hugomcp@di.ubi.pt, 2022/23

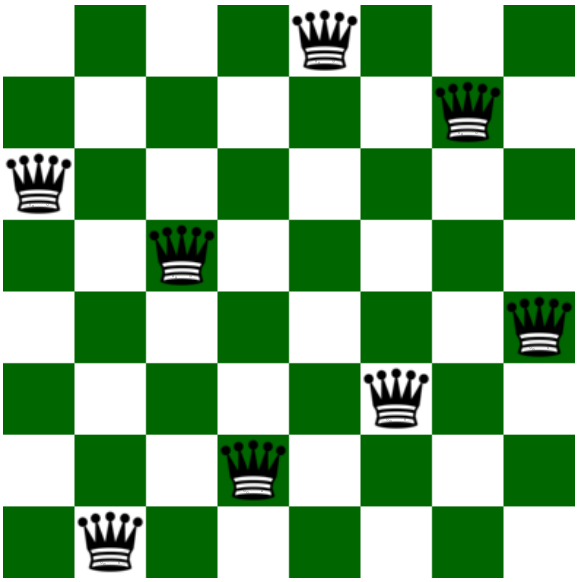
Artificial Intelligence

Practical Sheet 3: State Space Search

1. **Cell Coloring.** Suppose you have a 9×9 grid of squares, each of which can be colored **red** or **blue**. The grid is initially colored all blue, but you can change the color of any square any number of times. Imagining the grid divided into nine 3×3 sub-squares, you want each sub-square to be all one color but neighboring sub-squares to be different colors.
    a. Formulate this problem in the straightforward way. Compute the size of the state space.

    b. Find the number of solutions of the problem, assuming that you cannot color a square more than once.

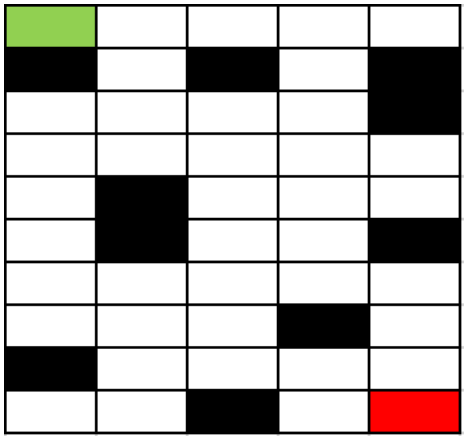
2. **N-Queens Puzzle.** This is the problem of placing **n** queens on an **n x n** chessboard such that no two queens attack each other.



    a. Given an integer n, return one solution to the problem, according to a search algorithm you find the most appropriate
    b. Return *all distinct solutions to the* ***n-queens puzzle***. You may return the answer in **any order**.

3. **Optimal Paths.** Suppose that you have to obtain the optimal path between two positions in a map, assuming that you are located in the green cell, and want to

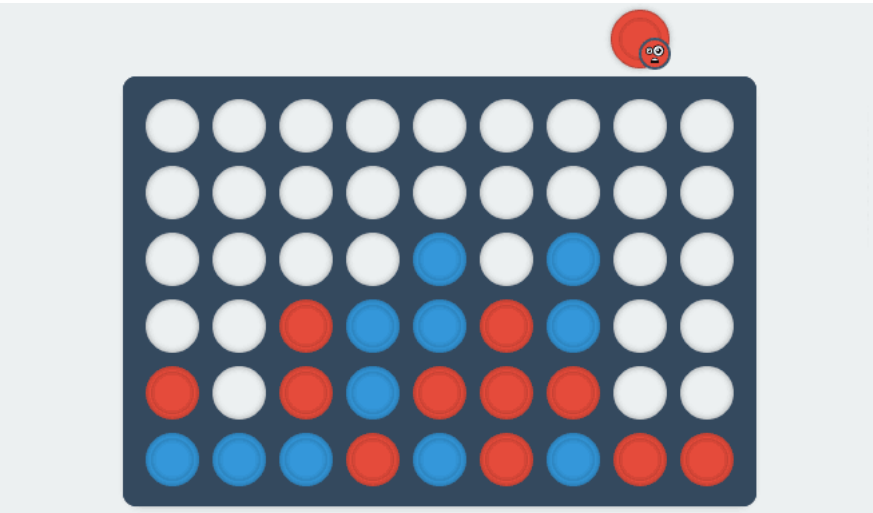
reach the red cell. You can only move between adjacent cells, using North, South, East and West Directions. You cannot pass over the black squares (walls)



a. Formulate the problem in a convenient way.

b. Compare the paths obtained for this kind of "generalized labyrinths", according to the:
   i. DFS
   ii. BFS
   iii. GS
   iv. A*

4. **Four-in-a-line.** This is a two-player game in which the players choose a color and then take turns dropping colored tokens into a seven-column, six-row vertically suspended grid. The pieces fall straight down, occupying the lowest available space within the column. The objective of the game is to be the first to form a horizontal, vertical, or diagonal line of four of one's own tokens.

