1.How does the turn-cost affects the path and the explored area for each search.

BFS is insensitive to costs; it merely expands nodes by their levels, seeking the shortest path to the goal state (counted by the number of steps, not the cost). Therefore, even if the cost of turning is increased, it will not affect the path length (i.e., the number of steps) found by BFS, but this might not be the most cost-effective path.

DFS is similarly insensitive to the cost of turning. It relies on a stack structure to delve deep into the search until the goal is found or there's nowhere else to go. DFS might find a path but not necessarily the least costly one, especially when the cost of turning significantly increases.

UCS is very sensitive to costs; it always prioritizes expanding the node with the lowest total cost. Increasing the cost of turning directly affects UCS's search strategy, making the algorithm prefer paths with straight movements over those with frequent turns, even if it means taking a longer path.

A\* combines the cost sensitivity of UCS and heuristic information (such as the estimated cost to the goal), aiming to find the least costly path more efficiently. Modifying the cost of turning affects A\*'s search efficiency because A\* considers the cost of turning to optimize path selection, avoiding costly turns.

Greedy Algorithm uses a heuristic function to decide the direction of the search, considering only the next step's contribution to the goal, not the total cost of the path taken. Increasing the cost of turning might affect the Greedy Algorithm's choices, making it prefer straight movements overturns, but this does not guarantee finding the best or most economical path.

2. Imagine it is going to cost twice as much to climb up the vertical than moving horizontal. How would you go about incorporating this into current implementation, and where and which functions would be affected? How many different ways this goal can be achieved and what are the benefits and weaknesses of each solution.

Affected functions are cost and node expansion functions. The cost function would need to be modified to account for the increased cost of vertical movements compared to horizontal movements.

For UCS and A\*, these algorithms would be directly affected by the change in cost calculation as they prioritize nodes based on the total path cost. Adjusting the cost function to account for different movement costs would directly influence their behavior, potentially leading to different paths being chosen.

For BFS and DFS, while these algorithms do not prioritize by path cost, incorporating movement costs could still affect implementations where the goal is to find the lowest-cost path, not just the first path found. However, by default, BFS and DFS do not consider path costs.

For greedy algorithms, it uses heuristics to guide search would also be influenced by changes in movement costs, especially if the heuristic involves cost estimation.

Strategies: Directly modify the path cost function to include different costs for vertical and horizontal movements. This requires identifying the direction of movement (vertical or horizontal) each time a new node is generated and applying the appropriate cost. Another approach is to adjust the problem representation so that the cost of vertical movements inherently reflects the increased difficulty. For example, if representing a grid, vertical transitions could be represented in a way that inherently has double the cost of horizontal transitions.