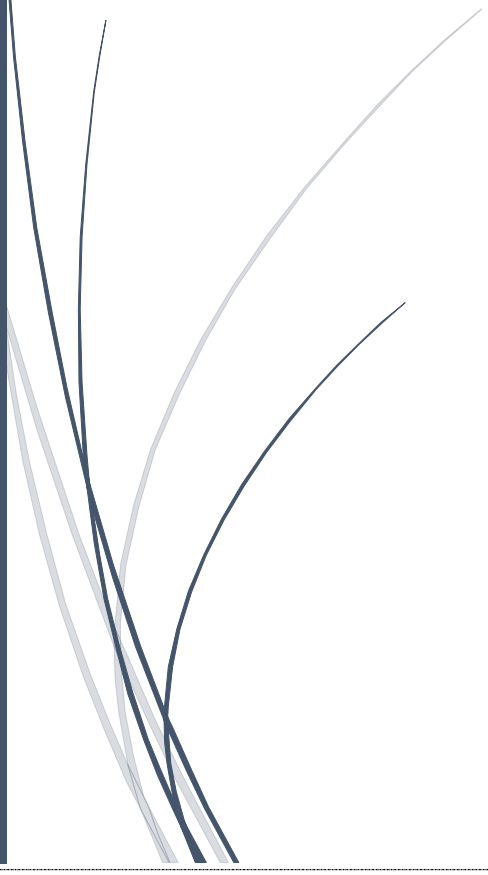
5/5/2023

Stage#1

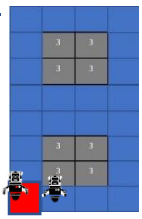
For Shelf Inventory Robot/Agent



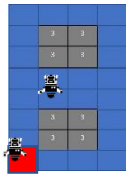
A. Information & assumptions needed to apply A*:

- To apply the A* Algorithm, We made the initial point of the agents to get the boxes dynamically, which means the agents can get the given boxes that are required to put into the shelves from any point in the grid world.
- We chose the MSE(Mean Squared Error) hueristic function but with few tweaks.
- We added two functions which are NVP(Nearest Vertical Position) and NHP(Nearest Horizontal Position) to the MSE in order to make the agents choose the minimum path cost if multiple paths have the same path cost.
- In Nearset Vertical Position method, the method becomes critical in finding the minimum path cost depending on the goal point which if the X-axis of the goal point is less than or equal to one, the value will be zero else it will be 3.
- In Nearset Horizontal Position method, the method also becomes criticial in in finding the minimum path cost depending on the goal point which if the Y-axis of the goal point is less than 2, the value will be zero else it will be 3. If the Y-axis of the goal point is less than 4, the value will be 3 else it will be 4. If the Y-axis of the goal point is less than 8, the value will be 7.

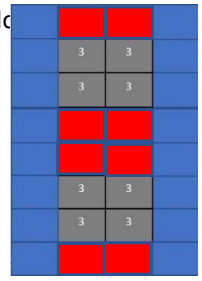
B. Applying A* search algorithm on the state diagram:



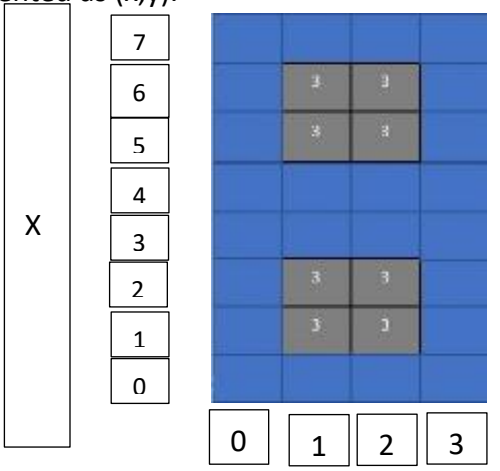
- **Solving using A***
  - **a) Assumptions and notes:**
    - Initial state is entered like the above figure where the agent standing on the blue block is the agent that will be moving.
    - Goal state is entered like the following figure.
    - Since our plan to implement DFS before knowing can't be performed, so we chose a heuristic function that somehow can be a replica



