cmput 355 2025 practice questions 4 (with answers)

1. For each tic-tac-toe opening move, give a **sub-optimal** reply. Check answers with `ttt/tt.py`.

```
. . .      x . .      . x .
. x .      . . .      . . .
. . .      . . .      . . .
```

Answer. Each opening move has minimax value draw, so a reply is suboptimal if and only if it has minimax value lose. Each losing reply is shown.

```
. o .      x o o      . x .
o x o      o . o      o . o
. o .      o o o      o . o
```

2. Consider this 2-player alternate-turn win/loss/draw game, played on a 3x3 board: *on a turn, a player colors any empty cell; once the board is full, the win/loss/draw condition (which we are not telling you) is used to determine the winner.*

   a) How does this game differ from tic-tac-toe?

   b) How many nodes are in the tree of all continuations of the game?

   c) Prove that the number of nodes in the graph of all continuations of the game is less than your answer to b).

Answer.

a) in ttt the game can end before the board is full.

b) $1 + 9 + 9 * 8 + 9 * 8 * 7 + \ldots + 9 * 8 * 7 * 6 * 5 * 4 * 3 * 2 * 1 = 986410$

c) The difference between the tree and the graph is that the graph brings transpositions back to the same node. Thus, for each j, the number of nodes at level j in the graph is less than or equal to the number of nodes at level j in the tree. Also, at level three, there are $9 * 8 * 7$ nodes in the tree but only $9 * 8 * 7/2$ nodes in the graph, so the number of nodes in the graph will be smaller than the number of nodes in the tree.

3. a) When you solve tic-tac-toe from the empty board using `ttt/tt.py`, how many nodes are in the search tree? Is this about what you expected? Explain.

   b) Repeat a) using the transposition table.

   c) Repeat a) using isomorphisms.

   d) Repeat a) using isomorphisms and transposition table.

   e) In `ttt/tt.py`, in function `negamax()`, uncomment this line

   ```
   if so_far == 1: break   # improvement: return once win found
   ```

   and repeat each of the previous questions.

   f) Based on your answer to e), how could you further improve `ttt/tt.py` (so that the number of nodes searched is further reduced)?

Answer.

a) 549946. The number of nodes in the tree of all continuations if the game never ended until the board is full would be 986411, so the number of nodes in the true tree is about 5/9 that, so about 4/9 of the leaves of the overfull tree are not reachable in the game (because it ended before the board was full), which seems plausible.

b) 16168. From our earlier answer, because of the number of transpositions that have already appeared by level 3, we know that the number will be at most 1/2 of that of a). This is about 16/550 of the number of nodes in a), smaller than I would have guessed.

c) 58524. At level 1 there are only 3 non-ismorphic moves, so the number will be at most 1/3 of that of a). This is about 58/550 of the number of nodes in a), which seems plausible.

d) 8307. As expected, this is smaller that answers to b) and c). This seems plausible: humans can easily solve this game, so the smallest proof tree, after pruning isomorphisms, should not be too big.

e) a 94978. b 9973. c 7885. d 2740. As expected these number are all smaller than the corresponding numbers before the code change, because at each point in the search, as soon as a winning child is found the remaining sibling nodes are pruned. Notice that the smallest number 2740 is still a lot larger than the number of nodes in the XKCD diagram showing the proof of tic-tac-toe, because there the author is picking a best move each time, whereas in this version of `tt.py` this does not always happen.

f) Use a better move order. E.g. in function `legal_moves()`, comment out `for j in range(Cell.n):` and uncomment `#for j in (4, 0, 2, 6, 8, 1, 3, 5, 7):`. Now you are considering cells in order center, corners, sides and the number of nodes becomes a 67182. b 8866. c 6731. d 2458.

