CS4725/CS6705 - Fall 2019

Assignment # 1

Due: Friday, September 20, by 12:30 pm (beginning of class)

[Note: This assignment is worth 3% of the final course grade.
Future assignments will have a higher weight.]

Submission Instructions:

- Please include your name and student number, the course number and the assignment number on the front page of your assignment.
- Solutions should be handed in on paper, by placing them in the CS4725/CS6705 bin on the E level of Gillin Hall (or by bringing them to class on the due date).

---

1. **(0 marks)** This is not an exercise to be marked, but is included here to encourage you to start looking at some sample code that is provided for the final programming project.

   - Take another look at the preliminary information about the project that is provided in the `Programming project` folder on Desire2Learn.
   - In one of the Faculty of Computer Science labs, download the six Java files provided in the `Programming project` folder and save these files together in their own directory in your account.
   - In that directory, type the command `javac *.java` to compile the code.
   - Type `java PokerSquares | more` to see a demonstration of games being played by the sample random player that has been provided. The demonstration includes:
     - a single game being played by the random player (using the British scoring system)
     - a batch of three games in a row being played by the random player, with a different sequence of cards dealt in each game
     - a tournament, in which two random players compete against each other, using three different scoring systems, with a batch of ten games being played under each scoring system
       (Note that tournament mode allows you to run two or more players against each other in a series of games, where each game will deal exactly the same cards to each player.)
   - Start looking at the code in the six `.java` files, to get an idea of how the implementation works.
   - Ask questions if you are struggling to follow the code.

2. **(13 marks)** This exercise deals with the formal specification of a task as a search problem. You are not being asked to **solve** this problem, but just to set it up as a formal problem so that it **could** be solved using a search algorithm.

   On the very last page of the assignment, I have provided an example of a formal specification for the 8-puzzle discussed in class. This should give you an idea of the level of detail that I am expecting.

   Given a game board with 11 holes, filled initially with 5 red pegs in the 5 leftmost holes and 5 blue pegs in the 5 rightmost holes (see the diagram below), the goal is to reach a configuration in which the red pegs are in the 5 rightmost holes and the blue pegs are in the 5 leftmost holes.

R	R	R	R	R		B	B	B	B	B

   In a single move, one of two things can happen:

   - A peg can be moved to the left or right by one position into the empty hole.
   - A peg can jump over exactly one peg of the opposite colour into the empty hole.

   Again, this question deals with formulating this task as a search problem. [You are not being asked to *solve* the problem, but feel free to find a solution.]

   Suppose that you have chosen to represent a state s as:

   - an 11-element array A, with each element $A[0], A[1], \cdots, A[10]$ being the character "R" (red), the character "B" (blue) or the character "E" (empty)
   - a single integer h ($0 \leq h \leq 10$), storing the current location of the empty hole

   (a) (1 mark) What is the initial state?

   (b) (1 mark) What is the goal test?

   (c) (1 mark) What would you choose as the path cost function?

   (d) (5 marks) For the successor function, suppose that you are going to define only one action with two parameters, $move(x, y)$, indicating that the peg in position x ($0 \leq x \leq 10$) is moved to position y ($0 \leq y \leq 10$).

      i. For a given pair of values x and y, what conditions must be checked in order to determine if $move(x, y)$ is a legal action in the current state $s = (A, h)$?

      ii. Provide pseudocode showing how the new state $s' = (A', h')$ would be calculated from the current state $s = (A, h)$ if the action $move(x, y)$ were performed.

(e) (5 marks) Note that the choice of successor function described in part (d) is not an ideal choice. There are 121 possible combinations of values for x and y, most of which would be illegal in a given state. (In fact, most of those (x, y) combinations – for example, $move(2, 7)$ – would **never** be allowed in **any** state.)

Using our 8-puzzle discussion in class as a guide, describe a different approach for defining the actions available to an agent trying to solve this problem. Provide a list of possible actions. (Give each action a name and explain in one sentence what it does.) For each of your actions, what conditions must hold in order for the action to be legal in the current state $s = (A, h)$? [You do not have to write the pseudocode for calculating the next state.]

3. **(12 marks)** Consider a state space where the initial state is the number 1 and where each state n has three successors (children), the numbers $3n - 1$, $3n$ and $3n + 1$.

(a) Draw the portion of the state space that contains the states numbered 1 to 25.

(b) Suppose that the only goal state is 21. For each of the following search techniques, list the order in which nodes will be selected for expansion and show the contents of the queue/stack after each step.

- **breadth-first search**
- **depth-limited search** with limit 3
- **iterative deepening search**

(Recall our discussion from class in which it was mentioned that breadth-first search terminates as soon as a goal node is **generated**, while the other search algorithms only terminate when a goal node is selected for **expansion**.)

Additional information for Question 2:

As mentioned in Question 2, here is an example of how we might provide a formal specification of the 8-puzzle as a search problem.

- **State representation:**
  - We could represent the state as a two-dimensional array A of integers (with 0 representing the empty cell). For example, the state pictured below could be represented with $A[0][0] = 6$, $A[0][1] = 2$, $A[0][2] = 0$, $A[1][0] = 3$, *etc.*

6	2	
3	4	7
8	1	5

  - For reasons that should become clear below, it might also be useful to keep track of two integers, $row_{empty} = 0$ and $col_{empty} = 2$, to easily access the location of the empty cell.

- **Actions:** $\{U, D, L, R\}$, representing the empty space "moving" up, down, left or right, respectively.

- **Successor function:**
  - For each state $(A, row_{empty}, col_{empty})$, we can check whether each action is legal and specify how the next state $(A', row'_{empty}, col'_{empty})$ would be computed.
  - For example, the D (down) action is legal if and only if $row_{empty} \neq 2$. If D is a legal action, then we can calculate the new state as follows:
    1. (1) Start by making A' an exact copy of A
    2. (2) $A'[row_{empty}][col_{empty}] = A[row_{empty} + 1][col_{empty}]$
    3. (3) $A'[row_{empty} + 1][col_{empty}] = 0$
    4. (4) $row'_{empty} = row_{empty} + 1$
    5. (5) $col'_{empty} = col_{empty}$
  - Similar details would have to be provided for the U, L and R actions.

- **Goal test:** Does the current state match the following?

1	2	3
4	5	6
7	8	

- **Path cost:** Each action has a cost of 1. The path cost is simply the number of actions taken from the initial state to the current state.