Agenda-based Search

2.1 Implementation DFS, BFS, and UCS

- For implementation of Breadth First Search, Depth First Search and Uniform Cost Search, I have used functional approach for coding.
- Main.py file has all the functions implementing BFS, DFS and UCS. The tubedata.csv file is present in the same folder as of main.py file and read_csv_file() is the function reading the csv file which contains tubedata and converting into the nxgraph.
- DFS, BFS and UCS are uninformed uniform search.
- The graph is created using first and the second column entry as the node value and its edge value is the average time travelling from 1st station to 2nd station which is the 4th column entry in the csv file.
- DFS() function is written to give the solution for depth first search. This function takes into account the graph, initial and goal node. Also this function gives us the number of explored nodes during path search.
- The output of this function is then given to construct_path_from _node functionalong with the starting node or the initial node. This function gives us the path which is followed from initial to final stage using DFS or BFS strategy based on the input.
- Compute_cost_path() is a function which is used to calculate the cost depending on the path provided to the function along with the graph.
- The above code flow is same for both the BFS and DFS. However, for UCS, to construct the path, construct_path() function is written which takes into accountsolution provided by UCS() function. And, compute_path_cost() is used to calculate the cost of the path.
- The cost which is calculated only depends upon the weight of the edge which is representing the average time required to travel from one station to other.

2.2 Compare BFS, DFS and UCS

BFS	DFS	UCS
It uses First-In-First-Out	It uses Last=In-First-Out	
Time complexity is equivalent to number of nodes traversed. ($O(n^s)$) where s is the depth of shallowest solution	Time complexity is equivalent to number of nodes traversed. ($O(n^d)$)	$T(n) = O(n^{C/\varepsilon})$
Space complexity ($O(n^s)$)	Space complexity O(n*d)	$S(n) = O(n^{C/\varepsilon})$
Complete if solution exist for a given search tree	Complete if the search tree is finite	Complete if states are finite and there is no loop with zero weight
Optimal as long as costs of all edges are equal.	Not optimal	Optimal if no negative cost

	BFS		DFS		USC	
	Nodes explored	Cost	Nodes explored	Cost	Nodes explored	Cost
Canada Water to Stratford	39	15	7	15	54	14
New Cross Gate to Stepney Green	25	14	33	27	17	14
Ealing Broadway to South Kensington	49	20	180	57	52	20
Baker Street to Wembley Park	15	13	4	13	90	13

As per the comparison given in the above table and asper the output of the code for given inputs, we can see that performance of UCS in terms of average time as the cost function is most efficient for all the inputs. After UCS, BFS and then DFS.

