Single-Source Shortest-Paths

CS 412-L3: Algorithms Design & Analysis *(Spring 2022)*
Team: *Planar Analysis*

AUTHORS

- Fahad Shaikh (05452)
- Aiman Haq (04318)
- Syed Muhammad Hamza (05192)

AFFILIATIONS

Habib University, Karachi, Pakistan

INTRODUCTION

This particular project shall concern itself with the **Shortest Path Problem**. Intuitively speaking, the problem in question ascertains that given a *source* and a *destination*, what is the shortest path between them. Indeed, the project focuses on the Shortest Path Problem but what was omitted earlier was the intent to focus solely upon the **Single-Source Shortest Path** variant.

For the project, we have focused on two algorithmic paradigms/design techniques to solve the problem in question. We considered a **greedy approach** (i.e., *Dijkstra's algorithm*) and a **dynamic-programming approach** (i.e., *Bellman-Ford algorithm*).

DESIGN TECHNIQUES

Two Design Techniques are used:

- Dynamic Programming (Bellman-Ford Algorithm)
- Greedy Approach (Dijkstra Algorithm)

Bellman-Ford Algorithm:

A dynamic approach to find a single-source shortest path. it can handle negative weights in a graph, where the graph must be connected.
It can also be used to find negative weighted cycle.

Dijkstra Algorithm:

A greedy approach to find the shortest path, where a graph must be connected and should have only positive weights.
Depending on the data structure used, its time complexity may varies.

COMPARISON ANALYSIS OF THEORETICAL COMPLEXITIES

Theoretical Complexities:

- Dijkstra
  - Array-based: O(|V|^2)
  - Heap-based: O(|E|log|V|)
- Bellman-Ford: O(|V||E|)

Comparison Analysis:

Dijkstra is faster than the Bellman-Ford algorithm, this analysis is based on their approaches to updating their shortest path distances. While Dijkstra greedily decides to relax the edges once, Bellman-Ford relaxes all edges |V|-1 times.
Contrary, Bellman-Ford can handle graphs with negative weights while Dijkstra is suitable only for positive weights.

METHODOLOGY

The experiments performed, evaluate the tested approaches in terms of their performance with respect to time. A singular, three-pronged "experiment" that is, a single test that is repeated thrice with variability in terms of two parameters in particular, that are as follows:

- Size of the graph (i.e., number of vertices)
- Density of the graph (i.e., number of edges)

As such, via the Erdos-Renyi model, random graphs were generated as follows:

- Dense Density Graphs: for $p = 1$
- Average Density Graphs: for $p = 0.5$
- Sparse Density Graphs: for $p = 0.1$

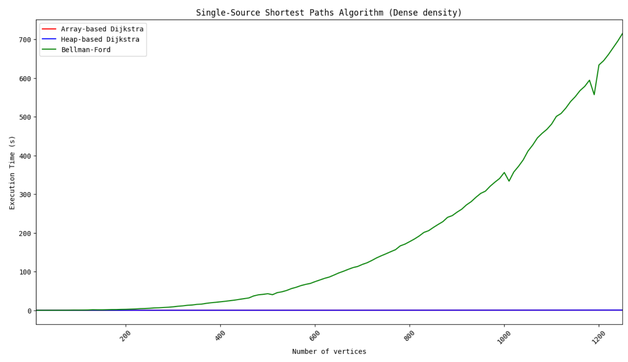
Where **p** is the probability for generating a random edge between two arbitrary vertices.

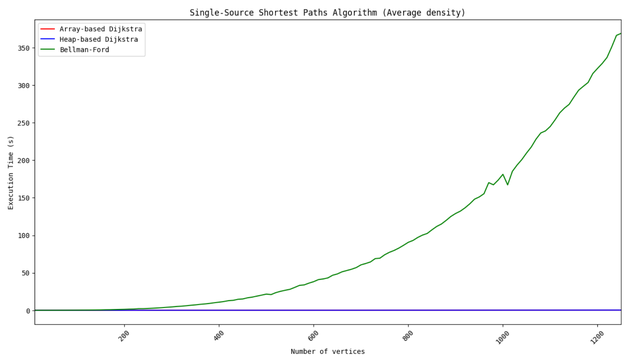
DEPENDENCIES

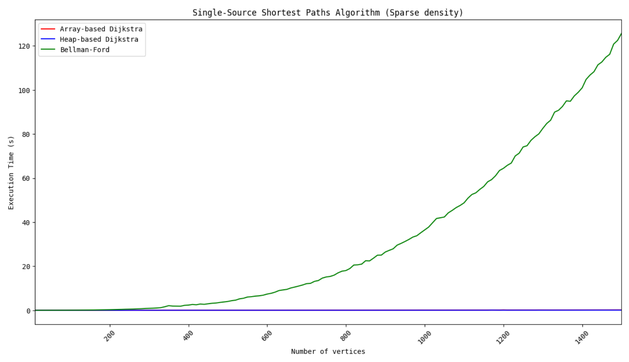
We used the Library **NetworkX** for the following:

