CS201: Data Structures and Algorithms **Autumn 2020**

Quiz 1 & 2 - Phase 2 - Questions

IIT Dharwad *Computer Science and Engineering*

Instructions:

- There are 22 questions in this document. Answer any 8 questions (including the question with the serial no. same as your team's serial no.)
- Due on 4th Dec 11:59 pm. With 25% penalty, you can submit up to 9 am on 5th December.
- If you answer more than that, we will consider only first eight of them.
- Every question has equal maximum marks. Please see attached evaluation scheme for the details of evaluation.
- There may be differences between the question you submitted and the the question with number same as your team's serial no. Make sure that you answer the question given here (rather than the question you submitted).
- While answering a question, clearly mention the question number. There will be a penalty of 10% if a question number is mentioned wrongly.
- For every algorithm design problem, the better the worst-case time complexity of the algorithm you design, the better the marks that you earn.
- For every algorithm design problem, it is expected that you briefly explain the correctness of the algorithm and derive the worst-case time complexity.
- Prepare the answers using any text editor. Please submit a pdf of the document after naming it with <team-no>.pdf, where <team-no> is the serial number assigned to your team. There will be a penalty of 2.5% if you violate the naming policy.

1. (a) Show the execution of merge sort algorithm for the array $\{38, 27, 43, 3, 9, 82, 10\}$.

   (b) What is the time complexity of merge sort in all the three cases (worst, average, and best)? Support your statement with a mathematical proof. Use the version of merge sort explained in CLRS 2.3.1 for your answer.

2. Suppose you are given a hash table, implemented using linear probing. The hash function used is $h(x) = x\%9$. The hash table is

index	0	1	2	3	4	5	6	7	8
value	9	18		12	3	14	4	21	

   (a) In which order could the elements have been added to the hash table? There are several correct answers, and you should give all of them (among the options given below). Assume that the hash table has never been resized, and no elements have been deleted yet.

      i. 9, 14, 4, 18, 12, 3, 21.

      ii. 12, 3, 14, 18, 4, 9, 21.

iii. 12, 14, 3, 9, 4, 18, 21.

iv. 9, 12, 14, 3, 4, 21, 18.

v. 12, 9, 18, 3, 14, 21, 4.

(b) Remove 3 from the hash table and write down how it looks afterwards.

(c) If we want a hash table that stores a set of strings, one possible hash function is the string's length, $h(x) = x.length$. Is this a good hash function? Explain your answer.

3. Manav runs an ice-cream shop that currently focuses on 3 flavors: Vanilla, Chocolate, and Strawberry. His shop is famous for mixing these above flavors to make new "ice-cream combos" recipe. An "ice-cream combo" is made by mixing v_req, c_req and s_req scoops of Vanilla, Chocolate, and Strawberry respectively.

   Manav currently has $curr\_v$ scoops of Vanilla, $curr\_c$ scoops of Chocolate, and $curr\_s$ scoops of Strawberry ice-cream in the shop kitchen currently. Besides, his ice-cream supplier Pratik has all three ingredients, the prices are v_cost, c_cost, s_cost for a scoop of Vanilla, Chocolate, and Strawberry ice-cream, respectively.

   Manav has with him a total of $total\_money(1 \leq total\_money \leq 10^{12})$ rupees. He would like to know the maximum number of "ice-cream combos" he could make using the current ingredients in his shop and buying the required ingredients using the money he has. Design an algorithm to print the maximum number of "ice-cream combos" that Manav can make with all these constraints. Write pseudocode and derive the worst-case time complexity of your algorithm. Briefly explain the correctness of your algorithm.

   Note: One can assume that Manav cannot slice any of the ice-cream scoop. Pratik has an unlimited number of ice-cream scoops for each flavour. All input values are non-negative integers.

   Please find below some sample input and output. This is only for better understanding of the problem. Your algorithm can assume that the variables (such as v_req etc.) have the input values - you don't have to write pseudocode to read the input values.

