CISC 235: Topic 6

Game Trees

Outline

- Search Trees
- Transposition Tables
- Game Trees
- Minimax Method
- Alpha-beta Pruning

Game Program Strategy

- How can we calculate the best move to make?
- How can we take into account the opponent's strategy?

We need to do both when designing a program that plays a competitive game against a human opponent.

Search Trees

Many problems, not only games, can be formulated as a tree of possible outcomes.

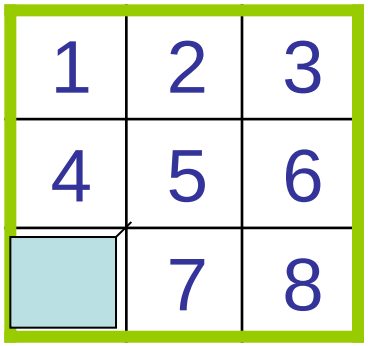
Note that the tree is not constructed in memory. It's a tree of recursive calls.

A state space formulation includes:

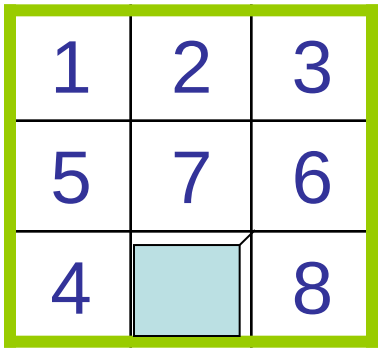
- a starting space
- an operator that generates one state from another
- an evaluation of states
- a goal state

Example: The 8-Puzzle Problem

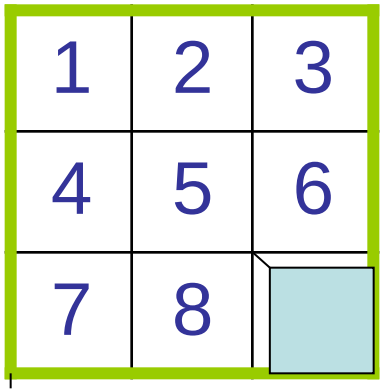
- Starting state:
  - Initial Board State
- Operator:
  - Generates up to four possible board states from parent node
- Evaluation of States
  - Count total (city block) distance that squares are from destination
- Goal State
  - Squares are lined up in numeric order