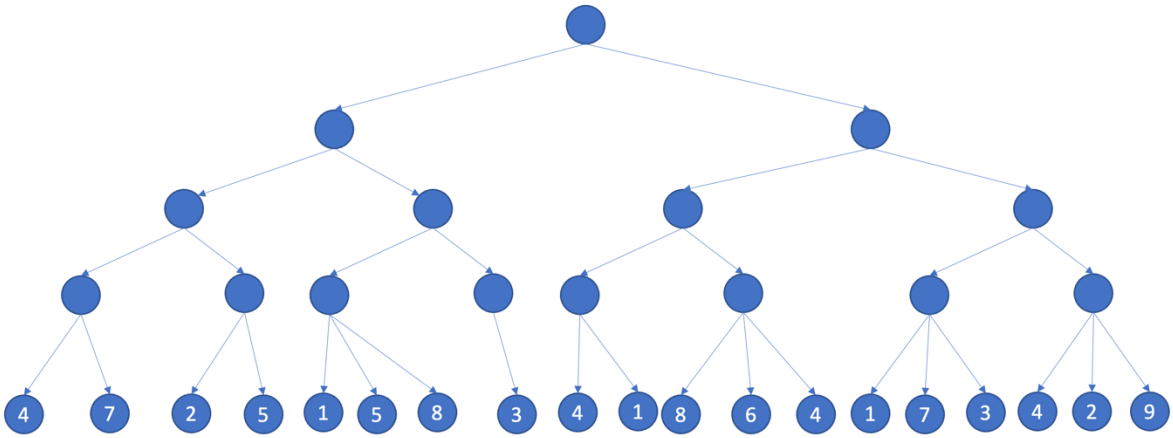


a. Formulate the problem in a convenient way. Generalize your solution, such that any "row x col" boards can be created.

b. Use the minimax algorithm to design an intelligent agent able to play this game.

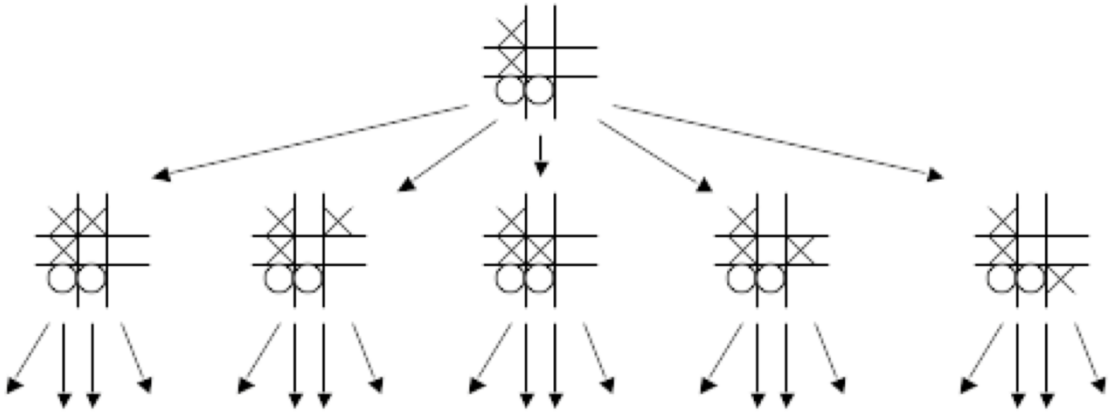
c. Also implement the Alfa-Beta pruning strategy and compare the number of nodes visited with/without pruning.

5. **Min-Max.** Consider the following tree. Perform the minimax algorithm on it, at first "without" and then "with" Alfa-Beta pruning.



6. **Tic-tac-toe.** Suppose that you are the "X" player looking at the board shown below, with five possible moves. You want to look ahead to find your best move and decide to use the following evaluation function for rating board configurations:

```
def objective_function(board):
    ret = 0
    for all columns, rows, diagonals T in board:
            ret = ret + pow(10, total_X(T))
            ret = ret - pow(10, total_O(T))
    return ret
```



Draw the configurations possible for the next level in the tree. Next, use the above objective function to rate the board configurations and choose X's best move.