COMP5600/6600: HW1

Assigned on 09/10; Hand in on 09/20 (Friday)

September 6, 2019

Requirement

Please submit your homework on Canvas before the deadline. If you need to use your delay coupon, you can hand in your homework to my office. Include your name, the student ID number, and how many days late it is (if handed in late) in the headline. If you have any questions, please contact our TA. (*), (**), or (***) indicates the difficulty of each question.

Policy

We apply the late policy explained in syllabus to all homework. Any grading questions must be raised with the TA in two weeks after the homework is returned. The homework must be completed individually. However, you are encouraged to discuss the general algorithms and ideas with classmates in order to help you answer the questions. You are also allowed to share examples that are not on the homework in order to demonstrate how to solve problems. If you work with one or more other people on the general discussion of the assignment questions, please record their names over every question they participated. However, the following behaviors will receive heavy penalties (lose all points and apply the honest policy explained in syllabus)

- explicitly tell somebody else the answers;
- explicitly copy answers or code fragments from anyone or anywhere;
- allow your answers to be copied;
- get code from Web.

1 (**) Advanced Approaches: 10 points

This question asks you to use the advanced search algorithm to solve the 25-Queens problem. You can use any approach (hill climbing, simulated annealing, or their combination you think out of) you learned from the class. The goal is to find a solution which has the least number of conflicting queens. The initial state is

$$(1, 2, 3, \cdots, 24, 25).$$

You can define your "neighbors" and your score function. Please report the following things

- your definition for the score function;
- your definition for neighbors (or how to generate neighbors?);
- the best state you can find; 16
- how many conflicting queens in your best result;
- discuss the approach you used (or designed) and your result.

2 (**) A* Algorithm: 10 points

This question asks you to implement A* algorithm to solve the following puzzle problem. The initial state is given on the left of Table 1, while the goal state is given on the right of Table 1. You can move a "digit" block to the white block next to it per step. The black block is the "black hole". Everything moved into it would be absorbed. In this question, you are asked to use the Manhattan distance as the heuristic function. (In the bonus question, you are asked to design a better heuristic function.) Please report the following things:

2d matrix with tuple defining invalid states

2	3	7	4	5
1	■	11	■	8
6	10		12	15
9	■	14	■	20
13	16	17	18	19

1	2	3	4	5
6	■	7	■	8
9	10		11	12
13	■	14	■	15
16	17	18	19	20

Table 1: Left: initial state. Right: goal state.

- how many steps you used to achieve the goal state;
- how many states you explored before achieve the goal state;
- the 5th and 5th last states (only two states).

You can follow Algorithm 1 for A* algorithm.

Algorithm 1 A* Algorithm

```
1: Put the start state S on the priority queue, called OPEN
2: while OPEN is not empty do
3:    Remove from OPEN and place in CLOSED a state n for which f(n) is the minimum
4:    if n is a goal state then
5:       exit(trace back pointers from n to S)
6:    end if
7:    Expand n, generating all its successors and attach to them pointers back to n.
8:    for each successor n' of n do
9:       if n' is not in OPEN or CLOSED then
10:         Estimate h(n'), g(n') = g(n) + c(n, n'), f(n') = g(n') + h(n')
11:         Place n' on OPEN.
12:      else
13:         if g(n') strictly less than its old g value in OPEN or CLOSED then
14:            Update pointers backward from n' to n
15:            if n' is in OPEN then
16:               Update g(n') in OPEN.
17:            else
18:               Remove n' from CLOSED and place it on OPEN with new g(n')
19:            end if
20:         end if
21:      end if
22:   end for
23: end while
24: Exit with failure.
```

3 (**) A* Algorithm: 10 points

Prove A* algorithm is optimal, given the heuristic function h(s) is admissible.

4 (*) UCS Algorithm: 5 points

Clarify the difference between uniformed cost search algorithm and Dijkstra's algorithm.

5 (***) Open question: 5 points

This is an open question. Consider the same problem as Question 2. You are asked to design a better heuristic function to reduce the number of explored states comparing to the Manhattan distance. Please report the following things:

- the definition of your "better" heuristic function;
- how many steps you used to achieve the goal state;
- how many states you explored before achieve the goal state;
- the 5th and 5th last states (only two states);
- explain why it improves the Manhattan distance.