PROBLEM OF THE MONTH

PHILLIP ROGNERUD

GAME RULES. There are two players, P1 and P2. The game starts off with the numbers 1 - 30. P1 chooses a number and gets that number added to their score; however, the sum of all the available factors of that number get added to P2 score, and all the numbers involved are crossed out from play. Repeat until entire board has no more available options.

Contents

1. Approaching the Problem

As I looked at the problem for the first time, one of the first things that came to mind was that I could create a program to help me find the best possible solution. I programmed the entire game in Java using their graphical user interface, and an artificial intelligence (AI) with a strategy to always win. I ran through a number of different strategies to get what I think is the best overall strategy. The first thing I thought of when thinking of a strategy was picking all the prime numbers since they have no other factor than one and itself. However, I soon realized that it was not the best way to win at the game.

2. A Winning Strategy

2.1. **Calculate.** The sum of proper divisors (that is, the divisors excluding n itself) can be expressed as

$$\sum_{d|n} d = \prod_{i=1}^{k} \frac{P_i^{m_i+1} - 1}{P_i - 1} \tag{1}$$

where $d|n$ is shorthand for "d divides n", P_i is the i^{th} prime factor, and m_i is the maximum power of P_i by which n is divisible.

Date: 2/22/2016.

Using this series above we can then use a simple equation to find the next best move.

$$netDifference = (numberChosen - sumOfProperDivisors) \tag{2}$$

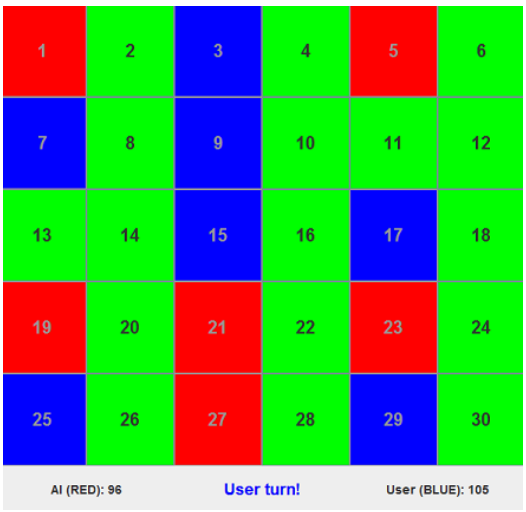
However, we would need to calculate the net difference for every single available number and choose the one with the biggest value. Which is simply done by my program, using a loop to store each calculated value into an array and then finding the max value in the array.

This strategy makes sense because it's a way to maximize the distance between the two player's scores. But is that all there is to it?

2.2. **Thinking Ahead.** There is more to this winning strategy than just calculating the difference between two numbers. Consider playing the game against my AI who's only strategy is to get the net difference. You'll find that there are multiple ways to beat the AI, regardless if you go first or second. Which tells us the best strategy isnt going to be as trivial as doing some algebra. Using (2) to figure out your next move is the most logical move, but there are certain cases where following that logic will end up losing you the game.

2.2.1. *Turning Point.* Following the strategy above works up until a certain point, where we'll have to make a different move that seems counter-intuitive. I'm calling this a 'turning point' in the game. There is only one way to efficiently play the game up until this turning point.

Consider this scenario (assuming we went 1^{st}). Now let's say both players are using the same exact strategy of finding the net difference and making their moves accordingly. If that's the case then the player who goes first will always lose, so whoever goes second will always win. Up until the scenario below, the only way to efficiently play the game using that strategy. Any other move made out of this sequence will lessen the gap between the two scores.



2.2.2. *Winning Moves.* There is a number of different ways to approach this turning point that could result in your winning, but only one perfect move to make. The logical move to make is choose 13; however, we will most certainly lose if we choose 13. In order to win we have to make a move that seems illogical because we will be choosing a number that yields are smaller net gain.

Here's a list of moves we can make at the turning point.

Moves @ turning point			
Number choice	User Final Score	AI Final Score	Score Difference
26	243	222	21
22	242	223	19
30	241	224	17
28	241	223	17
20	235	230	5
13	231	234	-3

2.2.3. *Number of Turns.* So why did choosing these numbers actually end up making us win? Simply because of the number of turns it took to end the game. Since we went first if we end on an odd number of turns we will win because landing on an odd number of turns eliminates the opponents last turn.

Number of turns		
Number choice	Score Difference	Number of turns
26	21	17
22	19	17
30	17	17
28	17	17
20	5	17
13	-3	18

2.3. **Strategy Stealing.** It's worth noting that this factor game is probably non-losing for player1, because of strategy-stealing.

In combinatorial game theory, the strategy-stealing argument is a general argument that shows, for many two-player games, that the second player cannot have a guaranteed winning strategy.

Player1 always has the option of taking 1 on the opening move, so if all other options would lead to losses (under perfect play), then player1 should take the 1 which in effect turns the tables on player2, forcing player2 to open with a losing move.

3. Artificial Intelligence

As for the AI that I programmed, it is only programmed to win at this game. The program can scale to games of smaller or bigger size; however, once a new turning point appears, if any, then the AI will not make the appropriate move. It will just play the entire game of any size using (2) to make its moves. Making

an AI that can scale with different games would take some serious coding abilities with deeper knowledge of machine learning, predictive modeling, and artifical intelligence. Something my feable mind cannot comprehend at this point in my life.

4. Growth

If we want to try and get a grasp on how fast this game grows in terms of number of playable options then we can do it by sizing down the game. If we scale it down we have: [1, 2, 3, 4, ... , N].

For N = 3, the possible sequence of moves of the playing the game can be written as 32, 23, 132, and 123, where "abcd..." means that player1 takes a, and player2 responds by taking b and so forth. Thus there are 4 ways to play the game at this size, and the total number of possible moves is 10.

For N = 4, here's a list of all the games:

43, 34, 324, 243, 234, 143, 134, 1324, 1243, 1234

We get a number of 10 total sequences with a number of 31 total possible moves.

Here's half the list for N = 5:

543, 534, 5324, 5243, 453, 435, 354, 3524, 345, 3254, 3245, 2543, 2534, 2453, 2435, 2354, 2345

The other half repeats this list with a "1" at the beginning. Note that half the list shown has 18 sequences and 66 total moves. The second half of the list will have all these moves again plus another 18 corresponding to the 1's, which gives an overall total of moves of 2 x 66 + 18 = 150.

I tried to make a program that could go through every single sequence for our game of 30 numbers, but doing that is a little to complex at my level of coding. However, we can assume that the total number of moves is something less than 30!

$$30! = 2.65 \times 10^{32} \tag{4.1}$$

5. Conclusion

To conclude the best strategy for this game is to calculate the net difference and make your moves accordingly, and once the turning point hits, depending on if you went 1^{st} or not, then try to force an odd number of turns. This game probably gets more complex when we increase the size of the game and the basic strategy that will scale to all game sizes is to calculate the net difference. There probably won't be the same turning point and would need a different strategy. However, if you want to play this game against a friend and always want to win then just go first.