Using A* Algorithm in Onion Routing Pathfinding

Naufal Alexander Suryasumirat / 13519135
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13519135@std.stei.itb.ac.id

***Abstract*— Online security is top priority in this day and age where people can easily connect and communicate between each other disregarding distance unlike ever before. For that reason, online security, personal data, and online privacy should be uphold to the same standards as people keeping their real life personal identity safe. One way of keeping personal data and privacy safe online is to anonymize yourself while browsing using onion routing, yet some onion routing take too long to respond and use on a daily basis, this is where the A* algorithm can be applied with its heuristic function to find the shortest connection while still maintaining security and anonimity online.**

Keywords—Onion Routing, A* algorithm, online security, online privacy;

I. INTRODUCTION

In this present age, the modern era, using the internet to browse for information, entertainment, news purposes, and generally staying informed about current events in the world is a common activity people engage in everyday, and looking at the trend, which is only improving and the amount of people engaging in this activity is only going up each and every single day, it will become an even more common thing to do to pass the time. For this very reason, keeping your online security, online privacy, and personal data safe is a must for people to do, which is why it is one of the reason for onion routing being founded.

Normal or common web browsers such as Google Chrome, Mozilla Firefox, Opera GX, or Microsoft Edge usually don't provide the best online privacy and security for users who use them, some of them even collect your data for their purposes and gain, mostly to cater their users with advertisements that suits them better, or more nefariously a small percentage of them might even sell your data. To avoid these disadvantages of using normal web browsers to browse the internet, some people use anonymized connectivity such as a VPN or onion routing.

Onion routing is a way to provide anonymous connection for internet browsing which allows anonymous communication between computers or servers by encapulasting a message that was sent by the user between each intermediary nodes and decrypting the message when receiving them between intermediary nodes. It acts as layers of an onion.

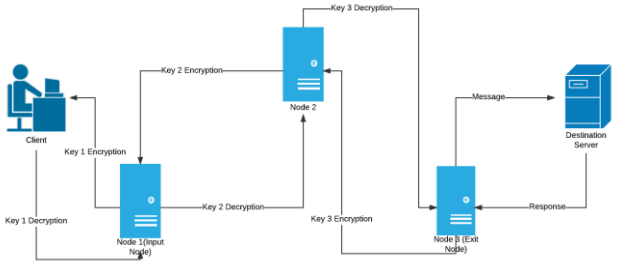

Image 1. An illustration of how onion routing works by encrypting between intermediary nodes before reaching the destination (source: https://media.geeksforgeeks.org/wp-content/uploads/Onion-Routing-Page-1.png)

Onion routing protects the privacy and anonimity of internet communicators against agents or people that might want to analyze the TRAFFIC between websites, eavesdrop on connecitons, or take a peek at what a person is doing on the internet. The method or algorithm for route selection that is used for selecting nodes to connect to or path to use before reaching the goal and destination might take too long relative to normal web browsers because of the need to use intermediary nodes to encapsulate the message sent by the user and connect between other intemediary nodes.

The algorithm mentioned above works by non-deterministically choosing nodes as intermediary nodes to connect to as a method of privacy preserving to avoid repeating routes. This algorithm is defined as a route planning algorithm but one that only preserves privacy and does not fully optimize connection time while using onion routing method and might be bad for people that are concerned by a slow connection time when accessing websites or communicating to sites.

This route planning algorithm that onion routing uses traverses nodes or computers in the network that is viewed as a graph that contains those nodes with the connection that represents real life connection between those computers or servers. While the method mentioned above best preserves anonimity of users for the reason that it is non-deterministically chosen, the method might not be great enough for people expecting fast connection, which is where the A* algorithm comes to play to solve the issue and slightly improve the algorithm while still maintaining adequate anonimity of the users. This A* algorithm will be used to search for the fastest and shortest path between random nodes to optimize connection time and make the experience for users generally better for daily usage.

II. THEORY

A. *Onion Routing*

Onion routing is a method of computer network communiation in internet browsing method that enables the users to appear as anonymous in the computer network itself to spying eyes and uses a different technique of communication over a computer network than what the normal internet browsing method. Onion routing works by encapsulating the message with layers of encryption, which can be visualized as an onion with layers that is sent by the client or users which is then passed around through the intermediary nodes between the starting node or the user's computer and the destination which would be the node that the user is trying to access before reaching it.

