COMP3211 Fundamentals of AI
Fall 2020 Midterm
21/10/2020
Time Limit: 80 Minutes

Name: ____________________

Stu ID: ____________________

Instructions:

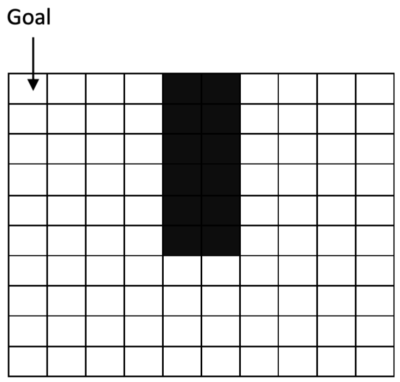
1. This exam contains 11 pages (including this cover page) and 9 questions.
2. Upload your answer to canvas as a pdf file (preferred) or a zip archive of image files (problem1.jpeg, problem2.jpeg,...) named by your student ID.
3. Observe the honor code - it's open book but no discussions or outside help are allowed.
4. Sign on the class zoom and turn on the video but mute the audio.

Grade Table (for teacher use only)

Question	Points	Score
1	8	
2	5	
3	5	
4	10	
5	5	
6	10	
7	10	
8	10	
9	10	
Total:	73	

Question 1 [8 points]

(Simple Agents) Consider the robot with the same specification as our boundary-following robot discussed in class: eight sensors $s_1, ..., s_8$ and four actions (*North*, *South*, *East*, and *West*). Now suppose the environment is a 10x10 grid with an obstacle inside as shown below:



Suppose we want the robot to go to the **top left** corner, wherever its initial position is. Does it exist a reactive agent to achieve the task? If not, explain why not. If yes, give a reactive production system to achieve the task - noted that your production system is not allowed to call the boundary-following production system in the lecture note.

Solution: There is no reactive agent to achieve this task as the goal position and the top right free cell next to the obstacle are isomorphic as far as the agent's sensors are concerned.

Question 2 [5 points]

(Perceptron) Which Boolean function does the following TLU implement? The TLU has five inputs. Each input is either 0 or 1. Its weight vector is $(1, 3, 1, 2, 0.5)$, and the threshold is 5.

Solution: x_2 needs to be 1. either x_4 is 1 or both x_1 and x_3 are 1: $x_2(x_4 + x_1x_3)$.

Question 3 [5 points]

(Error-Correction Procedure) Given the following training set:

ID	x_1	x_2	Label
1	1	1	0
2	1	0	1
3	0	1	1
4	0	0	0

Run the error-correction procedure for 2 iterations (i.e. going through all examples for two times). Use the learning rate 0.5, and the initial weight vector $(0, 0, 0)$. Notice that the last weight is the negative of the threshold as we add a third input that is set to 1, and fix the threshold of the training TLU to be 0 - see the lecture note just before the description of the procedure.

Will the procedure converge for this training set?

Solution: Error-correction procedure:

Iteration	$w_{old} = (w_1, w_2, w_3)$	$X = (x_1, x_2, x_3)$	d	sum	f	$W_{new} = (w_1, w_2, w_3)$
1	0,0,0	1,1,1	0	0	1	-0.5,-0.5,-0.5
2	-0.5,-0.5,-0.5	1,0,1	1	-1	0	0,-0.5,0
3	0,-0.5,0	0,1,1	1	-0.5	0	0,0,0.5
4	0,0,0.5	0,0,1	0	0.5	1	0,0,0
5	0,0,0	1,1,1	0	0	1	-0.5,-0.5,-0.5
6	-0.5,-0.5,-0.5	1,0,1	1	-1	0	0,-0.5,0
7	0,-0.5,0	0,1,1	1	-0.5	0	0,0,0.5
8	0,0,0.5	0,0,1	0	0.5	1	0,0,0

No it will not converge as the dataset is not linearly separable.

