Part 2: Minimax

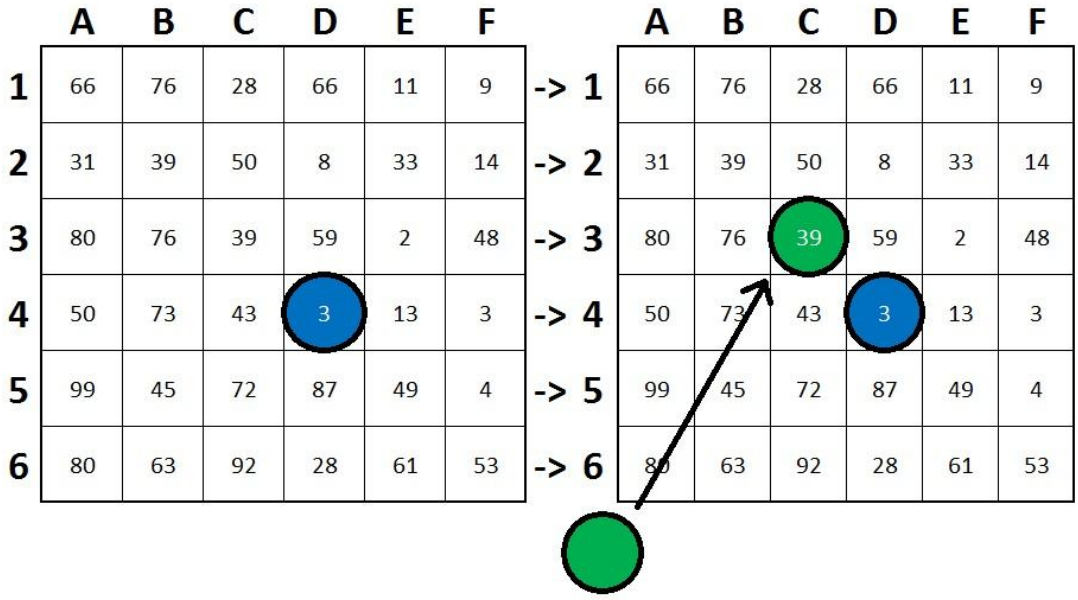
Candy game (12 points)

The goal of the game is to implement an agent to play a simple “candy game.”

Rules of the game:

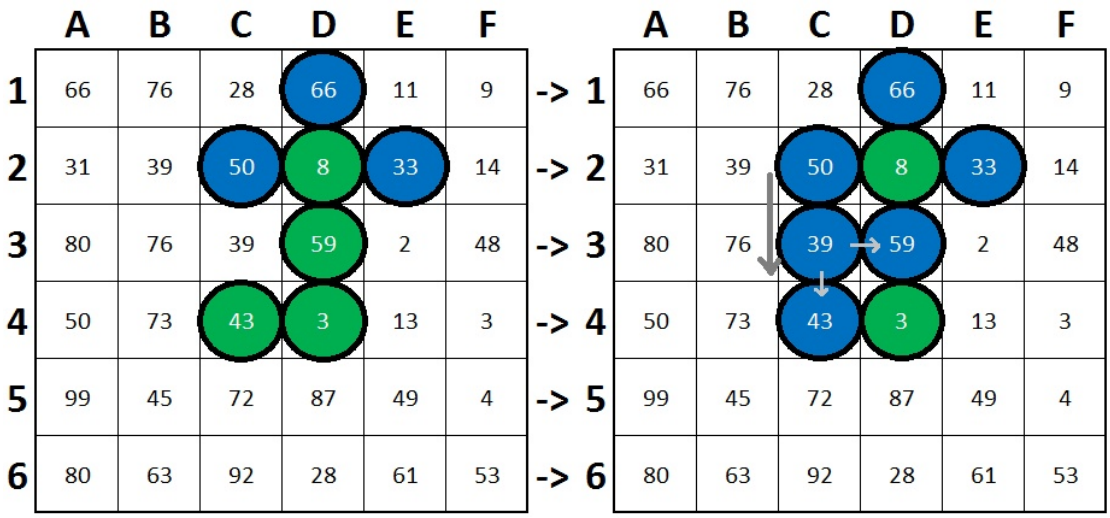
1. The game is played on a board of size 6x6.
2. Each cell of the board has a specified positive **value** (1 to 99).
3. There are two players, Blue and Green. Blue takes the first turn and Green takes the second and so on.
4. Each time Blue/Green “own” a cell, they place a colored candy piece in the cell to represent their ownership of the cell.
5. Blue/Green never run out of candy!
6. The objective of the game is to own cells with maximum total **value.**
7. The game ends when all the cells are occupied.
8. The values of the cells are variables that can be changed for each game, but stay constant during a game.
9. On a turn, a player can make one “**Drop and own”** move as follows: If any cell is unoccupied, Blue or Green can own it by placing a candy.
10. **Candy capture:** If a “Drop and own” move places a candy horizontally or vertically adjacent to other candies of the same color, then enemy candies that are horizontally or vertically adjacent to the placed candy are converted. Diagonal candies are not affected.

Here is a picture showing some example moves:



The figure above shows two “drop and own” moves. On the left, Blue makes a move, dropping its candy on [D,4] and getting 3 points. Then, as shown on the right, Green drops its candy on [C,3] and gets 39 points.

The following figure shows what happens when Blue drops its candy onto [C,3]. Because this is horizontally adjacent to the blue candy at [C,2], the green candies at [D,3] and [C,4] are “Candy Captured” and converted to blue, as shown on the right. Notice that in its next move, Green will not be able to “Candy Capture” any of Blue's candies.



Your task is to implement different agents to play this game, one using **minimax search** and one using **alpha-beta search**. Your program should use depth-limited search with an evaluation function -- which you need to design and explain in the report. Try to determine the maximum depth to which it is feasible for you to do the search (for alpha-beta pruning, this depth should be larger than for minimax). The worst-case number of leaf nodes for a tree with a depth of three in this game is roughly 42,840. Thus, you should at least be able to do minimax search to a depth of three.

Run the following matchups on the five game boards in (with blue, indicated on left, moving first): http://www.tamaraberg.com/teaching/Fall_15/HW/HW2/game_boards.zip

1. Minimax vs minimax;
2. Alpha-beta vs alpha-beta;
3. Alpha-beta vs minimax;
4. Minimax vs alpha-beta;

Carefully mention in the report:

1. The sequence of moves (e.g. “Blue: drop A3, Green: drop C1, etc. ”), the final state of the board (who owns each square) and the total score of each player.
2. The total number of game tree nodes expanded by each player.