DSA CSF211 Comprehensive examination

The respondent's email (**null**) was recorded on submission of this form.
* Required

1. Email *

____

2. Let the height of a binary tree be the number of nodes in the longest path from root to any leaf. What is the maximum number of nodes that can be there in a binary tree of height h? * 1 point

Mark only one oval.

- ◯ 2^h-1
- ◯ h^2-1
- ◯ h
- ◯ 2h
- ◯ 2^h-2

DSA CSF211 Comprehensive examination

3. Let us suppose you are given a natural number N in the form of a singly linked list with each node holding a digit. The least significant digit (LSD) is at head and MSD is at tail, and all the digits in the list are in the same order as they are in N. What would be the time complexity of the optimum algorithm that can check whether N is a multiple of 25 or not? * 1 point

Mark only one oval.

- O(1)
- O(n) where n is the number of digits in N
- O(n^2) where n is the number of digits in N
- O(n^n) where n is the number of digits in N
- If N is a power of 10, then O(1), otherwise depends on the number of digits in N

DSA CSF211 Comprehensive examination

4. Suppose that you have a singly linked list of length n, holding integer values between 1 and 500. What would be the time complexity of an optimum algorithm that can detect duplicates in this list? * 1 point

Mark only one oval.

- O(1)
- O(n)
- O(n^2)
- O(n^3)
- O(n^4)

DSA CSF211 Comprehensive examination

5. Let the running time T(n) of an algorithm be expressed using the polynomial 3n^2+5n . Which of the following statements are True. (A) T(n) is O(n^2) (B) T(n) is O(n^3) (C) T(n) is O(n) * 1 point

Mark only one oval.

- A and B
- B and C
- A alone
- B alone
- A and C

DSA CSF211 Comprehensive examination

6. What is the minimum time required to construct a max heap from an array sorted in descending order, assuming array size to be n? * 1 point

Mark only one oval.

- O(1)
- O(n log n)
- O(log n)
- O(n)
- O(n^2)

DSA CSF211 Comprehensive examination

7. Let us suppose that you are given a graph G with 2 connected components both having one cycle each. What is the minimum number of edges in G * 1 point

Mark only one oval.

- n
- n-1
- n+1
- n-2
- 2n

DSA CSF211 Comprehensive examination

8. Given a graph G with exactly one spanning tree, which of the following statement is true * 1 point

Mark only one oval.

- ◯ G has two vertices with degree 1
- ◯ Any graph with n-1 edges can be G
- ◯ Any graph without cycles can be G
- ◯ Any connected graph can be G
- ◯ G has exactly one cycle

DSA CSF211 Comprehensive examination

9. Which of the following recurrence relation captures the worst case time complexity of merge sort? * 1 point

Mark only one oval.

- ◯ T(n)= 2T(n/2)+cn (for a constant c)
- ◯ T(n)=T(n/2)+cn (for a constant c)
- ◯ T(n)=2T(n-1)+cn (for a constant c)
- ◯ T(n)=2T(n/4)+cn (for a constant c)
- ◯ T(n)=2T(n/2)+cn^2 (for a constant c)

DSA CSF211 Comprehensive examination

10. Let us suppose that you performed a deletion from an AVL tree containing n nodes. Which of the following statements are true. * 1 point

Mark only one oval.

- ◯ If the node deleted had both its children and a sibling which is a leaf, then we can do restructing in O(1) time
- ◯ Restructuring can always be done in O(1) time
- ◯ If the node deleted had a sibling which is not a leaf then we can never do restructing in O(1) time
- ◯ Restructuring always takes O(log n) time
- ◯ If the node deleted didn't have a sibling, then we don't have to restructure.

DSA CSF211 Comprehensive examination

11. Let rear and front denote the front and rear indices for an array implementation of a circular queue, where the size of the array is N, Considering the standard implementation, function size() given below has to return the total number of elements in the queue. Which of the following fits most to fill-in the the missing instruction in the function size()? * 1 point

```
int size()
{
    ????
}
```

Mark only one oval.

