#____

CSE 5311 Design and Analysis of Algorithms

Midterm Exam 12:30-1:40pm, 10/20/2022 (Thursday)

IT'S A CLOSED BOOK EXAM, NO CHEATSHEET OR COPIES ARE ALLOWED DURING THE EXAM.
DO NOT USE ANY ELECTRONIC DEVICES.

[TRY YOUR BEST TO WRITE YOUR ANSWERS on the question sheets]

Name: Sai Rohit Kolayan **Student ID:** 100 2070724

Part A: Fill in the blank to answer the questions: (4 X 10 = 40 points)

1. The statement "In the merge-sort recursion tree, roughly the same amount of work is done at each level of the tree." is Correct (correct or wrong).
2. The solution to the recurrence T(n) = 2T(n/8) + $\sqrt[3]{n}$ is $\Theta(n^{1/3}\log\log(n))$.
3. The recurrence for the best case of QuickSort with an input size of n is T(n) = $\Theta(n)$ if all No. are Sorted.
4. A divide-and-conquer algorithm of powering a number (a^n, where n is an integer) has a time complexity of $O(\log(n))$
5. The worst-case running time for building a binary search tree of n elements is $O(n)$.
6. Given an open-addressed hash table that is 50% full, the expected number of probes in an unsuccessful search is $p=1-1/2=(1/2)$ So the No. of probes $= 1/(1-i) = 1/(1-1/2)$
7. "The best worst-case running time that we've seen for a sorting algorithm is O(nlgn)". This statement is false (true or false).
8. Running time of the randomized order statistics algorithm with an input of n elements is upper bounded by $\text{fun}(A, q, \delta-k)$ k = randomization.
9. The load factor of a hash table refers to Number of Elements / Slots (i.e. definition of load factor).
10. The objective of randomly building a binary search tree of n elements is to bound $O(1)$ to O(lgn).

Part B: Solve the problems [write your answers on the question sheets if possible]

(1) Use recursion tree to guess a solution of recurrence T(n) = T(n/5) + T(3n/4) + n, and then use induction to prove it. (10 points)

Sol)

Given $T(n) = T(n/5) + T(3n/4) + n$ ↳①

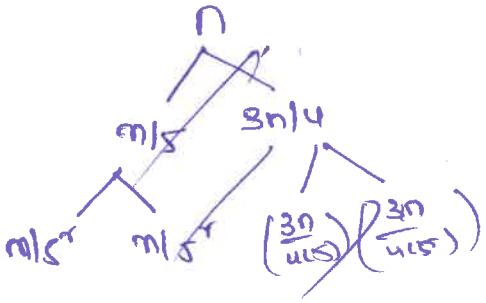
$(n = n/5)$

$T(n/5) = T(n/5^2) + T(3(n/5)/4) + n/5$

$= T(n/5^2) + T(3n/4(5)) + n/5$ ↳②

$T(3n/4) = T(3(3n)/4(5)) + T(3^2n/4^2) + 3n/4$

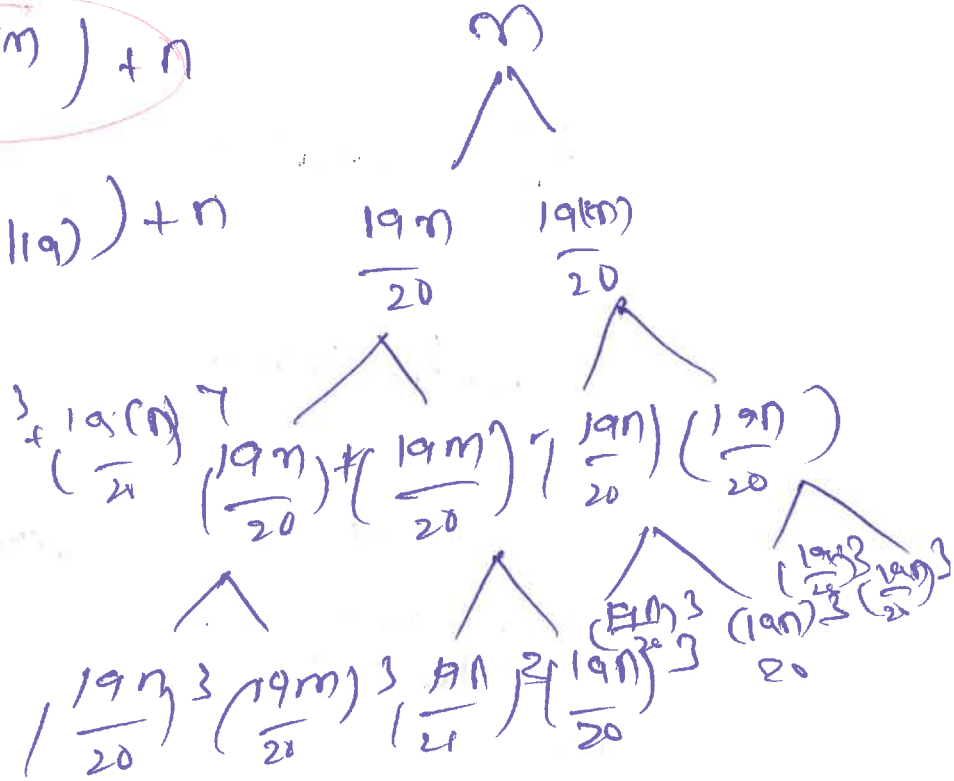
$= T(3n/4(5)) + T((3n)^2/4^2) + 3n/4$ ↳③



