Annotations

Jonathan Nguyen, id:000918228

March 27, 2019

Dijkstra's Algorithm

I chose dijkstra's algorithm due to the simplicity of the algorithm, and the variability of it. I also chose it because initially it seemed the simplest to implement given the requirements for the project. I would simply make the nodes, and use the algorithm to find the shortest/fastest route between them.

Strengths of this algorithm

One of the strengths of this algorithm is the flexibility it brings. No matter how large or small the data size is, the algorithm will be able to produce an adequate result. Because the algorithm searches for the smallest distance between graphs, as long as the data is correct, it will not matter the size.

Another strength of this algorithm is the delivery speed. Although is is not computationally the quickest, given that the complexity is $(O(n^2))$, we can rest assured in the fact that if a package has an early deadline it will more likely than not be delivered on time.

Other possible algorithm choices

One other algorithm that could be used is a **heuristic algorithm**. Although this may not necessarily yield the most optimal or accurate route, computationally it is one of the fastest. One could simply search the packages by deadlines, and put them into the trucks based on the maximum *"weight"*, or capacity of the trucks. Such an algorithm would choose the highest value package, which in our case would be the one with the earliest deadline.

Another option could be a **breadth-first** search algorithm. This algorithm has a starting vertex, and subsequently visits the vertexes with the smallest distance to the vertex. We could set a list of items in each truck, and the BFS would start from the hub and move on to the closest hubs and mark the current distance once the node is found to be in the trucks list of packages.

The difference between the heuristic algorithm and the one I chose is that the heuristic chooses based on speed, but djikstra's algorithm sacrifices computational speed for efficiency.

Although the breadth-first search algorithm does look for a somewhat shortest path by visiting the closest nodes compared to the starting node, my algorithm looks through and searches for the shortest path without visiting unnecessary points.

WHAT I WOULD DO DIFFERENTLY

The major thing I would do differently is utilizing the hash table more. I used a JSON file to hold all the information, which worked out fine. But, if I had to go back I would update and pull data from the hash table, rather than updating and pulling from the JSON file.

If I felt it was possible and within my wheelhouse, I would utilize multiple algorithms in order to change the computational speed and make it more efficient.