- ◯ return ((N - front + rear) % N)
- ◯ return ((N + front - rear) % N)
- ◯ return ((rear-front) % N)
- ◯ return ((rear-front+1)%N)
- ◯ return N

BITS PILANI, DUBAI CAMPUS

DUBAI INTERNATIONAL ACADEMIC CITY, DUBAI

SECOND SEMESTER 2020 - 2021

COURSE : Data Structures & Algorithms (CS F211)

Comprehensive Examination

Date:23/05/21 **Maximum Marks:**60 (30%)

CLOSED BOOK **DURATION:**120Minutes

1. Let height of a binary tree be defined as the *maximum* of the number of nodes in a longest path from root to any leaf. Consider a binary tree T containing n nodes having exactly one node with two children. What is the *maximum and minimum* height that T can attain, given n is odd. **[2M]**

2. Let height of a binary tree be defined as the *maximum* of the number of nodes in a longest path from root to any leaf. Suppose that you are given root pointers of two binary search trees, say $root_1$ and $root_2$. Further, it's given that all the key values are distinct in both the trees, neither they have anything in common.

   Given the largest element in the first tree is less than the second smallest in the second tree, device a procedure Merge($root_1, root_2$), that merges both the trees and returns the root of the combined tree in $\mathcal{O}(h)$ time, where $h = \max(h_1, h_2)$. Further, argue the running time of your procedure. **[6+2M]**

3. Let us suppose that you are being provided with a procedure findMaxFactor(x) that takes a positive integer x as input and returns the largest factor smaller than x. Note that if x is a prime number, the function returns 1. Design a procedure findPrimeFactorization() that represents a natural number N ($N \geq 2$) as product of prime factors, in $\mathcal{O}(n)$ time, where n is the number of prime factors of N. Further, argue the running time of your algorithm. **[3+2M]**

4. Suppose that you are given the pointer $start$ to the first node in a singly linked list containing *distinct* integers. Further, let the field that points to the next node be labeled $next$. As usual, the next field of last node is $NULL$. Unfortunately, we don't have access to the $data$ field. What we have is a function countBigs($temp$). This function takes the starting address, $temp$, of a linked list as input and returns *the number of nodes* with key value greater than that of $temp$. Design a procedure isSorted($temp$) that takes $start$ as input and returns 1 if the list is sorted in ascending order and 0 otherwise. Note that we don't have access to the data field and the only handle available is the function countBigs(). You do have access to the $next$ field to move around the list. Furthermore, argue the correctness of your procedure. **[4M]**

5. Suppose that you are given an $m \times n$ matrix $H[0 \ldots m-1][0 \ldots n-1]$ with values at each index being initialized to -1. We need to use this matrix $H[][]$ as a hash table

with *partial linear probing* to store *positive* integers. The hashing mechanism and the associated partial linear probing can be described as follows:

Given a key, The row index for an appropriate slot is $key\%m$. The column index for an appropriate slot is $key\%n$. In case, a collision happens, the insertion procedure probs over the same column downwards (wrapped around at the end) searching for a free slot. If it fails to do so, the function throws an error "Insertion Failure:key". The procedure Search(key) works accordingly. Write down the pseudo code for both insert() and search(), given the setup mentioned above. Note that you dont have to bother about Delete(key) function. **[4M]**

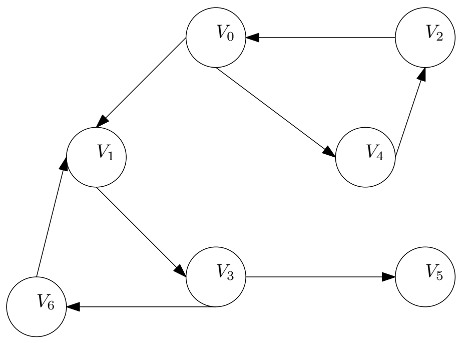
6. Draw a Standard Trie to represent the following dictionary D of strings. Count the number of nodes in the trie. Give an estimate of the space saving in data representation caused by the use of a Trie, compared to the traditional way of consuming one byte per character, per word. You may neglect pointer storage costs. $D = \{ BERLIN, COPENHAGEN, PARIS, SYDNEY, TOKYO, STOCKHOLM, BELGIUM \}$ **[4M]**

