

Optimizing the Universal Studios Singapore Experience: Solving the Traveling Salesman Problem using Dynamic Programming

Jeffrey Chow - 13521046
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13521046@std.stei.itb.ac.id

Abstract—Universal Studios Singapore is a popular theme park that attracts millions of visitors every year. With numerous of attractions and limited time, visitors often struggle to plan their visit and maximize their enjoyment and experience. In this paper, the author propose a novel approach to optimizing the Universal Studios Singapore experience by formulating it as a traveling salesman problem and solving it using dynamic programming. The author shows that the approach can significantly improve the visitor experience by providing an optimal itinerary that maximizes the number of attractions visited while minimizing the time spent in mobility.

Keywords—universal studio singapore, dynamic programming, travelling salesman problem

I. INTRODUCTION

Singapore is an island country and city-state located at the southern tip of the Malay Peninsula in Southeast Asia. Known for its multicultural population, vibrant economy, and stunning skyline, Singapore is a popular destination for travelers from around the world. One of the most popular attractions in Singapore is Universal Studios Singapore, a theme park located within Resorts World Sentosa on Sentosa Island. Featuring 27 rides, shows, and attractions in six themed zones, the park offers visitors a wide variety of exciting experiences based on blockbuster movies and television series.



Source: <https://anekatempatwisata.com/wp-content/uploads/2021/03/Universal-Studio-Singapore-.jpg>

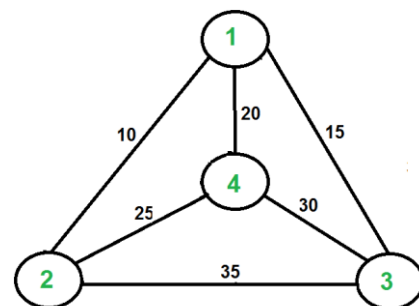
As Southeast Asia's first and only Universal Studios theme park, Universal Studios Singapore attracts millions of visitors every year with its adrenaline-pumping rides and interactive shows. With attraction for everyone at this world-class theme park, visitors can experience everything from the high-speed roller coasters of Sci-Fi City to the enchanting fairy tale kingdom of Far Far Away.

In this paper, the author focuses on analyzing Character Meet & Greet events at Universal Studios Singapore. The decision to limit the scope of the study to this specific venue was made due to the rising complexity of including all venues in the analysis. This complexity arises from the use of the traveling salesman problem (TSP), which can be time-consuming to solve. Further details on this decision and its implications will be provided in the implementation and discussion section.

II. THEORETICAL FOUNDATION

A. Travelling Salesman Problem (TSP)

The traveling salesman problem (TSP) is a well-known problem in the fields of operations research and theoretical computer science. The objective of the problem is to find the shortest possible path to complete a round-trip route starting from the origin city, visiting all other cities only once, and returning to the origin city. In solving the TSP, a set of cities and the distances between every pair of cities are given.



Source: <https://media.geeksforgeeks.org/wp-content/uploads/Euler12.png>

The TSP can be viewed as a problem of finding a Hamiltonian cycle with minimum weight. For instance, consider a graph with four cities and the distances between them as shown in the figure above. A TSP tour in this graph would be 1-2-4-3-1, with a total cost of $10+25+30+15 = 80$. The TSP is a famous NP-hard problem because no polynomial-time solution is known for this problem.

The Travelling Salesman Problem could be solved by using several algorithms. Two of them are dynamic programming and brute force which will be explored further in this paper.

B. Dynamic Programming

Dynamic programming is a computer programming technique where an algorithmic problem is first broken down into sub-problems, the results are saved, and then the sub-problems are optimized to find the overall solution which usually has to do with finding the maximum and minimum range of the algorithmic query.

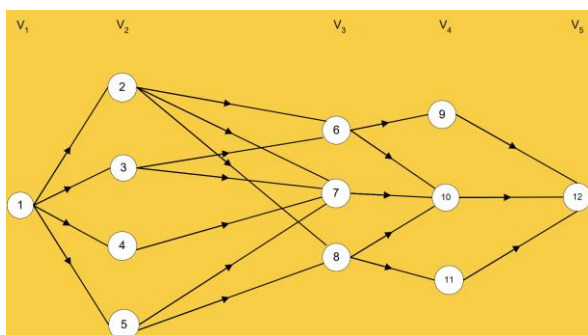
When a more extensive set of equations is broken down into smaller subproblems, and if there are overlapping among these subproblems, then the solutions to these subproblems can be saved for future reference. In this way, the efficiency of the CPU can be enhanced.