18.01) $T(n) = T(n/5) + T(3n/4) + n$

$= T\left(\frac{4n+15n}{20}\right) + n$

$T(n) = T(n/(20/19)) + n$



$\Rightarrow \dots$

$\Rightarrow c\left(\frac{n-4}{2}\right)$

$\Rightarrow O\left(\frac{n-c4}{2}\right)$

$\Rightarrow O(n)$

#____

Because for i to j the slots filled are can hold Equal Chance of filling it again so the mean(n) have Equal probability So we can Say $h(i) = h(j)$

(2) [10 points] Suppose we use a hash function h to hash n distinct keys into an array T of length m. Assuming simple uniform hashing, what is the expected number of collisions? and prove your conclusion. More precisely, what is the expected number of elements of the set $\{\{i, j\} : i \neq j$ and $h(i) = h(j)\}$? (10 points)

Given Hash function(h) of (n) distinct keys Array(T.) = m.

Say Hash function = H(n) (i mod n).

$\sum H(n)\,dn = H(n_1) + H(n_2)\,dn + H(n_3) + H(n_4) \ldots$

* The No. of collisions of the Hash function probably (1/2)

(3) Show how to sort n integers in the range 0 to $n^3 - 1$ in O(n) time. (10 points)

Given n integers in radix Sort $\frac{b}{r}(n+2^r)$ Now $(n \to n^3) \Rightarrow b = \log^3 n$,

$r = \log n$ => Now in Equation we keep the $\left(\frac{\log^3 n}{\log n}(n+2^{\log n})\right)$

=> $3\log n\,(n+2^{\log n}) = O(3(n+2^{\log n}))$ $O(3n+2^{(3)})$

=> $O(n)$

(4) Radix sort takes multiple passes, each on a digit position with an auxiliary stable sort, starting from the least-significant digit position,

(a) What is stable sorting? Name an example stable sorting algorithm. (7 points)

a)

Stable Sorting : if we Sort an array By not comparing algorithm and when Sorting the index of the first position comes first then the second index position

Ex (3, 4, 1, 2, 1, 5) when we Need the array in (1, 1, 2, 3, 4, 5) Such kind of

index 0 1 2 3 4 5 index 2 4 3 0 1 5

of Sorting is Called Stable Sorting (Insertion Sort, Counting Sort) are two Best Examples

(b) Is quicksort a stable sorting algorithm? (5 points)

b) NO Quicksort is not a Stable Sorting algorithm Because Quick Sort have different kind of choosing of the pivot ① first Element as pivot ② last Element as pivot and ③ Random Number as pivot

* So according when we choose the pivot the positions of the numbers changes

* Take above Example 3, 4, 1, 2, 1, 5

index 0, 1, 2, 3, 4, 5

[3, 4, 1, 2, 1, 5]

Index = 0, 1, 2, 3, 4, 5

pivot as first Element

k = 3

i, j

3, 4, 1, 2, 1, 5 (3 > 4) jtt

index k: 0, 1, 2, 3, 4, 5 (3 > 1) * itt swap(i, j)

1,	4	3	2	1,	5
2	1	0	3	4	5

index:

* Here (1) is having 2nd index in Original array

pivot as last Element

k = 5

i, j

3, 4, 1, 2, (1) (5) 5 > 3

0, 1, 2, 3, 4, 5

k = 2

(1 < 3) (True) itt swap(i, j)

1,	4	1	2	3	5
4,	1	2	3	0	5

index

* Here (1) is having 4th index

* Hence we say Quick Sort is Not a Stable Sorting.

(4c) When we perform the unstable Sorting ie instead of rightmost Significate we use left most significate then

* Note when we do that in unstable way from left to right the Sorting is Not done.

329	329	329	720
457	355	720	355
657	457	436	436
839	436	839	457
436	657	355	657
720	720	457	329
355	839	657	839

* Hence auxiaby unstable Sorting of Radix Sort is always give's wrong option.

#____

(c) The following graph shows use of the radix sort on a list of integers with three passes. use it as example to demonstrate that if an auxiliary unstable sort was applied, the sorting would be wrong. (8 points)

-3

Sol) When we do unstable sorting ie Right most Significate Element (Left to right) left from

329	720	720	329
457	355	329	355
657	436	436	436
839	457	839	457
436	657	355	657
720	329	457	720
355	839	657	839

329
457
657
839
436
720
355

*So under the (4).

(5) We have a list of integers. Their total number of digits is n. Show that they can be sorted in O(n) time in each of the two cases:

-8

a. they have the same number of digits; (10 points)
b. they may have different numbers of digits [Note that (b) is a **bonus** question worth 10 points]

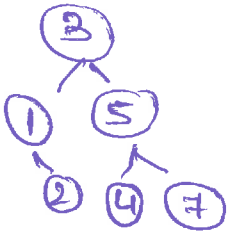
Sol) Given the a list of total No. of digit is (n)

We Need to Show that if they have Same number of digits the Sorting time Complexity is O(n)

Say: When we Sort an algorithm (array) using Binary Tree algorithm we can Sort the array in O(n)

Ex* let's we Same digits : [3, 5, 1, 2, 4, 7]

* So the list is Sorted in O(n) complexity.



5b) Take the different digits [30, 54, 12, 42, 24, 5]

* Again if we sort the array using Binary Tree we can solve By O(n) times irrespective of the Number of digits.