



American University Of Sharjah

Department of Computer Science & Engineering

CMP 333

Project 1 : Optimal Route Finding using Search Algorithms

**Prof. Omar Arif**

Fall 2025

Group member name	ID
Rana Riyaz	g00100761
Fathimamaaziya Mansoor	g00099913
Hamad Alteneiji	b00098197

## **Table of contents**

SN	CONTENT	PAGE NO
1	Abstract	3
2	Introduction	4
3	Methodology	6
4	Results	12
5	Conclusions	15
6	References	16
7	Contribution of each member	17

# **Abstract**

This project explores various graph-search algorithms to determine the most efficient path between two distant Japanese cities, Makurazaki and Nemuro using geographic data from the SimpleMaps japan dataset. The study implements and compares three algorithms: Breadth First Search as an uninformed approach, Greedy Best-First Search, and A\* Search as informed approaches. A graph was constructed with cities connected according to Haversine distances, and each algorithm was evaluated based on the number of nodes explored, path optimality, and computational efficiency. BFS explored a large number of nodes and produced a valid but relatively long route. Greedy Best-First Search reduced node exploration by using heuristic guidance but produced a suboptimal path, as it only considers the estimated distance to the goal without accounting for the total path cost. A\* combines path cost with heuristic estimates, ensuring both efficiency and optimality when the heuristic is admissible and consistent. The results show differences in efficiency, completeness, and solution quality, and highlight the importance of heuristic design in real-world navigation problems.

# Introduction

In artificial intelligence, search algorithms are fundamental tools for solving complex route-finding and decision-making problems. A search strategy determines the order in which nodes are expanded, directly influencing how the algorithm explores possible paths toward a goal. These strategies can be broadly categorized into uninformed and informed search methods.

## Uninformed search

Uninformed search algorithms, also known as blind search methods, do not use any domain-specific knowledge or hints about the goal's location. It explores all possibilities systematically and guarantees finding the shortest path in terms of the number of edges. However, because of lack of guidance, such algorithms often become computationally expensive and inefficient when applied to large or weighted graphs. The uninformed search algorithm used in this project is:

- Breadth-First Search (BFS) is the uninformed search algorithm used in this project. BFS explores all nodes at the current depth before moving to the next level. It is complete (guarantees finding a solution if one exists) and optimal (finds the shortest path in terms of edges), but it can be memory-intensive for large graphs.

## Informed search

In contrast, Informed search algorithms use heuristic functions to estimate how close a given node is to the goal, allowing the search to be more directed and efficient. A heuristic is a function that estimates the cost of reaching the goal from a given node. The quality of the heuristic significantly affects an algorithm's performance: a good heuristic can drastically reduce the number of nodes explored and the overall runtime.

Two important properties of heuristics are:

- Admissibility: A heuristic is admissible if it never overestimates the true cost to reach the goal. This ensures that algorithms like A\* can find the optimal path.
- Consistent: A heuristic is consistent if it satisfies the triangle inequality between connected nodes. This means the estimated cost from a node to the goal is never greater than the cost of moving to a neighboring node plus the neighbor's estimated cost to the goal.

Two key informed algorithms explored in this project are:

- Greedy Best-First Search relies solely on the heuristic value to choose the path that appears closest to the goal.
- A\* Search combines both the accumulated path cost ( $g(n)$ ) and heuristic estimate ( $h(n)$ ) to balance accuracy and performance.

This project applies these concepts to a real-world navigation problem in Japan, computing an optimal route from Makurazaki to Nemuro, two geographically distant cities. The graph is constructed from the SimpleMaps World Cities dataset, where cities are nodes and edges connect each city to its  $k = 8$  nearest neighbors. Edge weights are computed using the Haversine distance formula, which represents the geographic distance between two cities. BFS explores this graph in terms of edge count, while Greedy Best-First and A\* additionally use the Haversine distance as a heuristic to guide the search. By comparing the algorithms in terms of path cost, runtime, and node expansions, the project highlights how heuristic design influences both efficiency and solution quality in real-world route-finding.