Evaluation: 2



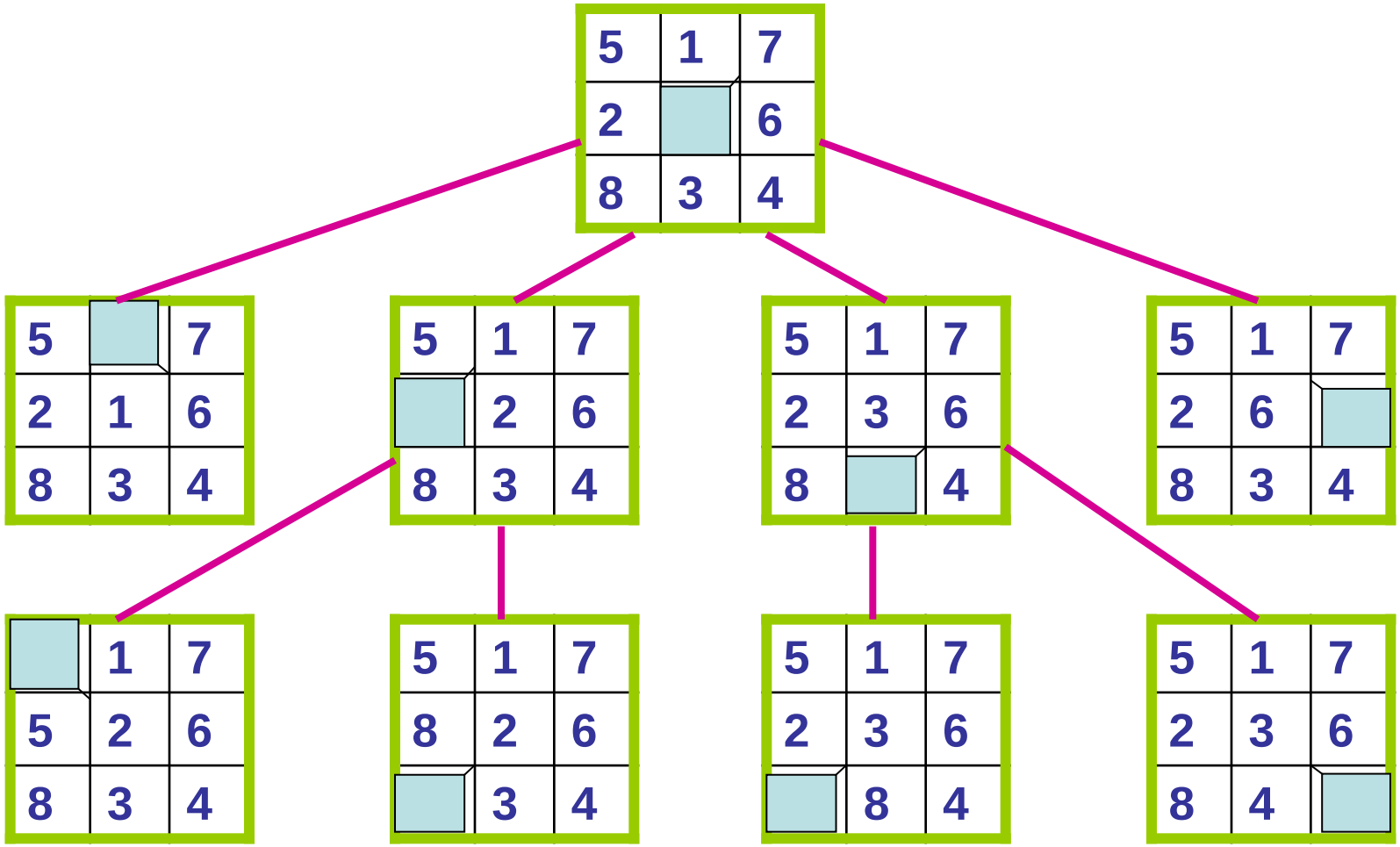
Evaluation: 5



Goal State

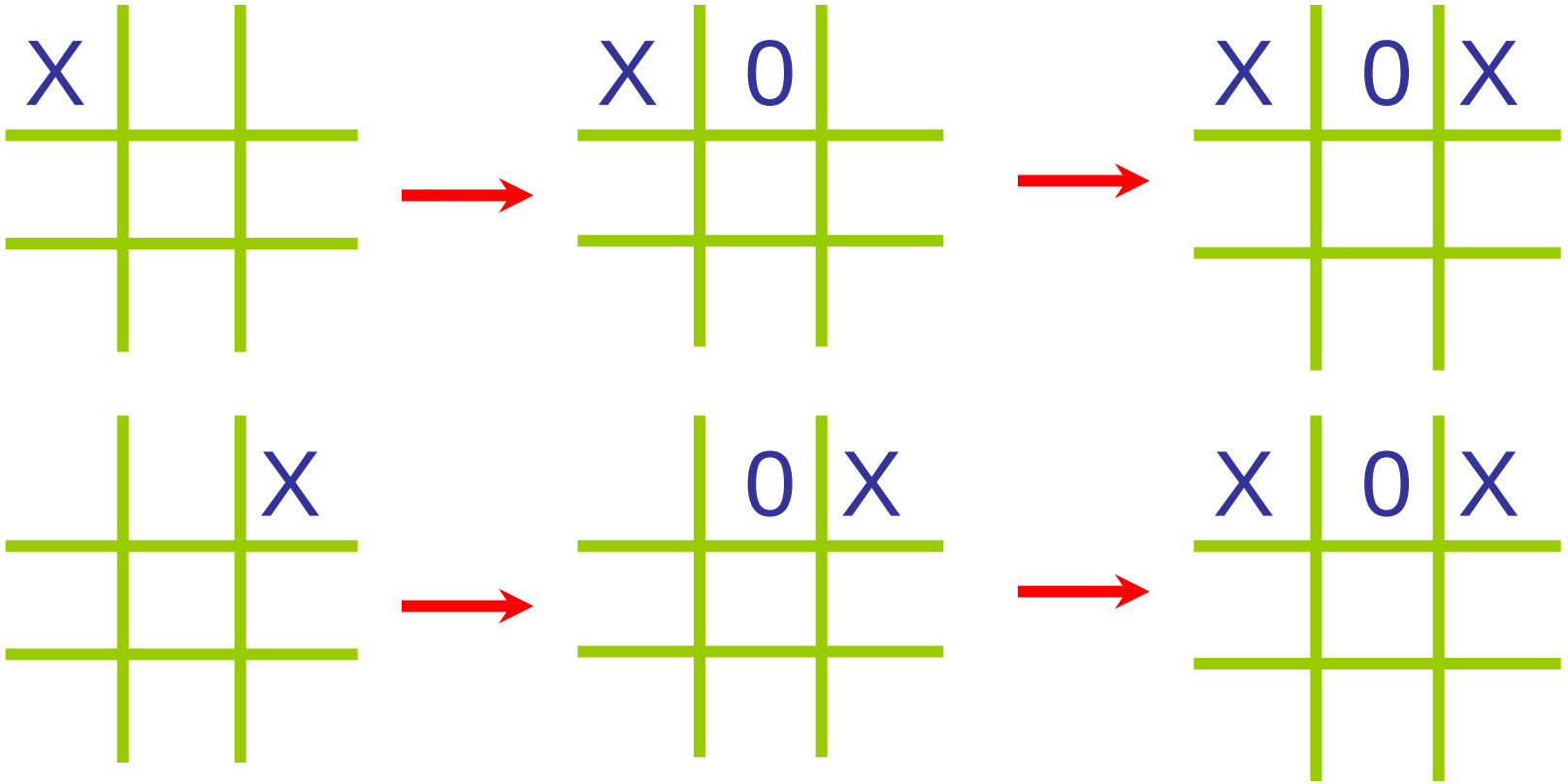
Part of 8-Puzzle Tree

How do we decide which move to make next?



Example: Tic-Tac-Toe

Two searches that arrive at identical positions:



Transposition Tables

Table to keep track of all positions in a search tree that have already been evaluated

If the values of the positions are saved, the second occurrence of a position doesn't have to be recomputed.

Store and retrieve position-value pairs (p,v) where

p is a board position

v is the value of that board position

What data structure should be used to store and look up values of positions?

Game Trees

State space formulations for competitive games, used in creating programs that play competitive games with human opponents

Difference from other Search Trees: One player cannot choose the entire path. Half the choices are made by the opponent.