   INPUT:

   The first line of the input contains three integers v_req, s_req, c_req.

   The second line contains three integers $curr\_v$, $curr\_s$, $curr\_c$ (Current available scoops of Vanilla, Strawberry and Chocolate, respectively)

   The third line again contains three integers v_cost, s_cost, c_cost (Cost of 1 scoop of each flavor). Finally, the fourth line contains integer $total\_money(1 \leq total\_money \leq 10^{12})$ - the amount Manav has in Rupees.

   OUTPUT: Maximum number of ice-cream combos Manav can make. If he can't make any combos, the algorithm should return 0.

   Write an algorithm for the above problem.

   Test Case 1:

   INPUT 3 2 1

   6 4 1

   1 2 3

   4

   OUTPUT

   2

   EXPLANATION:

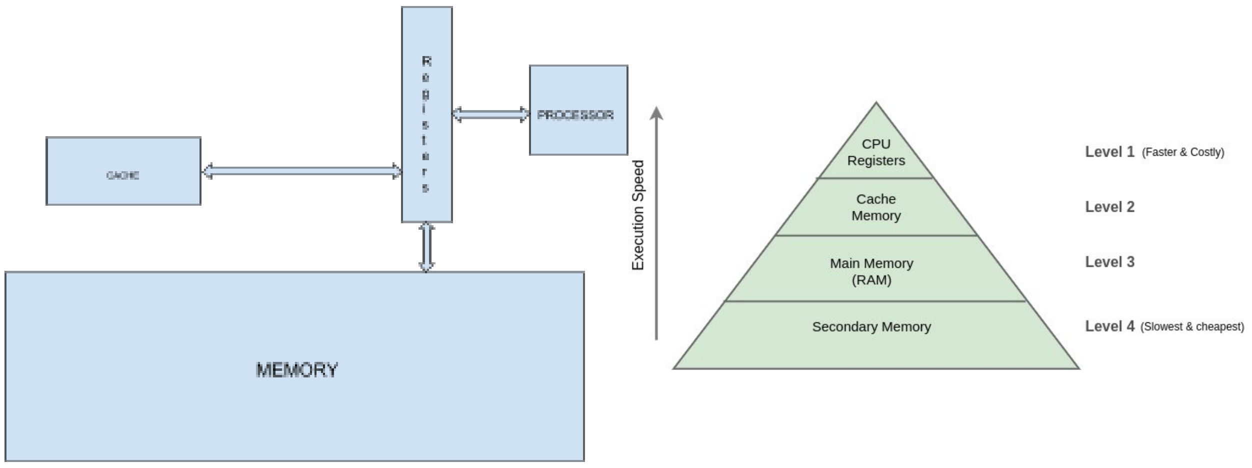

Figure 1

Using the scoops already present in the shop, he can make one complete ice-cream as $6 \geq 3$ and $4 \geq 2$ and $1 \geq 1$. So, after making 1 ice cream, $(curr\_v, curr\_s, curr\_c)$ = (3,2,0). For the second ice-cream he is short of chocolate scoops as he has 0 chocolate scoops now and for each combo 1 chocolate scoop is necessary. So, he will buy 1 chocolate scoop from the market using $total\_money$. Cost of 1 chocolate scoops is 3 rupees so $total\_money$ left is $4 - 3 = 1$ rupee.

Now, $(curr\_v, curr\_s, curr\_c) = (3, 2, 1)$ Now, he can make one more ice-cream as $3 \geq 3$ and $2 \geq 2$ and $1 \geq 1$. So remaining items $(curr\_v, curr\_s, curr\_c) = (0, 0, 0)$ and $total\_money = 1$ rupee. Can he buy something more? - No because with remaining money he can maximum buy 1 vanilla scoop which will not suffice. Thus, at max he can make 2 complete ice-creams satisfying the requirements.