# Methodology

## Breadth-First Search (BFS)

A graph is explored in layers by the ignorant algorithm known as Breadth-First Search (BFS), which ensures the shortest path in terms of hop count. Before proceeding to nodes at the following level, it starts at the start node and methodically visits each neighboring node at the current depth. A First-In-First-Out (FIFO) queue is used to manage this level-by-level expansion, guaranteeing that every node at depth  $n$  is visited before any at depth  $n+1$ . Although this ensures completeness and reduces the number of transitions, BFS is not appropriate for determining the shortest path in weighted graphs where distance is more important than hop count because it disregards edge weights.

---

⤓ Graph built with 1257 cities and 10056 edges

```
Breadth first search method:  
Path length : 52  
Cost (km): 2494.61  
Nodes expanded: 1246  
Execution time (s): 0.0043
```

This output shows the results of a **Breadth-First Search (BFS)** algorithm applied to a large road network graph containing 1,257 cities and 10,056 edges. The search found a path from Makurazaki to Nemuro spanning 52 hops with a total distance of 2,494.61 km. Notably, BFS expanded 1,246 nodes - nearly the entire graph - demonstrating its characteristic of exploring all possible paths level by level without heuristic guidance. Despite the extensive exploration, the algorithm completed very quickly in just 0.0043 seconds. However, the path appears suboptimal compared to A\* results (which found 2,225 km paths), showing that BFS finds the shortest path in terms of number of hops but not necessarily the shortest geographical distance.

## Greedy Best-First Search (GBFS)

Greedy Best-First Search (GBFS) always pursues the node that appears to be closest to the destination, relying solely on a heuristic function to direct its exploration. This guiding heuristic in this implementation is the straight-line geographic distance between cities, which is determined using the Haversine formula. By giving the algorithm a precise "as-the-crow-flies" estimate, this formula enables the algorithm to give priority to cities that are closer to the objective.

Even though it doesn't promise the shortest overall path, GBFS outperforms ignorant searches like BFS in terms of efficiency. Its application of a geographical heuristic significantly lowers the number of nodes investigated, demonstrating how performance can be significantly enhanced by even basic domain knowledge. As a result, GBFS is a useful benchmark against more complex algorithms such as A\* and an efficient, albeit not optimal, technique for quick pathfinding.

```
print("Greedy Best-First method:")

hops = len(path) - 1
total_dist = path_cost_km(path, coords)
print("Path length:", hops)
print("Cost (km):", round(total_dist, 2))
print("Nodes expanded:", len(explored))
print("Execution time (s):", round(runtime, 4))
```

---

→ Greedy Best-First method:  
Path length: 55  
Cost (km): 2575.32  
Nodes expanded: 62  
Execution time (s): 0.0039

This code demonstrates the application of the Greedy Best-First Search (GBFS) algorithm to a routing problem between Makurazaki and Nemuro, using a k-nearest neighbor graph ( $k = 8$ ). The algorithm identified a path consisting of 62 nodes, covering a total distance of approximately 2,575 km. GBFS achieved this with only 62 node expansions, demonstrating high computational efficiency. However, because the algorithm relies solely on the Haversine-based heuristic and ignores accumulated path cost, it sometimes selects routes that appear promising locally but are suboptimal overall. As a result, the final path was longer than the optimal route found by A\*, highlighting GBFS's tendency to trade accuracy for speed.

## A\*

The A\* algorithm is an informed search method that combines the advantages of uniform cost search and heuristic guidance to find optimal paths efficiently. It uses the evaluation function  $f(n) = g(n) + h(n)$ , where  $g(n)$  represents the accumulated travel cost and  $h(n)$  estimates the remaining distance to the goal. In this project, the Haversine distance was used as the primary heuristic, as it provides an accurate measure of great-circle distance between two cities based on their latitude and longitude. Because it consistently underestimates the true cost, this heuristic is both admissible and consistent, allowing A\* to guarantee an optimal route.