For example, when using the dynamic programming technique to figure out all possible results from a set of numbers, the first time the results are calculated, they are saved and put into the equation later instead of being calculated again. So, when dealing with long, complicated equations and processes, it saves time and makes solutions faster by doing less work.

There are several characteristics of dynamic programming:

1. A problem can be divided into several stages and for each stage, there will be only one decision.
2. Each stage consists of several different states related to this stage. In general, statuses are the various possible inputs that exist in a stage.

In a multistage graph, each vertex in the graph states the status, while V_1, V_2, \dots, V_n state the stage.



Source: [7]

3. The results of the decisions taken at each stage are transformed from the status concerned to the next status in the next stage.

4. The cost at a certain stage increases steadily with the increasing number of stages.
5. The cost of one stage depends on the cost from that stage to the next stage.
6. There is a recursive relationship that identifies the best decision for each state at stage k and gives the best decision for each status at stage $k+1$.
7. The principle of optimality applies to this problem.

There are two approaches to dynamic programming:

1. Forward or top-down dynamic programming

In the top-down approach, the solution is implemented naturally using recursion but modify it to save the solution for each subproblem in an array or hash table. Calculations are made from stages 1, 2, ..., $n-1$, n . A series of decision variables are x_1, x_2, \dots, x_n .

2. Backward or bottom-up dynamic programming

The bottom-up approach is just the reverse but an iterative version of the top-down approach. It depends on a natural idea: the solution of any subproblem depends only on the solution of smaller subproblems. The calculation for the bottom-up approach is made from stage $n, n-1, \dots, 2, 1$. A series of decision variables is x_n, x_{n-1}, \dots, x_1 .

In developing the algorithm of dynamic programming, there are several steps including characterizing the optimal solution structure, recursively defining the optimal solution value, calculating the value of the optimal solution whether using the top-down or bottom-up approach, and reconstructing the optimal solution.

The instance of problems that could be solved by using dynamic programming is Fibonacci numbers, subset sum problem, minimum cost path, traveling salesman problem, 0/1 knapsack problem, palindrome partitioning, word wrap problem, etc.

C. Brute Force

Brute Force algorithm is a straightforward approach to solve a problem that rely on sheer computing power and trying every possibility rather than advanced techniques to improve efficiency. This algorithm is usually based on problem statement and defined concept. Brute force algorithm solves the problem with a direct, simple, and obvious way.

Advantages of using brute force algorithm are its wide applicability, simple and easy to understand, produces a feasible algorithm for some problems, generates standard algorithm for tasks computing. Disadvantages of using brute force algorithm are rarely producing good and effective algorithms, generally slow, and not as constructive and creative as other problem solving strategy.

Instance of problem that could be solved by using brute force algorithm are searching for minimum or maximum value, sequential search, multiplying two matrixes, bubble sort, selection sort, etc.

III. IMPLEMENTATION AND DISCUSSION

As stated in the introduction section, the author's analysis will focus solely on Character Meet & Greet Events at Universal Studios Singapore. This decision is based on the time complexity of solving the traveling salesman problem using Dynamic Programming. Using dynamic programming, the time complexity would be $O(2^n n^2)$. This means that if all 27 active locations in Universal Studios Singapore were analyzed, it would take approximately 97,844,723,712 computations. Assuming one computation takes one millisecond, it would take approximately 3 years to complete. Therefore, the author has decided to analyze only nine Character Meet & Greet Events. Further details about the Dynamic Programming algorithm used will be provided in Section B.



Source: [12]

The figure above displays a map of the Universal Studios Singapore theme park, which is divided into six areas: Far Far Away, which has 6 active attractions; The Lost World, which has 5 active attractions; Ancient Egypt, which has 3 active attractions; Sci-Fi City, which has 5 active attractions; New York, which has 4 active attractions; and Hollywood, which has 3 active attractions.



