Application of Travelling Salesman Problem for Optimizing Island Exploration Routes in Sea of Thieves

Eduard Daniel Ariajaya - 13524129
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: eduard.ariajaya@gmail.com , 13524129@std.stei.itb.ac.id

***Abstract*—Sea of Thieves is a pirate adventure game released on March 20, 2018. Players can take various available missions and try to explore various locations to complete the mission and get the treasure. So that, route planning in exploring various locations becomes an important aspect so that time and resources can be used optimally. This study models the game map as an undirected weighted graph, with islands as nodes and shipping lanes as edges. By applying the Travelling Salesman Problem (TSP) algorithm, to find the shortest path that visits all islands exactly once and returns to the starting point. The results of this approach provide an optimal solution for island route exploration, while also demonstrating how graph theory can be used in a strategic decision-making game.**

Keywords— Sea of Thieves, Travelling Salesman Problem, Graph, Route Optimization

I. INTRODUCTION

Sea of Thieves is an online open-world pirate adventure game that allows players to explore the oceans to complete quests, collect hidden treasures, and fight or form alliances with other players. In the game, players can take on various missions at various Outposts on the map. An Outpost is the island location where players first spawn in the game, and where players can take on a number of missions and purchase various cosmetic items before setting sail. By taking on a mission, players must visit several islands to complete the mission before returning to the Outpost to sell the treasure or other valuables that the player has obtained. Since players are required to visit several islands, players must be able to determine the most optimal route to save time and resources.

The problem in this game is very similar to the Travelling Salesman Problem (TSP), which aims to find the shortest path by visiting each island location exactly once and returning to the starting point (Outpost). In the game, the distance between islands varies where each 1 grid on the map represents 1 nautical mile and without a systematic approach, players may choose a longer or roundabout path. By using an optimal route approach, the quality of the player's playing experience will be improved, and travel time will be more efficient.

Graph theory can be used to model and solve the exploration route problem in this game. With each island represented as a node in an undirected weighted graph, while the sea routes between islands are represented as edges with weights in the form of distances. With this modeling, the TSP algorithm can be used to find the optimal exploration route.

The purpose of this paper is to optimize the island exploration route in the game Sea of Thieves with the Travelling Salesman Problem approach. Hopefully, this approach can be an example of the application of discrete mathematics that is relevant and applicable in a digital game.

II. THEORETICAL FOUNDATION

A. Graph Theory

Graphs in discrete mathematics are used to represent objects (vertices) and the relationships between these objects (edges). According to Rinaldi Munir (2024), a graph is defined as an ordered pair G = (V, E), where V is a non-empty set of vertices, and E is a set of edges connecting a pair of vertices.

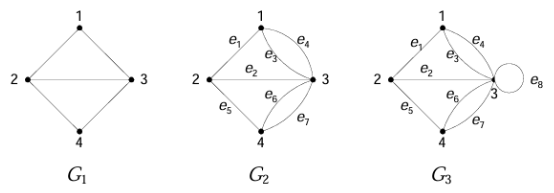


Figure II-1. (a) simple graph, (b) dual graph, and (c) pseudo graph
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

Based on the orientation of the direction on the side, graphs are divided into 2 types:

- Undirected graph: A graph whose edges have no directional orientation is called an undirected graph.
- Directed graph: A graph in which each edge is given a directional orientation is called a directed graph.

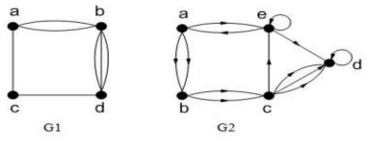


Figure II-2. a) undirected graph, b) directed graph
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

1) *Terminology in Graphs*

a) *Adjacent*

Two vertices are said to be adjacent if they are directly connected by an edge.

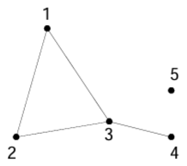


Figure II-3. Graph G3
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

From graph G3: Node 1 is adjacent to nodes 2 and 3. Node 1 is adjacent to nodes 1, 2 and 4.

b) *Incidency*

Incidence refers to the relationship between nodes and edges in a graph.