4. Actual time analysis of sorting algorithms:

   - Background:

     At times, the time complexity of the program may not accurately represent its actual execution time. Sometimes, when a program is executed by a processor the time taken for it to access memory elements is much greater than time required for other operations. The objective of this question is to use a simplified processor memory model and analyze the time taken by input/output operations to sort an array using Quicksort.

     A computer is architectured such that there are different memory levels: The main memory (RAM), caches, and registers. Main Memory is a very large storage location from which data can be read or written to, but it has a large delay associated with it. To overcome this, a faster storage location called cache is used which has comparatively lesser delay associated with them. A processor can only work upon the values which are stored in registers. So to use any value, a processor needs to fetch the value from cache or from the main memory (if not present in cache) and then write it into a register. See Figure 1.

     Operation for read and write from memory is $100ns$, and from cache it is $10ns$. The computer is designed such that it has unlimited memory space but it has a limited number of cache which is 1 unit.

   -

Memory Type	Delay Associated	Lifetime of Data	Size Limit
Registers	0 ns(no delay)	1 iteration only	As many as needed
Cache	10ns	Lifetime of program	Only 1 integer value can be stored
Main memory	100ns	Lifetime of program	Infinite

- Question: For Quicksort, write down the sequence of operations along with the time delays and calculate the total time delay for sorting the input array [5, 2, 6, 1] in ascending order. The better the time delay, the better the marks you receive. Please make sure that you follow the algorithm mentioned in Lecture 9.

  You are free to use cache in any way you want. Also use as much main memory as needed.

- Assumptions and notes
  - Assume that there is no delay associated with accessing values from the registers.
  - Any comparisons / operations like addition, subtraction, multiplication etc are instantaneous (no delay).
  - please assume that the value from the register can only be used once for comparison/operation i.e., once used for comparison/operation, a value is needed to be read again from cache/the main-memory into the register for another comparison/operation but a value in a register can be used to write into cache and the main-memory. (values fetched from the main-memory or the cache are always stored in the register and then the comparisons are done on them.)
  - Please assume that working of registers is automatic and don't bother storing values in them explicitly, just fetch values from cache or the main memory and operate on them directly. Also please don't bother storing addresses, use any address you want directly.
    For example if you want to store value 5 at address 3, just use:
    "writing 5 in memory[3] - 100ns"
    or "fetch 4 from memory[2] - 100ns" and then operate on them "Compare ($4 < 6$)".
  - All the comparisons must be carried out according to the algorithm described even if the array seems sorted.
  - Temporary variables (for eg. itinerary variables i, j, temps) don't need to be described explicitly and can be assumed to be available all the time in registers. However you are advised to mention them to show the flow of the program and to make it easier for the examiner to check.
  - Don't bother storing addresses, assume that all addressing is taken care of and use the addresses directly. Also at all times it is known which memory is free and which one is not.
  - The first four elements of main memory are taken by the input array however all other memory is free for use.

  The current state of memory and cache is represented using:

  $\{5, 6, 2, 1, ...\}$: [1] $\longrightarrow$ Represents 5, 6, 2, 1 in the main memory array starting from index 0, and [1] represents that 1 is stored in cache.

- Example 1: Fibonacci series (0 and 1 are stored in memory[0] and memory[1]), store the next 6 elements of the series after them.

