# [5 pts] Please define the agent design problem for this mover’s prototype.

**Action space (A)**

Shift the robotic arm up, down, left, right, and rotate the arm clockwise and counterclockwise.

**Percept space (O)**

The robotic arm can push one moving box or movable and it cannot collide with static obstacles.

**State space (S)**

The position and angle for the robotic arm, and all other positions for moving box, movable obstacles and static obstacles.

**World dynamics (T: SXA → S)**

The position of moving boxes and movable obstacles will change to next state if the robotic arm coincide with them.

**Percept function (Z: S → O)**

The robotic arm can only push one moving or movable obstacles, and it cannot collide with static obstacles. In addition, any objects in the environment cannot collide with each other.

**Utility function (U: S → real number)**

1. To push moving boxes or movable obstacles, the arms must coincide three-quarter of the length to side of box/obstacles.
2. Each unit of moving distance is 0.001
3. The boundary of the environment is from [0,0] to [1,1]
4. The length of robotic arm is 0.1

# [10 pts] Please describe your search method at the conceptual level (i.e., pseudo code and what abstract data structure is used for the container). If you use sampling-based method, please describe the strategy you apply or develop for each of its four components. Otherwise, please describe the details of your discretization method.

//For broad-first search(BFS):

**For** (upper, lower, left, right position of object){

**If** (this position of object is empty){

**If (**the position hasn’t been visited before){

Set the current position to new position}

**}EndFor**

**//Main**

**Load** Input file

**Put** moving box position into list ‘unmoved\_box’

**Put** Movable obstacle positions into list ‘movable\_obs’

**Put** Static obstacle position into list ‘static\_obs’

**//Moving Box to the goal position**

Calculate the Manhattan distance from each moving boxes to arm

Order the list ‘unmoved\_box’ into array based on the Manhattan distance

**While** (The array is not empty){

Poll the first moving box position

**Do** BFS to find the best path to goal position}

**For** each BFS loop{

**If** (the moving box changed movement direction){

Shift and rotate the arm to the side vertical to moving direction}

}**Endfor**

Record arm path one

**//Initial arm position to the moving box**

**If** (arm is not vertical to first move direction of moving box) {

Rotate

**Do** BFS from initial place to side of target moving box}

Record arm path two

Total path of arm(i) = path one + path two **//I = 1,2,3….total # of moving boxes**

**Remove** moving box position from array ‘unmoved\_box’

**Add** moving box goal position to list ‘static\_obs’

}**EndWhile**

**Output** all total path of arm to output text file.

# [12.5 pts] Which class of scenarios do you think your program will be able to solve well? Please explain your answer.

* 1. The robotic arm cannot repeat same path for more than once. So the program can still be efficient even if there are many “dead end” in that scenarios.
  2. Our program can make the robotic arm always find the right side of box to coincide, so many collisions caused by rotation can be avoided.
  3. When the goal position of a moved moving box is on the path of other moving boxes, our program can avoid the collision between them under this scenario.
  4. Under the scenarios that there are numerous available solutions, the broad-first search method we applied can always guarantee our solution has lowest cost.

# [12.5 pts] Under what situation do you think your program will fail? Please explain your answer.

* 1. The moving boxes or robotic arm are surrounding by static obstacles, in that case, there is no solution no matter how this program is designed.
  2. There is limited space between static obstacles and it only has solution when robotic arm rotates between 0 to 90 degree. This is because in our program, the arm can only be moving under 0 degree or 90 degrees.
  3. The arm is possible to collide with other object while rotation. The rotation space that the robotic arm traveled is not considered in our program.
  4. It will fail if there are solutions exist only if the movable obstacles were moved firstly. In our program, the movable obstacles and static obstacles are treated as same objects.