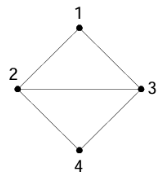


Figure II-4. Graph G1
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

From graph G1: edge (2, 3) is incident to vertex 2 and vertex 3, edge (2, 4) is incident to vertex 2 and vertex 4, but edge (1, 2) is not incident to vertex 4.

c) *Path*

A path of length n from a starting vertex v_0 to a goal vertex v_n in a graph G is an alternating sequence of vertices and edges of the form $v_0, e_1, v_1, e_2, v_2, \ldots, v_{n-1}, e_n, v_n$ such that $e_1 = (v_0, v_1)$, $e_2 = (v_1, v_2), \ldots, e_n = (v_{n-1}, v_n)$ are the edges of the graph G [1].

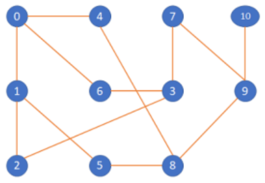


Figure II-5. Unweighted Graph
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

One of the paths in the graph is 0, 6, 3, 7, 9, 10. Where the sequence of the paths is the path from vertex 0 to 10

d) *Circuit or Cycle*

A path that starts and ends at the same node is called a circuit or cycle [1].

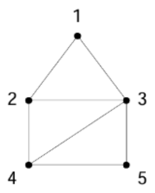


Figure II-6. Unweigthed Graph
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

One of the circuits in the graph is the path 1, 2, 4, 5, 3, 1.

e) *Weighted Graph*

A weighted graph is a graph in which each edge has a value called a weight. Weights can represent things like distance, cost, or other things.

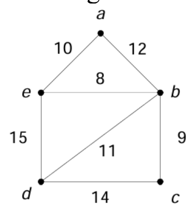


Figure II-7. Weighted Graph
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

f) *Complete Graph*

A complete graph is a simple undirected graph in which every distinct vertex is directly connected to every other vertex by an edge. In other words, if there are n vertices in the graph, then each vertex has edges to every other vertex, so the total number of edges in a complete graph is:

$$E = \frac{n(n-1)}{2}$$

[2]

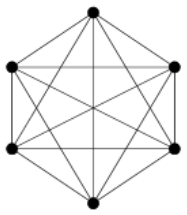
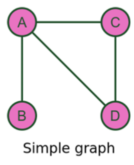


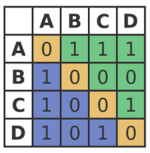
Figure II-8. Complete Graph
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

2) *Graph Representation*

a) *Adjacency Matrix*

In this form, a graph with n vertices is represented in the form of a $n \times n$ matrix, where the element $M[i][j]$ indicates whether or not there is an edge connecting the i-vertex to the j-vertex. $M[i][j] = 1$, if nodes i and j are adjacent. $M[i][j] = 0$, if nodes i and j are not adjacent



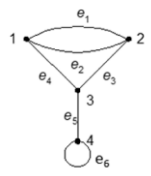


	A	B	C	D
A	0	1	1	1
B	1	0	0	0
C	1	0	0	1
D	1	0	1	0

Figure II-9. Adjacency Matrix
https://graphicmaths.com/computer-science/graph-theory/adjacency-matrices/

b) Incidency Matrix

A graph has n vertices and m edges, then the incidence matrix is represented in the form of a matrix of size $n \times m$ with $M[i][j]$. $M[i][j] = 1$ if node i is adjacent to edge j. $M[i][j] = 0$ if node i is not adjacent to edge j.



$$\begin{array}{c c} & \begin{array}{cccccc} e1 & e2 & e3 & e4 & e5 & e6 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

Figure II-10. a) Undirected Graph b) Incidency Matrix
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/21-Graf-Bagian2-2024.pdf

3) Hamilton Path and Circuit

A Hamiltonian path is a path in a graph that visits each vertex exactly once, without returning to the starting vertex. A Hamiltonian circuit is a path that visits each vertex exactly once, and returns to the starting vertex.