Unlike Greedy Best-First Search, which focuses only on  $h(n)$ , A\* balances estimated proximity with actual travel effort, leading to better decision-making and fewer unnecessary expansions. The algorithm proved highly effective for the Japan cities dataset, offering a strong balance between accuracy and computational efficiency.

```

# Heuristic 1: Straight-line Haversine
heuristic_values_h1 = [haversine(city_coords[v], city_coords[goal_city]) for v in vertices]

# Heuristic 2: Landmark-based (using Tokyo as landmark)
landmark_city = "Tokyo" # <-- change this if needed
dist_from_landmark = {v: haversine(city_coords[landmark_city], city_coords[v]) for v in vertices}
dist_landmark_goal = dist_from_landmark[goal_city]
heuristic_values_h2 = [abs(dist_landmark_goal - dist_from_landmark[v]) for v in vertices]

```

---

→ A\* with Heuristic 1 (Haversine):  
 Path length: 64  
 Cost: 2225.721992481162  
 Nodes Expanded: 1193  
 Execution Time: 9.5427 seconds

---

A\* with Heuristic 2 (Landmark – Tokyo):  
 Path length: 64  
 Cost: 2225.721992481162  
 Nodes Expanded: 1255  
 Execution Time: 7.3675 seconds

---

Using two distinct heuristic functions on the same routing problem, this comparison shows A search performance\*. The algorithm's optimality guarantee was confirmed by both heuristics, which discovered the same 64-node optimal path at a cost of 2,225.72 km. With 1,193 nodes expanded as opposed to 1,255 nodes using the Landmark heuristic (using Tokyo as reference), the Haversine heuristic (direct distance to goal) proved to be marginally more effective, finishing approximately 0.07 seconds faster. This implies that although the search was successfully led to the best answer by both heuristics, the direct Haversine distance offered superior direction for this specific routing issue, leading to fewer node expansions and marginally better performance.

## Implementation decisions

Haversine Distance Formula: Because this routing problem was geographical in nature, the Haversine formula was chosen. Emvin and Amsavalli (2024) describe the Haversine formula as a trigonometric method that calculates the great-circle distance between two points on the Earth's surface using their latitude and longitude coordinates. By taking into account the Earth's spherical curvature, it provides more precise distance measurements compared to the simpler Euclidean distance, which assumes a flat plane. This precision was essential for generating

accurate distance estimates between Japanese cities and for ensuring that both the heuristic values and edge weights in the graph accurately represented real-world geographic proximity. The formula was the right choice for this dataset because of its mathematical foundation in spherical trigonometry and its demonstrated reliability in geospatial distance estimation.

K-Nearest Neighbors (K=8): The K-Nearest Neighbors (KNN) concept is a simple yet powerful approach commonly used in machine learning for classification, regression, and clustering tasks. In this project, it is applied to construct graph connectivity between cities based on geographic proximity. Each city is connected to its K closest neighboring cities according to the Haversine distance, which ensures that the network reflects realistic regional relationships.

Selecting the appropriate value of K required balancing network realism with computational efficiency. A fully connected graph of 1,257 cities would have been computationally intensive, while choosing too few neighbors (for example, K = 2–4) could lead to disconnected regions or inefficient detours. Setting K = 8 provided an effective compromise: it maintained sufficient regional connectivity to simulate realistic travel routes while keeping the graph size manageable for search algorithms. This choice also aligns with the natural structure of transportation networks, where cities tend to be most strongly connected to their nearby urban centers rather than to distant ones, resulting in a graph that is both geographically coherent and computationally practical.

### **Heuristic functions used for Greedy and A\***

Two heuristic functions were implemented and compared in this project to evaluate their influence on search performance. The first heuristic, based on the *Haversine distance*, estimates the direct great-circle distance between the current city and the goal city using their latitude and longitude coordinates. This approach provides an admissible and consistent estimate because it never overestimates the true travel cost, making it suitable for both Greedy Best-First Search and A\*. It allows the algorithms to prioritize cities that are geographically closer to the destination, thereby improving search efficiency while maintaining path accuracy.

