CSE 512 – Homework Assignment 1 – Winter 2017

Instructor: Kerstin Voigt

This is a 2-part assignment. Part 1 consists of running a provided program and gathering several performance statistics. Part 2 is a Python programming assignment. Both parts are worth 5 points each. Total number of points: 10.

Part 1: Run a provided implementation of graphsearch for the 8-puzzle problem domain. You should use the Python code given in files

- hw1_graph_search.py
- hw1_puzz8.py

Run function graphsearch with the test cases provided as many times as needed to fill in the Following worksheet:

- hw1_performance_study.xlsx

You may fill the tables by hand and compute the averages with calculator. For greater convenience, using Excel or an equivalent software is recommended.

Part 2: Write the Python code necessary to solve the "Hobbits and Orcs" problem with graphsearch

"Hobbits and Orcs": *Three hobbits and three orcs are sitting on the left bank of a river. Conveniently, there also is a boat on the left bank. All hobbits and orcs are to reach the right river bank by boat. The boat can cross the river in either direction with one or two creatures inside (it cannot go empty). When orcs are in the majority, they will eat the hobbits. Therefore, the possible movements are constrained by the requirement that at any time, the number of orcs never outnumber the number of hobbits. What is the minimum number of boat trips, and what are the respective boat loads, that will ferry the three hobbits and the three orcs from the left bank to the right bank?*

This exercise will consist of your implementing the following functions (with exactly these names):

- `goal_fct(state)`
- `successor_fct(state)`
- `eval_fct(state)`
- `show_state(state)`

In file hw1_graph_search.py make sure to import your "hobbits and orcs" file (analogous to puzz8.py)

Test your code by running function `graphsearch` in all four modes: A-start, BFS, BEST, and DFS.

Your coding the for "hobbits and orcs" will depend on your representation of a problem "state". How will you present the fact that the j_1 hobbits and k_1 orcs are on the left bank, j_2 hobbits and k_2 orcs are on the right, and the boat is on the left bank?

The below is just an example of how one could go about representing states and produce successor states:

```
START = [[3,3],[0,0],'left']
GOAL = [[0,0], [3,3], 'right']
…
def successor_fct(state):
    succs = []
    if state[2] == 'right':
        succs.append(hh_2left(state))
        …
        succs.append(o_2left(state))
    else:
        succs.append(hh_2right(state))
        succs.append(oo_2right(state))
        …
    succs = [s for s in succs if s != None] # remove all None
    return succs

# 2 hobbits go right to left; +2 hobs on left, -2 hobs on right;
# watch for danger on right!

def hh_2left(state):
    state2 = copy.deepcopy(state)
    [[h1,o1],[h2,o2],b] = state2
    if b == 'left' or h2 < 2 or 0 < h2-2 < o2:
        return None # no legal successor exists
    return [[h1+2, o1],[h2-2,o2],'left']

# follow with analogous functions until you cover all possible boat
# loads of 1-2 hobbits, 1-2 orcs, or 1 hobbit and 1 orcs moving
# left to right or right to left.
```

Submit: (1) The completed performance statistics worksheet. (2) Your .py file with the "hobbits and orcs" domain specific functions. (3) Typescript(s) for the runs of graphsearch for hobbits in all 4 modes. You may comment or edit out the "`%d. From OPEN...`" lines to save on paper.

Due Date: Feb 7, 2017, beginning of class.