4. Convert decimal 29 into binary. Convert binary `1 0 1 1 1 0 1 1` into decimal.

Answer.

```
29    1         answer  1 1 1 0 1
14    0
 7    1         check  16 + 8 + 4 + 0 + 1 = 29 :)
 3    1                  or use method below
 1    1

 1   1       check ->  187 1
 0   2                  93 1
 1   5                  46 0
 1  11                  23 1
 1  23                  11 1
 0  46                   5 1
 1  93                   2 0
 1 187  <- answer        1 1   10111011 checked :)
```

5. For a nim position with pile sizes 15, 27, 14, 25, 7, find all winning moves.

```
a     15        1 1 1 1
b     27        1 1 0 1 1
c     14        1 1 1 0
d     25        1 1 0 0 1
e      7        1 1 1
```

Answer.

```
a     15          1 1 1 1
b     27        1 1 0 1 1
c     14          1 1 1 0
d     25        1 1 0 0 1
e      7            1 1 1
      xor-sum   0 0 1 0 0
```

A move wins if and only if the new xor-sum is 0. Here this happens only 3 ways: change 2nd bit (from left) in 15 (becomes 11), change 2nd bit in 14 (becomes 10), change 1st bit in 7 (becomes 3). Winning moves: remove 4 stones from 15-pile, 4 stones from 14-pile, or 4 stones from 7-pile.

Check your answer with `nimbig.py`.

6. Find a 3-pile nim position with exactly 2 winning moves, or explain why there is no such position.

Answer.

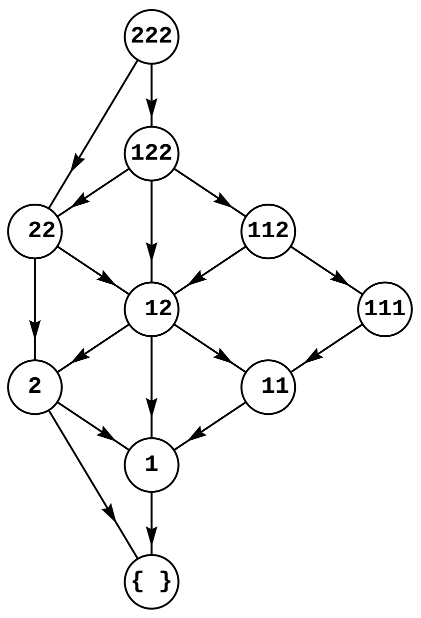
A pile has a winning move if the pile size in binary has a 1 in the same bit position as the leftmost 1 in the xorsum of all the piles. If there are exactly two winning moves, there are exactly two piles with a 1 in this bit position. But then the xorsum of the pile sizes in that position is 0, contradiction the assumption that that pile has xorsum 1. This argument works for any even number: for any position, the number of winning moves is either 0 or an odd number.

7. a) Draw the graph (dag) of all continuations of the game for nim(2 2 2). Group nodes by their multiset of non-zero pile sizes, e.g. group all 6 permutations of (1 0 2) as multiset {1, 2}, group all 3 permutations of (1 0 1) as multiset {1, 1}, etc. The root of your dag will be the node {2, 2, 2}, its children will be {1, 2, 2} and {2, 2}, and the lowest node in the dag will be {}, the position with all piles empty. Circle all losing nodes.

   b) Draw the top two levels (the root and all children) of the tree of all continuations (TOAC) of the game for nim(2 2 2). Also, for each multiset that appears as a node in the TOAC (so {}, {1}, {1,1}, {2}, {1,1,1}, {1,2}, {1,1,2}, {2,2}, {1,2,2}, {2,2,2}), give the number of nodes in the subtree of that multiset in the TOAC.

