Module 7: Critical Thinking

Brady Chin

Colorado State University Global

CSC506-1: Design and Analysis of Algorithms

Dr. Dong Nguyen

January 5th, 2025

Critical Thinking 7

This critical thinking assignment explored the Dijkstra algorithm to find the shortest path between two points in a weighted graph. We were tasked ways to optimize the algorithm to optimize a delivery app and to consider real-time traffic data.

Applicability

Using Dijkstra's algorithm to find the shortest path in a food delivery application is very beneficial. In the corresponding code, Dijkstra's algorithm uses a heap based priority queue to selected each node which is a very efficient way on finding the shortest path to the next node (Pizzo, B., 2024).

This algorithm is also not capable of handling negative weights. In an application that uses navigation, using Dijkstra's algorithm is great because there are no negative steps that can be taken. In a real-life scenario, if the wrong turn is taken, the application will simply reroute.

Compared to the Bellman-Ford algorithm, another shortest path algorithm, Dijkstra's algorithm is a better choice because it is more efficient and has a better time complexity, which I will discuss in a later section.

Time Complexity

Focusing on only the algorithm, the time complexity will be **O((V + E) log V)** where V is the number of vertices (nodes) and E is the number of edges (geeksforgeeks.org 2024, February 9). The following is a breakdown of the time complexity:

1. Initialization of the dictionaries:
    - Creating distances and parent dictionaries takes O(V), where V is the number of nodes.
2. The while loop:
    - The while loop runs as long as the priority queue has unvisited nodes, which means it runs O(V) times (each node is processed once).

3. Finding the minimum-distance node:
    - Extracting the smallest distance node from the priority queue using heappop takes O(log V) for each iteration of the while loop.
    - Over V iterations, this results in a total complexity of O(V log V).
4. The inner for loop:
    - The inner loop iterates over all neighbors of the current node.
    - Across the entire algorithm, each edge is processed exactly once, resulting in a total complexity of O(E) for edge relaxations.
    - Each heappush operation in this loop takes O(log V), so for E edges, the complexity is O(E log V).

While this is the time complexity of Dijkstra's algorithm, the path reconstruction and printing of the results add **$O(V^2)$** to the algorithm.

In a real-life scenario, the time complexity of Dijkstra's algorithm remains O((V + E) log V) when using a binary heap. However, frequent rerouting or updates due to changing traffic conditions can make the overall process more computationally expensive, as the algorithm may need to be rerun or partially recomputed multiple times.

Real-life Variables

A real-life variable that can impact the performance of Dijkstra's algorithm would be traffic updates. This would cause frequent updates to edge weights which will result in the algorithm to be run again to reroute and find a new shortest path.

Another real-life variable that would impact performance would be the scale of the delivery path. Larger graphs will require a large amount of memory and computational resources to carry out the same functions.

Lower Bound

System latency can impact the lower bound of route planning efficiency. The speed of the CPU, memory, and storage affects the ability to compute the shortest path efficiently. Additionally, delays in retrieving real-time data can slow down computation times.

Another external factor that can influence the lower bound would be environmental factors. In areas where there are often road closures because of heavy snow, storms, or other weather events can affect road usability which will alter route planning. The algorithm will have to be rerun more frequently to find new routes.

Corresponding Code

Since my portfolio project is also on shortest paths and Dijkstra's algorithm, I found this critical thinking assignment to be helpful because it allowed me to look at the algorithm from a different perspective, such as in a food delivery app. While my code stayed relatively the same, it was interesting to see how I could change the names of the nodes and how that would affect the code.

I believe that the portfolio project was a great way to understand Dijkstra's algorithm and this assignment was a great way to apply the algorithm.

Conclusion

Time complexity, external factors, and the lower bound are important factors to consider when using algorithms like Dijkstra's for navigational route planning. While this assignment focuses of food delivery apps, the same concept can be applied to several navigational applications.

References

geeksforgeeks.org (2024, February 9) *Time and Space Complexity of Dijkstra's Algorithm.* https://www.geeksforgeeks.org/time-and-space-complexity-of-dijkstras-algorithm/

Pizzo, B., (2024) *CSC506: Design and Analysis of Algorithms. 5.11 Heaps.* zyBooks. https://learn.zybooks.com/zybook/CSC506-1_5/chapter/5/section/11