Survival of the fittest

Hitesh Kumar
PES1201701511
PES University
Bangalore , India
hiteshkumarhk.info@gmail.com

***Abstract*—The problem survival of the fittest is a dynamic programming problem . There are 3 kinds of units in an arena. There are 'a' archers, 'p' pikemen and 'k' knights. The arena is large enough that the probability of two individuals meeting is equal. The archer kills the pikeman, the pikeman kills the knight and the knight kills the archer, if they meet. Only different units kill each other. In a given certain amount of time the problem is to find the probability that each unit would be the only kind left alive.**

I. Introduction

A Dynamic Programming is an algorithmic technique which is usually based on a recurrent formula and one (or some) starting states. A sub-solution of the problem is constructed from previously found ones. DP solutions have a polynomial complexity which assures a much faster running time than other techniques like backtracking, brute-force etc.

II. Implementation

The base cases for the problem has been handled initially which breaks the recursive call . A dp table has been created which stores the results of the previous value. The dp table has been created using a four-dimension vector. The problem has been solved using bottom up technique using dynamic programming. The problem has been used by implementing the following functions:

A. Solve

This function starts the recursive call from base cases and stores the result in a 4d vector and returns the final value after the end of the loop. The returned value is a one-dimensional vector which gives the probability of survival of each type.

B. Make Table

This function is called by solve which creates a four-dimensional vector which has initial values of zero.The vector size is based on the input provided . The four-dimensional vector has a size of 3*101*101*101 which is the maximum size defined in the problem where 101 is the maximum number of pike-man or archer or knight. The function on successful execution returns a four-dimensional vector.

C. cal

This is the major function which has all the cases . Initially this function handles all the base cases required and stores them in a single-dimensional vector which is later appended into the dp table . If the case is not the base case the new value is calculated using the previous value from the dp table using the recursive function : k*a*dp[a-1][p][k][i] + a*p*dp[a][p-1][k][i] + p*k*dp[a][p][k-1][i] . This function returns the probability . Initially to calculate the probability the total is calculated which is given by : a*p + p*k + k*a. Using the total the probability for each type is calculated and the value is stored in a vector which is returned by the function.