Algorithm 1 Fibonacci series

```
1: Fetching 0 from memory[0] - 100ns
2: Fetching 1 from memory[1] - 100ns
3: Add $(0 + 1 = 1)$ - 0ns
4: Writing 1 to memory[2] - 100ns
       $\{0, 1, 1, .....\} : [-]$
5: Writing 1 to cache - 10ns
       $\{0, 1, 1, .....\} : [1]$
6: Fetching 1 from cache - 10ns
       Here instead of writing into memory and fetching from memory (200ns), we have written it into
   cache and fetched from it (20ns).
7: Fetching 1 from memory[0] - 100ns
8: Add $(1 + 1 = 2)$
9: Writing 2 to memory[3] - 100ns
       $\{0, 1, 1, 2, .....\} : [1]$
10: Writing 2 to cache - 10ns
       $\{0, 1, 1, 2, .....\} : [2]$
11: Fetching 2 from cache - 10ns
12: Fetching 1 from memory[2] - 100ns
13: Add $(1 + 2 = 3)$
14: Writing 3 to memory[4] - 100ns
       $\{0, 1, 1, 2, 3, .....\} : [2]$
15: Writing 3 to cache - 10ns
       $\{0, 1, 1, 2, 3, .....\} : [3]$
16: Fetching 3 from cache - 10ns
17: Fetching 2 from memory[3] - 100ns
18: Add (2+3 = 5)
19: Writing 5 to memory[5] - 100ns
       $\{0, 1, 1, 2, 3, 5, .....\} : [3]$
20: Writing 5 to cache - 10ns
       $\{0, 1, 1, 2, 3, 5, .....\} : [5]$
21: Fetching 5 from cache - 10ns
22: Fetching 3 from memory[4] - 100ns
23: Add $(3 + 5 = 8)$
24: Writing 8 to memory[6] - 100ns
       $\{0, 1, 1, 2, 3, 5, 8, .....\} : [5]$
25: Writing 8 to cache - 10ns
       $\{0, 1, 1, 2, 3, 5, 8, .....\} : [8]$
26: Fetching 8 from cache - 10ns
27: Fetching 5 from memory[5] - 100ns
28: Add $(5 + 8 = 13)$
29: Writing 13 to memory[7] - 100ns
       $\{0, 1, 1, 2, 3, 5, 8, 13, .....\} : [8]$
```

Total time: 1400ns

- Example 2: Bubble sort the given numbers in ascending order: 8,0,4

Algorithm 2 Bubble Sort

```
1: Fetching 8 from memory[0] - 100ns
2: Fetching 0 from memory[1] - 100ns
3: Compare $(8 < 0) \longrightarrow$ Swap
4: Writing 0 to memory[0] - 100ns
5: Writing 8 to cache - 10ns
       $\{0, 0, 4, ...\} : [8]$
6: Fetching 4 from memory[2] - 100ns
7: Fetching 8 from cache - 10ns
8: Compare
9: Writing 4 to cache - 10ns
10: Writing 8 to memory[2] - 100ns
       $\{0, 0, 8, ....\} : [4]$
11: Fetching 0 from memory[0] - 100ns
12: Fetching 4 from cache - 10ns
13: Compare
14: Writing 4 to memory[1] - 100ns
       $\{0, 4, 8, ...\} : [4]$
```

Total time:740ns

5. Let $F(n)$ denote the n^{th} Fibonacci term, defined in the following way for all $n \geq 0$ with $F(0) = 0$, $F(1) = 1$, and any $F(n)$ where $n \geq 2$ is given by $F(n) = F(n-1) + F(n-2)$. Consider a function $f(n) = log_2(F(n))$ and another function $g(n) = n$, prove that $f(n) = O(g(n))$. Note: You are not allowed to use any closed-form formula for $F(n)$.

6. You are given 3 sets S_1, S_2, S_3 of integers, where the cardinality of each of the set is x and the elements of these sets can have a maximum value of x^p and minimum value of $-x^p$ where p is a constant integer greater than 1. Give an algorithm to check whether there exists three numbers one from each set such that the sum of the three numbers is 0. The worst-case time complexity of the algorithm must be $O(x^2)$. Note: Use of Hashing is not allowed.

7. Given 2 strings X and Y, design an algorithm to find a Longest Common Sub-sequence(LCS) such that order of the algorithm is not exponential in the length of X or Y. Write a pseudocode, derive the complexity, and briefly explain the correctness of the algorithm.

