Jules Voltaire

CS2223 – Algorithms

Project 1 Report

In this project, two methods are used to calculate the probability of a team winning a series given the number of games needed to win and the probability of winning a single game. The first one uses recursive method. It gets slow really quickly but doesn’t really use much storage. The second method uses dynamic programming to store previously calculated values to calculate new ones until the probability is found. Because values are stored this method is much faster as the number of games needed grows compared to recursive. The draw back to that is much more storage is used.

The recursive algorithm has a time efficiency of O(2n) and a space efficiency of O(2n) while the dynamic algorithm has a time efficiency of O(n2) and a space efficiency of O(n2). The dynamic algorithm uses a matrix to store previously calculated that explains why the space efficiency is what it is which is also the reason why it’s much quicker. The recursive one, however because it doesn’t really store anything, that’s the reason why it is much slower.

Testing both algorithms with n = 4 and p = 0.4, we get 0.29. In this scenario, both algorithms complete in about the same amount of time. But you’ll come to see later in the graph that the recursive method gets slower really quickly.

See data.xlsx for full data set

Pseudocode are as follow

Dynamic(i, j, p)

Create i x j matrix

Initialize all cells to 0

Set all cells in first row to 1 except the first one

q = 1 – p

for i in range(1, len(matrix)):

for j in range(1, len(matrix[i])):

matrix[i][j] = p\*matrix[i-1][j] + q\*matrix[i][j-1]

return the value of that bottom right most cell indexes

recursive(i, j, p)

if(i <= 0 and j <= 0):

return 0

elif(i <= 0 and j > 0):

return 1

elif(i > 0 and j <= 0):

return 0

else:

q = 1 - p

return p\*recursiveMethod(i-1, j, p) + q\*recursiveMethod(i, j-1, p)

Screenshots of terminal