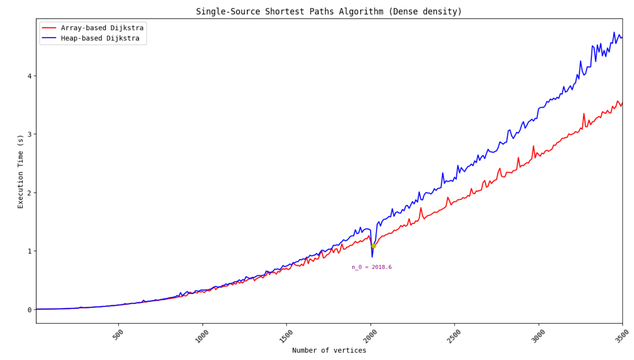
- **fast_gnp_random_graph** method was used to generate random inputs of *dense* and *average* density.
- **erdos_renyi_graph** method was used to generate random inputs of a *sparse density*.
- **is_strongly_connected** method was used to check if the generated graph was strongly connected or not.

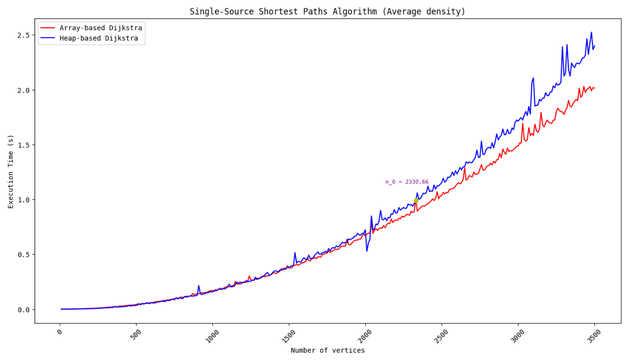
EMPIRICAL ANALYSIS

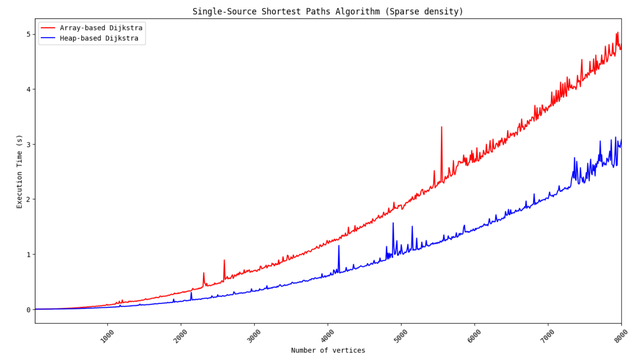












CONCLUSION

The authors come away with the following conclusion: there is no best/worst algorithm rather, an algorithm that is best suited to a specific task or configuration of input. Speaking strictly with respect to the single-source shortest-paths problem, we conclude:

1. **Bellman-Ford** is best suited to instances, where the graph has negative weights
2. **Array-based Dijkstra** is best suited to cases of the incredible density of the input
3. **Binary-Heap-based Dijkstra** is best suited to cases of sufficient sparseness. In short, very rarely is there a *swiss-army knife* situation where one algorithm suits all rather, what we have generally is a specific tool for a specific job.

REFERENCES

[1] Bisht, J. Erd¨os-reyni model (for generating random graphs). https://www.geeksforgeeks.org/erdos-renyl-model-generating-random-graphs/, 2021. Accessed: 2022-04-29.
[2] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. Introduction to Algorithms, 3 ed. The MIT Press, Cambridge, Massachusetts, 2009.
[3] Hagberg, A., Swart, P., and Schult, D. Networkx. (Version 2.7.1) [Source Code], 2022.
[4] Programiz. Dijkstra's algorithm. https://www.programiz.com/dsa/dijkstra-algorithm. Accessed: 2022-07-03.