   Example: X is the string CATCGA and Y is the string GTACCGTCA then LCS for X and Y is CTCA or TCGA

8. Suppose you are given 2 sets of numbers $S_1 = \{27, 13, 35, 89, 420, 69\}$ and $S_2 = \{3, 55, 81, 13, 29, 120\}$. You are now required to insert all elements from each of the given sets into separate hash-tables each of size 13. Assume that you are inserting the numbers in the order given in the set. Draw the final Hash-Tables for each of the given sets under collision resolution by linear probing and by double hashing (total four tables required). In each case, mention how many collisions happened.

   Note: Use hash function(s), $h'(x) = x$ for linear probing and $h_1(x) = x$ and $h_2(x) = 7 - (x\%7)$ for double hashing.

9. Kokila has two bahus(daughter-in-laws), Rashi and Gopi. Kokila wants to know who is the better daughter-in-law, Rashi or Gopi. So, she gives both of them, two different tasks to complete which are as follows:

   - Task for Gopi:

There are n workers in Kokila's house. Each worker has completed different number of tasks (i.e., no two workers have same number of tasks completed). They are standing in a row. Gopi is given an array p of size n , where $p[i]$ denotes the number of tasks completed by i^{th} worker. Now Gopi has to distribute salary among the n workers.

The workers are standing in a row such that worker with $p[0]$ completed tasks is standing in the left end of row and worker with $p[n-1]$ completed tasks is at the right end of the row.

Gopi has to start distributing salary from the left end of the row. For some reason, the workers say that they will accept salary equal to the number of tasks completed by the his/her best friend. A best friend of a worker A is the worker who is nearest to A among all workers standing left to A (with respect to the viewer) and completed less number of tasks than A. For example, assume that 9, 3, 5, 4 are the number of tasks completed (the array p) by the workers A, B, C, and D respectively. Then A has no best friend, B has no best friend, C's best friend is B, and D's best friend is B. Design an algorithm to help Gopi distribute the salary to all workers and provide the salary info (the array of distributed salary) to Kokila. Write the pseudocode, derive the worst-case complexity, and briefly explain the correctness of the algorithm. The better the worst-case time complexity of your algorithm, the better the marks will be.

- Task for Rashi: Kokila asks Rashi to go to the market to buy grocery. After reaching the grocery shop, Rashi realises that due to COVID, the shopkeeper has made certain number of slots in front of the shop, where the customers should stand. Rashi just came right now, so she does not know how many slots are there to stand. So, the shopkeeper provides this information, there are k people in front of the shop, and he also provides a binary representation of a number 'x'.

  It is known(given) that the binary representation of 'x' will have 'n' 1's in it (i.e., there will be 'n' bits which are '1' and the remaining bits will be '0'). The 'k' people can stand in any of the i^{th} slot if the i^{th} bit of binary representation of 'x' is '1'.(Also given that $k \leq n$).

  Also one available slot (i.e., with bit with value = '1' in the binary representation) can have at most 1 person. (For example: we are given binary representation of x as '10001000100', then here n=3 (as there are 3 bits equal to '1'), then this means that the k person can stand in these three positions (namely, i=1 , i=5 and i = 9) and one available slot (i.e. bit with value '1') can have at most 1 person.) (Here 'n' is the same as the one mentioned in task for Gopi)

  Due to COVID, the customers don't want to stand very close to each other. They want to stand in a way such that the minimum distance among any of the two customers is maximised. (The distance between two slots is the difference between their indices). For example, if the binary representation of x is 10100001001 and if $k = 3$, then the answer must be $[1, 8, 11]$ or $[3, 8, 11]$ both maximizes the minimum of all distances (i.e., 3). Help Rashi design an algorithm to find the largest possible minimum distance between any of the two customers. Write pseudocode, derive worst-case time complexity and briefly explain the correctness of the algorithm. The better the worst-case complexity, the better the marks will be.