      - ❖ F(x)=C(x) "path cost" +H(x) "heuristic function".
      - ❖ C(x)=1 which represents # of turns where each action (Left, Right, Up, Down, Put) consumes a turn to be executed, and it's neglected because it isn't the dominant effector in the final value of F(x).
      - ❖ Hence, $F(x) \approx H(x)$ where we always seek the minimum F(x) in each step.
      - ❖ Note that the put action is performed automatically in 8 after reaching the goal state so it won't be considered during the simulation for simplicity.
      - ❖ Note that the agent will go to the inventory 1st before going to the goal state and carrying the box will consume 1 turn, but it won't be considered/skipped in the simulation for simplicity as well.
      - ❖ Recall that put action is executed only in the follo

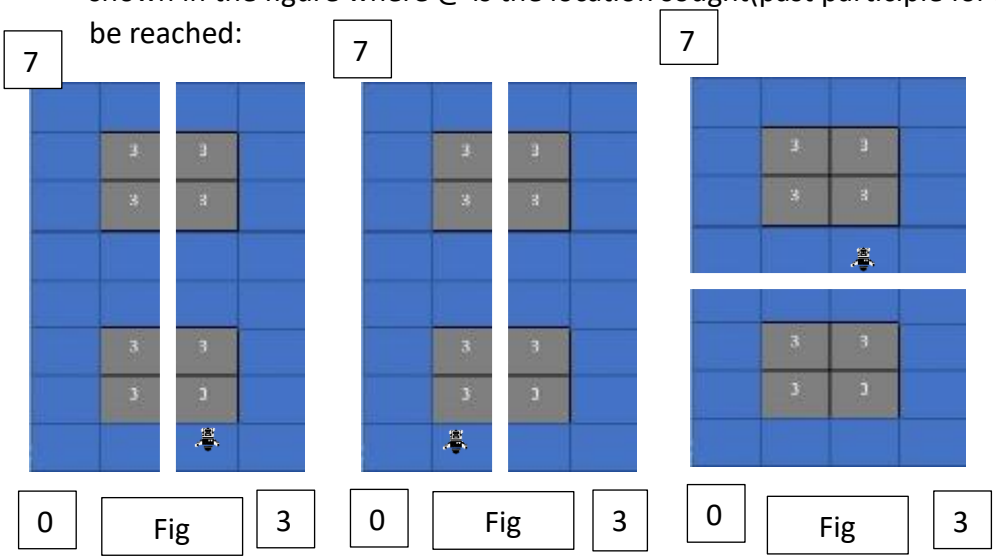


      - ❖ Our grid is a coordinate matrix as explained in the upcoming figure where X represents the rows and Y represents the columns so we have 4 cols and 8 rows, so any block b can be represented as (x,y).



      - ❖ $H(x)=\{ (X_1-X_2)^2+(Y_1-Y_2)^2 \} + \frac{c}{4} + \frac{r}{8}$ where:
        - X_1 is the X coordinate of the next state.

- Y_1 is the Y coordinate of the next state.
- X_2 is the X coordinate of the goal state.
- Y_2 is the Y coordinate of the goal state.
- $c = |num1 - \mathrm{Y1}|$, So c is the column difference between num_1 (which is explained later on) and Y_1.
- $r = |num2 - \mathrm{X1}|$, So r is the row difference between num_2 (which is explained later on) and X_1.
- To clarify what num1 is where num2 will be the same scenario but with a horizontal separator(fig 3).
- Imagine splitting the grid vertically and having 2 decisions to be taken as shown in the figure where @ is the location sought(past participle for seek) to be reached:

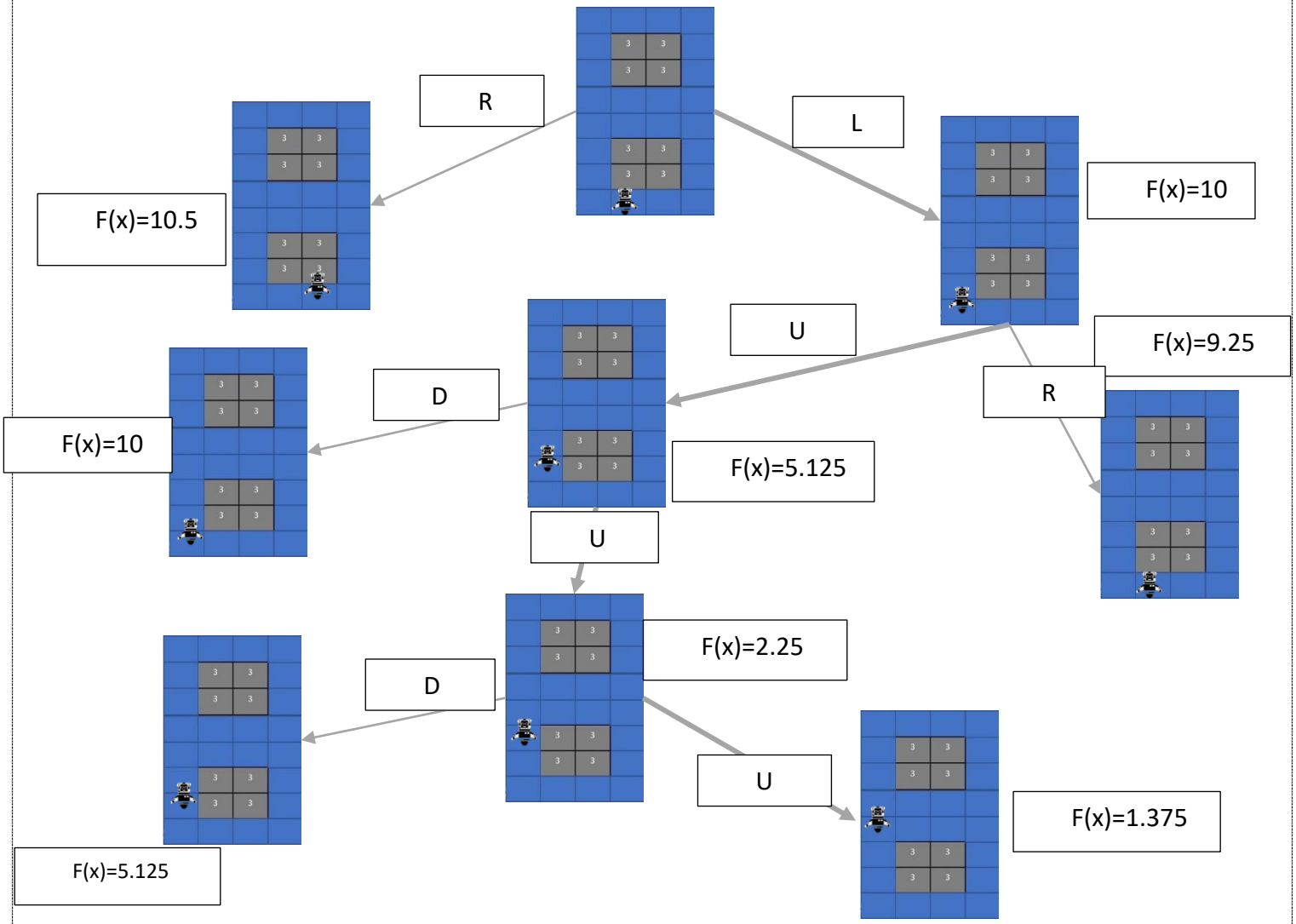


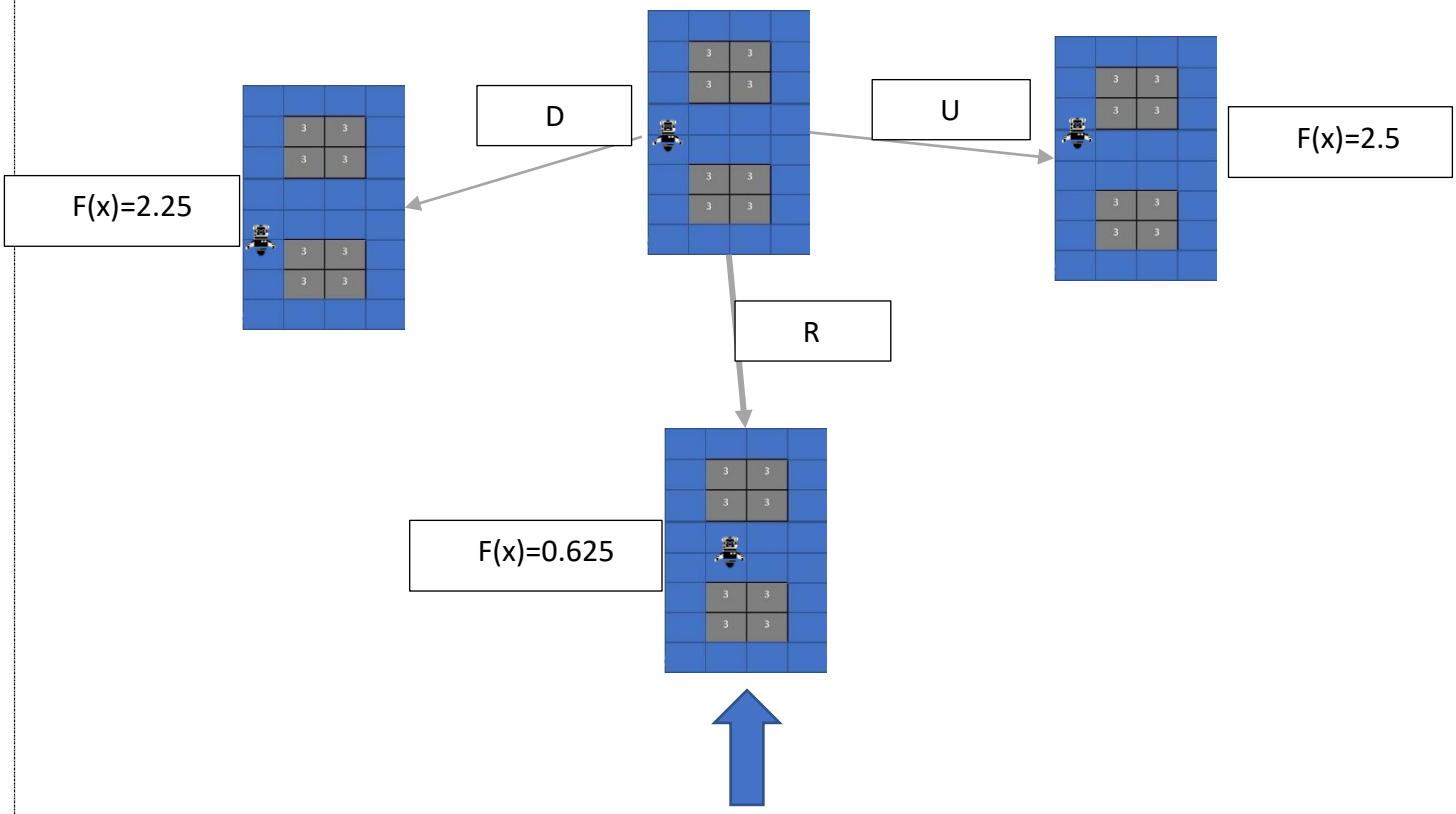
- As observed from moving left or moving right in:
  - Fig 1:It will result the same # of turns(=6).
  - Fig 2: It will result the same difference between{ $(Y_1-Y_2)^2$ } as well as { $(X_1-X_2)^2$}.
- To solve these issues and the agent heads toward a more logical direction, for further explanation consider for example Fig 2 :
  - The logical direction to us is moving left and as mentioned earlier it will have the same coordinated squared distance as moving right.
  - However, to achieve that mathematically a condition is added which is
    - If the @ is on the left side for example num_1 will be substituted with 0 (1^{st} col value).
    - If the @ is on the right side for example num_1 will be substituted with 3 (Last col value).
  - Consequently, the agent will move towards the logical direction which is towards the left concerning Fig 2 next state.

- Note that:
  - c is divided by 4 and r is divided by 8 to be always less than 1 as $\min(num_1)=0$, $\max(num_1)=3$, $\min(num_2)=0$, and $\max(num_2)=7$.
  - So c/4 and r/8 won't affect the normal case where the coordinate distances between the states are different, but they will be the separator between the minimum F(x) final value in the case where the coordinate distances between the states are the same.

b) <u>Example Simulation</u>

- The Simulation will be with 1 agent for simplicity.
- Recall $num_1=0$, $num_2=0$ according to the goal state assumed earlier.




D
U
F(x)=2.25
F(x)=2.5
R
F(x)=0.625

5/5/2023

Stage#2 (Implementation):

For Shelf Inventory Robot/Agent

A. Implementation:

Please find the implementation of our project in the following GitHub link:
DiaaEssam/Shelf-Managment-Agent (github.com)

B. A* was useful or not to:

- A* is extremely useful in our case as it optimizes the memory compared to our first choice of algorithms which was BFS.
- A* nearly results in optimal solutions.
- why 'nearly' because in A* we must eliminate some paths during processing as in our mentioned assumptions we can't let the two agent access the same slot at the same time.
- But in case of BFS we could use the heuristic function which is the number of turns and it will always be the optimal move for each agent and we won't need to eliminate any path, however it could allow the two agents to access the same slot at the same time.