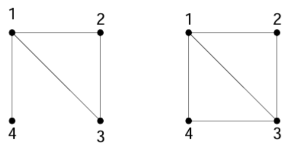


Figure II-11. a) Graph with Hamilton Path b) Graph with Hamilton Circuit
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/22-Graf-Bagian3-2024.pdf

B. *Travelling Salesman Problem (TSP) Application*

Traveling Salesman Problem (TSP) is one of the problems in the application of graph theory and discrete mathematics. TSP aims to find the shortest path by visiting each vertex in the graph exactly once and returning to the starting vertex. This goal is similar to determining the Hamilton circuit with minimum weight [3].

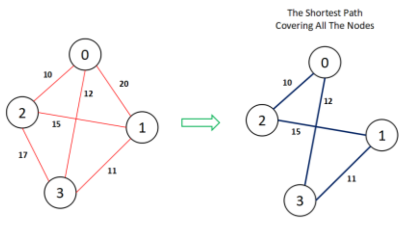


Figure II-12. TSP Problem
https://iq.opengenus.org/travelling-salesman-problem-brute-force/

C. *Dynamic Programming (Held-Karp) Algorithm*

Dynamic Programming is a method that breaks down a complex problem into subproblems, stores the results of the subproblems, and determines the most optimal final solution from the results. This method is very effective when the problem has two main properties, namely overlapping subproblems (the same subproblems appear repeatedly) and optimal substructure (the solution to the main problem can be built from the optimal solutions of its subproblems). According to Cormen et al. in Introduction to Algorithms, dynamic programming is used when simple recursive solutions result in many inefficient recomputations, and this approach can reduce the time complexity from exponential to polynomial in many cases [7].

The Held-Karp algorithm based on Dynamic Programming is an algorithm that can be used to solve the Traveling Salesman Problem (TSP). This algorithm stores the optimal results for a subset of vertices that have been visited and determines the minimum total weight to reach a particular vertex from the starting vertex. Held and Karp (1962) introduced this approach by stating that the TSP can be solved in time $O(n^2 \cdot 2^n)$, much better than exploring all route permutations totaling $n!$. Although the complexity is still exponential, this approach allows solving the TSP for small to medium graph sizes with an optimal approach.

By applying the Held-Karp algorithm to the Sea of Thieves game, the optimal island exploration route can be calculated so that players can complete their missions while saving time and resources.

D. *Euclidean Distance*

Euclidean Distance is defined as the distance between two points in Euclidean space. To find the distance between two points, the length of the line segment that connects the two points should be measured [9]. If there are two points $A(x_1, y_1)$ and $B(x_2, y_2)$. The Euclidean distance between the two points is calculated using the formula:

$$d(A,B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

[9]

In this study, Euclidean distance is used to calculate the weight of the edges in the graph to be created. This weight value reflects the distance that players need to travel to travel from one island to another directly. The use of Euclidean distance is because the map in the game is two-dimensional and has coordinates that can be seen directly, so the distance between vertices can be considered as a straight line between island coordinates.