Question 4 [10 points]

The cryptarithmetic puzzles are about finding mappings from letters to digits so that equations like the following hold:

```
   SEEN
 + SOME
--------
  BONES
```

Two further constraints are that each letter should be mapped to a different digit, and there should be no leading zeros. This puzzle can be naturally formulated as a constraint satisfaction (assignment) problem as following:

- Variables: B, S, O, E, N, M
- Domains: same for all variables 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Constraints:
  1. All variables are distinct.
  2. S and B cannot be equal to 0 (no leading zero)
  3. SEEN + SOME = BONES:
     a. B = (S + S) // 10 or B = (S + S + 1) // 10 (where // represents integer division)

b. O = (S + S) mod 10 or O = (S + S + 1) mod 10 (where mod represents modulus)
c. N = (E + O) mod 10 or N = (E + O + 1) mod 10
d. S = (N + E) mod 10
e. E = (E + M) mod 10 or E = (E + M + 1) mod 10

Now solve this puzzle using the constructive method (depth-first search with constraint propagation): start your search with the state $\mathcal{S} = \{B = 1, S = 5, O = 0\}$, and expand it using the following order: for the remaining variables, use the order $[E, N, M]$, and for the values, use the order $[9, 8, 7, 6, 5, 4, 3, 2, 1, 0]$. So the next assignment to consider is $E = 9$ unless this value is ruled out by the given starting state $\mathcal{S}$. Try to do as much propagation as you can.

Solution:

- $E = 9$: $N = 9$ or $N = 0$ by 3(c), both violates (1), deadend.
- $E = 8$: $N = 9$ by (1) and 3(c), violates 3(d).
- $E = 7$: $N = 8$ by (1) and 3(c), $M = 9$ (1 and 3(e)). A solution.

Question 5 [5 points]

Consider the following SAT problem:

$$\begin{aligned} &x1 + x4 \\ &x1 + x3' + x8' \\ &x1 + x8 + x12 \\ &x2 + x11 \\ &x7' + x3' + x9 \\ &x7' + x8 + x9' \\ &x7 + x8 + x10' \\ &x7 + x10 + x12' \end{aligned}$$

where X' is $\overline{X}$, the negation of X. Now consider the local search with min-conflict heuristic called GSAT (page 67 of the lecture note on search). Suppose the initial assignment is

$x1 = 0, x2 = 0, x3 = 0, x4 = 0, x5 = 0, x6 = 0, x7 = 0, x8 = 0, x9 = 0, x10 = 0, x11 = 0, x12 = 0$

Update this one by computing a best successor to it, breaking ties randomly.

Solution:

flips	$x1$	$x2$	$x3$	$x4$	$x5$	$x6$	$x7$	$x8$	$x9$	$x10$	$x11$	$x12$
# clauses satisfied	7	6	5	6	5	5	5	6	5	4	6	5

Choose to flip $x1$.

Question 6 [10 points]

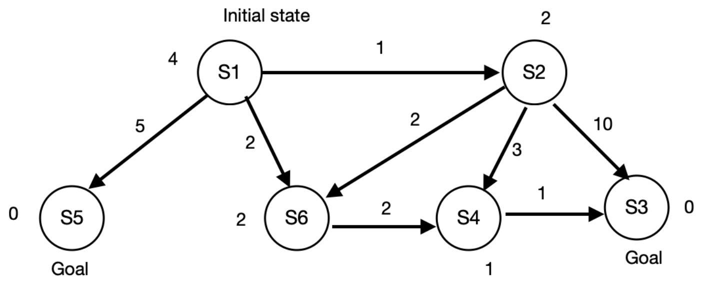
Consider modifying our A* search by tree algorithm (page 20) by checking if a node is a goal as soon as it is generated:

1. create a search tree T, consisting solely of the start node, n_0. *If it is a goal node, exit successfully with the corresponding solution.* Otherwise, put n_0 on a list called $OPEN$.
2. If $OPEN$ is empty, then exit with failure.
3. Select the first node on $OPEN$, and remove it from $OPEN$. Call this node n.
4. Expand node n, generating the set M of its successors that are not already ancestors of n in G. *If any of them is a goal node, exit successfully with the corresponding solution.* Otherwise, install these members of M as children of n in G, and add them to $OPEN$.
5. reorder the list $OPEN$ in order of increasing $g(n) + h(n)$ values. (Ties are resolved in favor of the deepest node in the search tree.)
6. go to step 2.