The second heuristic, the *landmark-based heuristic*, was designed to investigate an alternative method of distance estimation that relies on a fixed point of reference rather than direct calculation. Pereira et al. (2020) established that landmark-based heuristics can significantly improve search efficiency in pathfinding problems. They defined landmarks as "necessary properties that must be true at some point along all valid plans to achieve a particular goal" (p. 2). The authors created two landmark-based heuristics that demonstrated orders of magnitude faster performance than earlier cutting-edge methods while retaining high accuracy in a range of planning domains.

To investigate a different approach to distance estimation that depends on a frame of reference rather than direct calculation, Heuristic 2 (the landmark-based heuristic) was selected. The fundamental concept involves designating a single, strategically located city, in this case Tokyo, as a “landmark.” Using the triangle inequality principle and the landmark, the heuristic calculates the distance between any city and the objective. It can be expressed mathematically as  $|h(\text{landmark}, \text{goal}) - h(\text{landmark}, \text{current})|$ .

This strategy was chosen for two main reasons. In order for A\* to ensure optimality, it must first be a classic example of a consistent and admissible heuristic. Second, landmark heuristics can be very effective in specific computational situations. Estimating the heuristic for any search would be as easy as looking up and quickly subtracting the distances from the landmark to every other city, which could be quicker than repeatedly calculating the Haversine formula. However, the direct Haversine method outperformed the landmark heuristic in this particular implementation for the Japan cities network. Because it gave a rougher, less accurate estimate of the remaining distance, A\* had to explore more nodes before figuring out the best course of action.

### **Algorithm Comparisons:**

Each algorithm applied in this project has its own advantages and drawbacks, which influence how effectively it can find and evaluate possible routes.

#### **Breadth-First Search (BFS)**

BFS ensures the shortest path in hops by methodically exploring graphs level-by-level with a FIFO queue. Its strength is its simplicity and completeness; if there is a problem, it always finds a solution. However, BFS is not appropriate for distance-optimized routing since it disregards edge weights. During our tests, BFS expanded 1,246 nodes, or almost the entire graph, while discovering a 52-hop path that covered 2,494.61 km. Despite being incredibly quick (0.007s), the distance results were not ideal. BFS is ineffective for geographical pathfinding, where distance optimization is critical, but it works best in unweighted graphs or when minimizing transitions is more important than total cost.

#### **Greedy Best-First Search (GBFS)**

GBFS is quick and memory-efficient because it prioritizes nodes that appear closest to the goal using heuristic estimates. Direct goal-oriented search, which usually expands fewer nodes than BFS, is its strength. But because GBFS disregards accumulated path cost, it frequently produces less-than-ideal results. Geographically attractive but ineffective routes may deceive it, and it may become trapped in local minima. While GBFS is fast at finding reasonable routes, it rarely finds the best ones. It works best when speed can come at the expense of solution quality or when there is a strong correlation between the heuristic and the actual path cost.

## A Search\*

A\* ensures optimal solutions when employing admissible heuristics by combining heuristic estimates ( $h(n)$ ) with actual path cost ( $g(n)$ ). Its ability to effectively balance exploration and exploitation is one of its strong points; it usually expands fewer nodes than BFS while still being optimal. A\*, however, uses more memory and processing power per node. In our implementation, A\* expanded 1,193 nodes and used the Haversine heuristic to find the best route (2,225.72 km). Although heuristic selection has a big influence on performance, A\* works best when both solution quality and efficiency are important. For the best pathfinding in weighted graphs, it is still the recommended algorithm.

# Results

The results obtained from implementing and testing the three search algorithms (Breadth-First Search, Greedy Best-First Search, and A\* Search with two heuristics) are summarized and analyzed below. Each algorithm was tested on the same dataset to find the optimal route from Makurazaki to Nemuro using a graph constructed from Japanese cities connected through Haversine distance. Results are summarized in Table 1 and illustrated in Figures 1 to 3.