III. METHOD

A. Game Map Visualization

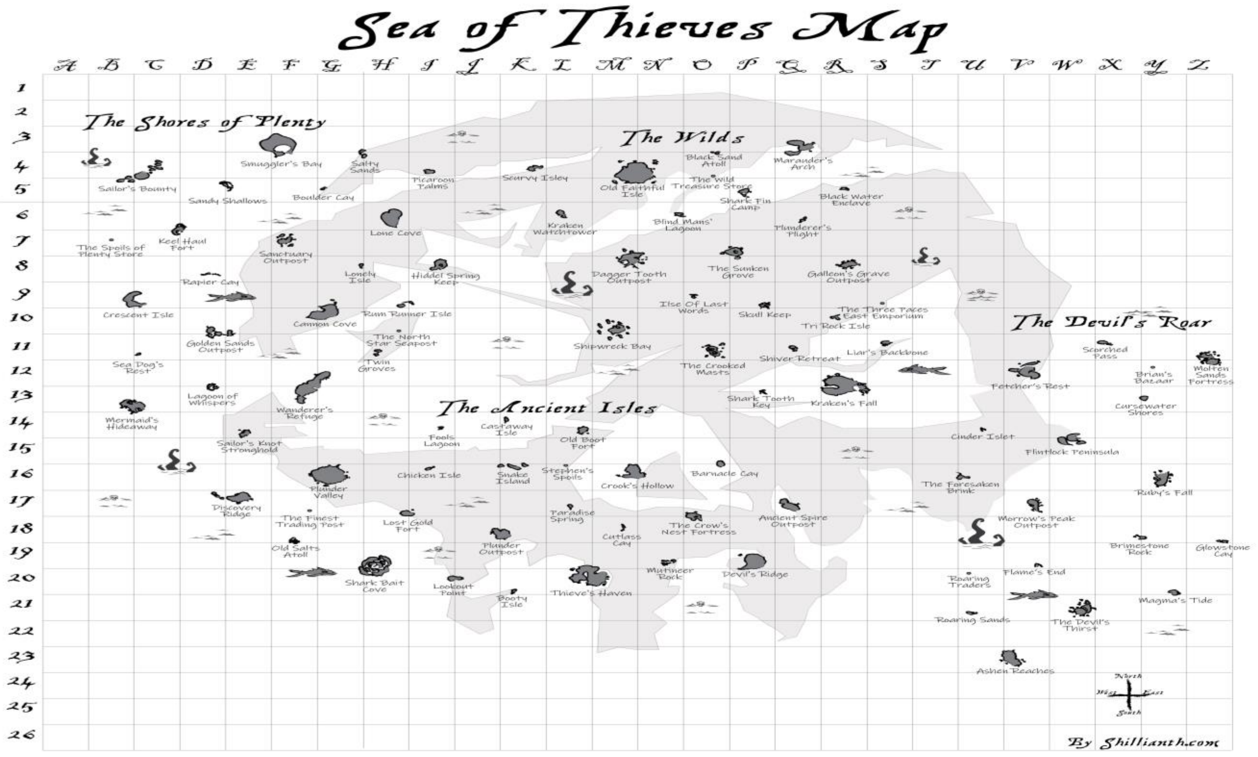


Figure III-1. Sea of Thieves Map
https://www.seaofthieves.com/community/forums/topic/133204/full-map

The Sea of Thieves map consists of a grid that divides the ocean and land areas with columns labeled A–Z (from west to east) and rows 1–26 (from north to south). Each grid on this map is 1 × 1 nautical mile in size.

B. Weighted Graph Formation

1) Island Nodes and Coordinates

There are 74 islands on the Sea of Thieves map. Therefore, for ease of calculation, only 10 islands on the map will be selected to be used as nodes in the graph to be created. The following is a list of the selected islands and their coordinates:

Table 1. Table Island Nodes and Coordinates

Index	Island Name	Grid	X (A=1)	Y
0	Plunder Outpost	J-18	10	18
1	Shark Bait Cove	H-19	8	19
2	Plunder Valley	G-16	7	16
3	Mermaid's Hideaway	B-13	2	13
4	Wanderer's Refuge	F-12	6	12
5	Cannon Cove	G-10	7	10
6	Shipwreck Bay	M-10	13	10
7	Kraken's Fall	R-12	18	12
8	Devil's Ridge	P-19	16	19
9	Thieve's Haven	L-20	12	20

2) Edges and Weights Between Nodes

Each edge in the graph is represented as a sea route between islands and is weighted using the Euclidean distance between its coordinate points (1 grid = 1 nautical mile). Here is the distance matrix between 10 islands:

