<u>Project Category -</u> Game Theory

<u>Program Name/ Purpose-</u> Implementation of a Nim Game

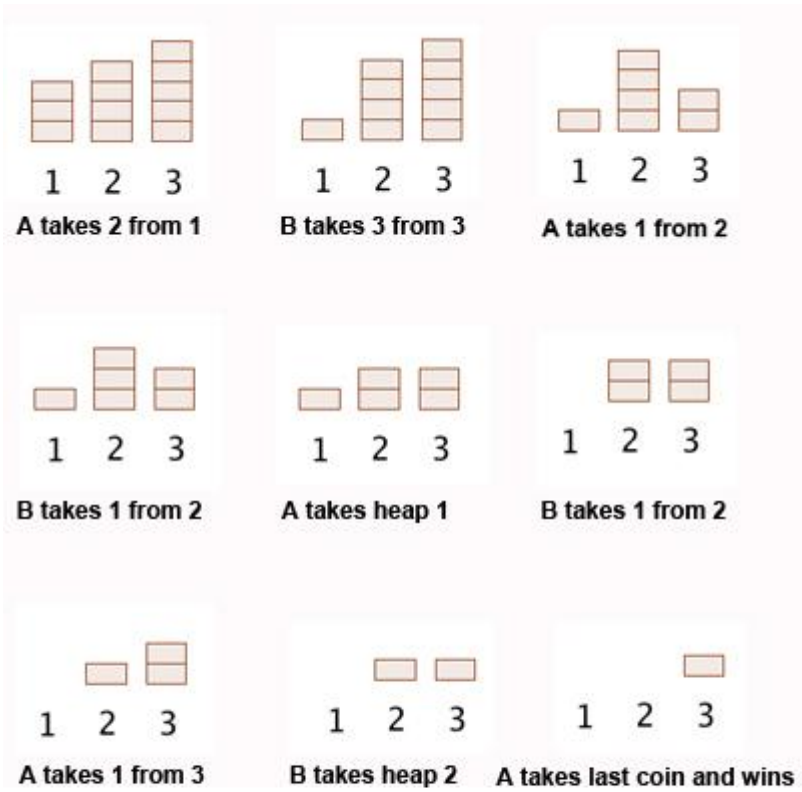
<u>Explanation-</u>

The Game of Nim is described by the following rules-

" Given a number of piles in which each pile contains some numbers of stones/coins. In each turn, a player can choose only one pile and remove any number of stones (at least one) from that pile. The player who cannot move is considered to lose the game (i.e., one who take the last stone is the winner). "

Note- There is another variation where player who cannot move is considered as winning which is knows as **Misere** play. Both are Game of Nim.

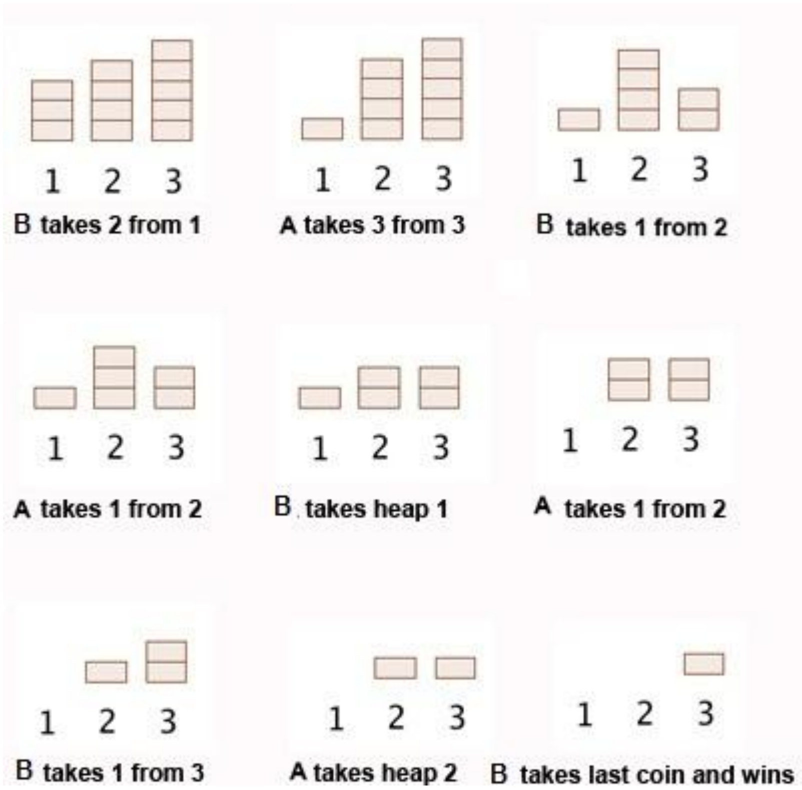
For example, we will consider that there are two players- A and B, and initially there are three piles of coins initially having 3, 4, 5 coins in each of them as shown below. We assume that first move is made by A. See the below figure for clear understanding of the whole gameplay.