Kokila will get to know who the best bahu(daughter in law) is depending on who completes her task first (i.e. whoever takes less time to complete their tasks - the comparison is done based on the worst-case time complexity of their algorithms). So Gopi and Rashi will both try to complete their tasks as fast as possible.

Rashi and Gopi consulted you to solve their problem. Since you do not have time to solve both you agreed to solve one problem. So, your task is to design and present the fastest (possible to you) algorithm either for Gopi's task or for Rashi's task. Explain briefly why your algorithm is correct and also derive the worst-case time complexity.

10. Summer vacation has started. So Ananya and Pj want to play, but their mom doesn't think so. She says that they have to read some amount of books before all entertainments. Ananya and Pj will read each book together to end this exercise faster. There are n books in the family library. The i^{th} book

is described by three integers: t_i — the amount of time Ananya and Pj need to spend to read it, a_i (equals 1 if Ananya likes the i^{th} book and 0 if not), and b_i (equals 1 if Pj likes the i^{th} book and 0 if not). So they need to choose some books from the given n books in such a way that:

- Ananya likes at least k books from the chosen set and Pj likes at least k books from the chosen set.
- the total reading time of these books is minimized (they are children and want to play as soon a possible).

The set they choose is the same for both Ananya an Pj (it's shared between them) and they read all books together, so the total reading time is the sum of t_i over all books that are in the chosen set. Design an algorithm to help them. Write pseudocode, derive worst-case time complexity and briefly explain the correctness of the algorithm. The better the worst-case time complexity, the better the marks will be.

The following example is only for understanding the problem. No need of implementing your algorithm.

Input:

0^{th} line - (n and k)

the i^{th} line contains three integers t_i, a_i, and b_i.

Example:

```
8 4
7 1 1
2 1 1
4 1 0
8 1 1
1 0 1
1 1 1
1 0 1
3 1 0
```

OUTPUT

```
2 3 5 6 7 8 (total time: 12)
```

11. There is a man named Harshad who likes mangoes a lot. Every year during Summer he use to drive to his farm in his village. For this drive he used to spend an amount of C(n) Rupees where n is age of his car.

    $C(n) = C(\sqrt{n}) + 1$, $C(2) = 1$.

    His expenses have been increasing still his love for mangoes could not stop him. When he went to his Farm his mangoes tree looked so good with full of mangoes. He wanted to relish the taste of it and searched for some mangoes but he didn't find any mango on the ground. Though he is about to cross his fifties, he decided to climb the tree as he used to do in his childhood and collect some mangoes. He took a thin basket where he could stack one mango over other and so on. He saw the tree and decided to only collect mangoes along one branch (only along one path from trunk to leaves). The tree looked similar to a binary tree. Each node has at most 2 branches. The mangoes are present at each node (Similar to the elements present nodes and leaves). He collected mangoes along one branch which has k nodes. He got down the tree with a stack of mangoes. He started to pop out the mangoes one by one. He noticed that the mangoes which are at the bottom of the basket are crushed by the weight of the mangoes above it. Similarly, the second bottom mango by the mangoes above it and so on. He assigned a value (in rupees) of t for each mango where t is the index of the mango in the stack (Bottom mango has $t = 1$ and mango above it has $t = 2$ and so on). Number of nodes k is equal to the age of the tree.

He then started to think is it profitable to come here every year if the mangoes become spoiled like this? You know Harshad is good at Stock Market but he is poor in asymptotic analysis. Help Harshad in finding the answer (Is it Profitable or not? - Profitable if the cost of driving is asymptotically smaller than value of mangoes and not profitable otherwise) and give him a good explanation such that he understands.

Note: The Car is bought on the same year as the mango tree is planted.

12. Consider the two algorithms mentioned below

    Note: In "**for** i *from* 0 *to* $n-1$", i takes the value $n-1$ at last
    The + in print indicates concatenation.