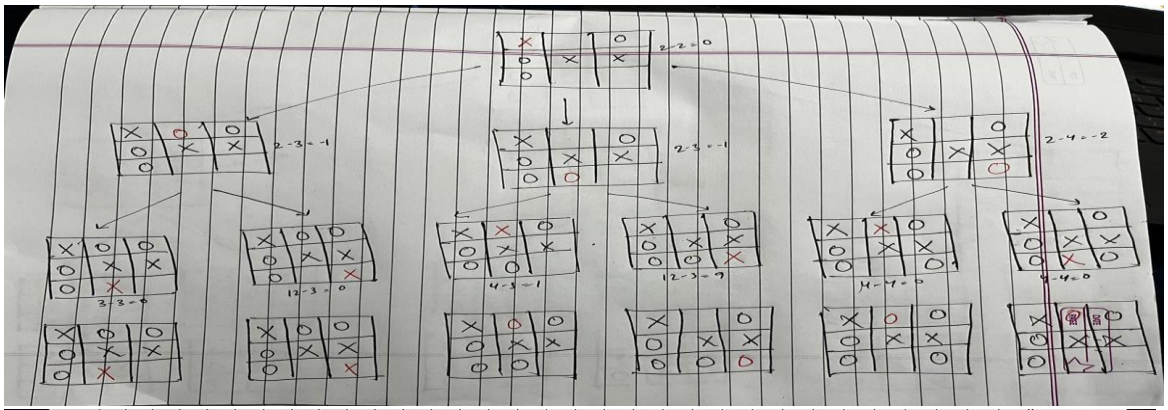
2.3 Extending the cost function

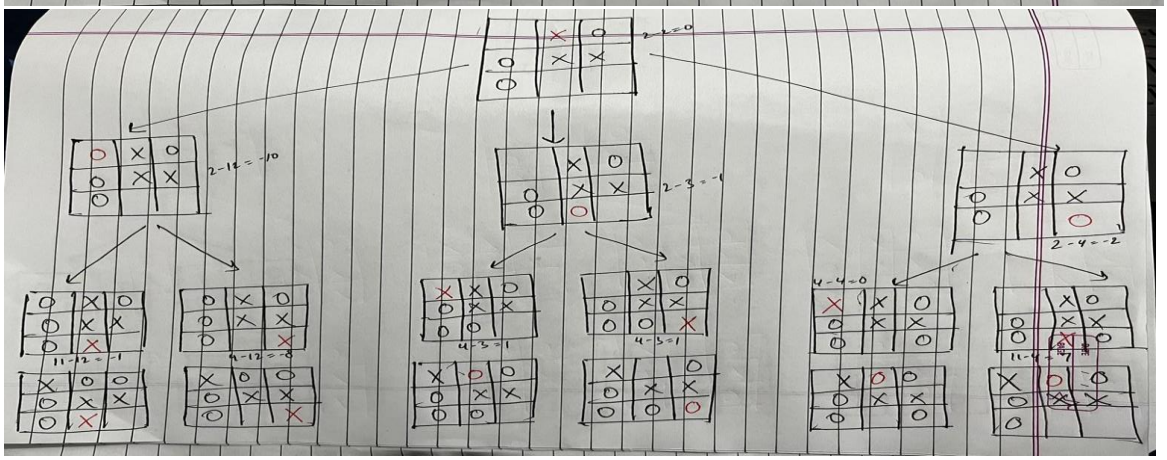
- To extend the cost function, 2 mins will be added to the average time if the tubeline is changed.
- The total cost for the 'Euston' to 'Victoria' path using normal UCS was 7 however this got increased by 2 mins as there is a line change in the path followed by UCS function to reach from 'Euston' to 'Victoria'.
- I have used nxgraph to create a graph which takes into account only 1 weight value. However, I have created a different csv vile which has headers and only 3 columns which includes startingStation, endingStation and tubeline.
- I have saved the tubeline for starting node and whenever the next station that is getting picked by the algorithm has a different tubeline, transit time (which is 2 mins) got added In the total cost of the algorithm.