Table 2. Distance Matrix for Edge Weights

	0	1	2	3	4	5	6	7	8	9
0	0.00	2.24	3.61	9.43	7.21	8.54	8.54	10.00	6.08	2.83
1	2.24	0.00	3.16	8.49	7.28	9.06	10.30	12.21	8.00	4.12
2	3.61	3.16	0.00	5.83	4.12	6.00	8.49	11.70	9.49	6.40
3	9.43	8.49	5.83	0.00	4.12	5.83	11.40	16.03	15.23	12.21
4	7.21	7.28	4.12	4.12	0.00	2.24	7.28	12.00	12.21	10.00
5	8.54	9.06	6.00	5.83	2.24	0.00	6.00	11.18	12.73	11.18
6	8.54	10.30	8.49	11.40	7.28	6.00	0.00	5.39	9.49	10.05
7	10.00	12.21	11.70	16.03	12.00	11.18	5.39	0.00	7.28	10.00
8	6.08	8.00	9.49	15.23	12.21	12.73	9.49	7.28	0.00	4.12
9	2.83	4.12	6.40	12.21	10.00	11.18	10.05	10.00	4.12	0.00

3) Graph Visualization

The graph visualization is based on the grid coordinates used in the game Sea of Thieves.

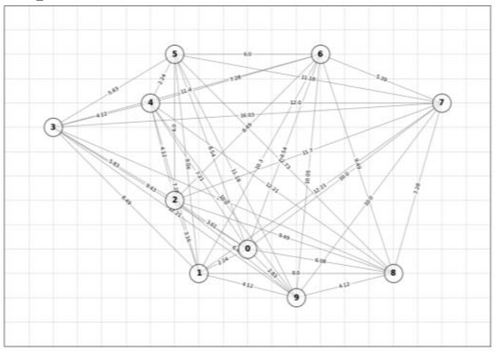


Figure III-2. Graph Visualization

C. TSP Algorithm Implementation

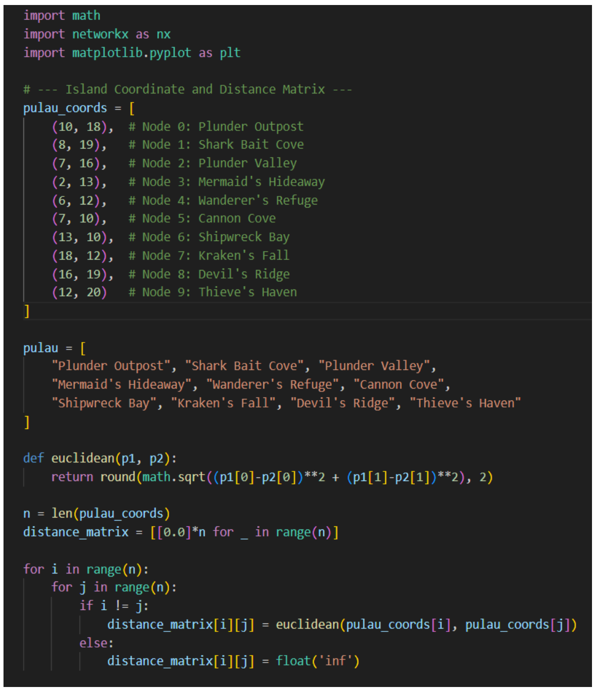
In this study, the implementation of the Traveling Salesman Problem (TSP) based on Dynamic Programming (Held-Karp) algorithm will use the Python programming language. This is because Python has a strong mathematical and graph library, and supports efficient coordinate-based visualization processes.

Visualization in this program will be created with the help of libraries:

- NetworkX: to build graph structures, add nodes, edges, and weight attributes between nodes.
- Matplotlib: to draw graphs in two dimensions, set layouts, colors, labels, and other visual details.

1) Data and Map Representation

The initial step is to use data (Review Table 1 and Table 2) in the form of Cartesian coordinates from 10 selected islands and the weights between nodes (islands) are determined based on the Euclidean distance between coordinates.