Algorithm 3 Pseudo code

```
function MYFUNCTION($n$)
    $sum \leftarrow 1$
    for $i$ from 1 to $n-1$ do
        $sum \leftarrow sum + pow(n, i)$                ▷ More about pow function given below
    end for
    $last \leftarrow (n-1) * sum$
    for $i$ from 0 to $last$ do
        $z \leftarrow$ NEWBASE($i, n$)                ▷ More about NEWBASE function is given below
        Copy the array $z$ into a new array $arr$
        Sort the array $arr$ using radix sort
        Create a new array $order$ of length $n$
        for $j$ from 0 to $n-1$ do
            $order[j] \leftarrow j$
        end for
        $flag \leftarrow$ COMPARE($order, arr$)        ▷ More about COMPARE function is given below
        if $flag = true$ then
            for $j$ from 0 to $n-1$ do
                print $(z[j]+1)$+" "
            end for
            print newline
        end if
    end for
end function
```

More about functions

(a) $pow(n, i)$ where n, i are positive integers returns an integer n^i. The worst time complexity of it $O(i)$.

(b) NEWBASE(i, n) where n, i are positive integers returns an array of length n that contains digits of number i in base n as its elements. The worst time complexity of it is $O(n)$.

(c) COMPARE($array1, array2$) where $array1$ and $array2$ are two arrays, returns boolean true if both have same length and have same elements in their respective positions. Else returns false. The worst time complexity of it is $O(n)$.

Note: The functions mentioned above could have been written explicitly in above pseudo code, but it would only lead to confusion and would be misleading so we have avoided writing them

Algorithm 4 Pseudo code

```
function PRINTARRAY($arr$,$n$)
    for $i$ from 0 to $n-1$ do
        print $arr[i]$+“ ”
    end for
    print newline
end function

function MYFUNCTION($arr$, $size$, $n$)
    if $size = 1$ then
        PRINTARRAY($arr$,$n$)
        return
    end if
    for $i$ from 0 to $size-1$ do
        MYFUNCTION($arr$, $size-1$, $n$)
        if ($size\%2 = 1$) then
            $swap(a[0], a[size-1])$
        else
            $swap(a[i], a[size-1])$
        end if
    end for
end function

$n \leftarrow$ positive integer input from user
Create an array $arr$ of length n
for $i$ from 0 to $n-1$ do
    $arr[i] \leftarrow i+1$
end for
MYFUNCTION($arr$, $n$, $n$)
```

- The above mentioned two pseudo codes essentially do the same task, but one is more efficient than other. What is the task that they perform?
- Find the worst time complexity of above pseudo codes in terms of n and based on time complexity which of them is more efficient?

13. Design an algorithm to concatenate the numbers from the given set to form the second largest number (possible in this manner). Briefly describe the correctness of your algorithm. Derive its worst-case time complexity.

    Example

    Input: $\{45, 97, 301, 2, 864, 23, 87\}$.

    Output: 978786445301223.

14. Consider the evaluation of polynomial equation $p(x) = a_0 + a_1x + a_2x^2 + ... + a_nx^n$ at point x.

    (a) Show that the following simple routine known as Horner's rule, does the job and leaves the answer in z.

i. $z = a_n$.
ii. for $i = n - 1$ down to 0.
iii. $z = zx + a_i$

(b) How many additions and multiplications does in this routine use, as a function of n?

15. A company wants to find the median of salaries of its employees. You have been given 2 lists. One list holds all distinct salary values for the employees in the company (unsorted). The other list is such that the i^{th} element holds the number of employees who have the salary given in the i^{th} index of the first list. The median of the salaries is such that if the salaries would be sorted in ascending or descending order, the salary which lies in the middle position would be the median. Design an algorithm which, in the worst case, finds the median given the 2 lists as input in linear time ($O(n)$), where n = size of the lists. For simplicity assume E(number of employees) is an odd numbers.

