Kevin Sekuj

1/18/2022

CS325: Analysis of Algorithms

Homework 2

1. Solve the recurrence relation using three methods

**Recurrence Relation**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Substitution Method**

Thus,

Since we arrived at the base case T(1) in the kth equation, we can say

Substituting this in our modified kth equation:

From our (Eq\*) we can say (n/2k) = 1

Substituting this for the value of k in the kth equation:

by log rule alogab = b

Finally,

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Recursion-Tree method**

Level 0 – T(n): c

Level 1 – T(n/2): c + c

Level 2 – T(n/4): c + c + c + c  
and so on until

Level i = T(1) + T(1) . . .

At this point, it becomes apparent that as the recursive tree deepens, nodes increase by a factor of two – 2^0, 2^1, 2^2, etc. Thus, at the ith level, 2i nodes will exist.

So, at the base case, level i, we have 2i nodes – T(1) =

The total cost of the tree will be cost at each level \* number of levels –

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Master Method**

Since f(n) = c, c is a constant, thus nd <<<

Thus

2.Solve the recurrence relation using any one method

a.

**Master Method**

Since n <<< n2, **T(n) = Θ(n2)**

b.

**Master Method**

Since n2 >>> √n, **T(n) = Θ(n2)**

3. Implement an algorithm using divide and conquer

1. Pseudocode

def kthElement(arr1: list, arr2: list, k: int) -> element:

if arr1 and arr2 are both empty return -1

sum length of arr1 and arr2 in variable n

if k is greater than sum of arr1 and arr2 or k is less than 0 return -1

Initialize empty array of new array lengths

create arr1\_len, arr2\_len to hold len of arr1 and arr2

create variables to hold indices of arr1, arr2 and new array; arr1\_idx, arr2\_idx, new\_idx

while arr1 < length of arr1 and arr2 < length of arr2

if element in arr1 index is less than arr2\_idx in arr2:

add the element in arr1 at arr1\_idx to the new array

increment the arr1\_idx by 1

else add element from arr2 at

increment arr1\_idx by 1

if k equals to new\_idx

return the kth indexed item in the new array

increment new\_idx by 1

while arr1\_idx is less than arr1\_len:

add the element in arr1 at arr1\_idx to the new array

increment the arr1\_idx by 1

increment new\_idx by 1

while arr2\_idx < arr2\_len:

add the element in arr2 at arr2\_idx to the new array

increment arr2\_idx by 1

increment new\_idx by 1

return kth element in new array

1. Implementation (KthElement.py)