cmput 355 2025 practice questions 5

Explain each answer, show all work.

1. For $n \times n$ hex for each $n \leq 4$, show all winning 1st black moves. Repeat for white.

2. Using negamax solver `hex/hex_simple.py` on your favorite machine, for `x`-to-play and then for `o`-to-play, solve 3×3, 3×4 (3 rows, 4 columns), 4×4 hex: you will need to change ROWS and/or COLS. **If this takes more than 1 minute, just kill the process and ignore that part of the question.** In each case, give

   a) who wins
   b) a winning move if 1st-player wins
   c) the number of calls made in solving
   d) the runtime, if it is given.

3. In `hex_simple.py`, uncomment the lines that assign a better move order to CELLS and repeat question 2.

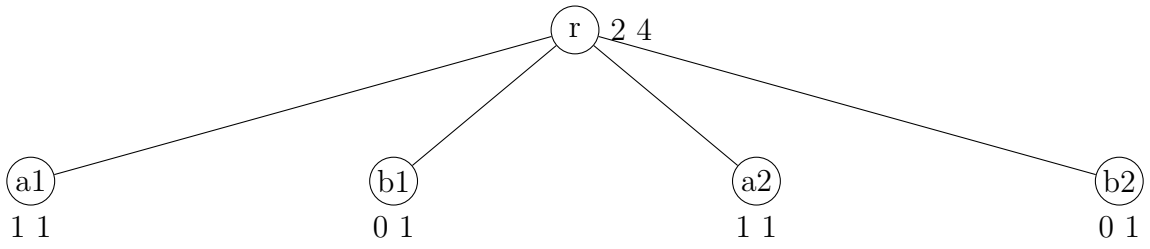
4. For the 4×3 hex board, give a CELLS move order that reduces the number of calls made by `hex_simple.py` both for solving `x` and for solving `o`.

5. Repeat question 2 for Monte Carlo Tree Search player `mcts/main.py`. Also:

   e) Why do different executions sometimes give different moves?

   f) How can this player solve 4×4 hex faster than minimax?

6. Below is the Monte Carlo Tree Search (MCTS) tree after the simulations (iterations) 1-4 from call *mcts x* in `mcts/main.py`, and also the output from simulation 5. Show the tree after simulation 5: explain each change.




```
trv_xpnd bu * 1.4 .4 1.4 .4 5
    xpnd_nd * 5 > 6
    xpnd_nd * 5 > 9
    xpnd_nd * 5 > 10
  sim 5. * 5 10 roll 6 9 parent win
```

7. Repeat the previous question for each simulation from 5 to 13: after simulation s, if you are given the mcts tree at that point and the output from simulation $s+1$ show the updated tree: explain each change.

8. a) Which of these features does `hex_simple.py` use? For each unused feature: would it significantly improve runtime?

   prune isomorphic positions transposition table alphabeta search

   b) In `hex_simple.py`, what algorithm does `has_win()` use to determine whether a player has won the game?

9. In `mcts1.py`, in `def best_uct(...`, change line `uct = mean_res + ...` to `uct = mean_res`. a) Explain how this changes the mcts algorithm. b) In the previous question, instead of output `trv_xpnd bu * 1.4 .4 1.4 .4 5`, what would the output be? c) With `MCTS_TIME = 1`, how does this change modify algorithm behaviour on 2×2 and 3×3 boards? On the 4×4 board?

10. `https://webdocs.cs.ualberta.ca/~hayward/355/mcts25.pdf` shows output from an execution of `mcts/main.py` (`mcts x`) for the empty 2×2 hex board.

    a) After search ended, what criterion was used to pick the move to play?

    b) Is the move from a) provably winning, probably winning, provably losing, probably losing, or none of these?

    c) Repeat the previous two questions if search had ended after simulation 10.

    d) Find a simulation in which the leaf selection was made by picking a most-likely-winning child at each step, or explain why there is none.

    e) Find a simulation in which the leaf selection was made by not always picking a most-likely-winning child, or explain why there is none.

11. a) Run `python3 main.py` in `mcts` on the 4×4 board. What move does it make? Is this a winning move? b) In `mcts/mcts1.py`, replace this line

    ```
    uct = mean_res+(self.c*sqrt(log(self.root_node.sims)/child.sims))
    ```

    with line `uct = mean_res` and repeat a). How do your results compare with a)?

12. Here is one way that the mcts algorithm in `mcts/` could be improved: at a node, whenever a true win (i.e. minimax win, so in hex once a player has joined their two sides) is detected, prune their siblings.

    a) For any two player game, what else could be done at this point?

    b) For hex, what else could be done at this point?

13. For each of the winning 3×3 hex 1st-player strategies below:

    a) is this a 1st-player or 2nd-player strategy? and for black or for white or for both?

    b) explain the strategy in words

    c) Draw this strategy as an and-or tree.

    $b2 \wedge (b1 \vee c1) \wedge (a3 \vee b3)$

    $a3 \wedge (a2 \wedge (a1 \vee b1) \ \vee \ c1 \wedge (b2 \vee c2 \wedge (b3 \vee c3)))$

    $a2 \wedge (a1 \vee b1) \wedge (a3 \ \vee \ c2 \wedge (b2 \vee c1) \wedge (b3 \vee c3))$

    d) Explain why this strategy notation is called opponent-oblivious.

14. See slide 32 in

    `https://webdocs.cs.ualberta.ca/~hayward/talks/hex.someques.pdf`

    a) Explain why the black stone is safely connected to the top. Write this safe connection in logical form using the notation from the previous question.

    b) Explain why the black stone is safely connected to the bottom. Write this safe connection in logical form using the notation from the previous question.

    c) Explain why the top is safely connected to the bottom. Write this safe connection in logical form using the notation from the previous question.

15. See slide 45 in

    `https://webdocs.cs.ualberta.ca/~hayward/talks/hex.someques.pdf`

    a) This position is white-to-play. If white's move is not in one of the 9 dark cells, explain how black can win.

    b) If white's move is not at the 1 dark cell in slide 48, explain how black can win.

    c) For a given hex position and given player-to-move, explain what a *mustplay region* is.

16. Go to `https://webdocs.cs.ualberta.ca/~hayward/355/asn/hexviz/`. Click *back*, click *4 by 4*, click *options*, click *rule decomp*, click *brain*, click *done*. You will see the 4×4 hex board with 1 blue cell, 1 brown cell, 3 pink cells, 5 green cells. What does the diagram show? What do the colored cell sets represent? Use the terms *safe connection* and *weak connection* in your answer.