Usually, in a general onion routing service, there are three intermediary nodes between the user and the destination, which by itself has a key and the instruction to encrypt a sent message and decrypt a received message before passing it on to the destination or the client. These messages are being sent through the computer network and can be visualized a graph of nodes, with the message being sent using a pathfinding algorithm to determine the path it takes to reach the destination. Onion routing itself has many implementations in the real world, but most of it works in the same way with the same method to encrypt the user's message and decrypting them after.

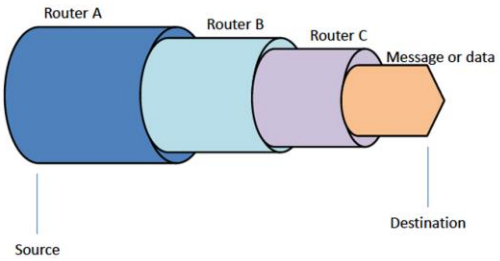

Image 2. An illustration of how onion routing encrypts a message or data that is passed around between intermediary nodes (source: https://1technation.com/wp-content/uploads/2016/12/tech-savvy.jpg)

In real life implementation, most onion routing uses three intermediary nodes before reaching its destination and uses an encryption method which encrypts the user's message with different layers of encryption everytime a message from the user is received and then passed on to reach the destination. Although some method might use more than three intermediary nodes to preserve security and safety better, using more than three nodes would make the user's experience significantly worse than it needs to be while not improving security as much, simply put the cost would be much more than the benefit. This method of encryption and routing has some vulnerabilities, but it is in general more safe and best anonymizes the user's message from people that might be spying on the network traffic. Though this routing method has many advantages, which is being more secure than normal routing methods, protects the user's personal data and privacy, this routing method has some disadvantages.

The disadvantages of an onion routing method is primarly being the time it takes for a user's message to be received by the destination and the time it takes for the response from the destination to the user. This problem existst because as has been mentioned before, onion routing uses intermediary nodes for the message being sent before it is then received, disregarding the time it takes to encrypt and decrypt the message with each passing of an intermediary node, the time it takes to even just make a connection between those nodes and send the message through those nodes might add a considerable amount of time for the message to be responded and being received again by the user and it has the possibility to make the user's experience of simply browsing the internet unbearable and painfully slow.

Another reason of onion routing being in general slower than normal browsing method is that the path that the message takes to the destination is non-deterministic, or in other terms the path it takes is randomized as to preserve privacy and make it harder for spying eyes to analyze the traffic to determine the identity of the user that is trying to connect to a computer network, since if it is randomized with no pre-determined pattern, it would make it harder for people trying to intercept your message. The onion routing method is better visualized as as a dynamic graph of nodes with each nodes that can have a connection between all other nodes or a complete graph or would be better visualized as a starting node and a destination node with a cloud of network between those nodes as shown below. One solution proposed by the author to this problem is to non-deterministically choose an amount of nodes in the network as a candidate to which the user can connect to and using the power of the A* algorithm and its heuristic function to choose the best, shortest, and fastest connection between the user's computer to the destination node while still maintaining adequate privacy for the user and still protect the user's personal data.

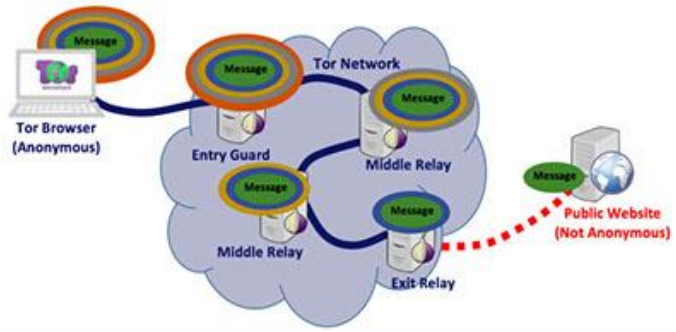

Image 3. A representation of an onion routing network or tor network as a cloud of nodes from the outside (source: https://www.axontechnologies.com/images/1__0zdvagxjbzdom2nrtr0zq393x196.jpg?crc=376330005)

B. Graph

Graph is an abstract data type or ADT for short which is used to represent the complex inner workings or complex relationship between objects that are represented as nodes or vertices in the graph with the connection between each nodes or vertices representation whether those nodes are related or not. There are many types of graphs such as an undirected graph with no pointing arrows used as its edges which is called an undirected graph that is different from graphs with directed edges known as directed graphs. There are other types of graphs but in general, graphs are used to solve problems by visualizing it as a network of related vertices. From this abstract data structure, a solution of shortest path can be found between nodes using many algorithms known as route planning algorithms or path finding algorithms. An example of a graph is one shown in Image 3 that represents real world problem that is a tor network as a graph that can be solved to find the shortest route.