Breadth-First Search (BFS) explored the highest number of nodes (1246) and produced a relatively large total distance (about 2494.61 km). Since BFS does not consider path costs or heuristic guidance, it expands many unnecessary nodes before reaching the goal. However, it provides a strong baseline for comparison because it always finds a valid solution if one exists.

Greedy Best-First Search performed efficiently in terms of nodes expanded (only 62), which shows how heuristics can greatly reduce the number of explored nodes. However, because this algorithm only considers the estimated distance to the goal and ignores the cost of the path taken, it did not produce an optimal route. Its total distance (about 2575.32 km) was higher than that of A\*.

A\* with Heuristic 1 (Haversine) achieved the most optimal route with a total cost of approximately 2225.72 km. It expanded 1193 nodes and successfully balanced path cost and heuristic information. Although it required more time than BFS and Greedy, it produced the best trade-off between optimality and efficiency.

A\* with Heuristic 2 (Landmark - Tokyo) produced the same total cost as Heuristic 1 but expanded slightly more nodes (1255 compared to 1193). This minor difference is likely due to variations in heuristic admissibility and consistency. Both heuristics were admissible, but the Haversine heuristic proved to be more consistent and effective for this dataset.

Table 1 presents the quantitative comparison of all algorithms, while Figures 1 to 3 visualize their performance across key metrics such as execution time, total distance, and number of nodes expanded.

Table 1: Comparing Results of different Algorithms

Algorithm	Path Length(hops)	Total Distance(km)	Nodes Expanded	Execution time(s)
Breadth first Search(BFS)	52	2494.61	1246	0.0043
Greedy Best-First Search	53	2575.32	1193	0.0039
A* Search (Heuristic 1: Haversine)	64	2225.72	1193	9.5427
A* Search (Heuristic 2: Landmark - Tokyo)	64	2225.72	1255	7.3675