Answer.

a) see also `https://webdocs.cs.ualberta.ca/~hayward/355/jem/nim.html#sol`, where the dag is drawn sideways



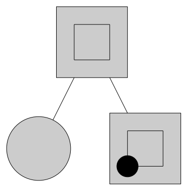
8. Here is output from recursive nim solver `nimsimp.py`. Give the missing number of states and messages: False, False dict, True dict, or True losing child.

```
 (2, 1)
   (0, 2)
     (0, 0) __________
   (0, 2) __________
   (0, 1)
     (0, 0) __________
   (0, 1) __________
   (1, 1)
     (0, 1) __________
     (0, 1) __________
   (1, 1) __________
 (1, 2) __________
___ states
```

Answer.

```
 (2, 1)
   (0, 2)
     (0, 0) False dict
   (0, 2) True losing child
   (0, 1)
     (0, 0) False dict
   (0, 1) True losing child
   (1, 1)
     (0, 1) True dict
     (0, 1) True dict
   (1, 1) False
 (1, 2) True losing child
5 states
```

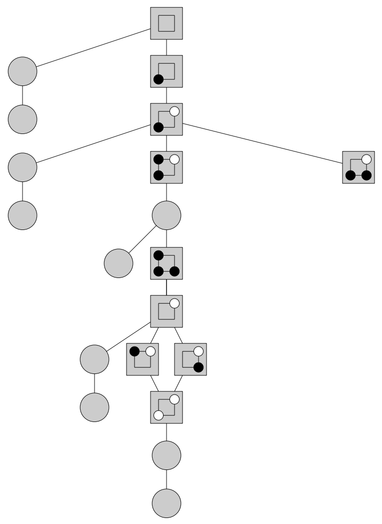
9. Starting from below, draw the next two levels of a 2×2 go second-player (White) strategy tree that shows how White can lose by at most 1. Draw each pass move as an empty circle. For the children of a game state at which no capturing has yet occurred, you can prune symmetric moves.



10. a) For a given game state S and player-to-move P, define a strategy tree.

b) The diagram shows part of a White strategy for 2×2 go. Give the number of nodes in the complete strategy tree. Explain.

c) Let m be the minimax score for Black for 2×2 go. Based on the diagram, what can you conclude about m? Answer like this: $m = 1$, or $m \geq -1$, or $m \leq 4$, etc. Explain.



Answer.

a) The root node of the tree corresponds to S. For each node in the tree corresponding to a state with P to move, the node has exactly one child, showing the state resulting after the by P that follows the strategy. For each node in the tree corresponding to a state with the opponent to move, the node has exactly one child for each possible opponent move.

b) Isomorphic moves have been pruned. Working from the bottom up, the number of nodes in the complete subtree rooted at position `.o..` is $1+2+4+4 = 11$, so the number of nodes in the subtree rooted at `*o*.` is 15. Symmetric sibling `.o**` also has 15 nodes in its subtree, so parent `.o*.` has $1+2+15+15 = 33$ nodes. Parent `..*.`'s subtree has 34 nodes, so root `....`'s tree has $1+2+34+34+34+34 = 139$ nodes.

c) This strategy shows that White loses by at most one, so Black cannot win by more than 1, so $m \leq 1$. (We saw in class that $m \geq 1$, so this answer together with what we learned from this question shows that $m = 1$. But the correct answer for this question is $m \leq 1$.)

11. What's wrong with the following argument? For each statement: state whether True or False; if False, explain.

    - For tic-tac-toe, the number of nodes in the tree of all continuations is about 500 000.
    - So ttt can be solved in a few seconds in python using minimax.
    - For 3×3 go, the number of legal positions is exactly 3^9.
    - Call a tic-tac-toe position legal if it is reachable in some game. For tic-tac-toe, the number of legal positions is exactly 3^9.
    - So we expect that 3×3 go can be solved in a few seconds in python using minimax.

Answer.

- true
- true
- false. 3^9 is the number of positions in which each cell is black or white or empty, but many of those positions are not legal go positions: a position is go-legal if and only if each block (connected component of same-color stones) has at least one liberty. For example, each position with no empty cells is go-illegal.
- false. each position in which both players have a winning line is not legal. each position in which the number of black stones differs from the number of white stones by two or more is not legal.
- TOAC for 3×3 go has more than 10^{1100} nodes (see the next question), too many to be solved with vanilla minimax.

