CSci 242: Algorithms and Data Structures  **Spring, 2020**

Instructor: Dr. M. E. Kim Date: 3/25 (Wed.), 2020

Due: by the end of day, April 1st (Wed.), 2020

**Home Assignment 6: Divide & Conquer and Recurrence Equation (84/100)**

**Q1. [35/30]** **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[1 .. *n*], 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. [10/10] 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) n<3

3T(2/3n) + O(1) n >= 3 -- inconsistent with the above T(n) = 3T(n3) + c, but correct.

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

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

T(n) = O(nlogba) if a > bd

A = 3, B = 3/2, D = 0

Bd = (3/2)0 = 1

A > bd / 3 > 1

Thus : t(n) = O(nlog3/23) = O(n2.7)

1. [10/10, Optional] Solve the recurrence equation in (1) 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  )

1. [3/5] Give the formula of recurrence equation for the running time, T(*n*), in this case.

we change line 5, then the recurrence equation becomes,

T(n)=3T(2/5n)+O(1)

#

1. [2/5] Using the Master’s Theorem, decide the asymptotic tight bound for T(*n*) in (A).

a=3,b=5/2,d=0

Therefore T(n)=O(nlog5/23)

# *a = 3, b = =*

*Case 1: vs. f(n) = c = O(n0)*

Since *log4/3 3 > log4/3  = 0, f(n) = c = O(n0)= O( )* -- Case 1

*( log4/3 3= =*

Thus,

**Q2. [30/30]** **Master’s Method**

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

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

T(n) = 4 \* T(n/2) + n2

Bk = 22 = 4

4 = 4

Theta(N2logn)

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

A = 6 B = 3 K = 1

Bk = 3

6 > 3

Theta (nlog36)

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

A = 4 B = 4

Bk= 43= 64

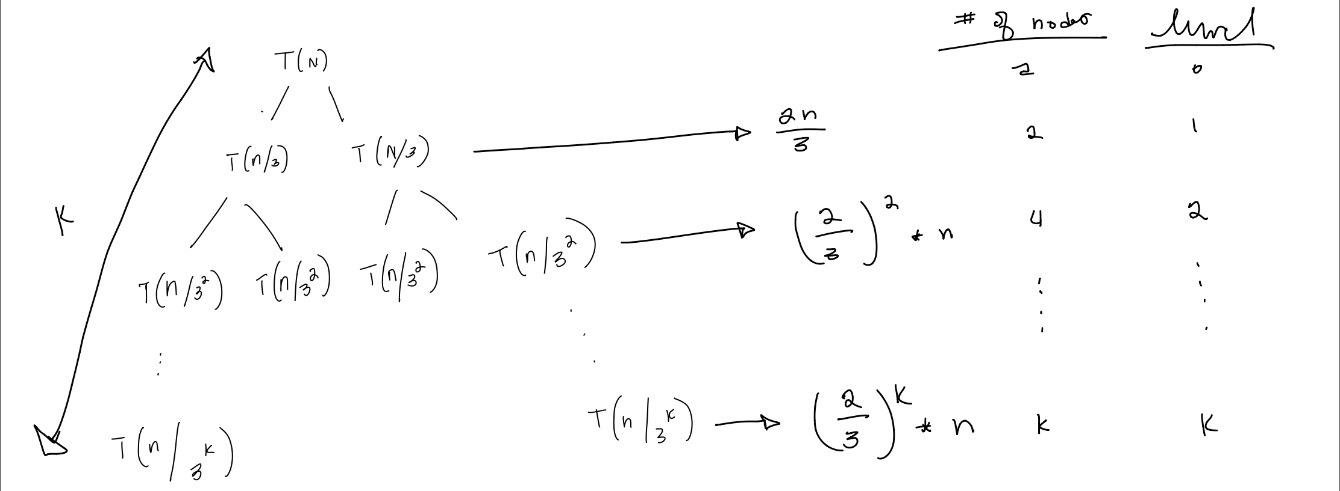
4 < 64

Theta(n3)

