Problem 3

> A cricket randomly hops between 4 leaves, on each turn hopping to one of the other 3
> leaves with equal probability. After
> $n \geq 0$ hops, what is the probability that the cricket
> has returned to the leaf where it started? Design a recursive algorithm to solve this
> problem. You need to give the main ideas of your algorithm and the recursive
> formula.

Thoughts:

Each turn hopping to 1 of 3 other leaves with equal probability

- 1/3 chance to each leaf

First hop there is no way for the cricket to hop to the same leaf so probability is 0

There is a 2/3 chance that the cricket will go to one of the other leafs

Over time the probability of jumping back to the starting leaf should approach 1

Algo:

Input:

n ← the number of hops to account for

Output:

the probability the cricket will return back to the starting leaf after n

def’n hopProb(n) # Returns the probability of a cricket returning to its original leaf

if n == 0 # if n is 0 then its already on it leaf, so its a 100% chance

return 1

elif n == 1: # there is no chance the cricket can return to the starting leaf

return 0

else # the prob of getting back to the goal leaf, is the prob of reaching the goal from the prev. leaf

return 1/3 + (2/3)*hopProb(n-1)

Recursive Formula (recurrence relation):

$$T(n) = \frac{2}{3} * T(n-1) + \frac{1}{3}$$