**Name – Rohila Gudipati UIN - 01058204**

**Compile and execution of the code:**

1. The system needs to have python version 2.7 or up
2. Download all the .py files into a folder.
3. input1.txt contains an example board, there are 6 example boards in the given set
4. Open command prompt and run the programs as follows –

python sokoban\_bfs.py <input1.txt >output.txt

python sokoban\_dfs.py <input1.txt >output.txt

python sokoban\_man.py <input1.txt >output.txt

python greedy.py <input1.txt >output.txt

python astar.py <input1.txt >output.txt

1. Result of each execution is stored in output.txt in the folder.

Program to solve sokoban using **Breadth first Search** is in the file **sokoban\_bfs.py**

Program to solve sokoban using **Depth first Search** is in the file **sokoban\_dfs.py**

Program to solve sokoban using **Greedy Search** is in the file **greedy.py**

Program to solve sokoban using **A\* Search** is in the file **astar.py**

**Input Format:**

**O - Wall**

**R – Robot**

**S – Storage space**

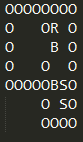
**B - Block**

**# - Robot in storage space**

**@ - Block in storage space**

**Solutions of programs by various searches:**

Board 1: Run command: python sokoban\_bfs.py <input1.txt >output.txt



**Breadth first Search**: E, S, S, S, S, W, N, E, N, N, W, W, W, N, W, W, S, E, E, E, E, N, E, S, S, S, W, N, E, N, W, W, W, N, W, W, S, E, E, E, E, N, E, S, S

**Depth first Search:** E, S, W, W, W, E, E, S, E, S, S, W, N, E, N, N, W, W, W, S, W, W, N, E, E, E, E, N, E, S, S, S, W, N, E, N, W, W, W, S, W, W, N, E, E, E, E, N, E, S, S

**Greedy Search:** E, S, S, S, S, W, N, E, N, N, W, W, W, N, W, W, S, E, E, E, E, N, E, S, S, S, W, N, E, N, W, W, W, N, W, W, S, E, E, E, E, N, E, S, S

**A\* Search:** E, S, S, S, S, W, N, E, N, N, W, W, W, N, W, W, S, E, E, E, E, N, E, S, S, S, W, N, E, N, W, W, W, N, W, W, S, E, E, E, E, N, E, S, S

Board 2: Run command: python sokoban\_dfs.py <input2.txt >output.txt



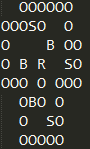
**Breadth first Search**: E, S, S, E, E, E, E, N, N, W, S, E, S, W, W, W, W, N, N, E, S, E, S, E, E, N, W, W, W, N, W, W, S, E, S, E, E, N, W, S, W, N, E, E, E, E, S, W, W, W, N, N, W

**Depth first Search:** S, S, E, E, E, E, E, N, N, W, S, E, S, W, W, W, W, N, N, E, S, E, S, E, E, N, W, W, W, N, W, W, S, E, S, E, E, N, W, S, W, N, S, E, E, N, E, E, S, W, W, W, N, N, W

**Greedy Search:** E, S, S, E, E, E, E, N, N, W, S, E, S, W, W, W, W, N, N, E, S, E, S, E, E, N, W, W, W, N, W, W, S, E, S, E, E, N, W, E, E, E, S, W, N, W, W, S, W, N, E, N, W, E, S, E, E, S, W, W

**A\* Search:** E, S, S, E, E, E, E, N, N, W, S, E, S, W, W, W, W, N, N, E, S, E, S, E, E, N, W, W, W, N, W, W, S, E, S, E, E, N, W, S, W, N, E, E, E, E, S, W, W, W, N, N, W

Board 3: Run command: python greedy.py <input3.txt >output.txt



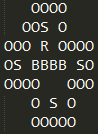
**Breadth first Search**: E, S, S, S, W, W, N, N, N, N, W, W, S, E, E, E, E, E, N, N, W, S, S, S, S

**Depth first Search:** W, N, W, W, S, E, E, E, E, S, S, S, W, W, N, S, E, E, N, N, N, E, W, W, W, S, N, E, E, S, S, S, W, W, N, N, S, S, E, E, N, N, N, W, W, S, S, S, E, E, N, N, N, E, N, W, W, S, W, N, W, W, S, E, E, S, S, S, E, E, N, N, N, W, E, S, S, S, W, W, N, N, N, W, N, E, E, W, S, S, S, S, E, E, N, N, N, E, N, N, W, S, S, S, S

**Greedy Search:** N, E, N, E, S, W, S, E, W, W, W, N, W, W, S, E, E, E, N, E, E, S, W, S, S, S, W, W, N, N, N, N, S, E, N, E, S, S, S

**A\* Search:** E, S, S, S, W, W, N, N, N, N, W, W, S, E, E, E, E, E, N, N, W, S, S, S, S

Board 4: Run command: python astar.py <input4.txt >output.txt



**Breadth first Search**: E, S, E, E, W, W, S, W, N, N, S, W, W

**Depth first Search:** S, W, W, E, E, N, E, S, E, S, S, W, W, N, S, E, E, N, W, E, N, E, W, W, N, W, W, S, E, N, E, S, E, S, S, W, W, N, S, E, E, N, W, N, N, W, W, S, E, N, E, S, E, S, S, W, W, N, S, E, E, N, N, W, S, W, N, N

**Greedy Search:** E, S, E, W, S, W, N, N, S, W, W, E, E, E, E, E

**A\* Search:** E, S, E, E, W, W, S, W, N, N, S, W, W

Board 5: Run command: python astar.py <input5.txt >output.txt



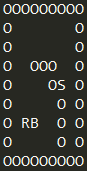
**Breadth first Search**: Solution not available for the board

**Depth first Search:** Solution not available for the board

**Greedy Search:** Solution not available for the board

**A\* Search:** Solution not available for the board

Board 6: Run command: python greedy.py <input6.txt >output.txt



**Breadth first Search**: N, E, E, S, W, S, W, N, N, N, N, W, N, E, E, E, E, N, E, S, S

**Depth first Search:** W, N, N, E, E, S, E, S, W, S, W, N, N, N, N, W, N, E, E, E, E, N, E, S, S

**Greedy Search:** E, S, E, N, E, N, W, S, W, N, E, N, W, S, W, N, N, W, N, E, E, E, E, N, E, S, S

**A\* Search:** S, E, N, N, E, N, W, S, W, N, N, W, N, E, E, E, E, N, E, S, S

**Analysis of Search Algorithms:**

b is the branching factor and d is the depth of the tree

**Breadth first Search**: Time complexity is O(bd+1) , Space complexity is O(bd)

**Depth first Search:** Time complexity is O(bd+1) , Space complexity is O(bd)

**Greedy Search:** Time complexity is O(bd+1) , Space complexity is O(bd)

**A\* Search:** Time complexity is O(bd+1) , Space complexity is O(bd)

For few example boards, I have tested the above algorithms and found out that Breadth first search is more efficient than depth first search.

But in few cases, where the board has many empty spaces and few walls, depth first search is more efficient whereas breadth first search takes lot of time to find solution as it stores many states in the queue and explores each and every state.

Furthermore, Greedy and A\* search give optimal solution and are more efficient as they use a heuristic and only the best state in the queue at an instant is explored first.

One case where my algorithms fail to come up with a solution is when the board has very less restrictions, my program using breadth first search goes into an infinite loop and it never ends. But for the same board, the other program using Depth first search gives a solution but takes considerable amount of time.