    For example:

    List1 = [60000,30000,25000,50000,80000]

    List2 = [9,14,6,2,8]

    The total number of employees is 9 + 14 + 6 + 2 + 8 = 39 = E

    Sorting the salaries, we have,

    25000 repeated 6 times,

    30000 repeated 14 times,

    50000 repeated 2 times,

    60000 repeated 9 times and

    80000 repeated 8 times.

    The median salary would be the salary in the (E + 1)/2 th position if E is odd. In this case, we choose the 20 th position which is 30000 here.

16. (a) Explain the time complexity of heap sort algorithm.
    (b) Why heap sort is not stable?
    (c) Imagine that instead of using an array to represent the heap, we use a singly linked list. Why might this be inefficient?

17. Consider the following modification to merge sort. Given an array of size n, we perform an insertion sort on (n/k) blocks (disjoint) of length k each and then merge the sorted blocks using the standard merging mechanism(pairwise merging as in merge sort), where k is a value to be determined.
    (a) Show that the total cost for performing insertion sort on all the n/k blocks takes $\theta(nk)$ worst case time.
    (b) Find the total cost for merging these n/k blocks to obtain a sorted array of size n.
    (c) What should be the largest value of k as a function of n for which the modified algorithm has the the same running time as standard merge sort, in terms of $\theta-$notation.

18. (a) Explain in detail about Open Addressing using Linear Probing.
    (b) Consider a Max-Heap. Design an algorithm to delete the entry having the largest $(key + value)$ sum (assume that keys are 1,2,...) from a given heap and restore the-max heap properties for the remaining list. Derive the time complexity of your algorithm.

19. Professor Bunyan thinks he has discovered a remarkable property of binary search trees. Consider a path from the root to a leaf in a binary search tree. Consider three sets: A, the keys to the left of the path; B, the keys on the path; and C , the keys to the right of the path. Professor Bunyan claims that if A, B, and C are nonempty sets then for any three keys $a \in A$, $b \in B$, and $c \in C$ it must satisfy that $a \leq b \leq c$. Is this true? If yes, prove it. If not, give a counter-example. If you are giving a counter-example, the smaller the example the more the marks will be.

20. Given pre-order of BST is $[k, a, i, g, b, j, c, e, h, f, d]$ and post-order is $[g, b, i, c, e, j, a, f, d, h, k]$ where $a, b, c, d, e, f, g, h, i, j, k$ are numbers which satisfies the BST property.

    (a) Is it possible to determine unique BST from given pre-order and post-order? If yes ,explain how we can draw BST from given pre-order and post-order? if no, explain why we cannot get unique BST from given pre-order and post-order?

    (b) Is it possible to determine unique BST from given pre-order and post-order, given all internal nodes have exactly two children? If yes, explain how we can draw BST from given pre-order and post-order? if no, explain why we cannot get unique BST from given pre-order and post-order?

21. Given to you is a list of inputs, and a hash function. Draw a hash table of appropriate size corresponding to these inputs and hash function. You should resolve collisions by chaining.

    Hash Function: $H(x) = x\%12$.

    Inputs: $71, 22, 25, 84, 10, 3, 37, 77, 1, 40, 45, 78, 30, 11, 64, 87, 76, 17, 80, 42$

22. Consider you have n disks and k poles, the n discs are of diameters 1,2,3. . . . .,n and the k poles are namely A_1, A_2, A_3,...., A_k. Initially all the discs are placed on pole A1 in the increasing order of diameter when viewed from top to bottom,the disc with diameter n is placed on the base and the lower ones are placed above them. The Task is to place all the discs on pole A_k with some manipulations involving all the poles, however we do have a constraint that a disc of larger diameter cannot be placed on a disc of smaller diameter. For given n discs and k poles derive a mathematical formula in terms of n and k which gives the least number of computations required to achieve the above task, Also determine the Time Complexity of the above task. Also Consider that we have way too many discs as compared to poles.