-negative-cycle-using-bellman-ford-algorithm