355 fall 2022 assignment 6

For this assignment, post CLARIFYING QUESTIONS ONLY on eclass. Posting suggestions or your answers or hints on eclass or any other site – e.g. a discord server or anywhere else – is plagiarism.

You can work on this assignment in groups of up to 5: within your group, you can discuss any questions, but you cannot copy answers. Each student must submit their own assignment. Discussing or copying with any student outside your group is plagiarism.

We might ask you later to explain your answers: if you are unable to do so, we might deduct some or all marks and report this to the faculty of science.

For this assigment, each student's secret number is the 4th and 5th integer of their student id, interpreted as a 2-digit number. E.g. if your id is `***91**`, then your secret number is 91. Some questions ask you for m, your secret number mod 3. E.g. if your secret number is 91, your m is 1.

Submit each answer on eclass.

1. **If you do not answer this question we will not mark the assignment and your assignment score will be 0.**

   (a) In your own words, state that you accept the plagiarism policy above.

   (b) Give the names and ccids of all members of your discussion group (including yourself). Explain how you worked together: e.g. discussed every question, discussed only questions 1 and 3 with group members X and Z, etc.

2.

	P0	P1	P2
pile A	0101101	0101111	0100101
pile B	1010010	1010000	1011010
pile C	1100111	1100101	1101111
pile D	1110100	1110110	1111100

Each problem P_m above shows in binary the number of stones in four nim piles. For your problem P_m, for each pile, give the number of stones that you would have to change that pile to in order for the new nim position to be losing, and explain whether that is possible (if the number goes down, it is possible, otherwise it's not a legal nim move).

Answer like this:

pile	new number of stones
A	27
B	32
C	411
D	999

3. If your m is 0, then t is 4 and k is 5. If your m is 1, then t is 4 and k is 4. If your m is 2, then t is 6 and k is 3. Consider a nim position with t piles: the number of stones in each pile is a uniform-random k-bit number. (There are 2 2-bit numbers, 10 and 11. There are 4 3-bit numbers, 100, 101, 110, 111. There are 2^{k-1} k-bit numbers.) What is the probability that your nim position is a winning nim position?

Example question: here $t = 4$ and $k = 3$.

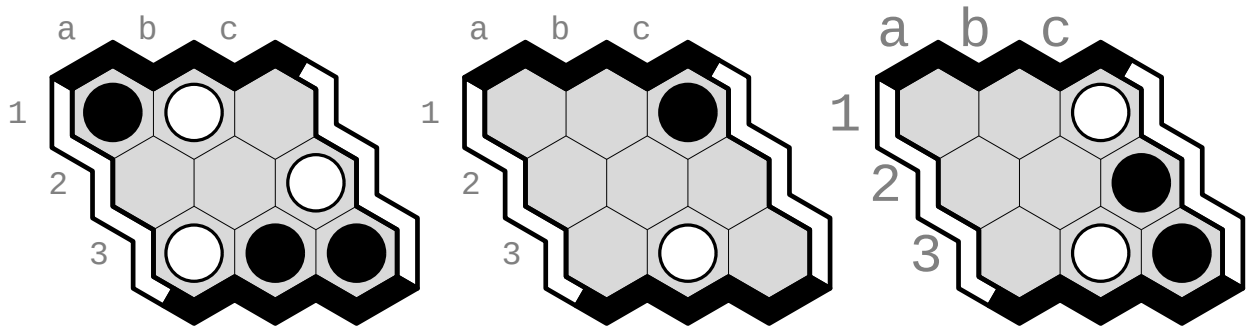
Example answer: The position is losing if and only if the xor-sum of the pile numbers is 0. Write the pile sizes in a bit array as shown on the course webpage, with each row corresponding to a pile size.