12. a) According to John Tromp, what is the number of legal 3×3 go games?

    b) What is the relationship between the number of legal 3×3 go games and the number of nodes in the TOAC.

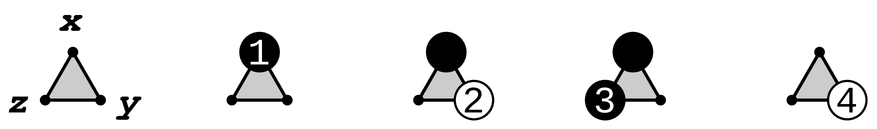
    c) Why is this number so much bigger than the number of nodes in the TOAC for tic-tac-toe?

Answer. a) About 10^{1100}.

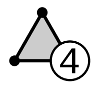
b) They will be within a constant multiplicative factor of each other. Usually, when legal go games are described, the final two pass moves are omitted. In this class, whenever we talk about the TOAC, we include all moves, including the final pass move.

c) The number of legal 3×3 go positions is 12675. A game corresponds to a path in the graph in which no node is visited more than once. The number of legal 3×3 ttt positions is about half as many (3139 + 2907 = 6046: use the psns-info option in class program `tt.py`) and in ttt, a legal move is much more restricted, because stones are never moved after they are placed.

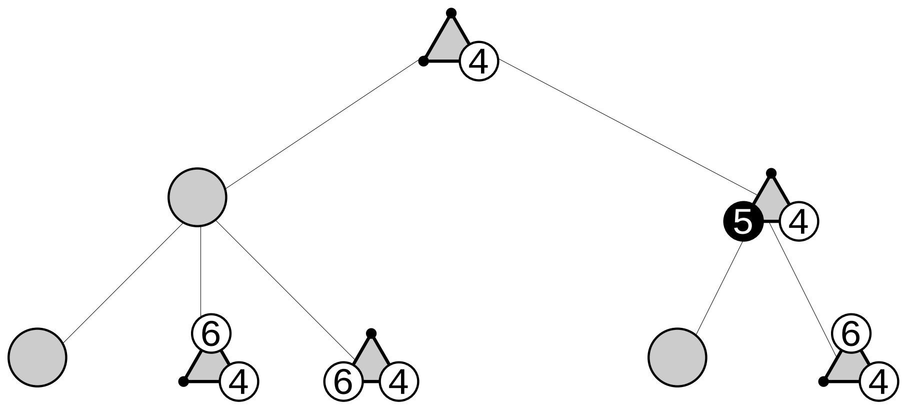
13. Trigo is go (with positional superko and no self-capture) played on a triangle.



Below, from the move 4 game state, draw the next 2 levels of the tree of all continuations of the game. Label each leaf node with its black minimax value (−3, 0 or 3).



Answer.



14. Here is output from `go/tromp.py`, which solves 2×2 go with alphabeta. Give the missing lines. Check your answer by running the program.

```
0 (-4,4)
 . .
 . .
1 (-4,4) pass
 -----
 -----
2 (-4,0)
 o .
 . .
3 (-4,0) pass
 o .
 . .  CUT -4 -4
```

```
3 (-4,0)
 -----
 -----
4 (-4,0) pass
 o *
 . .  CUT -----
4 (-4,0)
 -----
 -----
5 (-4,0) pass
 -----
 -----
```

```
6 (-4,-1)
 o .
 o o
7 (-4,-1) pass
 o .
 o o CUT -----
7 (-4,-1)
 -----
 -----
```

Answer.

```
0 (-4,4)
 . .
 . .
1 (-4,4) pass
 . .
 . .
2 (-4,0)
 o .
 . .
3 (-4,0) pass
 o .
 . .  CUT -4 -4
3 (-4,0)
 o *
 . .
4 (-4,0) pass
 o *
 . .  CUT 0 0
4 (-4,0)
 o *
 o .
5 (-4,0) pass
 o *
 o .
6 (-4,-1)
 o .
 o o
7 (-4,-1) pass
 o .
 o o CUT -4 -4
7 (-4,-1)
 . *
 . .
8 (-4,-1) pass
 . *
 . .
```

