CSE 573 Homework 1
Due Monday Oct 16th In Class

Please type or clearly write your answers to the problems below. Include your full name and student id at the top of each page. You may talk to other students in the class about the problems, but all final answers must be your own.

Search (30 pts)

1) Consider RDFS, a randomized version of DFS. RDFS performs identically to DFS except that each time RDFS expands a node, it chooses a new random ordering for which to consider that node's successors.

   (a) (2 pts) Is RDFS complete? Optimal? You can assume there are no cycles in the search space.

   (b) (2 pts) Consider an iterative deepening version of RDFS. How will the randomization affect the number of iterations required by the algorithm to reach the goal compared to non-randomized IDFS?

   (c) (2 pts) Why might one want to randomize DFS?

2) Searching with Negative Costs

   (a) (2 pts) R&N Problem 3.17a: Suppose that actions can have arbitrarily large negative costs; explain why this possibility would force any optimal search algorithm to explore the entire state space.

   (b) (2 pts) R&N Problem 3.17b: Does it help if we insist that step costs must be greater than or equal to some negative constant c? Consider both trees and graphs.

   (c) (2 pts) R&N Problem 3.17c: Suppose that there is a set of operators that form a loop, so that executing the set in some order results in no net change to the state. If all of these operators have negative cost, what does this imply about the optimal behavior for an agent in such an environment?

   (d) (2 pts) Provide an instance of a graph that shows how allowing heuristics to take on negative values can temporarily mislead A*. Be sure to show both g(n) and h(n).

   (e) (3 pts) Is A* still complete for graphs having negative h(n) values? Explain.

   (f) (3 pts) Is A* still optimally efficient for graphs having negative h(n) values? Explain.

3) Backpackers and Thieves

   Three tourists backpacking in the wilderness encounter three thieves on the left side of a river, along with a boat that can hold at most two people. We would like to find a way to transfer all six people to the right side of the river, without ever leaving a group of backpackers in any particular place outnumbered by thieves.

(a) (2 pts) Formulate the scenario as a search problem by defining the state space, goal, operators and path cost.

(b) (3 pts) Which of the following heuristics are admissible for performing A* search?

i) $h(n)$ = Number of people on the left bank
ii) $h(n)$ = Number of people on the left bank – 1
iii) $h(n)$ = h*(n) where $h^*(n)$ is the true remaining path cost
iv) $h(n)$ = 2
v) $h(n)$ = min(2, $h^*(n)$)
vi) $h(n)$ = max(2, $h^*(n)$)

(c) (5 pts) R&N Problem 4.7: Prove that if a heuristic is monotonic, then it is admissible. Assume that h(*goal*) = 0.

Constraint Satisfaction (15 pts)

4) Sudoku is a popular logic-based placement puzzle. The aim of the puzzle is to enter the digits 1 through 9 in each cell of a 9×9 grid made up of 3×3 regions so that each row, column, and region contains exactly one instance of each digit. A set of clues, or "givens", constrain the puzzle such that there is only one way to correctly fill in the remainder. An example of an unsolved Sudoku puzzle is given in Figure 1.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Figure 1: A Sudoku Puzzle

(a) (4 pts) Formally define Sudoku as a CSP. List the variables, their domains, and the set of ordered constraints.

(b) (6 pts) For a general Sudoku puzzle of size N x N (regions are √N x √N)
i) How many constraint arcs are there?
ii) Using forward checking and minimum remaining values heuristic, what determines the depth of the search tree? The branching factor at each level?

(c) (2 pts) One way we can enforce consistency among constraints is to take into account dependencies between rows, columns and regions of the game board. Consider the interaction between rows and regions as depicted in Figure 2. How might you simultaneously model original constraints between the row R and the region S to exploit interactions between them?

(d) (3 pts) Outline an approach that uses this framework to automatically generate legitimate Sudoku puzzles.