C. Route Planning Algorithms

Route planning algorithms are algorithms that are used to traverse nodes in a graph to search for and find a path from an initial node to a destination node. Most of the time, in computer science, route planning algorithms are used to find the shortest, and most cost-effective path between the initial node and the destination using many different algorithms.

There are two types of route planning algirhtms in computer science, which is uninformed search and informed search. Uninformed search, or called as its other name, blind search or unguided search, is a general purpose search algorithms or route planning algorithms that use no additional information about the state of the graph that it is trying to traverse and tries to find a solution based on the predefined workings of the algorithm, whereas an informed search algorithm have additional information of the goal node or destination node and it uses that information to find a better and more efficient path or solution to reach the destination, most of which use heuristic functions as the additional information being used to reach the destination.

Route planning algorithms that are classified as uninformed search or blind search are breadth first search, depth first search, depth limited search, iterative deepening search, bidirectional search, and uniform cost search. Each of the mentioned algorithms have its own signature or modifies the signature of another algorithm for cost-effective reasons to find a better solution, for example, the breadth-first search algorithm uses a queue as a mean to store expanded nodes to expand later between each iteration of the algorithm to find a solution which always produces a solution that takes the minimum amount of steps to reach the solution but would have disadvantages if those minimum amounts of steps turns out to cost more than another path that takes more steps with better efficiency, this is then improved in uniform cost search which uses another method to store its expanded nodes known as a priority queue with each expanded node having its own cost and the algorithm chooses the first node with the least cost first to find the solution.

On the other hand, route planning algorithms that are classified as informed search are greedy best first search and A* algorithm. The essence of greedy best-first search algorithm is to use an evaluation function for each of its node in the graph that comes from the heuristic function of the node or the estimation of cost from the expanded node to the goal. This method that the greedy best-first search algorihtm uses is then further improved in the A* algorithm which expands the evaluation function to not only account for its nodes' heuristic estimation but also accounts for the cost of the path overall on top of it, this algorithm also avoids expanding paths that are already expensive.

D. A* Algorithm

As has been mentioned before, A* algorithm is a route planning algorithm and graph traversal algorithm that is classified as an informed search algorithm that uses additional information to better find a more efficient and more cost-effective path to the solution, better than uninformed searches. The A* algorithm uses heuristic search that estimates the value of a node and an evaluation function unique to the algorithm that essentially avoids expanding paths that are already expensive by expanding paths with the least cost first.

The A* algorithm has an evalution function of:

$$f(n) = g(n) + h(n)$$

$f(n)$ represents the estimated total cost of path through n to the goal, whereas $g(n)$ is the cost so far to reach n, and $h(n)$ is the estimated cost from n to reach the goal. With this evaluation function, the result of a path search using A* algorithm always provides the best solution or a path with the least cost and with the most most efficiency and optimal if the heuristic function or $h(n)$ above always underestimates the true cost and is consistent with each and every node.

The most essential property of the A* algorithm is that the algorithm produces a complete solution if the graph has a non-infinite amount of nodes in it, with an exponential time complexity and an exponential space complexity, but provides an optimal solution or the best solution if the heuristic function is admissible as mentioned before. An admissible heuristics is an optimistic estimation that never overestimates the cost of a node reaching a goal and is consisten within each nodes.

E. Heuristic Value

A node in a graph with heuristic values are used for heuristic searches or an algorithm strategy that used the heuristic value to estimate or measure the cost from the node to the goal. A heuristic value for heuristic search has a purpose to estimate the value of a node by contemplating on four different factors which are the promise of a node, the difficulty of solving the subprolem, the quality of solution provided represented by the node, and the amount of information gained from the node. The heuristic values in route planning, especially A* algorithm will provide the best result only if it is admissible as mentioned above.

F. *Ping and Connection*

A ping is a signal that was sent to a host that is requesting a response from thhe user that contains bytes of data or an echo request. Ping is used to check the availability of the host that is expected to give a response and to measure how long it takes for the response to be processsed and received by the user. The response time of a ping is important to the user's experience to browse the internet, consume online entertainment, and even playing online games, because a high ping might cause a detraction in a user's overall internet connection experience.