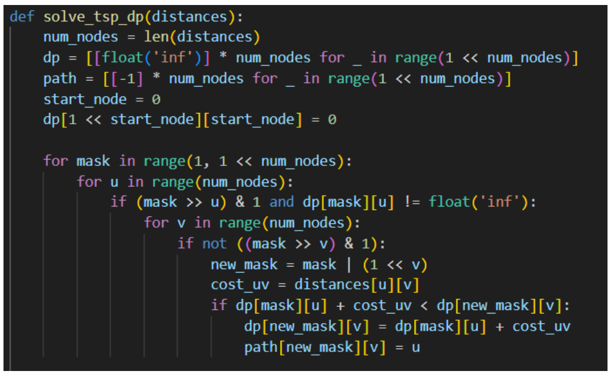
```python
import math
import networkx as nx
import matplotlib.pyplot as plt

# --- Island Coordinate and Distance Matrix ---
pulau_coords = [
    (10, 18),  # Node 0: Plunder Outpost
    (8, 19),   # Node 1: Shark Bait Cove
    (7, 16),   # Node 2: Plunder Valley
    (2, 13),   # Node 3: Mermaid's Hideaway
    (6, 12),   # Node 4: Wanderer's Refuge
    (7, 10),   # Node 5: Cannon Cove
    (13, 10),  # Node 6: Shipwreck Bay
    (18, 12),  # Node 7: Kraken's Fall
    (16, 19),  # Node 8: Devil's Ridge
    (12, 20)   # Node 9: Thieve's Haven
]

pulau = [
    "Plunder Outpost", "Shark Bait Cove", "Plunder Valley",
    "Mermaid's Hideaway", "Wanderer's Refuge", "Cannon Cove",
    "Shipwreck Bay", "Kraken's Fall", "Devil's Ridge", "Thieve's Haven"
]

def euclidean(p1, p2):
    return round(math.sqrt((p1[0]-p2[0])**2 + (p1[1]-p2[1])**2), 2)

n = len(pulau_coords)
distance_matrix = [[0.0]*n for _ in range(n)]

for i in range(n):
    for j in range(n):
        if i != j:
            distance_matrix[i][j] = euclidean(pulau_coords[i], pulau_coords[j])
        else:
            distance_matrix[i][j] = float('inf')
```

Figure III-3. Island Coordinates and Distance Matrix

2) Dynamic Programming Algorithm (Held-Karp)

The research will use the Dynamic Programming approach by implementing the Held-Karp algorithm.

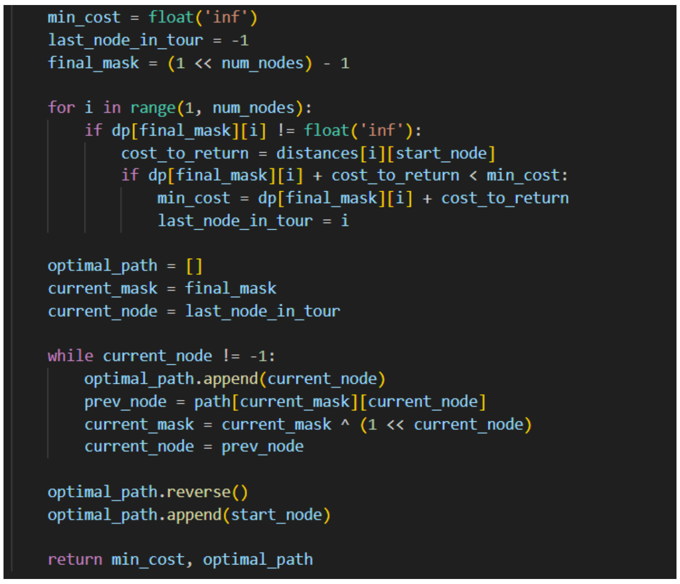
The Held-Karp algorithm uses bitmasking to represent the subset of nodes to be visited. For each combination of the subset mask and the end node u, it is stored in dp[mask][u] which represents the minimum cost to reach node u after visiting all nodes. In addition, the path[mask][u] structure is used to store the previous nodes so that the optimal route can be reconstructed at the end of the process.



```python
def solve_tsp_dp(distances):
    num_nodes = len(distances)
    dp = [[float('inf')] * num_nodes for _ in range(1 << num_nodes)]
    path = [[-1] * num_nodes for _ in range(1 << num_nodes)]
    start_node = 0
    dp[1 << start_node][start_node] = 0

    for mask in range(1, 1 << num_nodes):
        for u in range(num_nodes):
            if (mask >> u) & 1 and dp[mask][u] != float('inf'):
                for v in range(num_nodes):
                    if not ((mask >> v) & 1):
                        new_mask = mask | (1 << v)
                        cost_uv = distances[u][v]
                        if dp[mask][u] + cost_uv < dp[new_mask][v]:
                            dp[new_mask][v] = dp[mask][u] + cost_uv
                            path[new_mask][v] = u
```