The xor-sum will be 0 if and only if the xor-sum of each column of bits is 0. In each row, the bit in column 1 is 1. There are an even number of rows, so the xor-sum of column 1 is 0. For each other column j, there are t bits and each is equally likely to be 1 or 0. From this you can show that the probability that the number of 1s in this column is even is 1/2. So ...

answer like this (your arithemtic might be different):

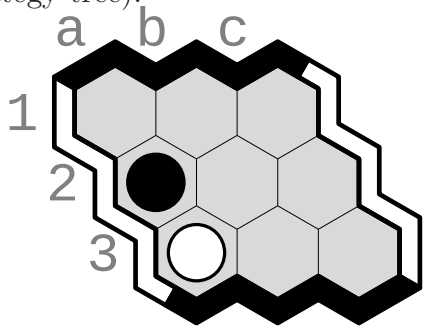
... the probability that all columns have an even number of 1s is $1 \times 1/2 \times 1/2$, so
the probability that the nim position is losing is 1/4, so
the probability that the nim position is winning is $1 - 1/4 = 3/4$.

4. a) Below from left are puzzles P_0, P_1, P_2. Your puzzle is P_m.



For a player, a strategy is *winning* if it wins against every possible opponent strategy, i.e. if the minimax value is win. A move is *winning* if it is the first move in a winning strategy. For this hex position with Black to play, give all winning moves. For one winning move, give a proof tree that shows how Black can win against any opponent move sequence. (Recall: a proof tree is strategy tree that proves or establishes some bound on the minimax value of the game. In your proof tree, each node with Black to play has 1 child; each node with White to play has t children, where t is the number of empty cells in the position for that node.)

b) For the puzzle below with Black to play, 1.Black[c2] is a winning move. Briefly describe a winning strategy, and give the number of nodes in a proof tree for this strategy (you do not have to draw the whole strategy tree).



5. Consider Pankratz's visualizer http://webdocs.cs.ualberta.ca/~hayward/355/asn/hexviz/ of Yang's winning 9×9 strategy. Below are move sequences 0, 1, 2 respectively. Use sequence n, where n is your secret number.

sequence 0	2.W[e4] 4.W[g2] 6.W[f2] 8.W[d7] 10.W[b8] 12.W[c7]
sequence 1	2.W[g1] 4.W[e2] 6.W[f4] 8.W[c3] 10.W[c9] 12.W[b8]
sequence 2	2.W[f4] 4.W[d9] 6.W[c8] 8.W[f2] 10.W[g3] 12.W[e8]

a) Against Black's winning strategy, assume that White starts with your sequence. Give Black's moves 1 to 13.

b) Continue this game from move 14. Assume that White suddenly decides to lose as quickly as possible. Find a line of play so that the game ends soonest. Give all moves after move 13. Prove that your answer is correct (hint: shortest path).

c) Assume that a Hex game continues after a) with Black continuing to follow Jing Yang's strategy and with White playing to prolong the game as long as possible. Give all moves (from 14 to the end) in such a game. (There are many correct answers.)

6. Consider Hex program `hex-3x3.py` from the class repo.

a) On my home machine, when I play **x b2** and then ask it who wins (**? o**), it answers **x can win, 677 calls**. Using this program, find all winning first moves for x, and for each give the number of calls.

b) 3x3-Hex is rotationally symmetric: if we rotate the board 180 degrees, it is identical to the original board. Why are the answers to a) not rotationally symmetric, e.g. why is the number of calls to solve the game after 1.B[a3] different from the number of calls after 1.B[c1]?

c) When I ask the program who wins from the empty board if x plays first (**? x**), it says **x can win, 678 calls**, so exactly 1 more call than after **x b2, ? x**. But if I change the move order (line 109) to the natural move order (108), **? x** takes 5677 calls. With this move ordering, what first winning move is the program finding for Black? Explain briefly. In your answer, give the number of calls made before it found the winning move, and the number of calls when it tried the winning move.

d) Replace line 109 with a bad move ordering, i.e. a move ordering that you think might *maximize* the number of calls needed to answer the question **? x** from the empty board. How many calls does your bad ordering achieve, and why do you think this might be the maximum (or close to the maximum)?

e) Program `hex-3x3.py` uses function `has_win` to check whether a player has won the game. Does `has_win` use breadth-first search, depth-first search, or neither? Explain briefly, and explain why I used this approach for this program.

f) Briefly explain the differences between program `hex-3x3.py` and program `hex-vc3.py`.