Pings are slightly affected by internet connection speed, but is mostly affected by physical distance between the source or initial node and destination node in the computer network, which means that physical distances correlates more to how high a ping is more than the quality of internet connection that the user has. In a perfect world, the response time of a ping would be the speed that light travels between the source and destination in the computer netowrk, but since it is impossible to achieve perfection, normal ping response times are much higher than the time it takes for light to travel between the destination and source. This ping response time is used to determine the cost of connection between hosts in a computer network or in the internet using real life distance between the source and destination. Ping calculations in the applied A* algorihtm would use the haversine distance between two coordinates in the globe and estimate the ping using the haversince distance of those two coordinates.

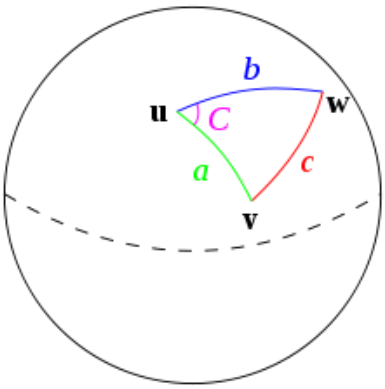

Image 4. Visualization of the haversine formula calculating the distance of two coordinates on a sphere or on earth (source: https://upload.wikimedia.org/wikipedia/commons/thumb/3/38/Law-of-haversines.svg/220px-Law-of-haversines.svg.png)

III. Application

The onion routing network or onion network that is used here is represented as nodes in a graph with the graph being the onion network itself and the nodes represents real world locations such as Indonesia, Singapore, Hong Kong, and other countries with its coordinates that contains the server or candidate host for connection to be made by the users whether as intermediary nodes or as the destination nodes. For these purposes, to make the problem simpler, the author decides to use the most important factor to determine the cost of traveling between each nodes in the graph as the calculation of the haversine distance between each nodes' coordinates and adjusted to match real world ping situation of the author. The ping in this matters represents the connection time between each nodes as accurately as is possible by the author to represent the ping in real life situations.

In most onion networks, the normal amount of intermediary nodes used is three, for this reason, the algorithm that is applied to the problem searches for the shortest or most cost-effective route using three intermediary nodes to replicate real life conditions trying to connect to a server using onion routing.

A. *How the A* Algorithm Works*

The A* Algorithm works by first selecting random candidates for possible connections as intermediary nodes between the source and destination, the amount of nodes chosen is based on the amount of nodes given and if it exceeds a certain amount the default amount of nodes is chosen. As mentioned before, the selection of the nodes is randomized to make it harder for spying eyes to analyze the network to determine your identity, in other words to preserve anonimity of the user connecting to the network.

The random selection must be non-deterministic to make it harder for interceptors to guess the path to the destination. The nodes chosen and the graph given as the problem is represented as a completed graph with every node connected to every other node in the graph to simplify the problem. The graph is given in a text file format containing every node in the network with its location to determine the ping between each nodes.

The steps on how the algorithm works are as follows:

1. Reads file and create graph object with nodes objects from the file.
2. If the amount of given nodes from the file is insufficient, the program is stopped and will prompt the user to give more nodes in the file.
3. Choosing the source node and destination node from the network.
4. Choosing candidate intermediary nodes non-deterministically based on the amount of nodes given from the read file.
5. Place the source node as path in the priority queue with the cost being the heuristic value calculated.
6. Pick the first path from the priority queue, which is the path with the least cost to expand and remove the picked node from the priority queue.
7. Pick every other connected nodes to expand and add the paths back to the priority queue with a newly calculated evaluation function.
8. Back to step 6
9. The algorithm always produces a solution to the problem if the amount of nodes given by the problem is sufficient, the algorithm stops when the destination is reached with the optimal cost for the path.
10. The program then outputs the result to the screen for the user to look at.

B. *Heuristic Evaluation Function*

The evaluation function used in the A* algorithm to find the heuristic value of the expanded node or path is as follows. The algorithm calculates the path cost so far to reach the node using the haversine distance between nodes and dividing it by the speed of light which is 299,792,458 meters per second and adjusting the result to match real life ping and connection situations based on the author's experience. Whereas the heuristic distance is calculated by pinging to the destination node from the expanded node to estimate the cost to reach the goal, which then is added with the cost so far and is used as the overall cost of expanding the path for the next iteration.

C. *Implementation*

The implementation of the program uses an object oriented paradigm to define the Graph class, Node class, and FileHandler class. The Graph class represents the onion network in the real world, whereas the node class represents the intermediary nodes in an onion network as well as the source and destination nodes. The FileHandler class handles problems given by the user.

