1. Problem Statement

Some of UTS Student come from Asia, especially Southeast Asia. Southeast Asia, also known as SEA, is the neighbourhood of Australia that works together in times of crisis to grow together (Morrison MP, 2021).

Southeast Asia consists of 11 different countries and approximately 674 million people. It consists of Indonesia, Philippines, Vietnam, Thailand, Myanmar, Malaysia, Cambodia, Laos, Singapore, East Timor, Brunei. Each of the countries has its language. There is a lot of reason to travel around Southeast Asia, some of them are cuisine, friendly, low cost (Shvili, 2021).

In this example, I will create a searching AI technique to travel around Southeast Asia. These below are the coordinate of Southeast Asia countries.

Country	Coordinate (x)	Coordinate (y)
Cambodia	12.57	104.99
Laos	19.86	102.50
Myanmar	21.91	95.96
Thailand	15.87	100.99
Vietnam	14.06	108.28
Brunei	4.54	114.73
Philippines	12.88	121.77
Indonesia	-0.79	113.92
Malaysia	4.21	101.98
Singapore	1.35	103.82
Timor-Leste	-8.87	125.73

*Coordinate may be inaccurate due to having multiple international airports and the huge size of a country.

*This coordinate is not created by me; Check the reference to find the sources

On this problem, I have also recorded the neighbour of each country. This neighbour shows the flight path available from a country to another country. The weight of each edge shows the on-flight time required to flight from a country to a specific country. For example: to fly from Singapore to Indonesia requires around 110 minutes of flight time. This is below all available paths and on-flight time required to travel from one city to another SEA city.

Cambodia	Indonesia 280 Laos 80 Myanmar 115 Thailand 65 Vietnam 45 Philippines 400 Malaysia 110 Singapore 105
Laos	Indonesia 360 Thailand 65 Vietnam 60 Malaysia 170
Myanmar	Indonesia 355 Thailand 85 Vietnam 125 Brunei 240 Philippines 405 Malaysia 150 Singapore 175
Thailand	Indonesia 205 Vietnam 90 Brunei 165 Philippines 210 Malaysia 210 Singapore 145
Vietnam	Indonesia 375 Brunei 140 Philippines 150 Singapore 140
Brunei	Philippines 125 Malaysia 85 Singapore 130
Philippines	Indonesia 235 Malaysia 225 Singapore 225
Malaysia	Indonesia 110 Singapore 55 Timor-Leste 255
Singapore	Indonesia 110
Timor-Leste	Indonesia 345

2. Selection of AI technique

To solve this problem, we need to give travellers multiple options, including optimal cost search. On this problem, I will use Depth-First Search (DFS), Breadth-First Search (BFS), and an optimal cost algorithm. Hopefully, using three algorithms will provide at least 2 different paths for a traveller to reach their destination. On the other hand, applying the optimal cost algorithm on this case will let the user know the lowest on-flight time taken to travel from one country to another.

Several candidates can be used as an optimal cost algorithm on this matter, including:

- Dijkstra's Algorithm, also known as Uniform Cost Search (UCS)
- A* Search

To decide optimal-cost search to solve this problem, I decide to test them. There will be five times test proceed on this selection process. On each test, I will randomly generate two countries. There are three main rules on this test:

- The first country selected will be the initial country.
- The second country selected will be the goal.
- The first country must be different from the second country.
- Each Algorithm tested must have the same initial and goal country.

I will look at the average number of iterations required for each Algorithm to find the optimal cost path. Selection prove will be provided in the screenshot section along with test cases detail.

Number of Tests	From	To
First Test	Cambodia	Singapore
Second Test	Indonesia	Vietnam
Third Test	Brunei	Laos
Fourth Test	Thailand	Malaysia
Fifth Test	Singapore	Myanmar

In the first case, we can see that the system will try to find the shortest distance from Cambodia to Singapore. For A* Search Algorithm, it requires 23 times of iteration to find this shortest path. Meanwhile, on UCS, it only cost 21 iterations. On the second test case, UCS takes 28 while A* need 34 iterations.

In the third test case, A* and UCS have 31 and 30 iterations. On the fourth test, A* won with 26 iterations, while UCS needed 27 iterations. In the last case, A* need 29 iterations and UCS only need 28 iterations. It can be concluded as shown in the table below.

Number of Tests	A*	UCS
First Test	23	21
Second Test	34	28
Third Test	31	30
Fourth Test	26	27
Fifth Test	29	28
Total	143	134

The table here shows that UCS won against A* to solve this specific problem. A small difference can see it in all of the test cases. With all the information provided on this specific problem, we can conclude that UCS is better than A*.

