Author: Prianna Ahsan

prianna@ucla.edu

SID: 704068040

TA: Emrah Kostem

Sec: 1D

Iterated Prisoner’s Dilemma

Description:

Project no. 4 consisted of constructing an iterated prisoner’s dilemma game out of a 10x10 array, subject to several constraints as detailed in the project specification.

Challenges:

One of the most notable challenges I faced was creating a concise and non-repetitive selection process for determining whether or not an element was an edge element, a corner element, or neither. I eventually managed to pare down the selection statements to four if statements, each with one condition.

The second challenge came with deciding whether or not to create a ‘mirror’ array to hold the previous iteration state. I decided against this, and ended up creating a second 10x10 array of integers to hold the total sentencing value per prisoner.

The third challenge I encountered involved the selection process when determining neighbor/accomplice states and prisoner sentences. I decided to create a four element string array, cell[4], to hold neighbor/accomplice states. Then, I used a switch statement and switched on the first character of the string held in each element of cell[] (which was only one character long anyway).

Results and Testing:

I created a test program, dilemmatest.cpp, and made the number of iterations variable based on user input. The test program created an output file called ‘yput.txt’ containing the percentage of cooperators per iteration, for every iteration from 1 to n. I noticed that, after a certain number of iterations, every grid eventually reaches an equilibrium state where it oscillates between several numbers. I ran my program several times, with varying iterations, and observed that the number of iterations until equilibrium is reached seems to be variable, dependent only on the initial state of the board (which is randomly determined).

Below are some plots (created in R) of the number of cooperators left on the board per iteration. It’s obvious when the board reaches equilibrium. Note that the values on the y-axis (vertical axis) are the number of cooperators left on the board, and the values on the x-axis (horizontal axis) are the number of iterations thus far:

Figure 1: Scatterplot of cooperators vs iterations(Run 1, 10k iterations).

Macintosh HD:Users:Prianna:Documents:Courses:CSC++:xcode:IteratedDilemma:Yput:Rplots:Rplot10k.pdf

Figures 2 and 3: Scatterplot of cooperators vs iterations (Runs 2 and 3 – both 20k iterations).

Macintosh HD:Users:Prianna:Documents:Courses:CSC++:xcode:IteratedDilemma:Yput:Rplots:Rplot20k.pdf

Macintosh HD:Users:Prianna:Documents:Courses:CSC++:xcode:IteratedDilemma:Yput:Rplots:Rplot20k2.pdf