Design Document
Assignment 1
James Gouveia
CSC 130 T/Th 10:15am-11:45am

Product Requirements:

A person is flipping a coin an unknown amount of times (0 <= number of flips < infinity) The person wants to know out of all of the possible outcomes how many do not contain 3 or more heads in a row.

How the problem is solved:

The probability of independent outcomes are calculated by multiplying the number of possible outcomes in each iteration. In this case each toss can only be head or tails, so there are only 2 possible outcomes from each flip. Our problem consists of the total number of outcomes possible, minus the number of outcomes that contain at least 3 heads in a row.

1 Flip
Total number of outcomes = 2
Number of 3 heads in a row = 0
return 2
For the 1st flip, the total number of valid outcomes are equal to the total number of possible outcomes as there is no possibility of 3 heads in a row.

2 Flips
Total number of outcomes = 4
Number of 3 heads in a row = 0
return 4
For 2 Flips, the total number of valid outcomes are equal to the total number of possible outcomes as there is no possibility of 3 heads in a row.

3 Flips
Total number of outcomes = 8
Number of 3 heads in a row = 1
Since 3 flips has 1 possibility of 3 heads in a row, 1 must be subtracted from the total number of outcomes. In this case the number of outcomes that do not contain 3 heads in a row are:
8 outcomes – 1 violation = 7
return 7

4 Flips
Total number of outcomes = 16
Number of 3 heads in a row = 3
There are 3 violations in this situation, 2 cases of HHH and 1 case of HHHH.
16 outcomes – 3 violations = 13

We could also calculate the return value by the following:
(1 Flip) 2 + (2 flips) 4 + (3 flips) = 13
or generally:
An-3 + An-2 + An-1 = 13
return 13

Why does the formula An-3 + An-2 + An-1 work?

1. If our first flip is T, then our next 3 flips = An-1 = A3
2. If our first two flips are HT, then our next 2 flips = An-2 = A2
3. If our first three flips are HHT, then our next (1) flip = An-3 = A1
4. Note that our first four flips cannot be HHHT as this violates our condition
5. In addition, the case of our first flip being H, the next 3 flips cannot be extrapolated to A3 because HHT would violate our condition. (First flip = H, next 3 = HHT, total flips = HHHT -> violation)

Implementation:

main:

Greeting: The program begins in the main and calls a print statement: CoinFlip! This program will return the number of possible outcomes of flipping a coin that do not result in three heads in a row.

User input: The program asks the user how many times they tossed the coin: Please enter the number of tosses in integer form

Input placed in variable: The user input is then placed into a variable named numberThrows

Output: The main calls the function threeInRow, passing in the variable numberThrows. This function then calculates the number of outcomes that do not contain 3 heads in a row then outputs that number to the screen.

Function threeInRow:

This function carries out the following algorithm:

1. 1 toss H/T results in 2 outcomes This sample space only has one slot so it can not contain any 3 heads in a row violations, add 2 to the memoization array

2. 2 tosses H/T H/T 2 * 2 = 4 outcomes This sample space only has two slots so it can not contain any 3 heads in a row violation, add 4 to the memoization array

3. 3 tosses H/T H/T H/T 2 * 2 * 2 = 8 outcomes This sample space contains one 3 head in a row series, subtract 1 from the number of outcomes and add 7 to the memoization array

4. 4 tosses H/T H/T H/T H/T This sample space is determined by the summation of the 3 previous tosses. The program retrieves the previous 3 results stored in the memoization array and sums them. Then the sum is returned as the result and added to the memoization array.

5. This pattern repeats for however many tosses the user inputs

Why use a memoization array?

Dynamic programming is all about avoiding calculating any given situation more than once. This problem can be solved by recalculating the number of possible outcomes but a solution like this would be much slower because it would have a lot more calculations. Since we are saving each of our calculations, we only have to make one calculation for every iteration of the for loop in threeInRow saving a lot of calculations and thus the program runs much faster.

Code:

1. The user input is accepted into the function threeInRow

2. An array is created to hold the memoization data

3. A for loop fills the memoization array with outcomes
    - A. The for loop iterates the number of times that the user has input
    - B. The first iteration places the outcome of 1 toss into the memoization array
    - C. The second iteration places the outcome of 2 tosses into the memoization array
    - D. The third iteration calls the power function then subtracts the 1 possible violating outcome and places that number into the memoization array
    - E. The fourth iteration adds up the 3 preceding values and places them into the memoization array.
    - F. All further necessary iterations will behave in the same manner as step E until they reach the number of tosses the user has input.
    - G. Return the number of outcomes that do not contain 3 or more heads in a row back to the main