Source: [13]

For the purposes of this paper, the author will analyze only the Character Meet & Greet Events as shown in the figure above and indicated on the map with location codes beginning with 'M'. There are 9 such events spread across the areas of Universal Studios Singapore:

1. M1 – Gru and the Minions in Hollywood
2. M2 – Po and Master Tigress in Hollywood

3. M3 – Madagascar in Hollywood
4. M4 – Shrek and Fiona in Far Far Away
5. M5 – Raptor Encounter in The Lost World
6. M6 – Hatched! Featuring Dr. Rodney in The Lost World
7. M7 – Egyptian Legends in Ancient Egypt
8. M8 – Voices of Cybertron in Sci-Fi City
9. M9 – Sesame Street in New York

The starting and ending location of the routes analyzed will be at the famous Universal Studios Singapore Globe, located at the Main Entrance.

In analyzing and finding solution for the problem, several steps will be taken such as obtaining location data, obtaining the weighted adjacency matrix, finding solutions using dynamic programming, finding solutions using brute force, and conducting an analysis.

A. Obtaining Locations Data

In order to analyze the shortest path between locations, it is essential to obtain accurate coordinates for each location. For this analysis, the author utilized the Python programming language and the OpenCV library to extract location data from an image of the map. The full version of the code used for this process can be found in the [repository](#).

Before extracting the coordinates, the author first resized the image to a dimension of 1600 x 900 pixels. After running the program, the author manually clicked on each location on the map and recorded the corresponding coordinates displayed in the terminal. The coordinates obtained are relative to the 1600 x 900 pixels dimension, not the original image dimension.

The resulting data was then manually converted into a structured JSON format for ease of analysis. This format includes information such as location code, name, area, and coordinates obtained through the process mentioned before. The coordinates are presented as an array, with the x-coordinate at the 0th index and the y-coordinate at the 1st index. Below is a snippet of the JSON data used for the analysis.

```

{
  {
    "code": "",
    "name": "Universal Studios Singapore Globe",
    "area": "Main Entrance",
    "coordinate": [728, 827]
  },
  {
    "code": "M1",
    "name": "Gru and the Minions",
    "area": "Hollywood",
    "coordinate": [736, 669]
  },
  {
    "code": "M2",
    "name": "Po and Master Tigress",
    "area": "Hollywood",
    "coordinate": [698, 672]
  },
  {
    "code": "M3",
    "name": "Madagascar",
    "area": "Hollywood",
    "coordinate": [744, 697]
  },
  {
    "code": "M4",
    "name": "Shrek and Fiona",
    "area": "Far Far Away",
  }
}

```

```

"coordinate": [559, 436]
},
{
"code": "M5",
"name": "Raptor Encounter",
"area": "The Lost World",
"coordinate": [697, 397]
},
{
"code": "M6",
"name": "Hatched! Featuring Dr. Rodney",
"area": "The Lost World",
"coordinate": [721, 386]
},
{
"code": "M7",
"name": "Egyptian Legends",
"area": "Ancient Egypt",
"coordinate": [953, 394]
},
{
"code": "M8",
"name": "Voices of Cybertron",
"area": "Sci-Fi City",
"coordinate": [969, 543]
},
{
"code": "M9",
"name": "Sesame Street",
"area": "New York",
"coordinate": [853, 618]
}
]

```

Source: [Author's Repository](#)

B. Obtaining the Weighted Adjacency Matrix

In section A, the author obtained JSON data containing information about locations in Universal Studios Singapore. The author then preprocessed this data to create a weighted adjacency matrix representing the cost of traveling between each pair of locations. The cost between locations a and b, denoted as c(a,b) is calculated using the Euclidean distance and is stored in the [a,b] position of the matrix.