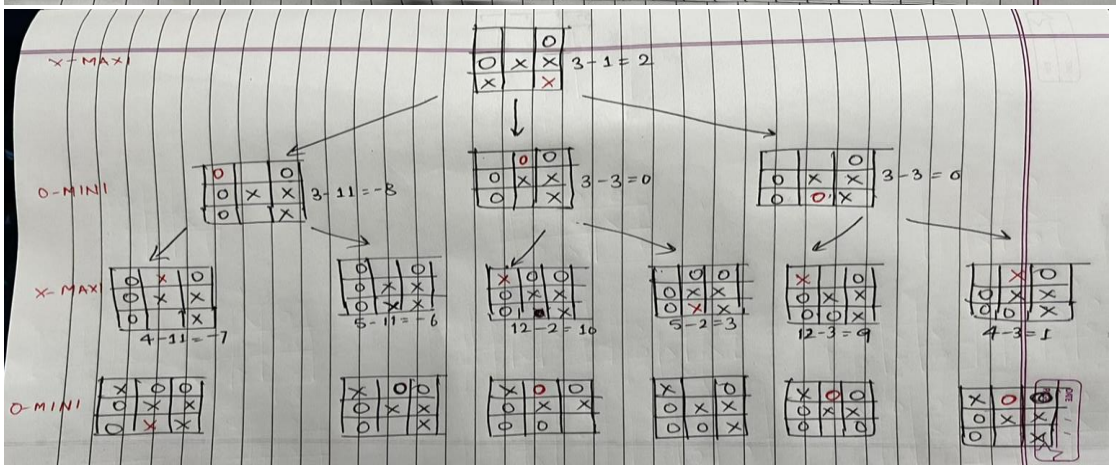
2.4 Heuristic Search

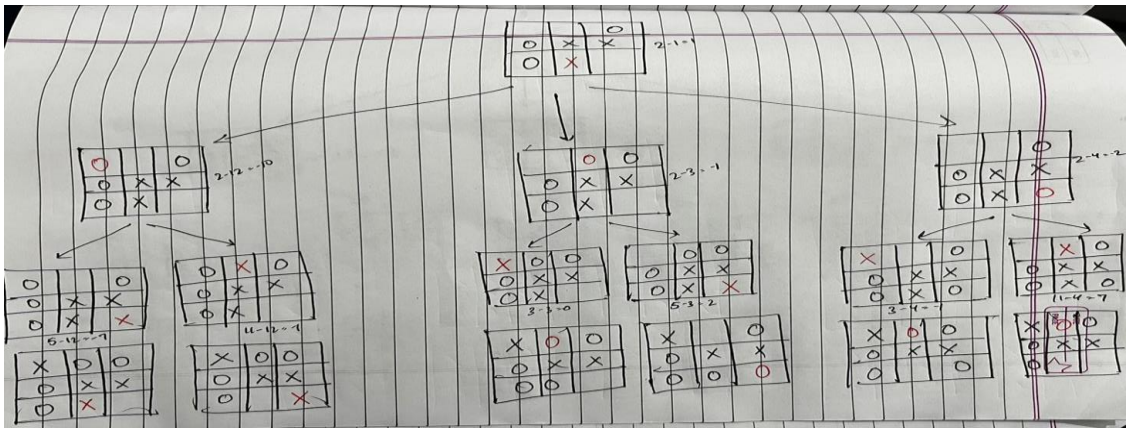
- Informed search algorithms have information about the goal state which helps in efficient searching.
- The information about the goal state is obtained by a function known as heuristics.
- Heuristics function estimates how close the state is to the goal. (Manhattan or Euclidean distance). In our case, I have taken the average time as my heuristics value.
- Depending upon the choices that we have to expand the path, priority is given to the node which is closest to the goal.
- For informed search, I have used Astar algorithm.
- We can improve the heuristic function with multiple ways –
    - If the tubeline is changing, then it is going to add cost to the total journey moneywise. We can consider this consideration.
    - Also, if the zone is changing then the total cost of the journey is going to increase moneywise. Hence, this can be also taken into consideration.
- I have used Astar as it does not expand the same node more than once.