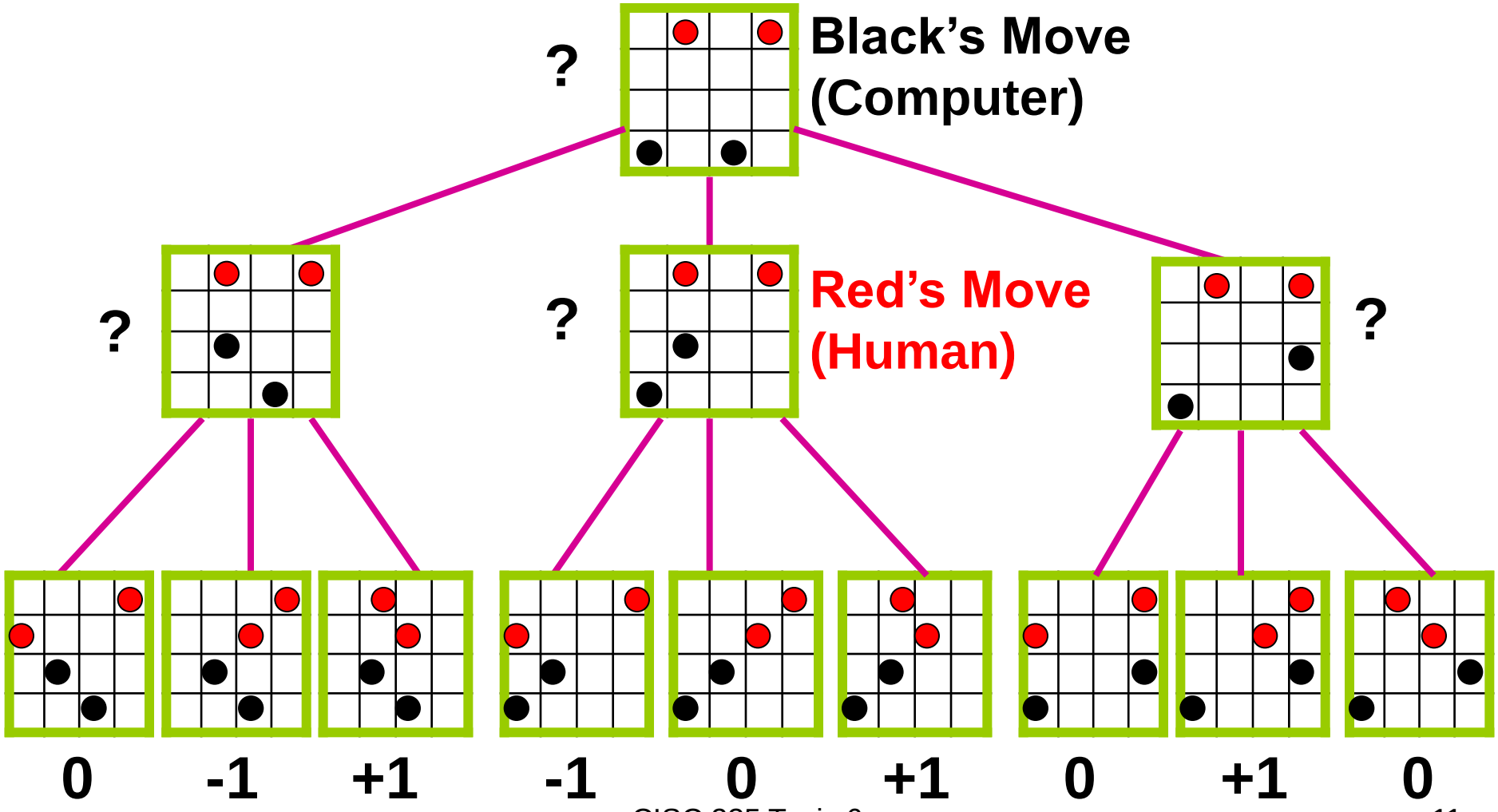
The levels of the tree alternate between the moves of the two players.

Minimax Method

- Assign an integer to each choice of move (board state), such that the largest integer represents the best move for the computer.
- Assume opponent will always choose their best move. Therefore, at their turn, opponent will choose move with smallest integer value.
- At computer's move, it chooses move with highest integer value.
- Assign values to each board state in tree, starting at bottom, according to evaluation criteria. At each level above leaves, choose max of children's values if it's computer's turn, min if it's opponent's turn.

Example: Mini Checkers

Evaluation: +1 if Computer can take a piece on next move
0 if neither player can take a piece
-1 if Opponent can take a piece



Alpha-Beta Pruning

Many techniques can be used to eliminate the need to look at each board configuration during look-ahead.

Alpha-Beta Pruning is a method in which we "prune" branches after only looking at part of them, once we can determine that they are worse than the choices we've already evaluated.

Alpha-Beta Pruning

Once the system has evaluated the left hand nodes, it is known that the opponent will select for -2 if that branch is taken. Once the -3 is selected on the right-hand branch, we don't need to evaluate the right side further. Why?