**Due: week 7, beginning of class**

**Instruction:**

* Submit your source codes and other requires files to Moodle
* Your source codes must contain your StudentId and Name
* Demonstrate your programs in class if asked by the instructor

**Question 1 (See Exercise 1 of Chapter 5) [10 marks]**

Write a Python program (8-puzzle.py) based on A\* search for solving the 8-puzzle problem that implements both the wrong tile heuristic (h1) and Manhattan distance heuristic (h2). Your program must be based on the A\* algorithm we discussed in class. Part of the source code is completed for you in Moodle.

For each case, report the running time for both heuristics as well as whether the solution has been found for the following problems

1. [1,2,3,4,0,5,6,7,8]
2. [5,1,3,4,6,8,7,0,2]
3. [6,4,3,1,8,0,5,2,7]
4. [1,7,4,2,3,8,6,0,5]
5. [4,7,2,0,8,1,3,6,5]

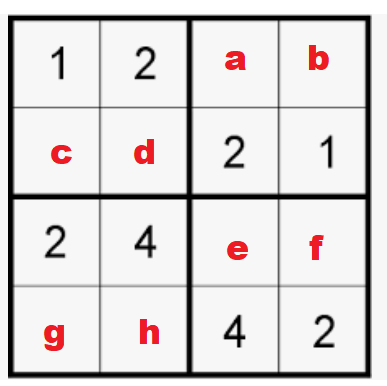
Submit your findings in the following format, as well as your program code to Moodle.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Problem | h1 (number of wrong tiles) | | h2 (Manhattan distance) | |
|  | Success? | Time (s) | Success? | Time (s) |
| 1. | Yes | 0.572 | Yes | 0.738 |
| 2. | Yes | 0.302 | Yes | 0.390 |
| 3. | Yes | 7.816 | Yes | 1.347 |
| 4. | Yes | 175.335 | Yes | 2.427 |
| 5. | Yes | 167.452 | Yes | 2.939 |

**Question 2 (See Exercise 2 of Chapter 5) [10 marks]**

Write a program (suduku.py) to solve the 4\*4 Sukuko problem as shown below. Your program must be based on the CSP algorithm we discussed in class.

(The program Australia.py may be a useful reference)



**Variable:** a,b,c,d,e,f,g,h

**Domain:** {1,2,3,4}

**Constraints:**

* Each smaller squares must contains 1 – 4:
* For example, constraint for the upper left squares can be wrriten as

sorted[1,2,c,d] = [1,2,3,4]

* Each row must contain 1 – 4
* Each column must contain 1 - 4

Note**: for the basic part, you don’t need to solve all 4\*4 suduku problem in general. Just solve the problem instance shown above.** The input can be hard-coded into the program if you wish.

**Bonus** **Part** (5 marks):

Change your program so that it can solve any 4\*4 suduku problem.

1. Ask the user to enter four rows of four numbers, and replace blank tiles with zero.
2. Show solution.

Example:

**Enter first row (enter 0 for space):** 2 0 0 0

**Enter second row (enter 0 for space):** 0 1 3 0

**Enter third row (enter 0 for space):** 3 0 0 1

**Enter fourth row (enter 0 for space):** 0 2 4 0

**Solution Found!**

**2 3 1 4**

**4 1 3 2**

**3 4 2 1**

**1 2 4 3**