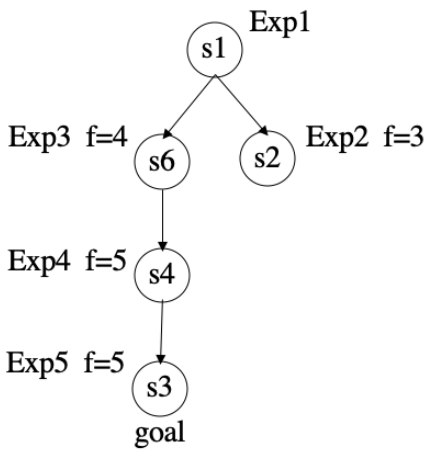
Give an example to illustrate the difference between this modification and the original one in the lecture note (page 20).

Solution: The modified algorithm may return a non-optimal solution. Example: three states, S_1 (initial state), S_2 and S_3 (goal state). Action A goes from S_1 to S_3 and costs 3, B goes from S_1 to S_2 and costs 1, and C goes from S_2 to S_3 and costs 1. The modified algorithm will return $[A]$ as the solution but the optimal one is $[B, C]$.

Question 7 [10 points]
Consider the following search problem, where the numbers on the edges are costs of the corresponding actions, and the numbers next to the states are their heuristic values. Apply A* search by tree on this problem and give the solution returned by it. You can answer this question by drawing a search tree with the sequence of nodes selected for expansion clearly indicated. You can use any tie-breaking rule.



Solution: The answer is not unique. For example, use ”Prefer newly generated nodes” as tie-breaking rule.



Question 8 [10 points]
Consider the following game tree: Perform a left-to-right alpha-beta pruning. Which nodes are pruned? Notice that Left-to-right means that whenever a node is expanded, it’s children are considered in the order from left to right.

Solution: M and E.

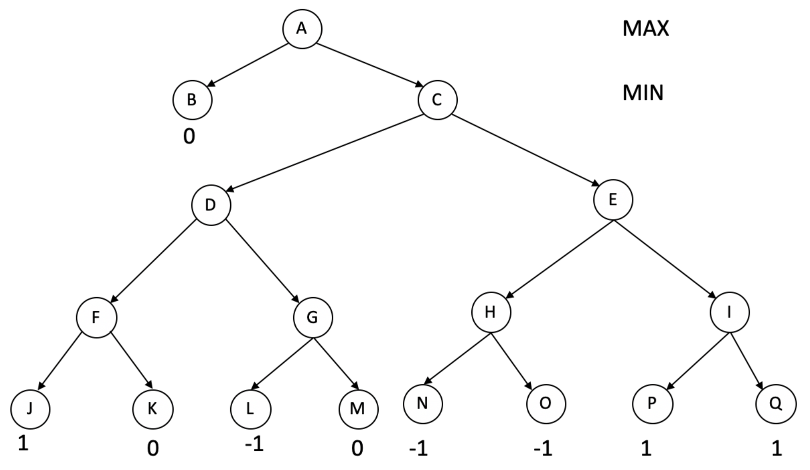


Figure 1: A minimax search tree

Question 9 [10 points]

Consider making N Queens problem into a zero-sum game between two players called **Odd** and **Even**: **Odd** goes first and then **Even**. Each is only allowed to mark a cell in the left most unoccupied column that will not have a conflict with the existing queens on the board. Thus **Odd** goes first and puts a queen in column 1. Then **Even** puts a queen in column 2 that will not have a conflict with the queen that **Odd** has placed in column 1. After that, it's **Odd**'s turn and she finds a non-conflicting cell to put the queen in column 3, and so on. The game ends when either all the queens are on the board or it cannot continue as no more legal moves are possible. In the first case, we have a solution to the queen's problem and both players get 0. In the latter case, the player who has just made the move loses the game and gets -1, and the other player wins the game and gets 1. An example game for 4-queens problem is below:

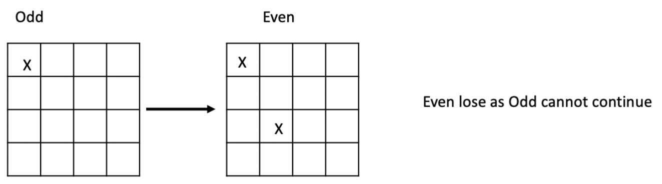


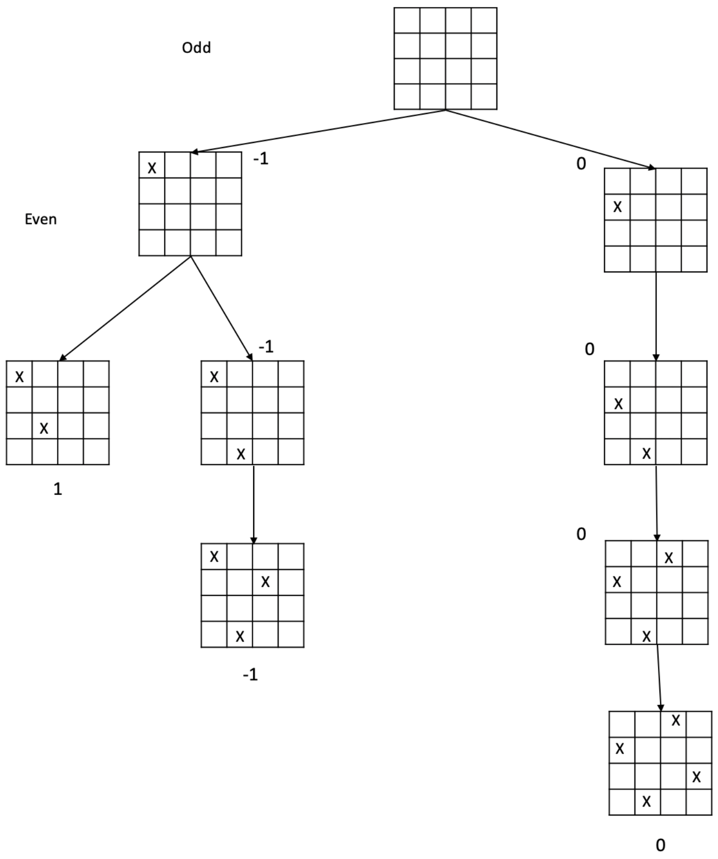
Figure 2: A Game on 4 Queens

The idea of the game is to encourage players to play to the end, and penalize the one that causes the game to be aborted prematurely.

- Apply the minimax algorithm with perfect decision (page 33 of lecture note) on the 4-queens problem to decide the best opening move for **Odd** and then the best response by **Even**. You can draw the minimax tree on your own or using the empty boards on the next page. You can also use a text notation with $move(i)$ to denote the move of the player to put a queen in the ith row. For example, the game in Figure 2 can be encoded as the move sequence $[move(1), move(3)]$.
- Do you think the minimax algorithm can solve the N-queen problem, in the sense that if the N-queen problem has a solution, and both players use the minimax algorithm to play it, then the game will end with a solution to the N-queen problem? Please explain your answer in the best way that you can.

Solution:

- Taking into the consideration of symmetric moves, the following is the minimax search tree for the opening move by **Odd**:



So **Odd** takes $move(2)$. Afterward, there is only one legal move by **Even**: $move(4)$.

- It works for 4-queens by accident. It will not work in general as the players will try to force their opponent to make moves that cannot be continued.