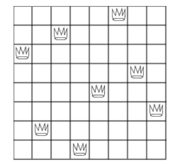
Computers playing games

One-player games

- Puzzle: Place 8 queens on a chess board so that no two queens attack each other (i.e. on the same row, same column, or same diagonal)



for $i_1 \leftarrow$ 1 to 8 // row of 1st queen
  for $i_2 \leftarrow$ 1 to 8 // row of 2nd queen
    ...
    for $i_8 \leftarrow$ 1 to 8 // row of 8th queen
      if (isValid($i_1,i_2,\ldots,i_8$)) print $i_1,i_2,\ldots,i_8$

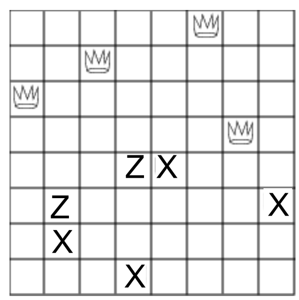
If we had a n x n board, what would be the running time?

Bactracking algorithm

- Idea: place queens from first row to last, but stop as soon as an invalid board is reached and backtrack to the last valid board
- Very similar to depth-first search

```
Algorithm placeQueens(partialBoard[8][8], row)
Input: A board with queens placed on rows 0...row-1
Output: Prints all valid configurations that can be reached from this board
  if (row=8) print partialBoard;
  else
  for i = 0 to 8-1 do
      partialBoard[row][i] = QUEEN;
      if (isValid(partialBoard)) then placeQueens(partialBoard, row+1)
      partialBoard[row][i] = EMPTY;  // reset board to original position
```

Backtracking algorithms

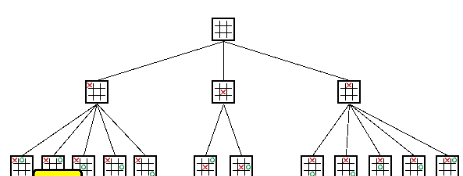


Only 2057 partial boards are considered, compared to 8^8 = 16 777 216 for the original algorithm

Two-player games

- Computers now beat humans in
  - backgammon (since 1980)
  - checkers (since 1994) (U. of Alberta)
  - chess (since 1997) (Prof. Monty Newborn)
  - bridge (since 2000 (?))
  - Go (since 2016)
- Human still beat computers in:
  - Rugby
- Human-computers are tied in:
  - 3x3 Tic-tac-toe
  - Rock-paper-scissor (but see http://www.rpschamps.com)

Game trees



Winning and Losing Positions

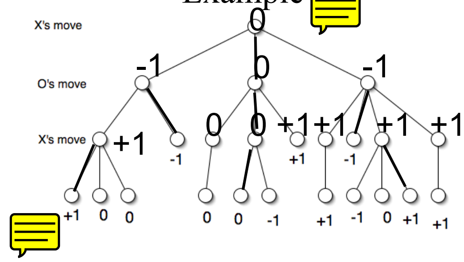
- A winning position for X is a position such that if X plays optimally, X wins even if O plays optimally
- A losing position for X is a position such that if O plays optimally, X loses even if it plays optimally.
- Recursive definition: *On X's move*,

  a position P is winning for X if
  - P is an immediate win (Leaf of game tree), OR
  - There exists a move that leads to a winning position for X

  a position P is losing for X if
  - P is an immediate loss (Leaf of game tree), OR
  - All moves available to X leads to losing positions for X

  a position P is a tie if
  - P is an immediate tie (Leaf of game tree), OR
  - No moves available to X lead to a win, but at least one leads to a tie

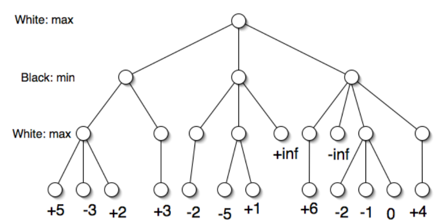
Example



Evaluation functions

- Game trees are too big to be searched exhaustively!
  - Chess has 10^{120} positions possible after 40 moves
- Idea: Look at most K moves ahead.
  - Tree has height K. Leaves are not final positions
  - Estimate the potential of the leaves
    - Good position for white: large positive score
    - Good position for black: large negative score
    - Undecided position: score near zero
    - For chess:
      - 1 point per pawn, 3 points for knights and bishops, ...
- Select the move that leads toward the most promising leaf.
- Start again next turn.

Minimax principle



Minimax principle

Algorithm white(board, depth)
Input: The current board and the depth of the game tree to explore
Output: The value of the current position
if (depth=0) **then return** eval(board)
else
return max { black(b' , depth-1): b' is one move away from board}

Algorithm black(board, depth)
if (depth=0) **return** eval(board)
else
return min { white(', depth-1): b' is one move away from board}