**ECE 448**

**Machine Problem 2**

Team Member:

Qianzhong Chen-3190110784-Section 1 & Section 2

Chentai Yuan--Section 3

Hao Ding--Section 4

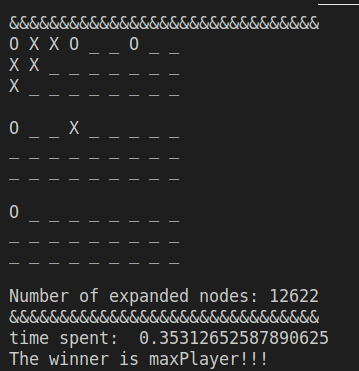
04/07/2023

**Section 1**

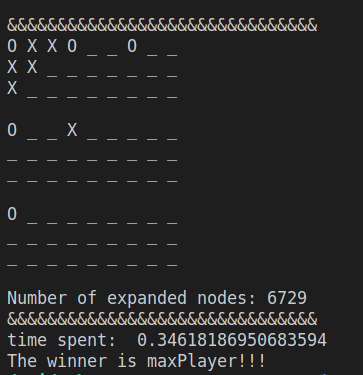
* For this exact cover problem, firstly, we find out all of the transformers of three types of blocks (domino, triomino, pentomino) by rotation and flipping. We keep a list to record all of the transformers for each shape. In total, there are 63 different pentominoes. Then, for each board, we only place single one transformer but try it for every place of the board. If no conflict occurs (no 0 is covered), we record the tilled board and the upper left coordination of the block transformer. So far, we have gained all of the elements we need to finish tilling task, if we overlap all of the boards we found before, we will find out that all of the 1s are covered. Now, we need to get rid of repetitions and pentominoes overlapping issues.
* We took advantage of Algorithm X’s great performance in solving exact covering problem. For every board we found before, we changed its shape to a single row. For instance, the board was initially a (x, y) matrix, but now being transformed into (1, x\*y), every column of the row represents an entry of the board. By data processing, 1 means that this entry is tilled and 0 means no object take this position. Afterwards, we put all of the transformed rows together from top to below, forming a big matrix. Now, the exact covering problem has been simplified to selecting several rows, forming a new matrix to make sure every column has and only has one 1.
* Algorithm X works in the following pattern: The algorithm selects a column in the matrix (referred to as the "pivot column") and tries to select one of the rows in that column to be included in the exact cover. If a row is selected, all other rows that contain elements in the same columns as the selected row are removed from consideration, and the process continues recursively with the reduced matrix until a valid exact cover is found. If at any point the algorithm reaches a dead end (i.e., it cannot find a valid row to select for the pivot column), it backtracks to the previous pivot column and tries a different row in that column. If there are no more rows to try in the previous pivot column, the algorithm backtracks further until it finds a pivot column where there are still rows available to try.
* While implementing Algorithm X, some heuristics are used naturally.
  + Column Ordering: To improve performance, Algorithm X selects pivot columns in a specific order that the algorithm selects the pivot column with the fewest 1s first, and then recursively selects pivot columns in increasing order of the number of 1s in each column.
  + Branch and Bound: Branch and bound involves pruning parts of the search tree that are known to lead to invalid solutions, which can greatly reduce the amount of backtracking required.

**Section 2**

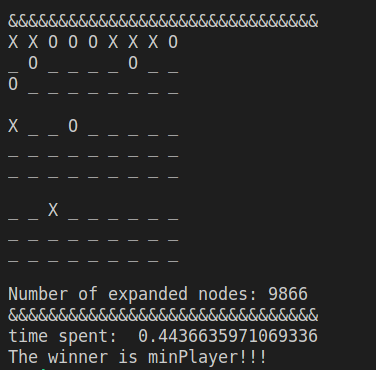
1. offensive(minimax) vs defensive(minimax), maxPlayer go first:



1. offensive(minimax) vs defensive(alpha-beta), maxPlayer go first:



1. offensive(alpha-beta) vs defensive(minimax), minPlayer go first:



1. offensive(alphabeta) vs defensive(alpha-beta), minPlayer go first:

