CS480/580 Introduction to Artificial Intelligence

## Assignment 1

Total Points: 100

Due Date: 9/28/2023

**Pukoban Puzzle**

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

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

While there is little information about Pukoban online, Sokoban can be played online at <https://www.sokobanonline.com/>.

Try to play a few games to familiarize yourself with the rules and objectives in Sokoban. The difference between Sokoban and Pukoban is Sokoban only allows push and Pukoban allows both push and pull.

**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 pushes or pulls 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 pulling or pushing a 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

OOOOOOOOO

OO OOOOO

OOB OOOOO

O B B O

OROSSSS O

O BS OO

OOBO OOOO

OO OOOO

OOOOOOOOO

**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 Pukoban puzzle.

3. Implement a Manhattan distance heuristic.

<https://xlinux.nist.gov/dads/HTML/manhattanDistance.html>

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 Pukoban 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 Pukoban puzzle by using the heuristic you proposed.

Programming Hints: Functions you probably need to have:

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

Successor() – generate the legal successor states.

Heuristic() – the heuristic function you implemented.

6. Benchmark your programs on the following three Pukoban puzzles. Compare the solution length and computational time.

1. Tiny

OOOOOO

O BR O

O BO O

OO O

OOSS O

OOOOOO

2. Medium

OOOOOOO

OS SO

O BBB O

O BRB O

OSBBBSO

OSS SSO

OOOOOOO

3. Large

OOOOOOOOO

OO OOOOO

OOB OOOOO

O B B O

OROSSSS O

O BS OO

OOBO OOOO

OO OOOO

OOOOOOOOO

**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.
4. Please send the program and the analysis report via assignment submission in canvas before the assignment due date.

Hints:

1. Good data structure representation can make your life easy.
2. You may want to start from a very simple Pukoban problem.