15. a) In `go/tromp.c` and `go/tromp.py`, how would you change the move ordering so that the pass move is considered last instead of first?

    b) With this change, is 2x2 go solved faster or slower? Why?

    c) In `go/tromp.c`, why does Tromp write code that depends on bit manipulation, instead of writing code that is easier to understand?

    d) What changes would you have to make to `go/tromp.py` so that it solves 3x3 go instead of 2x2 go?

Answer.

a) move the lines that consider the pass move to after the for loop.

b) slower. presumably because alpha and beta bounds are not changed until the search descends to a leaf node.

c) in the beginning of computer evolution, each computer had its own instruction set. the introduction of assembly language programming (applicable to multiple machines) was a big step forward. the next big step forward was to the more human-readable language FORTRAN (at the time programmers didn't believe that such a language's compiled code would run within a factor of 2 of hand-written machine code: they were wrong). progressively more human-readable languages came along (yay python :), but, especially for small programs, some programmers enjoy writing code in a language close to machine level such as C. John Tromp participated in competitions for writing deliberately unreadable (obfuscated) code. he is obviously an expert programmer: maybe he thinks that his 2x2 C code *is* human-readable `:)` also: bit manipulation can often yield code that executes faster than code written to be more readable. python is interpreted rather than compiled and so particularly slow, so code written to avoid any conversions that are only for readability purposes will usually execute faster.

d) this would take a lot of effort in python. even in `C++`, unless there is lots of clever pruning, solving 3×3 positions can be intractably slow.

16. a) Give the minimax value of 1×2 go.

b) Give the minimax value of 2×2 go.

c) Give the minimax value of 2×2 go after the move 1.B[pass].

d) For the game of go (positional superko, no self-capture, komi 0) played on any graph, prove that the minimax value is at least 0 (first player Black can always at least draw).

Answer.

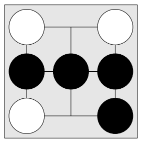
a) 0. If Black plays a stone on the first move, White can capture, and Black can't recapture because of superko.

b) 1. see the webnotes.

c) -1. After Black passes on move 1, White can follow Black's winning first-player strategy.

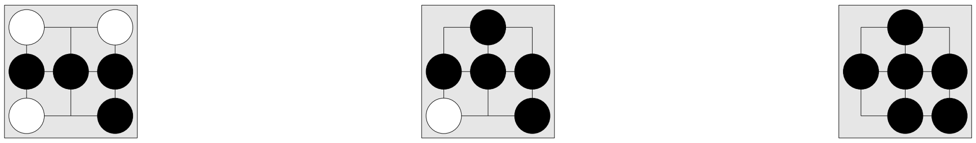
d) Assume by way of contradiction that the minimax score is $-t$ for some positive integer t: thus White has a second-player strategy where she always wins by at least t points. Then Black can do the following. On move 1, pass. If White passes, then each player scores 0, so Black does not lose. If White plays a stone, then Black can steal White's winning strategy, which guarantees that Black wins by at least t points. This contradicts the assumption that White can win by at least t points. So our original assumption is false: there is no winning second-player strategy. So there is a first-player not-losing strategy.

17. For 3×3 Go, a player has a *middle-3* position if they have 3 stones on the middle row and on at least one top-row point they don't have a stone, and on at least one bottom-row point they don't have a stone.



Prove that once a player reaches a middle-3 position they can win by 9.

Answer. They can play so that, on both the top row and bottom row, they have two stones or one stone in the middle column. From this point, the opponent has no legal moves.



hints

- see `https://webdocs.cs.ualberta.ca/~hayward/355/ssgo.pdf`