**School of Computers and Information Engineering**

**STUDENT LEARNING ACTIVITY**

Fall Semester 2021

**Computer Algorithms (CIE3090)**

Traffic Jam Elimination System

Student Learning Assignment

**Submitted by:**

**U1910243, Anna Pak (Group #003)**

**U1910253, Ulugbek Shernazarov (Group #003)**

**U1910256, Azamat Sultanov (Group #003)**

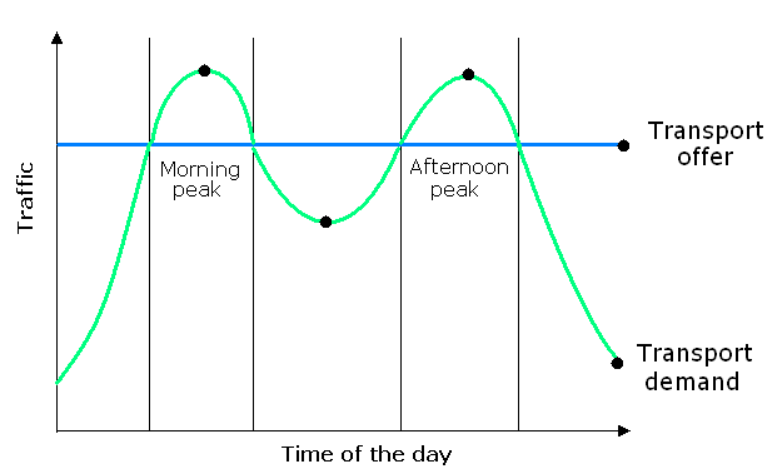
**Index Page**

|  |  |
| --- | --- |
| **Title** | **Page No** |
| Problem Description | **2** |
| Existing Solutions/Algorithms | **2** |
| Implementation | **3** |
| The Map | **4** |
| Algorithmic Steps (Pseudocode) | **7** |
| Complexity | **10** |
| Augmentation (space to improve) | **11** |
| Subject Importance | **11** |
| References | **13** |

**PROBLEM DESCRIPTION**

We know that all transportation means occur in the early mornings when every worker/student rushes to their destination in order to avoid being late, and in such cases, every minute counts. Because of the high number of vehicles on the road, traffic backups might linger for many hours. That will add another tension to an already nervous morning and cause stress in the work/study.

By studying the occurrence of traffic jams, our team figure out that most movements are made on the same route (traffic flow via one road) and cause traffic stuck while other possible routes to the same destination are free and comfy to use. Of course, it might take a few additional times to drive via longer path routes, but this would be much faster and safer in terms of the customer’s mental state, right? Furthermore, it is up to drivers to decide which road to take when driving, although this may be misleading if the majority of car owners make the same decision.



People may believe it is OK to arrive 15 minutes sooner at their destination, but it is far worse to wait 15 minutes or more on a stopped movement. In other words, it is considerably more beneficial for all passengers to arrive on time.

As a result of our understanding of the situation, we devised the following method to solve traffic congestion. Our technology will improve the efficiency of traffic movements, resulting in further advantages in the future.

Even while our algorithm may increase the distance a driver must go, the benefits of this strategy (time savings for passengers) may easily outweigh the higher costs for greater patrol and, more importantly, the driver's time (they can take another passenger rather than staying in traffic jam for more time).

**EXISTING SOLUTIONS/ALGORITHMS**

In the most nowadays’ transportation services, it is up to drivers to decide optimal route between to the passenger’s destination. Of course, all of them choose the shortest path, that will minimize fuel spent. It is beneficial in some extent, but such attitude leads to the traffic jam’ occurrence. It will happen because of problem we’ve mentioned above. The existing algorithms (Yandex’s algorithm, My-Taxi’s algorithm and others) do not calculate the traffic jam occurrence chance, they just display the possible route to the destination.

**IMPLEMENTATION**

1. All points of interest (vertexes) are identified and stored in the list of “paths”. Each vertex contains list of adjacent vertexes with cost to reach them.
2. Afterwards based on data contained in vertexes matrix with values to reach adjacent vertexes is built. Each entry in the matrix is a route which contains paths with all the costs to the vertexes.
3. Optimal routes are calculated using Floyd-Warshall algorithm (enhanced version of Dijkstra’s algorithm, that calculates optimal ways among all the vertexes)
4. Besides, all the results are stored in routes so that could be easily extracted later (e.g.: initial point A 🡪 D 🡪 K 🡪 J 🡪 destination F).
5. Based on time (morning, evening, night) timer for recalculation (5 mins, 15 mins, 25 mins etc.) is set up to repeat steps 3 and 4. There is also a trigger to monitor route overall cost based on number of cars that will activate emergency recalculation to avoid traffic jams.
6. User asks for a route and goes by the most optimal in time way. At the same time program tracks the number of cars passing by particular road.

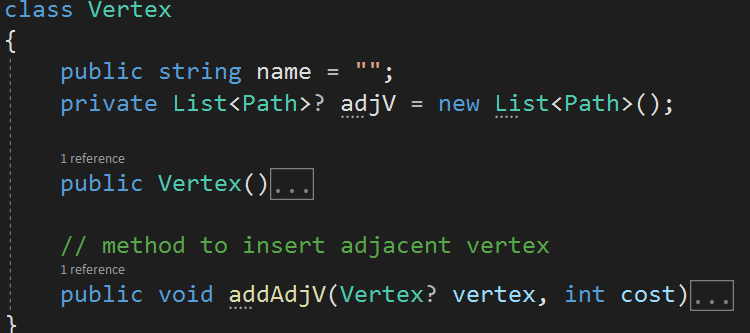
**Client-Server architecture – Server side**

The algorithm runs only in server side and send the results to users.

