CSci 242: Algorithms and Data Structures  **Fall, 2019**

Instructor: Dr. M. E. Kim

**Home Assignment 6: Divide & Conquer and Recurrence Equation (100)**

**Q1. [30]** **Divide and Conquer**

There is a sorting algorithm, “Stooge-Sort” which is named after the comedy team, "The Three Stooges." if the input size, *n,* is 1or 2, then the algorithm sorts the input immediately. Otherwise, it recursively sorts the first 2*n*/3 elements, then the last 2*n*/3 elements, and then the first 2*n*/ 3 elements again. The details are shown in Algorithm below.

**Algorithm** **StoogeSort**(A, *i, j* ):

***Input:*** An array, A, and two indices, *i* and *j*, such that 1 ≤ *i* ≤ *j* < *n*

***Output:*** Subarray, A[*i..j*] ,sorted in nondecreasing order

1. *n* ← *j* – *i* + 1 // The size of the subarray we are sorting

2. if *n* = 2 then

3. if A[i] > A[j] then Swap A[i] and A[j]

4. else if *n* > 2 then

5. *m* ← ⎣*n*/3⎦

6. StoogeSort(A, *i, j-m*) // Sort the 1st part.

7. StoogeSort(A, *i+m, j*) // Sort the last part.

8. StoogeSort(A, *i, j-m*) // Sort the 1st part again.

9. return A

1. [10] Give the formula of recurrence equation for the running time, *T* (*n) ,* of Stooge-sort.

T(n) = T(n/3) + T(n/3) + T(n/3) +c

T(n) = 3T(n/3) +c

T(n) = 3 \* (3\*T(n/9) + c) + c = 32 \* T(n/32) + 3 \* c + c

T(n) = 3k \* T(n/3k) + c(1 + 3 + … + 3k-1)

When, n/3k = 1 or n =3 k

Assume T(1) = c0

T(n) = n\* c0 + c \* ((3k – 1)/2)

T(n) = n \* c0 + c \* ((n-1)/2)

T(n) = O(n)

1. [10] By means of Master’s Theorem, determine an asymptotic bound for *T* (*n).*

T(n)=aT(n/b)+f(n) where a >= 1 and b > 1

T(n)=3T(2n/3) + O(1) where a=3, b=3/2, c=0, and f(n)= O(1)

Applying Masters Theorem case c<logba, where a=3, b=3/2, and c=0

N^(logba) = n, which is asymptotically than constant factor, so case 1 gives

T(n)= O(n^logba) = O(n)

1. [10, Optional] Solve the recurrence equation in (2) by means of ‘**Iterative Substitution**’ method.

T(N) = 3T(2/3n) + 1

=1 +3+9t(4/3n)

1+3+32+…+3(log3/2n)

=(3(log3/2n)-1)/(3-1)

=O(3(log3/2n))

=O(3(log3/2n)/(log3/2 3/2))

=O(n(1/(log3/2 3/2))

=O(n2.71)

1. [10] Suppose we change the assignment statement for *m* (on line 5) to the following:

*m* ← *max* (1, ⎣ *n*/4 ⎦ )

Array A will be divided into three different arrays, of size 2n/3.

1. [5] Give the formula of recurrence equation for the running time, T(*n*), in this case.

From likes 1-5 it is in constant time, from lines 6-8 the function is called recursively, thus each time the size of the array is ¾ of the original array size. Concluding that formula is,

T(n) = 3(T)(3/4)n + O(1)

1. [5] Using the Master’s Theorem, decide the asymptotic bound for T(*n*) in (A).

T(n) = a(T)(n/b) + f(n), where a >= 1 and b > 1 == equation 2

Equation(3) == T(n)=3T(3n/4) + O(1) where a = 3 and b = 4/3, c = 0 and f(n) = O(1) ; making c < logba

N(logba)=n which is asymptotically than constant factor, so case 1 of the master theorem gives T(n)=O(nlog4/3 3)

=O(N3.81)

=O(n)

**Q2. [30]** **Master’s Method**

Solve the given recurrence equation by Master’s method. Justify your solution clearly.

1. [10] Assume that *n* = 2*m* where *m* ≥ 0.