$$c(a, b) = \sqrt{(b_x - a_x)^2 + (b_y - a_y)^2}$$

Source: Author's Private Document

Below is the snippet of the resulting weighted adjacency matrix.

```

[[ 0.          158.202402  157.87653404  130.98091464  425.96009203
  431.11599367  441.05555206  487.96926133  372.47416018  243.52823245]
 [158.202402    0.          38.11823711  29.12043956  292.60553652
  274.78173156  283.39724769  350.30558089  264.88676826  127.63228432]
 [157.87653404  38.11823711    0.          52.35456045  273.89231461
  275.00181818  286.92333471  377.23865125  300.13663555  164.13713778]
 [130.98091464  29.12043956  52.35456045    0.          319.91561387
  303.65934861  311.84932259  368.08966299  272.65546024  134.617978 ]
 [425.96009203  292.60553652  273.89231461  319.91561387    0.
  143.40502083  169.54055562  396.23225512  423.73222677  345.77449299]
 [431.11599367  274.78173156  275.00181818  303.65934861  143.40502083
    0.          26.40075756  256.01757752  308.70698081  270.51247661]
 [441.05555206  283.39724769  286.92333471  311.84932259  169.54055562
  26.40075756    0.          232.13789006  293.51831289  266.92320993]
 [487.96926133  350.30558089  377.23865125  368.08966299  396.23225512
  256.01757752  232.13789006    0.          149.85659812  245.3079697 ]
 [372.47416018  264.88676826  300.13663555  272.65546024  423.73222677
  308.70698081  293.51831289  149.85659812    0.          138.13399292]
 [243.52823245  127.63228432  164.13713778  134.617978  345.77449299
  270.51247661  266.92320993  245.3079697  138.13399292    0.          ]]

```

Source: Author's Private Document

C. Dynamic Programming for Solution Finding

Dynamic programming is a powerful approach for solving complex problems by breaking them down into smaller, more manageable subproblems. One such problem that can be effectively solved using dynamic programming is the traveling salesman problem (TSP). In this section, the author will

explore how dynamic programming can be used to find optimal solutions to the TSP and discuss the key concepts and techniques involved in this approach.

The approach of dynamic programming used is top-down or forward. The recursion function defined for this analysis:

$$f(i, S) = \begin{cases} c(i, j) + c(j, 0), & \text{length}(S)=1 \\ \min\{c(i, k) + f(k, S - \{k\})\}, & \text{length}(S)>1 \end{cases}$$

Source: Author's Private Document

In this approach, a recursive function f(i,S) is defined, where i represents the source location index and S represents a set of destination indices to be visited. Indexing is based on the JSON data.

The base case of the recursion occurs when there is only one item j left in the set S, in which case the function returns the cost of traveling from i to j and then back to the starting position. In the recursive case, the function iterates over each item k in the set S, calculating the cost of traveling from i to k and then recursively calling itself with k as the new source and S - {k} as the new set of destinations. The minimum result from these recursive calls is returned as the final solution.

In the implementation of the analysis, the recursive function has been modified to return a tuple containing a list of the visited route and the total cost of that route. This allows the function to keep track of both the optimal route and its associated cost while recursively solving the problem. The cost between two locations a and b is represented by a weighted adjacency matrix (adj_mtx) which was discussed in section B. Below is a code snippet for solving the traveling salesman problem using dynamic programming.

```

def tsp_with_dp(i, S):
    ''' Base : if S has only 1 item left, return distance from it to the ending point '''
    if (len(S) == 1):
        j = S.pop()
        return ([i, j, 0], adj_mtx[i][j] + adj_mtx[j][0])

    # Set default value for the result
    result = ([], 10**9)

    # Iterate through all item in S
    for j in S:
        ''' Recursion '''
        recursion = tsp_with_dp(j, S-{j})

        # Insert source to the result array
        recursion[0].insert(0,i)

        # Create new tuple for the result and add distance from source(i) to destination (j)
        current = recursion[0], recursion[1] + adj_mtx[i,j]

        # Take result only if minimum, compared by the distance
        result = min(result, current, key=lambda x: x[1])

    return result

```

Source : [Author's Repository](#)

The time complexity of solving the travelling salesman problem using dynamic programming is $O(2^n \cdot n^2)$ where $O(n \cdot 2^n)$ is the maximum number of unique subproblems/states and $O(n)$ for transition in every state.

The result of solving the traveling salesman problem using the tsp_with_dp function defined above is displayed below. The function was called with i=0 (the starting location index) and S as a set of numbers from 1 to 9 (the destination location indices).

```
Using Dynamic Programming algorithm
Execution time : 0.5717 seconds
```

```
Route to travel :
Universal Studios Singapore Globe in Main Entrance
M3 - Madagascar in Hollywood
M1 - Gru and the Minions in Hollywood
M2 - Po and Master Tigress in Hollywood
M4 - Shrek and Fiona in Far Far Away
M5 - Raptor Encounter in The Lost World
M6 - Hatched! Featuring Dr. Rodney in The Lost World
M7 - Egyptian Legends in Ancient Egypt
M8 - Voices of Cybertron in Sci-Fi City
M9 - Sesame Street in New York
Universal Studios Singapore Globe in Main Entrance
```

```
Approximate distance to travel : 1405.57 meters
Assume : 1 pixel in map equals to 1 meter
```

Source: Author's Private Document

The result shows that by using dynamic programming, the problem was solved in 0.5717 seconds. The obtained route is Start – M3 – M1 – M2 – M4 – M5 – M6 – M7 – M8 – M9 – End, with a total travel distance of approximately 1405.57 meters, assuming 1 pixel is equal to 1 meter. Further analysis about the result will be presented in section E.

