# Title: Performance Analysis of Dijkstra’s and Bellman ford shortest path algorithms

Abstract:

This research paper explores and compares two fundamental algorithms, Dijkstra's algorithm and Bellman-Ford's algorithm, used to solve the shortest path problem in graphs. The paper provides an in-depth analysis of the pragmatic differences between these algorithms, focusing on their implementations, time complexities, and performance characteristics. Additionally, benchmark results are presented to illustrate the empirical differences between the algorithms. The findings of this study contribute to a better understanding of the trade-offs involved in selecting the most appropriate algorithm for solving the shortest path problem in various real-world scenarios.

## 1. Introduction:

The shortest path problem is a fundamental problem in graph theory, with applications in various domains such as transportation, network routing, and logistics. Dijkstra's algorithm and Bellman-Ford's algorithm are two commonly used methods to solve this problem. This paper aims to compare the pragmatic differences between these algorithms, including their implementations, time complexities, and performance characteristics.

## 2. Background:

**Dijkstra's Algorithm:** Dijkstra's algorithm is a greedy algorithm that finds the shortest path from a single source vertex to all other vertices in a weighted graph with non-negative edge weights.

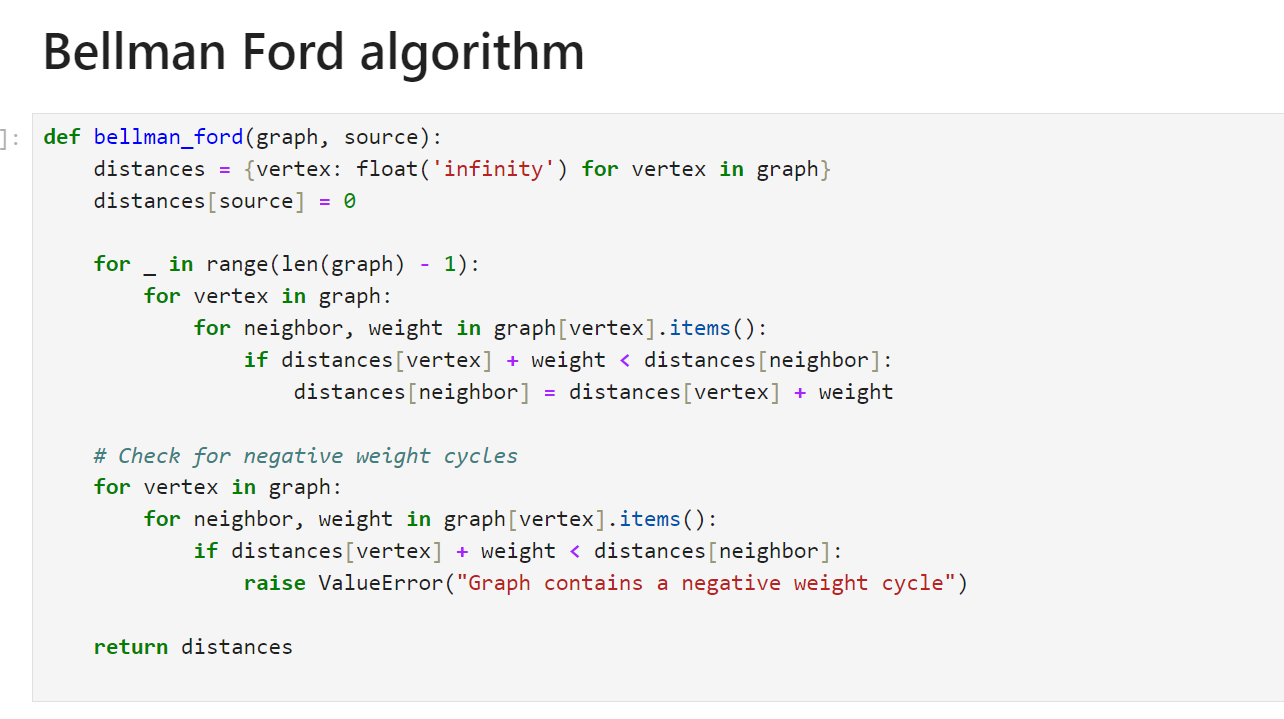
**Bellman-Ford's Algorithm:** Bellman-Ford's algorithm handles graphs with negative edge weights and can detect negative weight cycles. It iteratively relaxes all edges multiple times to compute the shortest paths.

## 3. Implementation:

Dijkstra’s



Bellman Ford



## 4. Benchmarking:

To benchmark the algorithms, I will generate random graphs with varying numbers of vertices and edge weights.

I will measure the runtime performance of both algorithms for different graph sizes and analyze the results.

First pass of benchmarking –

On the random graph with following properties

* num\_vertices = 100
* edge\_probability = 0.3
* min\_weight = 1
* max\_weight = 10

A screenshot of a computer

Description automatically generated

A graph with a bar and a number of different colored squares

Description automatically generated with medium confidence

## 5. Analysis

Comparison of Time Complexities: Dijkstra's algorithm has a time complexity of O((V + E)logV) using a priority queue, while Bellman-Ford's algorithm has a time complexity of O(VE).

Empirical Performance Analysis: Benchmark results will be presented and analyzed to compare the runtime performance of Dijkstra's and Bellman-Ford's algorithms for different graph sizes and edge weight distributions.

## 6. Results:

Benchmark Results: Tables and graphs illustrating the empirical performance of both algorithms in terms of runtime for various graph sizes and edge weight distributions.

Discussion of Findings: Interpretation of benchmark results and analysis of the pragmatic differences between Dijkstra's and Bellman-Ford's algorithms.

## 7. Conclusion:

## Summary of Key Findings: Recap of the main findings from the analysis and benchmarking of both algorithms.

## Practical Implications: Discussion of the practical considerations and trade-offs involved in selecting between Dijkstra's and Bellman-Ford's algorithms for solving the shortest path problem in real-world applications.