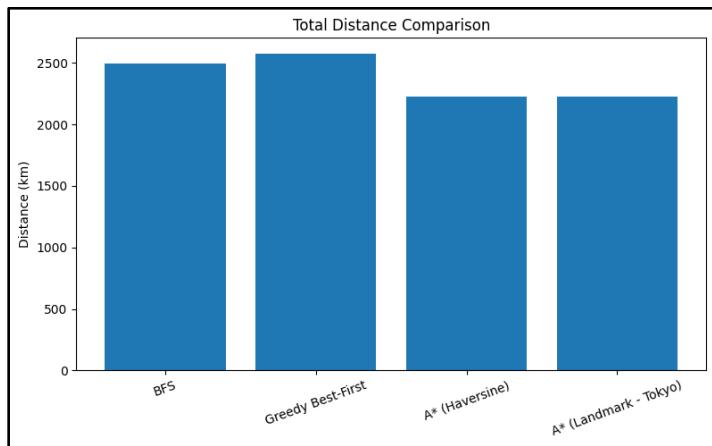


Figure 1 – Comparison of execution time between algorithms

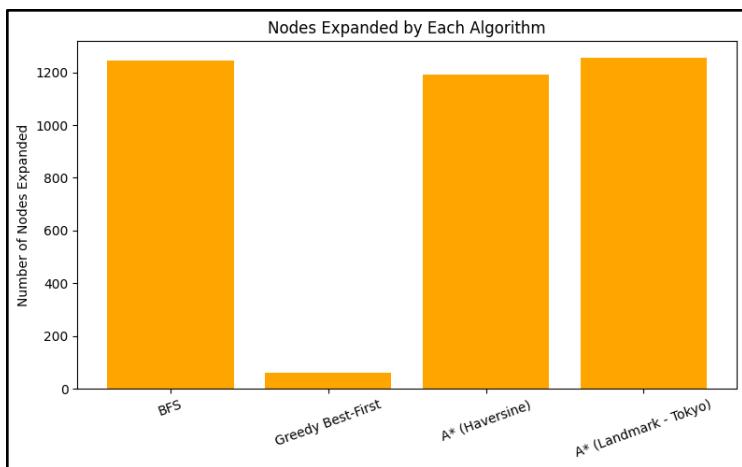


Figure 2 – Comparison of path cost

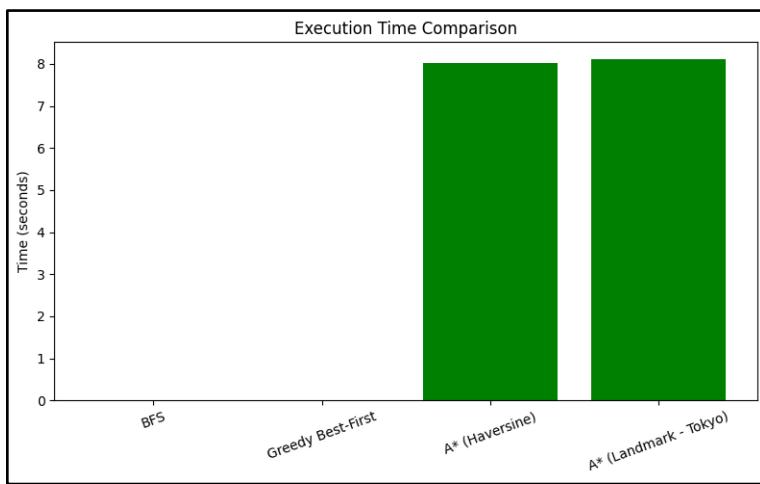


Figure 3 – Comparison of number of nodes expanded

# Conclusion

In this project, different search algorithms were used to find the best route from Makurazaki to Nemuro using real data from Japanese cities. The algorithms tested were Breadth-First Search (BFS), Greedy Best-First Search, and A\* with two different heuristics, the Haversine and Landmark (Tokyo) heuristics.

The results showed that each algorithm has its own strengths and weaknesses. BFS explored the most nodes and took longer, but it worked well as a baseline since it always finds a solution. Greedy Best-First Search was much faster and expanded fewer nodes, but it did not give the shortest path because it only focused on getting closer to the goal without checking the total distance.

A\* performed the best overall. Using the Haversine heuristic, it found the shortest path (about 2225.72 km) and balanced both cost and efficiency. The Landmark heuristic gave the same total distance but expanded slightly more nodes, showing that the Haversine heuristic was more reliable in this case.

Overall, this project showed how heuristics can greatly improve search performance by guiding the algorithm in a smarter way. It also proved that the quality of the heuristic directly affects the efficiency and accuracy of the results. In the future, it would be interesting to test with other heuristics, adjust the number of neighbors (K), or use real road data to make the routes even more accurate.

## References

- Fraga Pereira, R., †, Oren, N., ‡, Pontifical Catholic University of Rio Grande do Sul, Brazil, & University of Aberdeen, United Kingdom. (2017). *Landmark-Based heuristics for goal recognition*. <https://ojs.aaai.org/index.php/AAAI/article/view/11021/10880>
- Emvin, C., & Amsavalli, Y. (2024). *Estimation of Accurate Path Length with Geospatial Data Analysis*. Research.chalmers.se; Chalmers University of Technology.  
<https://research.chalmers.se/en/publication/541772>
- Arif, O. (2025). *Japan cities dataset (based on SimpleMaps World Cities Database)* [Unpublished dataset]. American University of Sharjah, Department of Computer Science & Engineering
- Google colab  
link:[https://colab.research.google.com/drive/1Fk2o9sUEyRsE7XBjpdW6TgdakqNYfI\\_L?usp=sharing](https://colab.research.google.com/drive/1Fk2o9sUEyRsE7XBjpdW6TgdakqNYfI_L?usp=sharing)

## Contribution of each member

Team member	Contribution
Rana Riyaz	<ul style="list-style-type: none"><li>• Code: Greedy search</li><li>• Report: Introduction, abstract, references</li></ul>
Fathimamaaziya	<ul style="list-style-type: none"><li>• Code:A* Search(with 2 heuristics)</li><li>• Report: Results,Conclusion</li></ul>
Hamad	<ul style="list-style-type: none"><li>• Code: BFS</li><li>• Report: Methodology</li></ul>