D. Brute Force for Solution Finding

The brute force algorithm is a straightforward approach to solving problems, including the traveling salesman problem (TSP). In this analysis, the brute force algorithm is used as a baseline for comparison with the dynamic programming approach. By comparing the performance of these two algorithms, the author can gain insight into the effectiveness of dynamic programming in solving the TSP.

To solve the TSP using the brute force algorithm, the following steps are taken:

1. Consider one city as the starting and ending location
2. Generate all $(n-1)!$ permutations of cities.
3. Calculate the cost of every permutation and keep track of the minimum cost permutation.
4. Return the permutation with minimum cost.

The time complexity for solving TSP using brute force is $O(n!)$. In the implementation, the algorithm is defined in a function called `tsp_with_bf(weighted_adjacency_matrix, start_index)`, which takes as input a weighted adjacency matrix (defined in section B) and a start index (0 in this case, representing the starting location index).

```
Using Brute Force algorithm
Execution time : 0.6730 seconds
```

```
Route to travel :
Universal Studios Singapore Globe in Main Entrance
M9 - Sesame Street in New York
M8 - Voices of Cybertron in Sci-Fi City
M7 - Egyptian Legends in Ancient Egypt
M6 - Hatched! Featuring Dr. Rodney in The Lost World
M5 - Raptor Encounter in The Lost World
M4 - Shrek and Fiona in Far Far Away
M2 - Po and Master Tigress in Hollywood
M1 - Gru and the Minions in Hollywood
M3 - Madagascar in Hollywood
Universal Studios Singapore Globe in Main Entrance
```

```
Approximate distance to travel : 1405.57 meters
Assume : 1 pixel in map equals to 1 meter
```

Source: Author's Private Document

Using the brute force algorithm, the problem was solved in 0.673 seconds. The obtained route is Start – M9 – M8 – M7 – M6 – M5 – M4 – M2 – M1 – M3 – End, with a total travel distance of approximately 1405.57 meters, assuming 1 pixel is equal to 1 meter.

E. Analysis

From the results obtained in sections C and D, it can be seen that both the dynamic programming and brute force approaches yield the same route and cost. The brute force solution is simply the reverse of the route obtained using dynamic programming, indicating that both solutions are equivalent. Therefore, the author has successfully determined the optimal route for visiting all Character Meet & Greet locations in Universal Studios Singapore. This route is Universal Studios Singapore Globe in Main Entrance - Madagascar in Hollywood (M3) - Gru and the Minions in Hollywood (M1) - Po and Master Tigress in Hollywood (M2) - Shrek and Fiona in Far Far Away (M4) - Raptor Encounter in The Lost World (M5) - Hatched! Featuring Dr. Rodney in The Lost World (M6) - Egyptian Legends in Ancient Egypt (M7) - Voices of Cybertron in Sci-Fi City (M8) - Sesame Street in New York (M9) - Universal Studios Singapore Globe in Main Entrance. The total travel distance is approximately 1405.57 meters. The visualization of route is displayed below.



Source: Author's Private Document

Assuming each Meet & Greet takes 30 minutes and the walking speed is 0.5 m/s, it would take less than 6 hours to visit all Character Meet & Greet events. However, this calculation is not entirely reliable due to the assumption that 1 pixel is equal to 1 meter and the assumption that Character Meet & Greet events will be available at any time. Despite the assumptions made in the analysis, the main objective of finding an optimal solution to the TSP using dynamic programming has been accomplished.

The main difference between dynamic programming and brute force algorithms is the time taken for execution. Dynamic programming solves the problem in 0.5717 seconds, which is faster than the brute force algorithm which takes 0.673 seconds. This difference is due to the time complexity of both algorithms: $O(2^n n^2)$ for dynamic programming, which is more efficient than $O(n!)$ for brute force. In this analysis, n is 9, which is relatively small and the difference is not significant. However, if n were larger, there would be a significant difference in performance between the dynamic programming and brute force approaches.