T(n)=1 n<2

= 3T(n/2) + n2

Master Method = T(n)= aT(n/b) + f(n)

A >= 1 and b>1

Compare both equations

A=3, b=2, f(n)=n2 logba = log23

F(n) = n2 Nlog2 3 == nlogba= nlog2 3 = n1\*5

So it satisfies case 3 of Master Method

F(n) = O(nlogba + E) E> 0

N2 = O(n1\*5+E) E= 0\*5 > 0

2nd condition in 3rd case

A\*F(n) <= c \* f(n) with c <1

3\*f(n/2)=3\*(n/2)2 =3\*(n2/4)

= <= (3/4)\*n2

<= c\* n2 c=3/4 <1

<= c\*f(n)

Thus concluding,

T(n)= O(f(n))

O(n2)

1. [10] Assume that *n* = *3m* where *m* ≥ 0.

T(n) = 4T(n/3) + n

A =4 ,b =3, f(n)=n

nlog3 4 =n(1\*1)

So it satisfies 1st case that is f(n)= O(nlogba – E) E>0

N = O(n(1\*1-0\*1)) E=0.1 > 0

So solution is T(n) = O(nlogba)

= O(nlog3 4)

1. [10] Assume that *n* = *4m* where *m* ≥ 0.

T(n) = 16T(n/4) + n2logn

A = 16, b=4, f(n)=n2logn

Nlog4 16 = n2

It satisfies 2nd code of Master Theorem

F(n)= O(nlogba logkn k>=0

N2logn = O(n2logn) k =1

So the solution is o(n^logba logk+1 n)

= O(n2 log2 n)

**Q3. [20]** **Recurrence Tree (Handout 7)**

Assume that *n* = 3*m* where *m* ≥ 0. Using the *recursion tree* method, solve



Draw your recursion tree, specifying

1. the height of tree

Log3n

1. the number of leaves

3^nm

1. a time per level at each depth

It continues to increase at each depth by 3^(levelNumber)

1. the total time of tree (not in the asymptotic bound), and

N log3n

1. determine the **smallest** asymptotic upper bound (O) of the total time in 4.

O(n log3n)

U

/|\

M M M

/|\/|\/|\

…

K = 3km

**Q4. [10]** **Iterative Substitution**

In the recurrence equation,



1. Solve it by the iterative substitution method. Clearly show the steps of derivation of the solution.

T(n) = 2\*T(n-1)+1

N=n-1

T(n-1)=2\*5(n-2)+1

Substituting second equation for first

T(n) = 2\*[2\*T(n-1)+1] +1

= 22 \* T(n-2)+1+2

For n=n-2

T(n-2) = 2\*T(n-3)+1

Substituting fourth for third

T(n)=22 \* [2\*T(n-3)+1]+1+2

=23 \*T(n-3) +1+2+22

……

For n=2

T(2)=2\*T(1)+1

When n=1, T(n) = 2(n-1) \* T(n-(n-1))+1+2+2^2+…+ 2(n-2)

=2(n-1) \* T(1) + (1+2+22+…+2n-2)

T(1)=1

Sum of n terms is

Sn = (a(rn -1)/(r-1)

=1(2n -1)/(2-1) = 2n -1

1. Give the smallest asymptotic upper bound (O) of your solution in (1).

The algorithm will run for a maximum 2n steps. Concluding the solution of O(2n). The smallest asymptotic upper bound is O(2n)

**Q5. [10]** **Maxima Set**

By applying Maxima Set algorithm, find the maxima set from the following set of points:

{ (7, 2), (3, 1), (9, 3), (4, 5), (1, 4), (6, 9), (2, 6), (5, 7), (8, 6) }.

**Q5B [10, optional] Maxima Set in 3 Dimension**

A point in 3-dimension is a maxima point in 3-dimension in S if there is no other point, (x’, y’, z’) in S such that x ≥ x’, y ≥ y’, and z ≥ z’, and the maxima set is the set of all of those maximum points.

Design the recursive algorithm **Maxima3Set(S)** based on Divide and Conquer paradigmthat returns the Maxima Set of S, in the pseudo code.

Maxima3Set(S)

Input: a set s, of n points in the plane

Output: the set, M, of maxima points in I

If n<= 1 then

Return S

Let p be the median point in S By (x,y,z) coordinates

Let L be the set of points less than p in s

Let G be the set of points greater than or equal to P in S

M1 <- MaximaSet(L)

M2 <- MaximaSet(G)

Let Q be the smallest point in M2

For each point n, in M, do

If (x(n) <= x(q) and y(n) <= y(q) and z(n) <= z(q) ) then

Remove n from M1

Return M1 U M2