3. Explanation of AI technique

a. BFS (Breadth-First Search)

Breadth-First Search, also known as BFS, is a common searching algorithm that starts with the root node and explores the adjacent node before exploring the next level node. BFS need to use Queue to store all of the visited nodes. Queue uses a principle of FIFO (First-in–First-Out), which will execute the data based on the first data into the last ("Breadth-first search (BFS) algorithm:: AlgoTree," n.d.).

b. DFS (Deep-First Search)

Deep First Search or DFS is a very common searching algorithm that uses Stack to search. While Queue uses FIFO (First-in–First-Out), Stack uses LIFO (Last-in–First-Out), which executes the data inside the list from the last data to the first data in. DFS explore from root node to all the node along with the depth of the path. If it is stuck to the end, it will be backtracking to the top to explore the next selected node ("Depth first search (DFS) algorithm :: AlgoTree," n.d.).

c. UCS (Uniformed Cost Search)

Uniform-Cost search or also known as Dijkstra Algorithm is useful for the large graph. This Algorithm will work from any chosen state, then visit the adjacent state. On this occasion, the rule to choose an adjacent state is selected from the lowest cost state and must be an un-visited state. On this searching algorithm, it will still search for the even after finding the goal state. There will be a priority queue that will provide the cheapest way to get into a specific state ("Uniform-cost search (Dijkstra for large graphs)," 2021).

4. Solution Result

This AI technique could be used to find travel paths in Southeast Asia. In a certain circumstance, this agent may show up to 3 different ways to travel from a specific country to another county within Southeast Asia. This AI applies three different searching techniques to give an optional path to the user. Moreover, this AI applies UCS, which will provide the lowest-cost path in one of the options.

5. References

Breadth first search (BFS) algorithm :: AlgoTree. (n.d.). Algotree :: AlgoTree. https://algotree.org/algorithms/tree_graph_traversal/breadth_first_search/

Cohen, E. (2017). *Counties geographic coordinates*. Kaggle: Your Machine Learning and Data Science Community. https://www.kaggle.com/eidanch/counties-geographic-coordinates?select=countries.csv

Depth first search (DFS) algorithm :: AlgoTree. (n.d.). Algotree :: AlgoTree. https://algotree.org/algorithms/tree_graph_traversal/depth_first_search/

Morrison MP, H. S. (2021, June 11). *Southeast Asia*. Australian Government. https://www.dfat.gov.au/geo/southeast-asia

The random choice generator online tool. (n.d.). TextFixer. https://www.textfixer.com/tools/random-choice.php

Shvili, J. (2021, April 11). *Southeast Asian countries*. WorldAtlas. https://www.worldatlas.com/articles/which-countries-are-considered-to-be-southeast-asia.html

Uniform-cost search (Dijkstra for large graphs). (2021, August 25). GeeksforGeeks. https://www.geeksforgeeks.org/uniform-cost-search-dijkstra-for-large-graphs/

6. Screenshot

a. A* Search

Test 1

```
display(SEA_problem, AStar(heuristic_fun_euclidean))
```

```
Algorithm: AStar
 Iteration times : 29  Path : [<Node Singapore>, <Node Myanmar>]
```

Test 2

```
display(SEA_problem, AStar(heuristic_fun_euclidean))
```

```
Algorithm: AStar
 Iteration times : 23  Path : [<Node Cambodia>, <Node Singapore>]
```

Test 3

```
display(SEA_problem, AStar(heuristic_fun_euclidean))
```

```
Algorithm: AStar
 Iteration times : 34  Path : [<Node Indonesia>, <Node Singapore>, <Node Vietnam>]
```

Test 4

```
display(SEA_problem, AStar(heuristic_fun_euclidean))
```

```
Algorithm: AStar
 Iteration times : 31  Path : [<Node Brunei>, <Node Vietnam>, <Node Laos>]
```

Test 5

```
display(SEA_problem, AStar(heuristic_fun_euclidean))
```

```
Algorithm: AStar
 Iteration times : 26  Path : [<Node Thailand>, <Node Cambodia>, <Node Malaysia>]
```

b. UCS

Test 1

```
display(SEA_problem,UCS_tree())
```

```
Algorithm: UCS_tree
 Iteration times : 21  Path : [<Node Cambodia>, <Node Singapore>]
```

Test 2

```
display(SEA_problem,UCS_tree())
```

```
Algorithm: UCS_tree
 Iteration times : 28  Path : [<Node Indonesia>, <Node Singapore>, <Node Vietnam>]
```

Test 3

```
display(SEA_problem,UCS_tree())
```

```
Algorithm: UCS_tree
 Iteration times : 30  Path : [<Node Brunei>, <Node Vietnam>, <Node Laos>]
```

Test 4

```
display(SEA_problem,UCS_tree())
```

```
Algorithm: UCS_tree
 Iteration times : 27  Path : [<Node Thailand>, <Node Cambodia>, <Node Malaysia>]
```

Test 5

```
display(SEA_problem,UCS_tree())
```

```
Algorithm: UCS_tree
 Iteration times : 28  Path : [<Node Singapore>, <Node Myanmar>]
```