Assignment 1

Name: Diwakar Yalpi

UIN: 01127089

Github link: https://github.com/diwakaryalpi/AI-Assignment1

I have considered N as North, S as South, E as East and W as West while coding in Python3. {'N':[-1,0], 'E':[0,1],'S':[1,0],'W':[0,-1]} to understand which direction the robot is moving. For each state, the robot can move North, South, East, or West. If the robot moves to the location of a box, the box will move one spot in the same direction, if there is space available. All the coding files are saved in the above github link.

Heuristic:

- **Manhattan Distance Heuristic:** I have programmed the Greedy and A Star search algorithms using Manhattan distance between each box that has yet to be stored and the storage nearest to it. Ignored the positions of obstacles in your calculations and assume that many boxes can be stored at one location.
- **My own Heuristic:** In addition to the above, I now consider the following for the algorithm to work better I have used the sum of manhattan distance heuristic and to improve the performance of the algorithm I have considered Distance of box and robot and Number of remaining spaces at any time.

*Modified function = Manhattan(box - storages) + Distance(box - robot) + storagesLeft * 2*

Since storagesLeft has the less value when compared to the other two values in the above function. I have used 2* storagesLeft to provide more weightage.

Computational times and Number of moves:

Algorithm Name	Computational Time	Number of moves
BFS	2.098	45
DFS	1.705	51
Greedy search using manhattan distance	0.478	45
Greedy search using own heuristic	0.284	45
A* search using manhattan distance heuristic	1.796	45
A* search using own heuristic	1.629	45

Greedy Search using own heuristic takes less out of all the algorithms for the given sokoban puzzle i.e., 0.284 and the number of moves to solve the puzzle is 45.

Analysis:

- BFS is slow but it always gives an optimal solution as it checks for the solution level by level. Number of moves is less than DFS.
- DFS is faster when compared to BFS but gives a longer path to reach target because DFS opens a node and continues among its children till it reaches a leaf node. Number of moves is more than BFS.
- Greedy Best first considers heuristic to reach the goal faster when compared to Greedy search using Manhattan distance.
- A* search algorithm is a combination of both BFS and greedy search, so it outputs the optimal solution and is faster when compared to BFS.

I have tested this code on different puzzle inputs from the internet I am able to run almost all the puzzles. For the below puzzle BFS and A* algorithms take forever to find the solution for the input puzzles. This may be due to the number of empty storage spaces, where most of the time the robot tries to move away from the blocks when there are many moves.

```
0000000
0     0
0     0
0S p  0
0S BB 0
0SBB  0
0S0  R0
0000000
```

References:

- http://sokobano.de/wiki/index.php?title=Solver
- https://github.com/KnightofLuna/sokoban-solver
- https://worksheets.codalab.org/worksheets/0x2412ae8944eb449db74ce9bc0b9463fe/
- https://stackoverflow.com/questions/4237462/sokoban-solver-tips
- https://github.com/gabrielarpino/Sokoban-Solver-AI/blob/master/solution.py