Econ 101 Honors Section

Generalized Pick-Up Sticks Game

In this lesson we will see how computers can be used to solve complex dynamic games. We will learn the concept of *dynamic programming*, which is a fancy way of saying something like backward induction.

Here is the game we want to solve:

There are N sticks and two players. The players take turns picking up a number of sticks. The player who picks up the last stick loses. However, the two players differ in the number of sticks they are allowed to pick up. Player 1 can pick up anywhere from n_1^{min} to n_1^{max} sticks, and player 2 can pick up anywhere from n_2^{min} to n_2^{max} sticks. Who wins the game and what is the optimal strategy?

Solution Strategy

We want to solve the game backward. Starting with $n = 1$ and going to $n = N$, we want to construct the following functions:

- $p_1^{win}(n)$ and $p_2^{win}(n)$, which tells us whether player 1 and player 2 are expected to win *if they are the mover when there are n sticks left*
- $p_1^{take}(n)$ and $p_2^{take}(n)$, which tells us how many sticks player 1 and player 2 should take *if they are the mover when there are n sticks left*

We can define $p_i^{win}(n)$ **recursively** according to the following definition:

- If $n \leq n_i^{min}$, then $p_i^{win}(n) = 0$. Because the number of sticks left is less than the minimum the player has to take, the moving player has to take the last stick.

- If $n > n_i^{min}$, then $p_i^{win}(n) = 1$ if there exists t, with $n_i^{min} \leq t \leq n_i^{min}$, such that $p_{-i}^{win}(n - t) = 0$; and $p_i^{win}(n) = 0$ otherwise. This condition states if the moving player can take a number of sticks t such that the moving player on the next turn would lose, then the player can ensure a victory by taking t sticks.

- If there is a winning t above, then $p_i^{take}(n) = t$. If there is no winning move, i.e. $p_i^{win}(n) = 0$, then $p_i^{take}(n)$ could be anything, and we simply set it to n_i^{min}.

See if you can implement the solution using the following code as a base.

```
import numpy

# How many sticks at start of game?
N = 100

# min amd max sticks for each player
n1min = 1
n1max = 4
n2min = 2
n2max = 6

# Initialize piwin and pitake for each player
p1win = numpy.zeros(N)
p2win = numpy.zeros(N)
p1take = numpy.zeros(N)
p2take = numpy.zeros(N)

# Initialize the piwin and pitake when there is 1 stick left
p1win[0] = 0
p1take[0] = 1
p2win[0] = 0
p2take[0] = 1

# For n going from 2 to N
for n in range(2,N+1):

      # if n is <= nimin, the player loses
      # fill in the code

      # if there is t, nimin<=t<=nimax such that the next player loses
      # fill in the code


# Print the full results
for n in range(1,N+1):
      print("n=%d, p1win=%d, p1take=%d, p2win=%d, p2take=%d" % (n, p1win[n-
      1], p1take[n-1], p2win[n-1], p2take[n-1]))
```