The pseudocode of the A* algorithm and the evaluation function of the algorithm for onion route planning uses C# language and the full code is available at the linked github repository. The following is the pseudocode for the A* algorithm implemented.

```
procedure AStar(from, to: string):
    start ← findNode(from)
    goal ← findNode(to)
    selected ← selectRandom(from, to)
       {Selects random candidate nodes}
    if (selected = Empty) then stop
    PQueue[Path, Cost, EstimatedCost]
       {Declares new PriorityQueue}
    PQueue.Add(start, 0, 0)
       {Adds source node to PriorityQueue}
    while not PQueue.IsEmpty()
       Head ← PQueue.Pop()
       if (Head.Path.Last() = Goal) then
          output Head
          stop
       else
          if (Head.Path.Count = 4) then
             PQueue.Add(GoalPath)
          else
             expanded_path ← expand(Head)
             calcHeuristic(expanded_path)
             PQueue.Add(expanded_path)
```

The implementation of the A* algorithm in C# language is as follows

```
public void AStar(string from, string to)
{
    Node StartNode = this.findNode(from);
    Node GoalNode = this.findNode(to);
    List<Node> selected = this.SelectRandom(from, to);
    if (selected == null) return;
    List<Tuple<List<Node>, double, double>> PQueue = new List<Tuple<List<Node>, double, double>>();
    Console.WriteLine("Finding connection from " + from + " to " + to);
    List<Node> first = new List<Node>();
    first.Add(StartNode);
    PQueue.Add(new Tuple<List<Node>, double, double>(first, 0, StartNode.CalcPing(GoalNode)));
    while (PQueue.Count > 0)
    {
        var Head = PQueue[0]; // Head of PQueue
        PQueue.RemoveAt(0); // Popping Head
        if (Head.Item1.Count == 5) // Shortest Path
        {
            Console.WriteLine("Found path connection:");
            for (int i = 0; i < 5; i++)
            {
                Console.Write(Head.Item1[i].GetName());
                if (i != 4) Console.Write(" ↔ ");
                else Console.WriteLine(".");
            }
            PQueue.Clear();
            return;
        }
        else
        {
            if (Head.Item1.Count == 4)
            {
                List<Node> toAdd = new List<Node>(Head.Item1) { GoalNode };
                double pingDis = Head.Item2 + Head.Item1.Last().CalcPing(GoalNode);
                PQueue.Add(new Tuple<List<Node>, double, double>(toAdd, pingDis, pingDis));
            }
            else
            {
                foreach (var node in selected)
                {
                    if (Head.Item1.Contains(node)) continue; // Already in path, continue
                    List<Node> toAdd = new List<Node>(Head.Item1) { node };
                    var ping = Head.Item1.Last().CalcPing(node);
                    PQueue.Add(new Tuple<List<Node>, double, double>
                        (toAdd, Head.Item2 + ping, Head.Item2 + ping + node.CalcPing(GoalNode)));
                }
            }
            PQueue.Sort((x, y) => x.Item3.CompareTo(y.Item3));
        }
    }
    return;
}
```

Image 5. Implementation of A* Algorithm to find the optimal path in an onion network (source: screenshot of algorithm implementation in C# language)

D. *Experiments*

The algorithm is tested without a Graphical User Interface, instead only using the command line, and using a text file to read the nodes contained in the graph problem. Here is an example of the text input file used in the program.

```
Indonesia -0.7893 113.9213
Singapore 1.3521 103.8198
Switzerland 46.818188 8.22751
```

The example above is the format for the text file, each line must contain the name of the node with the location of the node in decimal latitude and decimal longitude, which is the information used to calculate the ping for the heuristic function.

Here are the results of the program after reading 50 nodes representing countries the user's trying to connect to with the same format as shown above. The A* algorithm in this case uses Indonesia as the source node and Singapore as the destination node. Here are the results of running the algorihtm.

```
Randomized selected nodes:
Location: Netherlands
Location: Brazil
Location: HongKong
Location: Canada
Location: Myanmar
Location: Austria
Location: Brunei
Location: UK
Location: Israel
Location: Japan
Finding connection from Indonesia to Singapore
Found path connection:
Indonesia ↔ Brunei ↔ HongKong ↔ Myanmar ↔ Singapore.
Cost/Ping: 114.993231428119 ms
```

Image 6. Result of onion route pathfinding with A* algorithm from Indonesia to Singapore (source: screenshot from command line)

From the results shown above, we can concur that the algorithm successfully found the optimal path for onion routing after randomizing available nodes and selecting them randomly the optimal path is shown above with the cost in the response time the path takes for the user.

