CS480/580 Introduction to Artificial Intelligence (Fall, 2017)
Assignment 1

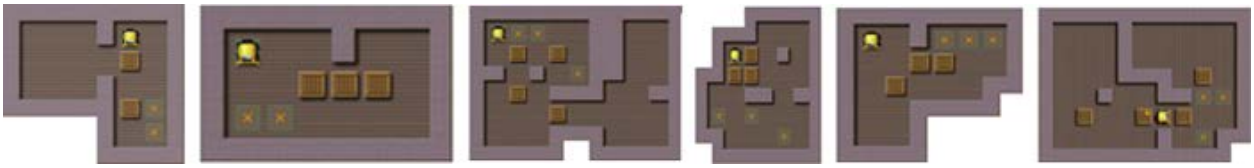
Due: Thursday, Sept. 28, 2017
Total Points: 100

Sokoban Puzzle

In this assignment, you are asked to implement a working solver for the Sokoban puzzle game. Sokoban is a puzzle game where a robot must push boxes into storage spaces. Here are the rules:

1. One box can be moved at a time.
2. Boxes can only be pushed by a robot and not pulled.
3. Neither the robot nor the box can pass through obstacles, walls, or other boxes.
4. A robot cannot push more than one box.
5. The goal is achieved when all boxes are in their storage spots.

Sokoban can be played online at https://www.sokobanonline.com/. Try to play a few games to familiarize yourself with the rules and objectives.



Setup of the puzzle

The puzzle is played on a board that is a grid board with N squares in the x-dimension and M squares in the y-dimension. Each state contains the x and y coordinates for the robot, the boxes, the storage points, and the obstacles.

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.

Each movement is of equal cost. Whether or not a robot is pushing an box does not matter for the cost.

Each state has a standard format. O for obstacle and wall, S for storage, B for block, and R for robot. For example, the leftmost puzzle can be represented as

```
00000000
0    0R 0
0     B 0
0    0  0
00000BS0
     0 S0
     0000
```

Tasks:

1. Your program should be able to process the above format. We will supply several puzzles for you to test. Your program must be able to parse several additional test cases to get full marks.

2. You need to write programs to implement breadth-first search and depth-first search to generate solutions for a given Sokoban puzzle.

3. Implement a Manhattan distance heuristic. This heuristic will be used to estimate how many moves a current state is from the goal state. Your implementation should calculate the sum of Manhattan distances between each box that has yet to be stored and the storage nearest to it. Ignore the positions of obstacles in your calculations and assume that many boxes can be stored at one location.

4. Implement a non-trivial heuristic for Sokoban that improves on the Manhattan distance heuristic. Explain your heuristic in your comments in under 500 words.

5. Implement programs using greedy-best search and A* search algorithms to generate the solution for a given Sokoban puzzle by using the heuristic you proposed.

Hinds: Functions you probably need to have:

Initstate() – initialize the Sokoban state by reading in the state file.

Successor() – generate the legal successor states.

Heuristic() – the heuristic function you implemented.

What to Hand in

1. Well documented codes implementing breadth first search, depth first search, greedy search, and A* search. A README file should provide instructions on how to compile and execute the code.
2. Provide the solutions generated by your programs using BFS, DFS, Greedy search and A* search.
3. Compare the computational times and results of the BFS, DFS, Greedy Search, and A* search algorithms. Analyze your results, for example, try to find out when your algorithms fail to come up with a solution.

Please send your programs and documents to dmfeng8898@gmail.com before the assignment due date.