Management System
ISO 9001:2015
www.tuv.com
ID 9108653506

Republic of the Philippines
CAMARINES NORTE STATE COLLEGE
F. Pimentel Avenue, Brgy. 2, Daet, Camarines Norte – 4600, Philippines

INSTITUTE OF COMPUTER STUDIES

ACTIVITY NO. 5

Code: IT 114
Subject: Quantitative Methods (Modeling and Simulation)

Activity Title: MONTE CARLO SIMULATION OF CHUTES AND LADDERS

Objectives:

Upon completion of the design brief, students will be able to do the following:

1. Understand how Monte Carlo simulation work using a program;
2. Identify the IDE environment of python; and
3. Utilize programming and analytical skills using python program in performing simulation-based problems.

Furthermore, students learn the concept of Monte Carlo simulation to estimate the value of a probability. They gain more programming experience by adapting the simple children's game into a script in Python/MATLAB that simulates one game and then collecting data from playing that game millions of times.

Instructions

Chutes and Ladders[*] is a simple children’s game where two or more players take turns moving along a path of numbered squares trying to be the first to land on square 100. As such, the player who goes first has a small advantage. Since the rules say that the youngest player goes first, older children want to know how big is this advantage?

Activity: Write a function in Python/Matlab that simulates one game of Chutes and Ladders between two players and returns whether the winner is Player One or Player Two and the number of moves it took the winner to win. Use this function in a Monte Carlo simulation to estimate the likelihood that the player who goes

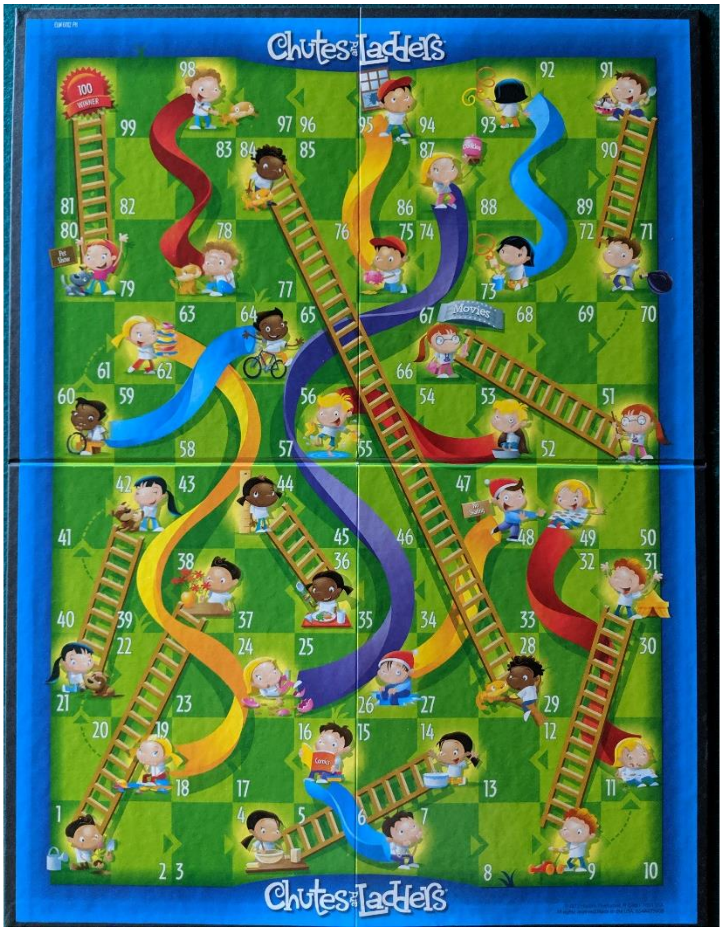
[*] The game originated in India, where it was used to teach how virtues (the ladders) and vices (the chutes) affect us in our daily walk. It came to Victorian England in 1892 under the name Snakes and Ladders, which remains its most common name in the English-speaking world outside of the United States. The Milton Bradley Company introduced the game to American children as Chutes and Ladders in 1943. The game is now made by Hasbro Gaming, and for this assignment, we will use the rules and board layout of their most recent version. Different version of the game over the centuries have featured different board sizes, winning conditions, and numbers and locations of chutes and ladders as well as different vices and virtues that they represent.

first will win and the expected number of moves required for Players One and Two, respectively, to win.

Rules of Chutes and Ladders:
The board consists of squares numbered from 1 to100. Some pairs of squares are connected by chutes and other pairs by ladders.

Players start off the board (square 0) and spin a spinner which returns 1, 2, 3, 4, 5, or 6 with equal probability. The player then moves that many squares ahead along the path. If a player lands on the bottom of ladder, the player climbs to the square at top of the ladder. However, if the player lands at the top of a chute, the player slides down to the bottom.

Play continues until a player wins by landing on square 100 exactly. If a spin would move a player beyond square 100, the player does not move and must try again on the next turn.