So, we can see that A won the match and B lost it.

So was A having a strong expertise in this game ? or he/she was having some edge over B by starting first ?

Let us now play again, with the same configuration of the piles as above but this time B starting first instead of A.

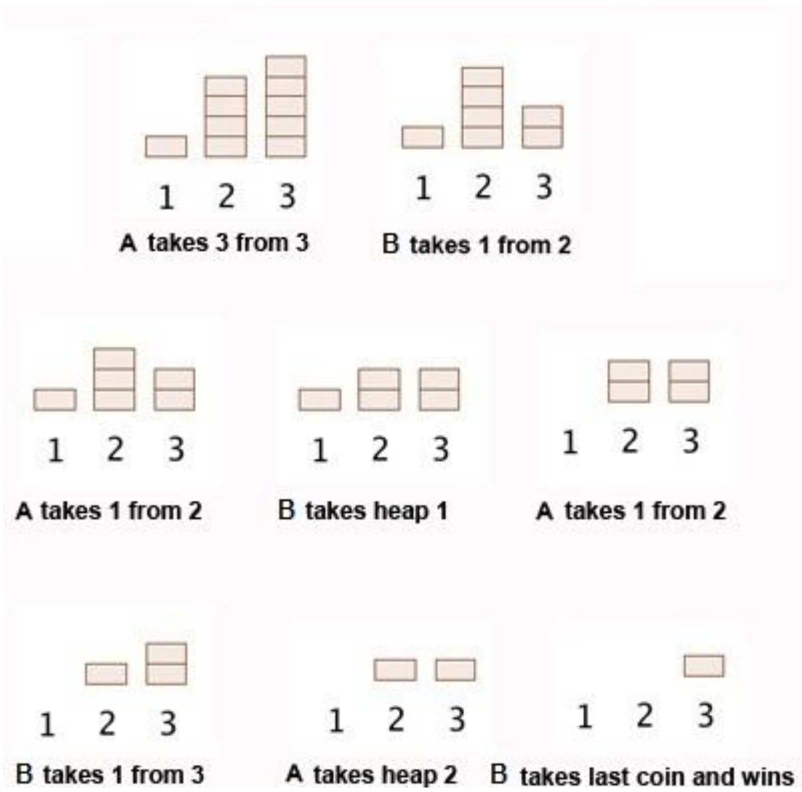


By the above figure, it must be clear that the game depends on one important factor – **Who starts the game first ?**

So does the player who starts first will win everytime ?

Let us again play the game, starting from A , and this time with a different initial configuration of piles. The piles have 1, 4, 5 coins initially.

Will A win again as he has started first ? Let us see.



So, the result is clear. A has lost. But how? We know that this game depends heavily on which player starts first.

Thus, there must be another factor which dominates the result of this simple-yet-interesting game.

That factor is the initial configuration of the heaps/piles .

This time the initial configuration was different from the previous one.

So, we can conclude that this factor depends on two factors-

1) The player who starts first.
2) The initial configuration of the piles/heaps.

In fact, we can predict the winner of the game before even playing the game !

Before going to the final part of this article and the coding part, we introduce a term –**" Nim-Sum".**

Nim-Sum -

The cumulative **XOR** value of the number of coins/stones in each piles/heaps at any point of the game is called ***Nim-Sum*** at that point.

Hence, we can predict the winner before even playing the game by stating the most fundamental theorem of Nim-Game-

" If both A and B play optimally (i.e- they don't make any mistakes), then the player starting first is guaranteed to win if the Nim-Sum at the beginning of the game is non-zero. Otherwise, if the Nim-Sum evaluates to zero, then player A will lose definitely."

Let us apply the above theorem in the games played above. In the first game A started first and the Nim-Sum at the beginning of the game was- 3 XOR 4 XOR 5 = 2, which is a non-zero value, and hence A won. Whereas in the second game-play, when the initial configuration of the piles were 1, 4, and 5 and A started first, then A was destined to lose as the Nim-Sum at the beginning of the game was- 1 XOR 4 XOR 5 = 0 .

For the proof of the above theorem , see-
https://en.wikipedia.org/wiki/Nim#Proof_of_the_winning_formula

In the program below, we will play the Nim-Game between computer and human(user)
The below program uses two functions- knowWinnerBeforePlaying() which will tell the result before playing and playGame() which will play the full game and finally declare the winner.

The function playGame() doesn't takes input from the human(user), instead it uses a rand() function to randomly pick up a pile and randomly remove any number of stones from the picked pile. The below program can be modified to take input from the user by removing the rand() function and inserting cin or scanf() functions.

Time Complexity-

We just have to make a pass over the input array to know the Nim-Sum. Hence the function knowWinnerBeforePlaying() has a linear **O(N)** time complexity, while the function- playGame() has a time complexity of **O(total_no_of_moves * N).**

Space Complexity-
No auxiliary/extra space needed.

References-

https://en.wikipedia.org/wiki/Nim