Figure III-4. Initialize dp, path, and loop Subset Masks

After all subsets are computed, the algorithm finds the best end node to return to the starting point. Then, the optimal route will be updated by tracing the path backward from the end node to the starting node.



```python
    min_cost = float('inf')
    last_node_in_tour = -1
    final_mask = (1 << num_nodes) - 1

    for i in range(1, num_nodes):
        if dp[final_mask][i] != float('inf'):
            cost_to_return = distances[i][start_node]
            if dp[final_mask][i] + cost_to_return < min_cost:
                min_cost = dp[final_mask][i] + cost_to_return
                last_node_in_tour = i

    optimal_path = []
    current_mask = final_mask
    current_node = last_node_in_tour

    while current_node != -1:
        optimal_path.append(current_node)
        prev_node = path[current_mask][current_node]
        current_mask = current_mask ^ (1 << current_node)
        current_node = prev_node

    optimal_path.reverse()
    optimal_path.append(start_node)

    return min_cost, optimal_path
```

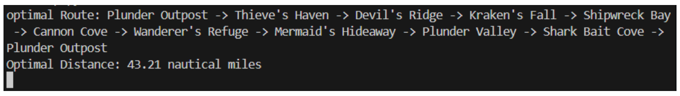
Figure III-5. Optimal Route Update

3) Result

The result of the implementation of the Traveling Salesman Problem (TSP) using the Dynamic Programming algorithm in this study is the optimal exploration route that visits all 10 islands exactly once and returns to the starting point. By using distance matrix data calculated using the Euclidean distance between island coordinates, the program successfully calculates the total minimum distance needed to complete the island exploration route.

The final output produced:

- Optimal Route: a route with a minimum total distance represented by the order of island visits
- Optimal Distance: The minimum total distance traveled by the player in nautical miles.



```
optimal Route: Plunder Outpost -> Thieve's Haven -> Devil's Ridge -> Kraken's Fall -> Shipwreck Bay
 -> Cannon Cove -> Wanderer's Refuge -> Mermaid's Hideaway -> Plunder Valley -> Shark Bait Cove ->
Plunder Outpost
Optimal Distance: 43.21 nautical miles
```

Figure III-6. Route and Minimum Distance Output

The program successfully produces a route with the minimum total distance, thus showing that the Dynamic Programming (Held-Karp) algorithm can solve the island exploration problem in Sea of Thieves. These results confirm the suitability of dynamic programming for small to medium-scale TSP scenarios with spatial constraints, such as map-based game environments like Sea of Thieves.

