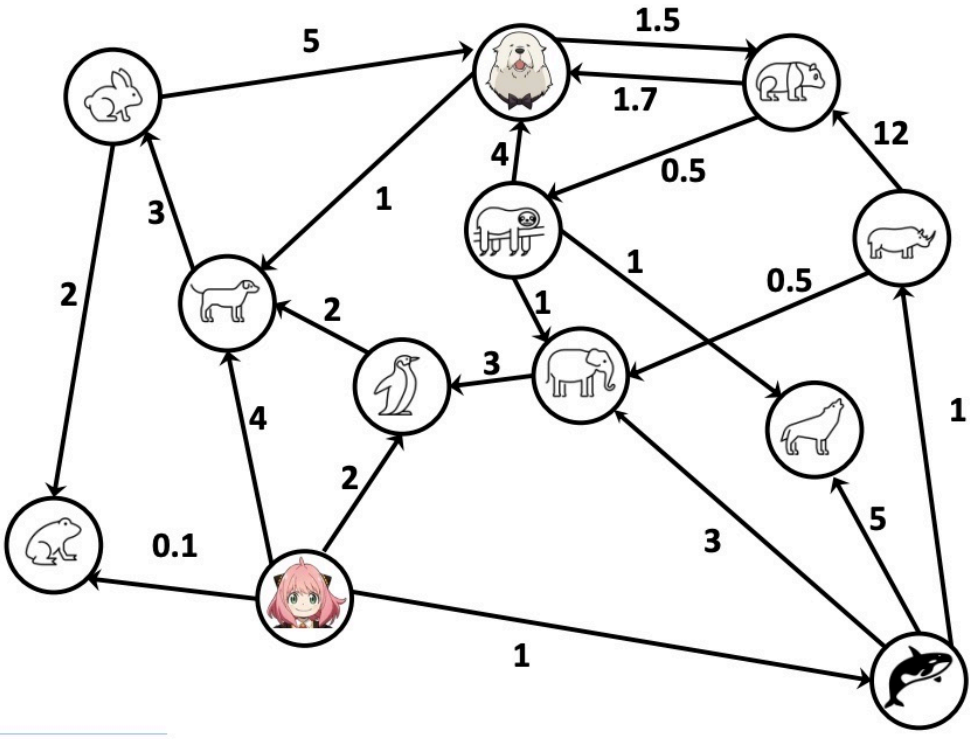
Problems:
Use the graph below for problems 1 and 2



1 (text) Dijkstra [15 points] Find the shortest path from Anya (Human) to all other nodes using Dijkstra's (Show your work and intermediate steps)

2 (text) A* [15 points] Find the shortest path from Anya to all other nodes using A* (Show your work and intermediate steps) **Hint:** Use the geographical location of each node to derive the heuristic to use.

3 (text) Comparison [5 points] Which algorithm found the shortest path to H in less iterations Dijkstra or A*? (Explain your answer)

1.)

Anya	0	Anya → Frog 0.1 Anya → Whale 1 Anya → Penguin 2 Anya → Dog 4
Frog	0.1	→
Whale	1	Whale → Human (1) + Whale 1

whale 1	whale → Rhino (+1) = Rhino 2 whale → Elephant (+3) = Elephant 4 whale → Wolf (+5) = Wolf 6
Rhino 2	Rhino → Elephant (+0.5) = Elephant 2.5 Rhino → hamster (+12) = Hamster 14
Penguin 2	Penguin → Dog (+2) gives 4
Elephant 2.5	Elephant → penguin (+3) = 5.5 (>2) ignore
Dog 4	Dog → bunny (+3) = bunny 7
Wolf 6	——
Polar bear	polar bear → hamster (1.5) = hamster 15.5 polar bear → dog (+1) = ignore
hamster 15.5	hamster → sloth (0.5) = sloth 14
sloth 14	sloth → polar bear (+4) = 18 (ignore) sloth → Elelephant (+1) = 15 (ignore) sloth → Wolf (+1) = 15 ignore

Final Shortest Distance:

Anya	0
Frog	0.1
whale	1
Penguin	2
Rhino	2
Elephant	2.5
Dog	4
Wolf	6
bunny	7
Polar bear	12
hamster	13.5
Sloth	14

7 (text) Algorithm Analysis [25 points]
For each of the algorithms you wrote for problems 4-6, explain their time complexity and space complexity using Big-O notation. Explain how you arrived at your answer.

Problem 4:

the time complexity I think is $O(n^2)$ because it checks every pair of nodes in the matrix to make an undirected version of the graph, which takes $O(n^2)$ time. Then it uses DFS to visit all connected nodes, which also takes up to $O(n^2)$ in worst case.

I think the space complexity is $O(n^2)$ too, because the undirected graph is stored as a list of lists and its worst case can hold n^2 edges.

Problem 5:

I think the time complexity is $O(n^4)$ because it uses four nested loops to check all possible 4 node combinations. For each combination, it checks if the edges form a 4-cycle.

the space complexity is $O(n^2)$ because the program stores a set of cycles, and in the worst case, there would be around n^2 different Y-cycles.

Problem 6:

The time complexity is $O(n^2)$ because the program prints an $n \times n$ matrix, which takes most time.

The space complexity is $O(n^2)$ too since it uses a 2D array to store all connections between nodes.