7. Demonstrate the construction of an AVL Tree by inserting the elements 64, 1, 44, 26, 13, 110, 98, 85 in the order of their occurrence. Now, from the constructed AVL Tree, show the deletion of node 26 followed by 44. **[4M]**

8. Construct an AVL tree of height 5 with as few nodes as possible. **[3M]**

9. Design a $\mathcal{O}(1)$ algorithm that determines the 3rd largest element in a max-heap, assuming n to be the number of elements in the heap. Given a max-heap and a value k, can we compute the k^{th} largest element in the max-heap in $\mathcal{O}(k)$ time? Justfy. **[4M]**

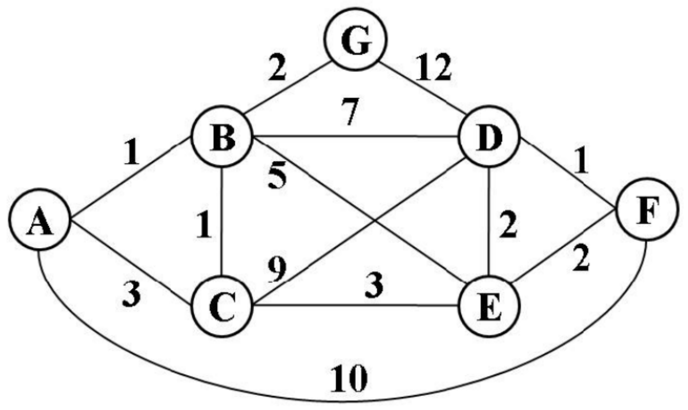
10. List the vertices in the following graph in the order they being visited assuming:



   (a) a DFS (Depth first Search) starting from V_0
   (b) a BFS (Breadth first Search) starting from V_0

[2+2M]

11. Consider the graph given below. Assuming every undirected edge (x, y) to be a pair of directed edges $x \rightarrow y$ and $y \rightarrow x$ of equal weight, determine the length of the shortest path from A to all the remaining vertices using Dijkstra's algorithm. Finally, list the vertices in the shortest path from A to F.



[3+2M]

12. Let the contents of a linear queue $Q[]$ be 5, 2, 8, 6, 7 with 7 at the rear end and 5 at front. Let Q.empty(), Q.enqueue() and Q.dequeue() be the standard empty, enqueue and dequeue functions respectively. Let Q.dequeue() returns -1 if there are no elements in the queue and otherwise it returns the element removed from the queue. Consider the following pseudocode. Assume the array Q to be sufficiently large (queue never becomes full). Describe the output of the pseudocode including all print statements.

```
Procedure fun1()
  while !Q.empty() do
        a= Q.dequeue()
        b= Q.dequeue()
        print a
        print b
        if(b==-1)
          return a
        else
          print a+b
          Q.enqueue(a+b)
EndProcedure
```

[5M]

13. Convert the following infix expressions to postfix expression, assuming standard precedence and associativity(^ is right associative). $P - Q - R\hat{}S\hat{}T * U * V$

    Further, evaluate the postfix expression obtained using a stack for the given value assignments P=20,000, Q= 20, R=10, S=2, T=2, U=1, V=2. **[4M]**

14. Consider an array $A[0..6]$= 40, 50, 70, 22, 34, 50, 60 which is to be sorted using the Quick Sort Algorithm. Assume you are provided with the following QuickSort() and PlacePivot() procedures which sorts a given array based on Quick Sort Algorithm.

```
QuickSort(A[], begin, end){
  if(begin<end)
  {
    pivot=PlacePivot(A[], begin, end);
    QuickSort(A[], begin, pivot -1);
```

```
      QuickSort(A[], pivot+1,end);
    }
}
```

```
PlacePivot(Inpt[],low,high){
        x=Inpt[low];
        i=low+1;
        j=high;
        while(j>=i)
        {
                if(Inpt[i]<=x)
                        {i++;}
                if(Inpt[j]>=x)
                        {j--;}
                if((Inpt[i]>x)&(Inpt[j]<x)&(j>i))
                 {
                        swap(Inpt[i],Inpt[j]);
                        i++;j--;
                 }
        }
  swap(Inpt[j],Inpt[low]);
}
```

Explain the working of Quick sort algorithm for the function call QuickSort(A[], 0, 6) for the above given array A[].

