Assignment 1:
Solving Ballsort Puzzle using Search

Due Sunday, 31 October, 11pm

Ballsort puzzle is a game where the goal is to collect the balls of the same color in one bottle by moving a ball from a bottle to another at a time. Such moves cannot occur if the color of the ball being moved is not of the same the color as the topmost ball of the destination bottle. There can be at most 4 balls in each bottle. This game is available at

`http://crazygames.com/game/bubble-sorting`.

Please implement in Python

- a Uniform Cost Search (UCS) algorithm, and
- an A* search algorithm

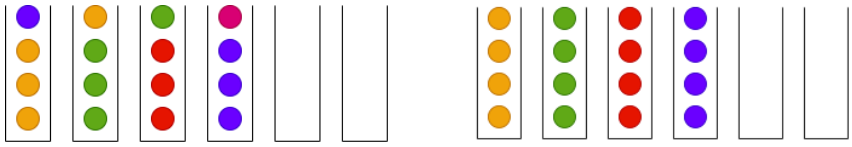
to solve the following version of this puzzle.

Input of the problem The input is represented by two numbers: f, which represents the number of full bottles, e, which represents the number of empty bottles and an $f \times 4$ matrix which represents the colors of balls in corresponding bottles.

For instance, the input below represents the input depicted in Figure 1(a) where the first line contains f and e, respectively. The rest of the lines represent the $f \times 4$ matrix (i.e., initial setting of the balls).

4 2
2 2 2 1
3 3 3 2
4 4 4 3
1 1 1 4

Output of the problem A shortest sequence of legal states and moves that sort the balls. For instance, Figure 1(b) shows a goal state where all the balls are sorted.



(a) (b)

Figure 1: An example for Ballsort, with its initial state (a) and goal state (b).

What to do The assignment consists of five parts:

1. (10 points) Model the puzzle as a search problem: specify the states, successor state function, initial state, goal test, and step cost.
2. (30 points, provided that the first part is completed) Implement in Python a UCS algorithm to solve the puzzle.
3. (10 points, provided that the first part is completed) Extend your search model for A* search: Find a heuristic function and prove that it is admissible.
4. (30 points, provided that the third part is completed) Implement in Python an A* search algorithm to solve the puzzle.
5. (20 points) Compare your UCS and A* codes on some sample puzzle instances. Construct a table that shows, for each instance, time and memory consumption. Discuss the results of these experiments: Are the results surprising or as expected? Please explain.

Submit

- A pdf copy of a description of your solutions (search model and heuristic function), and experimental evaluation (table and discussion).
- Your Python code for each algorithm, with comments describing your solution.
- Several test boards you created for the fifth part of the assignment.

In each one of the deliverables above, please include your name and student id.

Demos You are expected to make a demo of your solution so that we can grade parts 2, 4, and 5. The demos are planned for November 1 and will be scheduled later on.