**Q3. [10/20]** **Recursion Tree (Handout 8)**

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

Draw your recursion tree, specifying



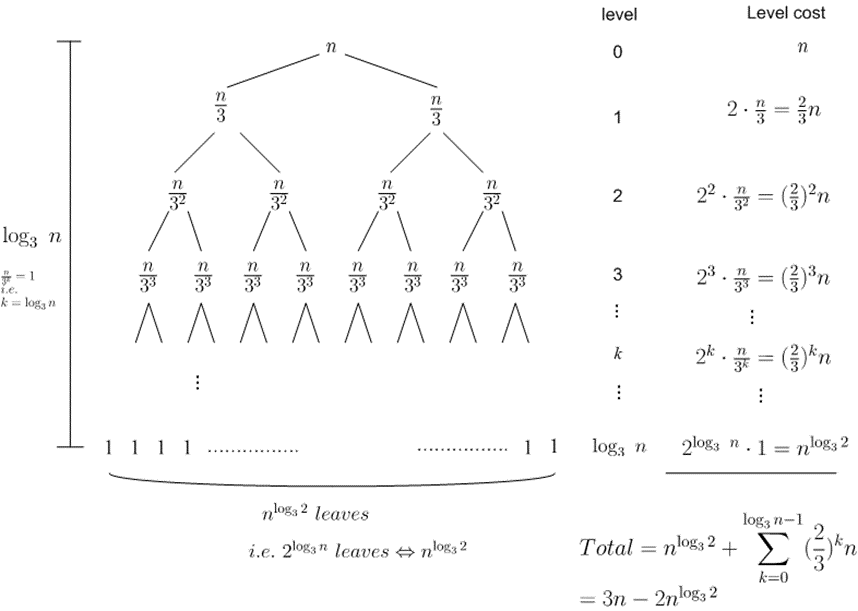
1. the height of tree
   1. M

# *log3 n*

1. the number of leaves
   1. 2m

#

1. a time per level at each depth
   1. (2/3)K \* n (K represents the level)
2. the total time of tree (not in the asymptotic bound), and
   1. = n + 2n/3 + (2/3)2 \* N + … + (2/3)m \* n
   2. = n( 1 + 2/3 + (2/3)2 + … + (2/3)m)
   3. = 3n(1 - (2/3)m + !) #
3. determine the **smallest** asymptotic upper bound (O) of the total time in 4.
   1. T(n) = O(n)



**Q4. [0/10]** **Iterative Substitution**

In the recurrence equation,

1. 0/ Solve it by the iterative substitution method. Clearly show the steps of derivation of the solution.

T(n)

= 2T(n-1) + 1

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

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

= 23T(n-3) + 22 + 21 + 1

= 2^nT(1) + 2^(n-1) + ... + 1

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

= 2^(n+1) - 1

T(n) = 2^(n+1) - 1

# T(n) = T(n-1) + n2

= T(n-2) + (n-1)2 + n2

= T(n-3) + (n-2)2 + (n-1)2 + n2

= ……

= T(1) + = 1 +

=

=

1. 0/ Give the smallest asymptotic upper bound (O) of your solution in (1).

T(n) = 2^(n+1) - 1

= 2\*2n - 1

= O(2n) # O (n^3)

**Q5. [9/10]** **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)}.

Min – Max : (1,4), (2,6), (3,1), (4,5), (5,7), (6,9), (7,2), (8,6), (9,3)

Median : (5,7)

G : points less than median : (1, 4) (2,6) (3,1) (4,5)

H : points greater than median : (6,9) (7,2) (8,6) (9,3)

Compare all points in “less than medium” to (6,9) thus it proves that the maximus set is {(6,9), (7,2), (8,6), (9,3)}

# Maxima Set: {(6, 9), (8, 6), (9, 3) }

**Q6 [0/10, optional] Minima Set in *n*-Dimension**

A point in *d*-dimension is a minimum point in *n*-dimension in S if there is no other point, (*x1*’, *x2*’, … *xd*’) in S such that *x1*  *x1*’, *x2*  *x2*’, … and *xd*  *xd*’, and the minima set is the set of all of those minimum points.

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