Homework 3
Due 9th October. In class. (Solutions will be posted on 9th and no submissions will be considered after that).

Qn. I. This is a variation on Homework 2, Problem 5—the CSP problem. As you may recall, in that problem you could not use the constraint graph representation, because one of the constraints:
`"J2 has to be done either before j3 or before j4"`
was a non-binary constraint. Suppose we change this constraint to read
`"J2 has to be done before j4"`
making it a binary constraint. Assume all other constraints remain same. Answer the following questions:

1.1. Draw a constraint graph representation for the problem
1.2. Is the problem 1-consistent? 2-consistent (arc-consistent?)?
1.3. Enforce arc-consistency (2-consistency) on this constraint graph. *Explain the process.* Show the resulting constraint graph. Is the constraint graph in 1.3 1-consistent? 2-consistent? 3-consistent? 4-consistent? Is it strongly 1-consistent? Strongly 2-consistent? Strongly 3-consistent? Strongly 4-consistent? Is it possible to do backtrackless search on this graph?
1.4. Using the JAVA applet at URL http://www.cs.ubc.ca/labs/lci/CIspace/csp.html specify the CSP problem to the applet (see the instructions in the help menu). Then go into solution-mode and do auto-arc-consistency. See if you get the same answer as you showed in 1.4. Give a screen dump of the original constraint graph and the one after auto-arc-consistency. (You can also use this tool to improve your understanding of the arc-consistency… you can step through the arc-consistency procedure).
1.6. Go back to the status of CSP as it was at 1.1. Suppose you start with the complete assignment
`J1=2,J2=2,J3=3,J4=3`
Starting with this assignment show an iteration of the min-conflicts based hill-climbing procedure for following two cases:

Case 1. Neighborhood defined in terms of *all* assignments that differ from the current assignment in the value of any single variable.

Case 2. Neighborhood defined in terms of just the assignments that differ from the current assignment in one specific randomly chosen variable that takes part in a constraint violation in the current assignment

Qn II. Consider the 8-puzzle problem where the goal-state is (note that this is different from the goal state in project 1)

1	2	3
4	0	5
6	7	8

Suppose we choose the top row [1,2,3] as the fringe pattern.

Consider the following initial state—call it S0

3	4	5
1	0	2
7	6	8

A. What is the heuristic value associated with S0 if we use

A.1 Misplaced tile heuristic

A.2 Manhattan distance heuristic

A.3 Pattern Database heuristic—assuming top row as the fringe pattern. You will have to compute the value for A.3 –since the puzzle is small enough you should be able to do it by hand (kind of good exercise playing the sliding puzzle games ☺

A.4 Compare the heuristic values. How close is the PDB value to the perfect heuristic value in this case?

(*It is okay if you use your project 1 code to get answers for this problem. Note however that the goal-test function needs to be changed so it only cares about the fringe tiles).*

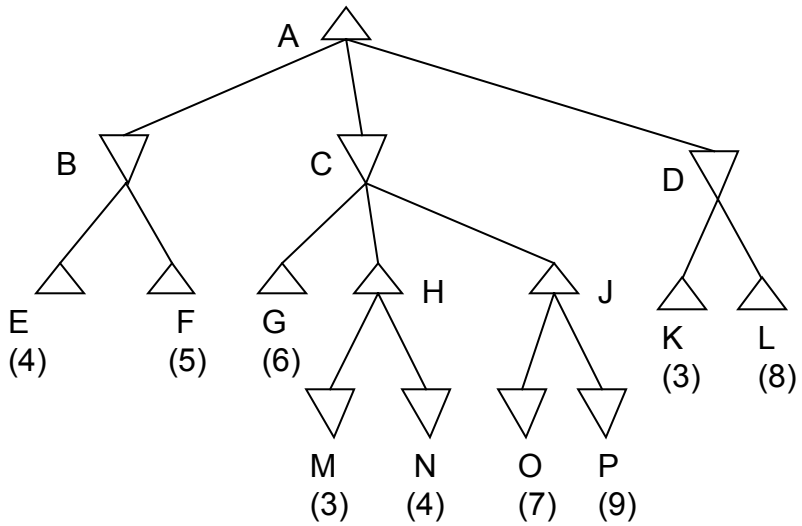
B Assuming that we use top row as the fringe pattern, how many distinct entries will you need in the pattern database? What fraction of the total number of states in 8-puzzle is this database size?

Qn III. [From midterm last year] Here is an extremely silly two-person game—it is even sillier than tic-tac-toe (if such a thing is possible). In this game, there are some number of stacks of coins on the table (each stack contains possibly different number of coins). When it is a player's turn, he/she needs to choose a stack of coins and split it into *two* ***unequal*** sized stacks (e.g., if you pick a stack of 7 coins, you can split it into two stacks of 5 and 2 or two stacks of 3 and 4 or two stacks of 6 and 1). When the game reaches a stage such that the player whose turn it is now cannot pick a stack from the table and split it this way, the player is considered to have **lost** the game (and the other wins).

Suppose **that initially there is a stack of 5 coins** on the table. You (MAX) is playing first. **PROVE** using min-max strategy, MAX will ALWAYS win this game, whatever the opponent (MIN) does. (You will have to explain what is the representation that you are using for the game-tree state, and generate the full min-max tree which is possible since it is such a small game).

The coin game min-max

Qn IV. [From midterm 98] Consider the following game tree, where the numbers in the parenthesis correspond to what the evaluation function would have returned if we subjected that node to the evaluation function. Assume that the upward arrows correspond to board positions where it is Rao's turn to make the move, and the downward arrows correspond to board positions where it is Tom's turn to make the move.



Part A. If Rao uses min-max analysis, what move would he wind up choosing? What is the value of that move?

Part B. If Rao uses Alpha-Beta pruning to improve his Min-Max search, what nodes would he avoid evaluating? Please show your work on the figure. Assume left-to right consideration of children nodes.

Part C. Assuming left-to-right consideration of children nodes, is there a (possibly different) assignment of evaluation function values to the nodes in the tree such that some of the nodes under **B** would have been pruned?
