Assignment #3

Student #:250626065

Name: Zaid Albirawi

UWO email: zalbiraw@uwo.ca

1. F(x) = x mod 7  
      
   Hashtable:

|  |  |
| --- | --- |
| **n** | **F(x)** |
| 0 | > | Null |
| 1 | > | 15 |
| 2 | > | Null |
| 3 | > | Null |
| 4 | > | Null |
| 5 | > | 47 | > | 12 | > | 19 |
| 6 | > | 27 |

1. F(x) = x mod 7  
      
   Hashtable:

|  |  |
| --- | --- |
| **n** | **F(x)** |
| 0 | 12 |
| 1 | 47 |
| 2 | 15 |
| 3 | Null |
| 4 | Null |
| 5 | 19 |
| 6 | 27 |

1. F(x) = x mod 7, F’(x) = 5 – ( x mod 5 )  
      
   Hashtable:

|  |  |
| --- | --- |
| **n** | **F(x)** |
| 0 | Null |
| 1 | 12 |
| 2 | 15 |
| 3 | Null |
| 4 | 47 |
| 5 | 19 |
| 6 | 27 |

1. F (1) = 1  
   F (n) = F (n-1) + 2 (n-1)

F (1) = 1

F (2) = F (2-1) + 2 (2-1)

= F (1) + 2 (1)

= 1 + 2 = 3

F (3) = F (3-1) + 2 (3-1)

= F (2) + 2 (2)

= 3 + 4 = 7

F (4) = F (4-1) + 2 (4-1)

= F (3) + 2 (3)

= 7 + 6 = 13

Therefore, since F (4) = F (3) + 2 (3)

Then, = F (2) + 2 (2) + 2 (3)

= F (1) + 2 (1) + 2 (2) + 2 (3)

= 1 + 2 (1) + 2 (2) + 2 (3)

Therefore, F (4) = 2 (1+2+3) + 1

Therefore, F (n) = 2 (1+2+3…+n-1) +1

= 2 ∑ +1 \*∑ = (1+2+3…+n-1), the sum of all values from 1 to n

Since ∑ = n(n-1)/2

Therefore, F (n) = ~~2~~ [n(n-1)~~/2~~] +1

= n(n-1)+1

= n2-n+1

1. **Algorithm count (root r, int k)**

**Input:** root r of a proper Tree T, int k an integer that represents the leaves distance from the tree.

**Output:** the number for leaves.

int leaves=0

If (k==0)

If (r.isLeaf ())

leaves++

Else

If (r.hasChild ())

leaves+=count (r.getLeft (), k-1)

leaves+=count (r.getRight (), k-1)

return leaves

**Time Complexity:**

**count (root r, int k)**

int leaves=0 1

If (k==0)

c’

If (r.isLeaf ())

c

leaves++ 1

Else

c’’’

If (!r.isLeaf())

c’’

leaves+=count (r.getLeft (), k-1) 1

leaves+=count (r.getRight (), k-1) 1

return leaves 1

We divide the problem into two parts, the algorithm and the recursive calls

F (n) for the algorithm, F (n) = 1 + c’\*c + c’’’\*(2\*c’’) + 1

= (2c’’’\*c’’)+(c’\*c)+2

Since this algorithms recursively visits every node once, then the number of recursive calls should be equal to the number of nodes of this tree. Furthermore, the algorithms is a traversal algorithm which implies that every node has to be visited once. Therefore, F (n) = n (2c’’’\*c’’+c’\*c+2), O(n) = n.