Basic Algorithm Project – Game
May 2, 2020
Frank Zhou
Layla Liu
Lily Wang

Basic Information about the Game

There are n moves, alternating between Paul (first) and Carole. At each move, the player selects a bit, zero or one. The starting node, denoted e, is the empty string. After u moves the intermediate node will be a binary string of length u. At the end of the game, the leaves are the 2^n strings of length n. The values VALUE(x) for the leaves x are set in advance as randomly picked values from the interval [-1, +1]. The result of the game is VALUE(e), the value of the game to Paul.

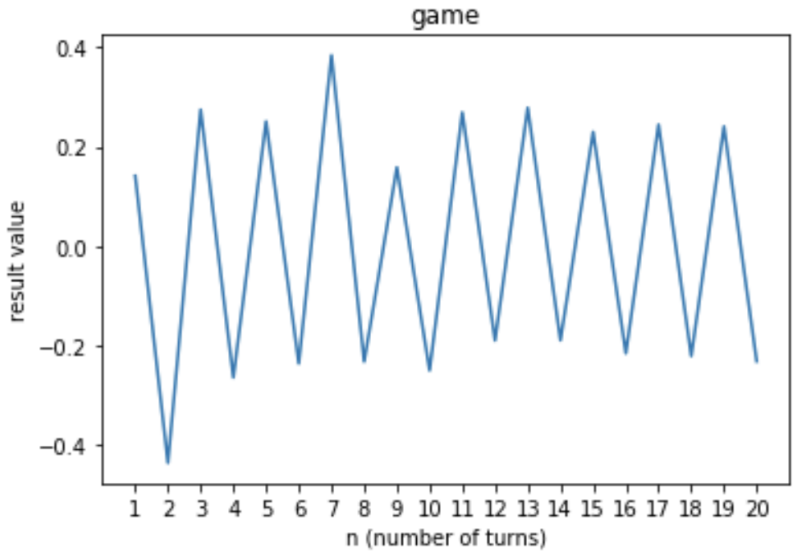
General Idea of the Algorithm

Since in each turn, the player who moves can only choose 0 or 1, then the whole game can be understood as a full binary tree with n+1 levels where n is the total # of moves.

So, we apply a simplified version of DFS – the Backward Introduction in Game Theory: Moving from the bottom to top, given the k_{th} level, we calculate the value of each node in $(k-1)_{th}$ level by choosing the larger (if it's Paul's turn) or the smaller (otherwise) value of its two children. Repeat doing this at each level until we reach the root, and now the root contains the result of the game.

In practice, instead of building a whole tree, we use arrays to represent each level and free the memory when one level is used, so that the RAM doesn't explode in running the algorithm. The result for each number n is also the average of 30 runs to avoid contingency. For more information, one can check the Jupyter notebook attached in the zip.

General Case (Paul goes first, n changes by stride 1)



Graph1: General Case

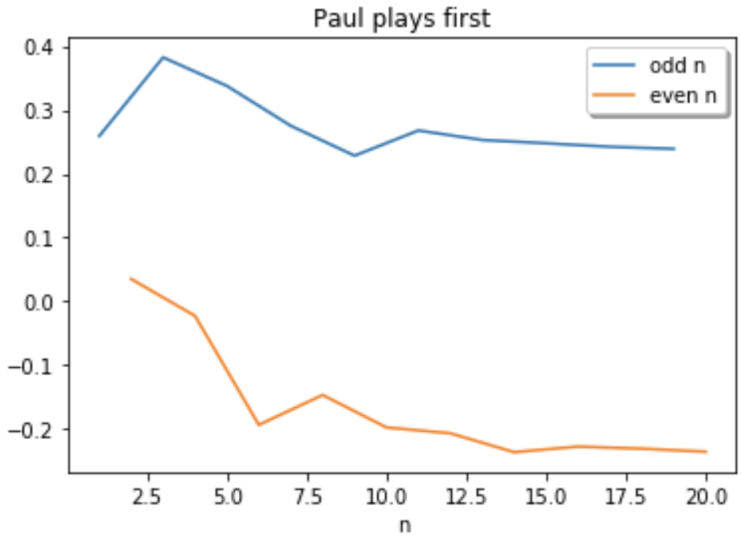
We ran the program with several numbers of n, and the results oscillated between a negative value and a positive value. There seemed to be a certain pattern of the result when n is very big (about 0.23 or -0.23), but not quite obvious.

We can make a guess for now that it is due to who plays last and then shall continue our experiment.