Suppose if the array A[] had n elements and all the n elements were same. (i.e., A[0, 1, 2] = 22, 22, 22 for n=3). What will be the time complexity of the given Quick Sort algorithm for the given QuickSort() and PlacePivot () procedures? **[4M]**

********END********

BITS PILANI, DUBAI CAMPUS

DUBAI INTERNATIONAL ACADEMIC CITY, DUBAI

SECOND SEMESTER 2020 - 2021

COURSE : Data Structures & Algorithms (CS F211)

Date:20/04/21 **Maximum Marks:**40 (20%)

Test-II(OPEN BOOK) **DURATION:**50Minutes

1. Consider the following array $A[0...7]$:

3	4	2	0	0	2	4	3

and $B[0...7]$:

13	24	52	10	20	12	04	63

(Don't have to write down anything) **[0M]**

2. Consider the following procedure given.

```
PlacePivot(A[],low,high)
{
    x=A[low];
    i=low+1;
    j=high;
    while(j>=i)
    {
        if(A[i]>=x)
        {i++;}
        if(A[j]<=x)
        {j--;}
        if((A[i]<x)&&(A[j]>x)&&(j>i))
        {
            print i
            print j
            print A[i]
            print A[j]
            t=A[i];A[i]=A[j];A[j]=t;//swap A[i] and A[j]
            i++;
            j--;
        }
    }
    t=A[j];A[j]=A[low];A[low]=t;//swap A[low] and A[j]
    print A[]//prints the contents of the whole array
}
```

Let Array $A[0...7]$ be the one in Question 1. Identify the output associated with the function call PlacePivot$(A[], 0, 7)$. Assume each print statement to print the value associated with the variable in a separate line. **[6M]**

3. Use substitution method to argue that (assuming $\log$ to the base 2 by default and $T(1) = k$):
   (a) $T(n) \in O(4^{log\ n})$ where

$$T(n) = 4T(\frac{n}{2}) + k(\text{for a positive constant } k)$$

[4M]

   (b) $T(n) \in O(n \log n)$ where

$$T(n) = T(\frac{n}{4}) + T(\frac{3n}{4}) + kn(\text{for a positive constant } k)$$

[5M]

4. Let Array $B[0...7]$ be the one in Question 1. Consider the standard Binary Search Tree (BST) holding *distinct* integers where every node x has the property that all values to the left of x are *less than* x and those to the right are *greater than* x. Design a procedure/pseudocode insertToBST$(head, new)$ where $head$ points to the first node in the BST and new points to the node to be inserted. Assume we start with an empty BST with $headPtr$ being the root pointer,

```
headPtr=NULL;//points to the root node of the BST
for(i=0;i<=7;i++)
{
    xPtr=createNode(B[i]);//creates a node with B[i] being
                            data, pointed by xPtr
    insertToBST(headPtr,xPtr);//don't write the procedures
}
```

   (a) Draw the BST after each iteration of the for loop. $i.e.$ you would be drawing 8 nonempty trees where each tree would contain one node more than the previous tree. **[7M]**
   (b) List the in-order traversal of your final tree **[3M]**

5. Consider the standard BUILD-MAX-HEAP() and MAX-HEAPIFY() procedures given below.

```
BUILD-MAX-HEAP(A)
{
    for(i =3;i>=0;i--)
    {
        MAX-HEAPIFY(A,i,7)
         print A[0...7]// prints the contents of A[0...7]
    }
}

MAX-HEAPIFY(A,i,n)
{
    left=2*i+1
    right=2*i+2
```

```
        if(left <= n and A[left]>A[i])
            largest=left
        else
            largest = i
        if(right<=n and A[right]>A[largest])
            largest=right
        if(largest!=i)
        {
            exchange(A[i],A[largest])//swaps the values
            MAX-HEAPIFY(A, largest ,n)
        }
}
```