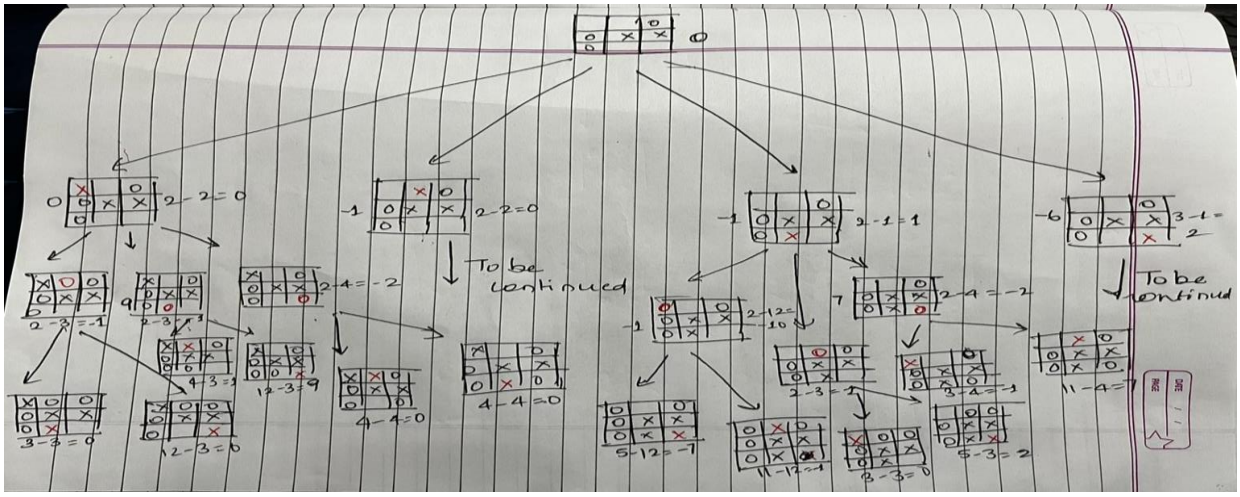
3.1 MiniMax

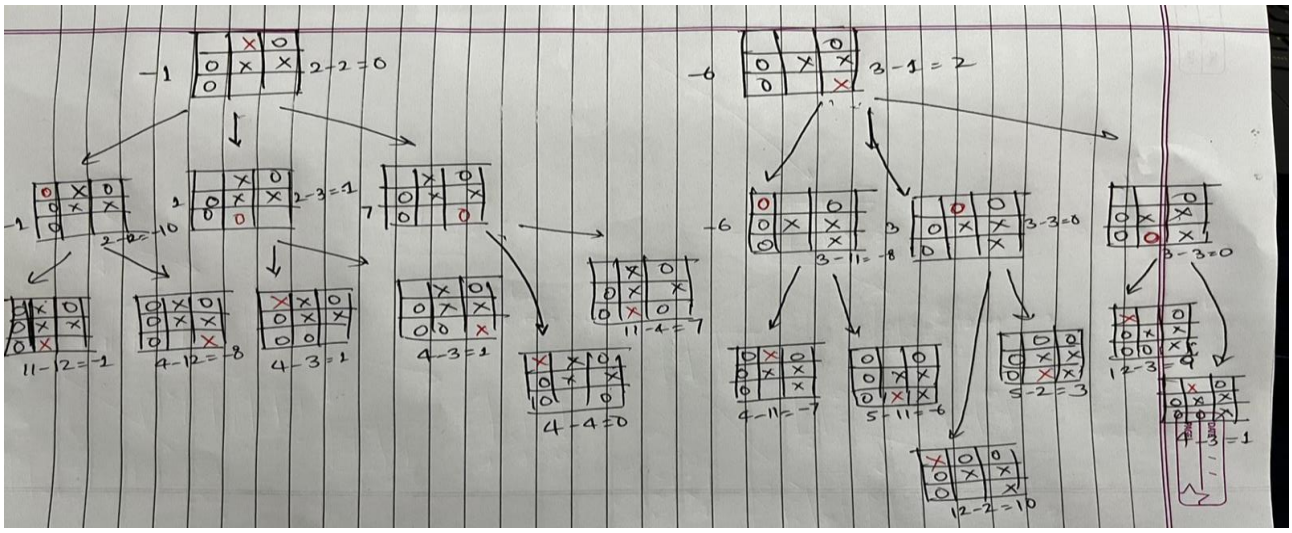


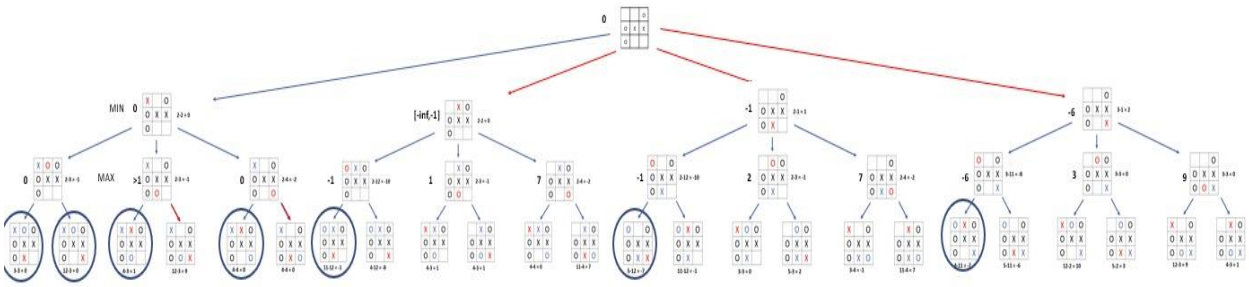







To be continued
To be continued




MIN
MAX