IV. Analysis

From the Application section (Section III), the author has shown that an onion routing that maintains security while still providing the user with acceptable experience or response time to connect is possible, but there are several vulnerabilities with the algorithm constructed by the author. The vulnerabilities are as follows:

1. The randomized selection of nodes in the graph to use as candidate intermediary nodes between the source node and destination node is not purely random as it is not non-deterministic and can be predicted because it uses the Random class given by the C# System.
2. As a result of the not fully randomized selection of nodes, this algorithm does not fully preserve the user's anonimity for the reason that if an outsider intercepts the algorithm and guesses the selected nodes for the algorithm to calculate the shortest path to, the interceptor could theoretically do the same calculations as the algorithm did and determine the path the message took to get to the source, which is the main reason people would be using onion routing in the first place.
3. Since the algorithm does not fully preserve the user's anonimity for the reasons mentioned above, using this algorithm is no better than just connecting to the destination normally using a normal routing technique which would result in a better user experience because the algorithm provided above just delays the inevitable moment someone intercepts the connection.
4. In some cases, if the randomized selection selects nodes that are too far away from each other, the algorithm would take exponentially more time to complete because the algorithm works on the ability for the node to ping other nodes to determine the cost of the path, which would take longer to complete the solution finding for the problem.
5. In real world solution, pinging other nodes to determine the cost to other nodes is inefficient because it takes real world time to calculate the ping of each other nodes and should be replaced with an estimated ping value instead of calculating the ping while the algorithm is running because it would make the algorithm slower.
6. The algorithm does not solve the real world problem of exit node vulnerabilities in an onion routing network.

V. Improvement Suggestions

To improve this A* algorithm to find the fastest route in an onion network while still maintaining security, privacy, and the user's anonimity, there are several improvements that could be made. The improvements are as follows.

1. Make the random selection of intermediary node candidates purely random or non-deterministic or at the least approaching purely random to avoid unwanted interceptors to guess the selected nodes.
2. Calculate or estimate ping from each and every node to each other nodes before running the algorithm, because in real world implementations, the calculation of ping between nodes would take too much time on top of the algorithm's exponential time complexity.

VI. Conclusion

Using onion routing to browse the internet means that a person cares about their privacy and online safety and more people should care about their privacy online, but while it is secure, the experience might not be the best if you are a user expecting a fast response time from using onion routing since the method itself requires time to pass the message around intermediary nodes, even disregarding the time it takes to encrypt and decrypt the message sent or received it takes more time than the normal method. The algorithm proposed by the author is not perfect but would make the connection faster for users. The algorithm still has a lot of vulnerabilities and still have much room for improvement but might be a foundation to a better algorithm for future improvements.

Video Link at Youtube

https://youtu.be/fw1mM8yA0IY

Source Code Link

https://github.com/naufalsuryasumirat/13519135_Makalah

ACKNOWLEDGMENT

The author thanks The One Almighty God,while this paper is far from perfection and far from the optimal quality of paper, the paper is finished with the best efforts from the author, which could not have been done without the help from The One Almighty God. This paper would also have not been completed without the help from friends, parents, and the honorable lecturers of Algorithm Strategy IF2211, Mr. Rinaldi Munir, Mrs.Nur Ulfa Maulidevi, and many more, especially Mr. Dwi Hendratmo Widyantoro, lecturer of Algorithm Strategy IF2211 Class 03.

REFERENCES

[1] Penentuan Rute (*Route/Path Planning*) Bagian 1: BFS, DFS, UCS, Greedy Best First Search, https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian1-2021.pdf, accessed on May 8, 2021.

[2] Penentuan Rute (*Route/Path Planning*) Bagian 2: Algoritma A*, https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian2-2021.pdf accessed on May 8, 2021.

[3] Onion Routing: Investigation of Route Selection Algorithms https://www.onion-router.net/Archives/Route/index.html accessed on May 8, 2021.

[4] Andriy Panchenko, Johannes Renner, Path Selection Metrics for Performance-Improved Onion Routing.

[5] Paul F., David M., Michael G. Anonymous Connections and Onion Routing.

[6] Ping Definition Tech Terms, https://techterms.com/definition/ping accessed on May 9, 2021.

STATEMENT

I hereby declare that the paper I write is my own writing, not an adaptation or translation of someone else's paper, and not a result of plagiarism.

Jakarta, 10 Mei 2021

Naufal Alexander Suryasumirat
13519135