Evaluation

For this laboratory activity, you should be able to show the following:

1. Flowchart Model
2. Algorithm
3. Code
4. Output of the program
5. What is your recommendation to improve the simulation?

References:

Teaching Computation Using MATLAB. Available at https://serc.carleton.edu/teaching_computation/workshop_2019/activities/230840.html

SAMPLE MATLAB CODE

```
clear;
n=10^6; % Number of games to be played
winners=zeros(2,n);
MaximumMoves=200; % Maximum number of moves per game. If exceeded, the game
is ignored.
OneWins=0; % Number of times Player One wins
TwoWins=0; % Number of times Player Two wins
MovesWhenOneWins=0;
MovesWhenTwoWins=0;
MaximumMovesExceeded=0;

for i=1:n
    winners(:,i)=ChutesAndLaddersGame(MaximumMoves);
    if winners(1,i)==1
        OneWins=OneWins+1;
        MovesWhenOneWins=MovesWhenOneWins+winners(2,i);
    elseif winners(1,i)==2
        TwoWins=TwoWins+1;
        MovesWhenTwoWins=MovesWhenTwoWins+winners(2,i);
    else
        MaximumMovesExceeded=MaximumMovesExceeded+1;
    end
end
OneWinsPercentage=100*OneWins/(OneWins+TwoWins);
AverageMovesPerOneWin=MovesWhenOneWins/OneWins;
AverageMovesPerTwoWin=MovesWhenTwoWins/TwoWins;
disp(['Player One wins ' num2str(OneWinsPercentage) '% of the time.'])
disp(['Average moves when Player One wins: ',
num2str(AverageMovesPerOneWin)])
disp(['Average moves when Player Two wins: ',
num2str(AverageMovesPerTwoWin)])
disp(['Maximum number of moves ' num2str(MaximumMoves) ' exceeded '
num2str(MaximumMovesExceeded) ' times'])

function toReturn = ChutesAndLaddersGame(MaximumMoves)

moves=1;
winner=0;
OnePosition=zeros(MaximumMoves,1);
TwoPosition=zeros(MaximumMoves,1);
while winner==0 && moves<=MaximumMoves
    OnePosition(moves+1)=OneMove(OnePosition(moves));
    if OnePosition(moves+1)==100
        winner=1;
    else
        TwoPosition(moves+1)=OneMove(TwoPosition(moves));
        if TwoPosition(moves+1)==100
            winner=2;
        end
    end
    moves=moves+1;
end
```

```
toReturn=[winner, moves];

end

function Next=OneMove(Start)
        s=randi(6); % Spin the spinner
        if Start+s>100  % Must land on 100 exactly
            Next=Start;
        else
            Next=Start+s;
            if Next==1
                Next=38;
            elseif Next==4
                Next=14;
            elseif Next==9
                Next=31;
            elseif Next==16
                Next=7;
            elseif Next==21
                Next=42;
            elseif Next==28
                Next=84;
            elseif Next==36
                Next=44;
            elseif Next==48
                Next=26;
            elseif Next==49
                Next=11;
            elseif Next==51
                Next=67;
            elseif Next==56
                Next=53;
            elseif Next==62
                Next=19;
            elseif Next==64
                Next=60;
            elseif Next==71
                Next=91;
            elseif Next==80
                Next=100;
            elseif Next==87
                Next=24;
            elseif Next==93
                Next=73;
            elseif Next==95
                Next=76;
            elseif Next==98
                Next=78;
            end % end if
        end %end if
end
```

RUBRICS FOR LABORATORY ACTIVITY 5

Laboratory (50 pts)	Poor	Fair	Good	Excellent
Program execution	Program does not execute (0-1)	Program executes with a minor (easily fixed error) (2-3)	Program executes properly (6-8)	Program executes correctly with no syntax or runtime errors (9-10)
Correct output	Output is incorrect (0-2)	Output has multiple errors (3-5)	Output has minor errors (6-8)	Program displays correct output with no errors (9-10)
Design of output	Output is poorly designed (0-2)	Program does not display the required output (3-4)	Program displays minimally expected output (5-6)	Program displays more than expected (7-8)
Design of logic	Flowchart and algorithm is incorrect (0-2)	Flowchart and algorithm has significant logic errors (3-5)	Flowchart and algorithm has slight logic errors that do no significantly affect the results (6-8)	Flowchart and algorithm is logically well designed (9-10)
Standards	Flowchart and algorithm is poorly written (0-1)	Flowchart and algorithm several has several inappropriate design choices (2-3)	Flowchart and algorithm has few inappropriate choices (4-5)	Flowchart and algorithm stylistically well designed (6-7)
Documentation	Most or all documentation missing (0-1)	Missing two or more required parts (2-3)	Missing one required part in documentation (4)	Simulation and modeling program output is well documented (5)