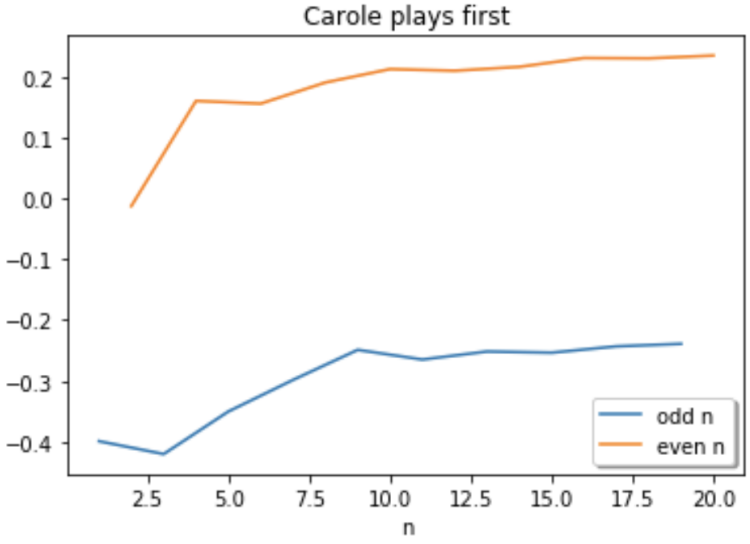
Discussion of n's parity and the first player

Let's continue our experiment with the questions: Is there an advantage to playing last? Is there an advantage to playing first?

Below are plots of even n and odd n for Paul playing first and Carole playing first respectively:



Plot1



Plot2

Below is the table of results:

	Paul First	Carole First
Odd n	Paul wins	Carole wins
Even n	Carole wins	Paul wins

Table1: results

As we know, the parity of n and the first person who moves determine the person who plays last:
if n is odd, whoever plays first plays last as well; if n is even, whoever plays first does not play last.

So, from Table1, we can see that whoever plays last wins the game, and the result is not determined only by n nor the first person who plays.

It is a tricky conclusion since it seems that whoever plays first can benefit from removing half of the total potential results, but it is actually the person who takes the last move that has more influence on the game.

Further discussions

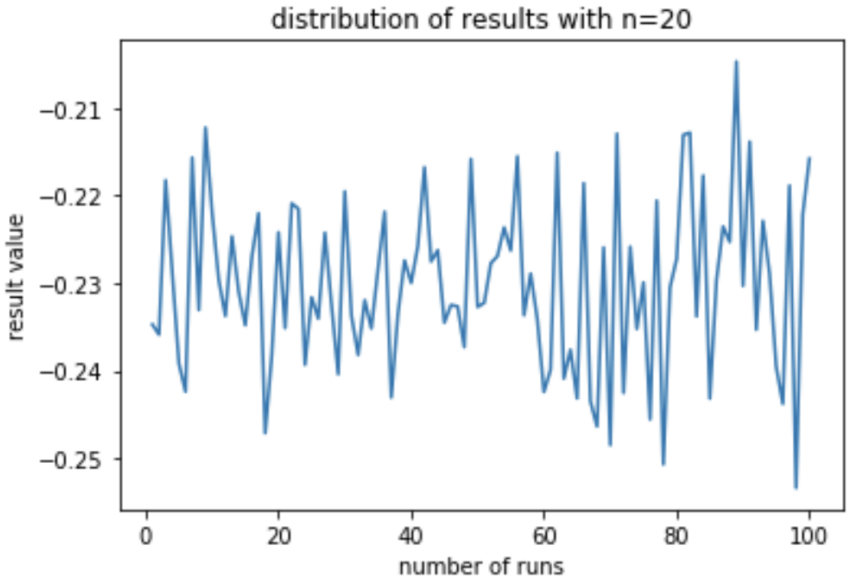
There are some other interesting findings from the results.
First, it seems that the peak of the winner's earning appears when n is between 3 and 7.
Second, when n grows bigger and bigger, the earnings of the winner converges to approximately 2.3 out of 1. So, we can guess, when $n \to \infty$, the winner gains approximately 2.3 out of 1.

Unfortunately, running the game consumes a lot of RAM and it is hard to go beyond 30 rounds on our machine. However, we can prove that the result doesn't come from one random case by looking at the variance of the data.

Variance and Distribution

We took n=20 with Paul going first and did 100 runs. The variance of the results is approximately 0.001, which is relatively low. So, this means it's trustworthy that the data will always converge to 0.23 or -0.23, not just the result of one occasional run.

Below is the plot for the results. With Paul going first and even-number of moves, the result values are always negative.



Conclusion

In this game, the person who makes the last move wins the game, which is not determined by parity of n nor who plays first. Also, the peak of the winner's earning appears when n is between 3 and 7. Finally, when $n \to \infty$, the winner's earning is about 0.23 out of 1.