IV. CONCLUSION

In conclusion, the Traveling Salesman Problem (TSP) can be solved using either dynamic programming or brute force algorithms, as explained in sections 3C and 3D. A comparison of these two approaches has shown that dynamic programming is a more robust and efficient method for solving the TSP in the context of visiting all Character Meet & Greet locations in Universal Studios Singapore, as analyzed in section 3E. By utilizing simple Python programming and libraries, real-world problems can be solved and valuable insights can be provided to the public about enhancing their experience while exploring Universal Studios Singapore.

Further analysis and approaches should be taken to address the complexity of the Traveling Salesman Problem with a large number of nodes. Scaling the map to its actual size could provide a more insightful and reliable analysis.

VIDEO LINK AT YOUTUBE

https://youtu.be/fPf_LtAM7BA

Explained in Bahasa Indonesia

ACKNOWLEDGMENT

The author would like to express his deepest gratitude to Dr. Nur Ulfa Maulidevi, S.T., M.Sc., the author's lecturer for the Algorithm Strategies course, for her invaluable guidance and support throughout the writing of this paper. The author would also like to extend his appreciation to Dr. Rinaldi Munir, S.T., M.T., and Ir. Rila Mandala, M.Eng., Ph.D. for sharing their knowledge and expertise. In addition, the author would like to thank his family and friends for their unwavering support and encouragement during the writing process.

REFERENCES

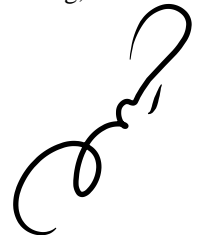
- [1] <https://www.britannica.com/place/Singapore>, accessed on 21th May 2023
- [2] [Universal Studios Singapore - Sentosa Island Ticket Price & Hours \(singaporetravelhub.com\)](https://singaporetravelhub.com), accessed on 21th May 2023
- [3] <https://www.rwsentosa.com/en/attractions/universal-studios-singapore>, accessed on 21th May 2023
- [4] <https://www.geeksforgeeks.org/travelling-salesman-problem-using-dynamic-programming/>, accessed on 21th May 2023
- [5] <https://www.guru99.com/travelling-salesman-problem.html>, accessed on 21th May 2023
- [6] https://www.spiceworks.com/tech/devops/articles/what-is-dynamic-programming/#:~:text=Dynamic%20programming%20is%20a%20compiler_range%20of%20the%20algorithmic%20query, accessed on 21th May 2023

- [7] <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian1.pdf>, accessed on 21th May 2023
- [8] <https://www.programiz.com/dsa/dynamic-programming>, accessed on 21th May 2023
- [9] <https://www.enjoyalgorithms.com/blog/top-down-memoization-vs-bottom-up-tabulation>, accessed on 21th May 2023
- [10] <https://www.freecodecamp.org/news/brute-force-algorithms-explained/#:~:text=Brute%20Force%20Algorithms%20are%20exactly,a%20advanced%20techniques%20to%20improve%20efficiency>, accessed on 21th May 2023
- [11] [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-\(2022\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf), accessed on 21th May 2023
- [12] https://www.rwsentosa.com/~jssmedia/project/non-gaming/rwsentosa/attractions/universal-studios-singapore/park-map/uss_park-map--3840x2160.jpg?sc_lang=en&rev=39eacc4bbfe44e0bbaafa85121063aec&extension=webp&hash=E5E33F3DA243516DDCF3BD252B4BBA21, accessed on 20th May 2023
- [13] <https://www.rwsentosa.com/~jssmedia/project/non-gaming/rwsentosa/attractions/universal-studios-singapore/meet-and-greet/uss-far-far-away-shrek-750x422.jpg?h=422&iar=0&w=750&rev=e8886bd4100246be9dc51fafd16aaab5&hash=DA0298E5C846C0D7E9FA1FB4EC908E6A>, accessed on 22th May 2023
- [14] JeffreyChow19, *Jeffreychow19/TSP-with-DP: Solving travelling salesman problem using dynamic programming, GitHub*. Available at: <https://github.com/JeffreyChow19/tsp-with-dp>, accessed 22th May 2023

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 22 Mei 2023



13521046 Jeffrey Chow