4) Visualization

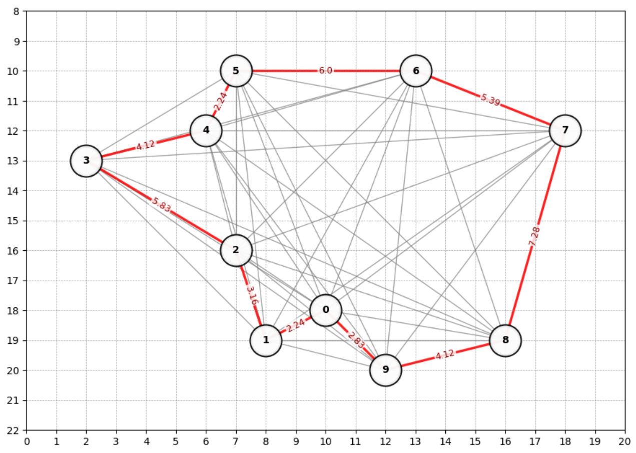


Figure III-7. Optimal Island Exploration Route Visualization

In the figure, the optimal route obtained from the application of the Dynamic Programming algorithm is highlighted using the red line, with the order of island visits: 0-9-8-7-6-5-4-3-2-1-0.

IV. CONCLUSION

This paper successfully demonstrates the application of the Traveling Salesman Problem (TSP) in optimizing island exploration routes in Sea of Thieves. By modeling the map as a weighted undirected graph and applying the Dynamic Programming (Held-Karp) algorithm, the program successfully determines the route with the minimum total weight that visits 10 islands exactly once and returns to the starting point (Outpost). The use of the Euclidean distance method as an edge weight also makes it easy to calculate the distance between islands so that the distance data used in the Distance Matrix is fairly accurate.

This paper highlights how the application of concepts from discrete mathematics, particularly graph theory, can be effectively applied in solving the island exploration problem in Sea of Thieves, demonstrating the relevance of discrete mathematics and computational concepts outside of the academic environment.

As a suggestion for future development, this approach can be extended by:

- Increasing the number of islands to increase the scale of the problem and test the performance of more sophisticated heuristics.
- Considering additional variables, such as wind direction, speed, or others to be more accurate with the game.
- Experimenting the algorithms that have been implemented into the game, and comparing them.

APPENDIX

The complete program implementation code can be seen at the following link: https://github.com/Minaqu28/TSP-SeaOfThieves

ACKNOWLEDGMENT

I would like to express my deepest gratitude to Dr. Ir. Rinaldi Munir, M.T., as a lecturer of IF1220 Discrete Mathematics, for his invaluable guidance during the semester, inspiring lectures, and useful sources of information during the lectures. His dedication and expertise have played an important role in deepening my understanding of graph theory and other discrete mathematics concepts, as well as the application of discrete mathematics itself in real life, which helped me in completing this paper.

REFERENCES

[1] R. Munir, Graf Bagian 1, STEI ITB, 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf. (Accessed on: June 17, 2025).

[2] R. Munir, *Graf Bagian 2*, STEI ITB, 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/21-Graf-Bagian2-2024.pdf. (Accessed on: June 17, 2025).

[3] R. Munir, *Graf Bagian 3*, STEI ITB, 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/22-Graf-Bagian3-2024.pdf. (Accessed on: June 17, 2025).

[4] "Islands," Sea of Thieves Wiki, 2022. https://seaofthieves.wiki.gg/wiki/Islands (Accessed June 20, 2025).

[5] Gutin, G. & Punnen, A. P. (Eds.). The Traveling Salesman Problem and Its Variations. Springer US, 2002.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, 3rd ed., MIT Press, 2009.

[7] M. Held and R. M. Karp, "A dynamic programming approach to sequencing problems," Journal of the Society for Industrial and Applied Mathematics, vol. 10, no. 1, pp. 196–210, 1962.

[8] GeeksforGeeks, "Euclidean Distance," *GeeksforGeeks*, Mar. 13, 2024. https://www.geeksforgeeks.org/maths/euclidean-distance/. (Accessed June 17, 2025).

STATEMENT

I hereby declare that the paper I wrote is my own writing, not an adaptation or translation of someone else's paper, and is not plagiarized.

Bandung, 17 Juni 2025

[illegible]

Eduard Daniel Ariajaya
13524129