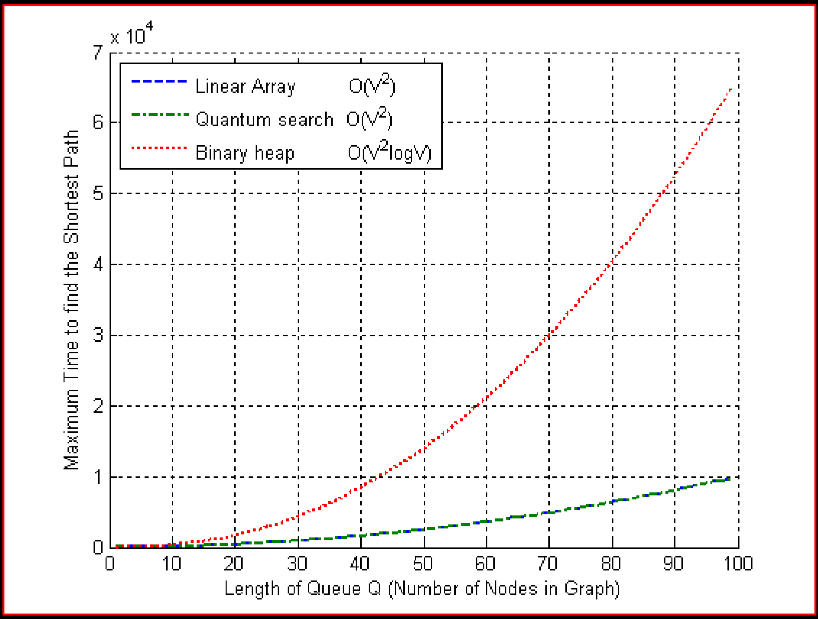
W04IST-SI4024G // REPORT ON SEARCHING ALGORITHMS

CASPIAN NORTH – 287776

REPORT AT 060425 1923HRS CEST

EXECUTIVE SUMMARY

1. Within our project to search through Wroclaw public transport data for the shortest tram route possible between two stops we implemented two algorithms. A* algorithm and Dijkstra's algorithm (DA). We found that A* algorithm was more efficient in terms of speed but significantly harder to implement and tune the heuristics to make our search data repeatable and effective. With Dijkstra's algorithm we found it taking 9.75 seconds on average to search between two fixed stops and ___ for A* algorithm. DA was significantly simpler to implement due to its simple logic. However, on a smaller dataset it would likely be more effective than A* to quickly set up. DA is a better lead for testing a program before implementing more advanced algorithms such as a*. We also considered Tabu Search as a comparison. While we didn't get it working it in this project, it's worth noting that Tabu Search focuses more on finding a "good enough" solution in a flexible way, rather than guaranteeing the shortest path like Dijkstra or A*. It's better suited for problems like route planning when conditions change or there are lots of constraints. However, it is more complex to set up and doesn't always give the perfect result. For live or real-time systems, though, its ability to adapt and avoid getting stuck on one idea makes it a strong option.
   In our first configuration we noticed we couldn't process polish characters in a string and decided to settle with the python models; Pandas for data manipulation and reading our CSV files, NetworkX for graphing and storing the data and Matplotlib for data visualisation. We know from our results that Dijkstra's in the most basic form has a time complexity of O(V^2) and A* at O(b^d) with d being defined as the depth of the solution and b is the number of successors for a state. Its visualisation is not ideal due to its defined varying factor but in most instances, it will prove better than a binary heap in terms of efficiency due to its guided nature.



*Graph showing time complexities. O(V^2) seen in green is that of DA REF*4*

LESSONS LEARNT

2. We have selected NetworkX as our module to handle data structure due to its effectiveness as being a premade module for handing of different data types and graphs with ease without requiring linking lists or new class building for nodes. The disadvantage of this is our programmer (myself) is now struggling to implement newer algorithms, and I would recommend someone builds classes to store in network before implementing them.

3. We selected matplotlib to visualize data. It is the industry standard and therefore highly effective for our program. There are still issues with the overlap of nodes in the visualisation of the graph. However, visualisation is not one of the requirements for our project so we will leave it unfixed.

4. We selected pandas to read our CSV file, it is simple and effective and is a well upkept and effective module for our requirements. The alternative of trying to parse our file with string splitting would likely be just as effective or a little slower so we decided on this approach.

OVERALL ASSETSMENT

5. Dijkstra's algorithm (DA) is a searching algorithm that is invasive. It works by checking the connections between every node to find the shortest path.
It is inefficient and slow for large data where only one small query is needed. This is likely to affect a large company negatively such as Jakdojade as it would affect user experience and requires a lot of power from servers.

6. DA is simple in theory in its logic. however, it was one of the first searching algorithms invented. This mean’s it is highly likely for beginner programmers learning searching algorithms to be able to implement it. This can be positive for small scale hobby projects and non-commercial use.

7. DA is negligible on graphs with few connections, this allows it to be good for some small businesses who may need to connect items such as company hierarchies. This can be positive for efficiency with a low cost to implement.

8. The A* algorithm is efficient due to its use of maths. A* algorithm uses a heuristic algorithm such as Manhattan distance to guesstimate the shortest path. It doesn’t always guarantee the true shortest path. This can lead to A* likely being somewhat unreliable when a non-fitting heuristic is used for a program. It requires some tailoring to the data it is searching. However, it can be repeatable and therefore also reliable.

9. A* algorithm uses targeted pathfinding while Dijkstra’s uses all paths from source. This gives both algorithms a use case however both have high consumption of power with A* storing less data.

10. Tabu Search (TS) is a smart way of finding good solutions to difficult problems, like scheduling or planning routes. It remembers what paths it has already tried and avoids repeating them. This helps it find better answers, but it can use a lot of computing power, which might not be ideal for smaller systems or devices.

11. TS is harder to understand than some other algorithms because it has more rules and steps. It’s not usually the first choice for beginners learning how to code, but it can be a great tool for experienced developers or bigger companies who need more advanced problem-solving.

12. Tabu Search works well for businesses with tricky or changing needs, like delivery services or airlines. It can help organize tasks or routes in a way that saves time and money, especially when there are a lot of things to consider at once.

13. TS doesn’t always find the perfect solution, but it usually finds one that’s good enough and does it quickly. This makes it useful in situations where a fast and solid answer is more important than a perfect one, such as live route updates or quick planning.

14. Unlike Dijkstra’s or A*, which focus on finding paths in a map or network, Tabu Search looks at the bigger picture of possible answers. It’s better for solving planning or optimization problems, and it can also be combined with other methods to be even more effective.

DATA GAPS.

- We do not have a good visualization for the effectiveness on our own data.
- We have not tested other searching algorithms or different data sets to come to our conclusion.
- We have a limited experience with changing the heuristics of A* that may lead to conflicting results with industry standards.

Historical context:

REFERENCES

1. https://en.wikipedia.org/wiki/A*_search_algorithm
2. https://en.wikipedia.org/wiki/Search_algorithm
3. https://gamedev.stackexchange.com/questions/178873/pathfinding-with-speed-based-turning-restrictions
4. https://www.researchgate.net/figure/Speeds-comparison-of-Dijkstras-algorithm-in-three-states-of-implementation-to-find-the_fig3_238492680