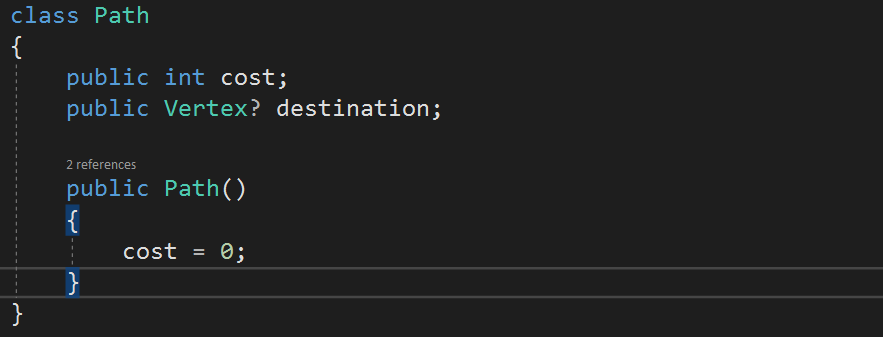
There is no need for the user in client-side calculations and separated server-side calculations (calculated once for all the users). Besides algorithm has adaptive running time based on intervals and trigger to monitor road costs (amount of time spent to pass the route based on traffic speed limitations).

**ALGORITHMIC STEPS (C# code)**

Vertex structure:



Path structure:



Route structure:



Floyd-Warshall Algorithm:

public void recount(int i, Matrix prev) // Floyd Warshall Algorithm

// i indicated the row throw which calculations are made

{

for (int j = 0; j < size; j++) // for every column

{

for (int k = 0; k < size; k++) // in every row

{

if (j == i || k == i || j == k) // copy diagonal + column and row of vertex, through which

// algorithm is looking for better path

{

matrix[j, k] = prev.getElement(j, k);

}

else

{

if (prev.getElement(j, k).overallCost() != Math.Min(prev.getElement(j, k).overallCost(), prev.getElement(j, i).overallCost() + prev.getElement(i, k).overallCost())) // checks which path is more optimal and stores the results

{

matrix[j, k].route.Clear();

foreach (Path instance in prev.getElement(j, i).route)

{

matrix[j, k].route.Add(instance);

}

foreach (Path instance in prev.getElement(i, k).route)

{

matrix[j, k].route.Add(instance);

}

}

else

{

matrix[j, k] = prev.getElement(j, k);

}

}

}

}

}

As can be seen all the vertexes that make up an optimal way are stored so that can be retrieved later.

Full version of the program is by this [link](https://github.com/Aidein/CA_Proj.git).

**COMPLEXITY**

Floyd-Warshall algorithm (dynamic programming) with time complexity **O(n3)**, where n is the number of vertices (|V|) in G.

Besides there is a possible reassigning of route so in the worst case there could be **n**. Also, to retrieve the trip in worst case there is **n** complexity.

Overall complexity is **O( n + n3 \* n ) = O(n4)**.

**AUGMENTATION (space to improve)**

**Map**

Using currently existing maps and communicating with them through their specified API will be the most cost-effective approach to develop a navigation system (map, in particular). Without a doubt, our example map cannot be superior than current ones (such as Google Maps). We may simplify and improve our algorithm by using Google map API methods to replace certain of its sections. Furthermore, Google Maps has already created maps for the majority of countries. If we apply it, our method and software may be used in many more nations in the future.

**Consider road traffic**

Taking into account traffic on the roadways might also help us choose the optimum path. Data from Google or Yandex maps may be used to implement it (through their API). Using that data we could derive formula to calculate the cost of the trip considering the time, distance and speed constraints.

**SUBJECT IMPORTANCE**

This topic offered us a broad understanding of what algorithms are and how they are employed in daily activities. We can obtain a notion of how to enhance a product by understanding how it is built. We learned how to study the heart of a program, assess its time and space complexity, and reuse and alter it in the Computer Algorithms course. Furthermore, our viewpoint on programming was hazy a year or two ago. We were acquainted with programming languages and how to choose the best solution for a given problem.

In computer programming, each work may be completed in an unlimited number of ways. Each algorithm has its own set of benefits and drawbacks in certain scenarios. Understanding the benefits and drawbacks of several algorithms helps you to choose the best one for your needs. The given input or the amount of data that must be submitted may have an impact on their behavior. You can select the best one for your scenario by examining their advantages and disadvantages.

It's vital to note that computer programs employ various algorithms that are executed on hardware that includes a processor and memory, both of which have restrictions. A processor's speed and the quantity of memory accessible are not infinite. As a result, they have a limited quantity of resources at their disposal.

Algorithms give a powerful framework for comprehending an issue. To see if an issue is solved, we can apply an algorithm. How fast and accurately can this be accomplished? We can use an algorithm to see if we can solve a section of the problem.

It's an essential factor in this course and in the IT industry in general so that you'll ought to be able to see the large picture and study the issue from a variety of angles.

**REFERENCES**

* <https://www.youtube.com/watch?v=oNI0rf2P9gE&t=296s>
* [The Path Optimization Algorithm of Car Navigation System considering Node Attributes under Time-Invariant Network (hindawi.com)](https://www.hindawi.com/journals/misy/2021/2963092/)