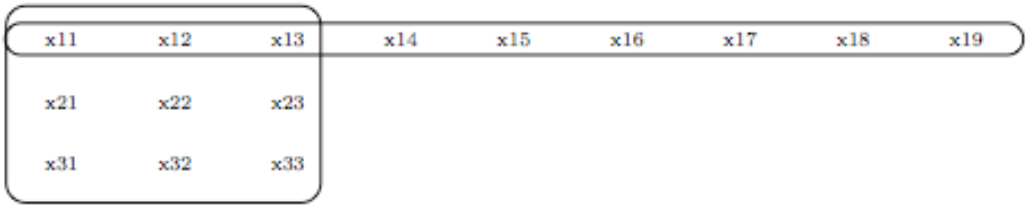


Figure 2: Row/Region Interaction

Adversarial Search (20 pts)

5) Consider an alternating two-player game such as Othello/Reversi, where the player with the most number of game pieces remaining on the board wins. (For more details on the rules of the game, see http://en.wikipedia.org/wiki/Reversi).

Let A and B be two evaluation functions defined as follows.

- A returns -1 if MIN has more pieces on the board, 0 if the both players have the same number, and 1 otherwise.
- B returns the difference between the number of pieces MAX has on the board and the number MIN has.

(a) (3 pts) Assume the game tree is small enough for us to be completely explored. If a player employs a MINIMAX search strategy with no pruning, is A or B a better strategy? Why?

(b) (3 pts) Now assume our player uses alpha-beta search. How does the runtime compare for A and B?

(c) (3 pts) Next, suppose we do not wish to explore the entire game tree and apply depth-limited cutoff strategy. Do you expect strategy A or B to have a better ability to win? Why?

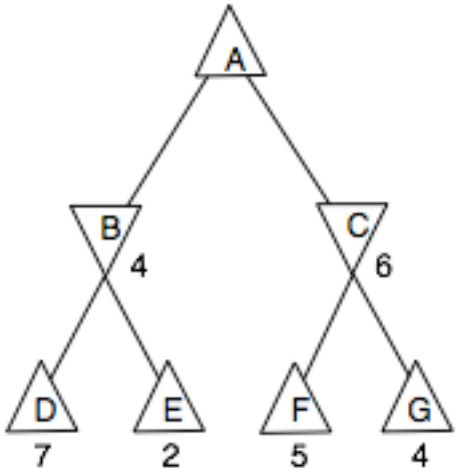
6) Best-First Minimax

While alpha-beta pruning, can reduce the number of game states minimax search explores, it does not eliminate the exponential nature of the search space. *Best-First Minimax* is a simple game-tree search algorithm; the basic idea is to always explore further the current line of play. As before, when given a partially expanded game tree with fixed utility values, the value of an interior MAX node is the maximum of its successors, and the value of an interior MIN node is the minimum of its successors.

The *principal variation* is a path from the root to a leaf in which every node has the same value. In best-first minimax, this leaf, called the *principal leaf*, determines the minimax

value of the root. The algorithm always expands next the current principal leaf node, since it has the greatest effect on the minimax value of the root.

(a) (3 pts) Trace best-first minimax through the first three expansions of the tree below, providing minimax values at each node at each step. Each node has been annotated with its utility.



(b) (2 pt) Assuming the simplest implementation of best-first minimax which maintains the search tree in memory, what is the space complexity of the algorithm? Explain.

(c) (4 pts) A more clever implementation of best-first minimax makes it possible to run in linear space. As with alpha-beta pruning, associated with each node on the principal variation is a lower bound α, and an upper bound, β. The root is bounded by $-\infty$ and ∞. A node remains on the principal variation as long as its backed-up minimax value stays within these bounds. Give, in psuedocode, the routine BF-MAX(*node,* α, β). You can assume that a corresponding routine BF-MIN(*node,* α, β) exists.

(d) (2 pts) What is the key difference between the decision making properties of alpha-beta and best-first minimax?