(a) Determine the output of the function call BUILD-MAX-HEAP(A) assuming Array $A[0...7]$ to be the one in Question 1. **[6M]**

(b) Consider the MAX-HEAPIFY() procedure given above. What exactly is the action performed by the procedure (with reference to input parameters)? What are the constraints to be satisfied by the input to the procedure? **[2M]**

6. Let array $B[0...7]$ be the one in Question 1. Let $H[0...9]$ be a hash table and $h(key) = key\%10$ be the standard hash function in use. Assuming linear probing to be the collision resolution technique in use, determine the output of the following segment of code,

```
Insert(B[0]);//Don't have to write down the body of
Insert(B[1]);// these functions
Insert(B[7]);
Insert(B[6]);
Delete(B[1]);
Search(B[6]);
Insert(B[1]);
```

Draw the contents of the hash table $H[0...9]$ after each of the above (insert/ delete/ search) function calls. With reference to the above segment of code, illustrate your mechanism to handle deletion along with linear probing. Which location does Search() find the key value at? **[5+1+1M]**

********END********

BITS PILANI, DUBAI CAMPUS

DUBAI INTERNATIONAL ACADEMIC CITY, DUBAI

SECOND SEMESTER 2020 - 2021

COURSE : Data Structures & Algorithms (CS F211)

Date:11/03/21 **Maximum Marks:**40 (20%)

Test-I(CLOSED BOOK) **DURATION:**50Minutes

1. Let the contents of a queue Q[] be $25, 32, 48, 55, 77$ with 25 at the *rear* end and 77 at *front*. Let $Q.enqueue()$ and $Q.dequeue()$ be the standard enqueue and dequeue functions respectively. Consider the following pseudocode. Assuming the array Q to be sufficiently large (queue never becomes full), describe the queue (contents) by the end of the following sequence of operations.

```
$temp \leftarrow Q.dequeue() + Q.dequeue()$
$temp \leftarrow temp/11$
$Q.enqueue(temp)$
$Q.enqueue(Q.dequeue())$
```

[4M]

2. Device an $\mathcal{O}(\log n)$ algorithm/procedure which computes 5^n for a non-negative integer n. You may use the following recursive formulation of 5^n.

$$5^n = \begin{cases} 1 & \text{if } n = 0 \\ 5^{n/2} \times 5^{n/2} & \text{if } n\%2 = 0 \\ 5^{n/2} \times 5^{n/2} \times 5 & \text{if } n\%2 = 1 \end{cases}$$

[5M]

3. Determine the validity of the following statements with proper justification (assume $\log$ to the base 2).
   (a) $3n^2 + 5n$ is **not** $\mathcal{O}(n)$
   (b) $4^{\log n} \times \log n + 45n \log n$ is $\mathcal{O}(n^2 \log n)$

[2M+3M]

4. Given an array A of size n, where each entry is either 2 or 3. Assume that all those 2s in A, occur before the sequence of 3s (if any). In other words, $A[0 \ldots n-1] = \{2, 2, \ldots, 2, 3, 3, \ldots, 3\}$. Design an algorithm/pseudocode to find out the smallest index i in A such that $A[i] = 3$, consuming at most $c \log n$ premitive operations for some positive constant c. **[6M]**

5. Consider the following Algorithm *Task(A,n)*. Assume array $A[0...n-1]$ contains n *distinct* positive values. Identify the best case and worst-case performance scenarios, estimate the number of primitive operations in both cases, and hence express the corresponding running times (both best and worst) of the algorithm using big-$\mathcal{O}$ notation.

```
function TASK(A,n)
    for $j \leftarrow 1$ to n-1 do
        $k \leftarrow A[j]$
        $i \leftarrow j - 1$
        while $i \geq 0$ and $A[i] > k$ do
            print $A[i]$
            $i \leftarrow i - 1$
        end while
        $j \leftarrow j + 1$
    end for
end function
```

[6M]

6. Consider the arithmetic expression $A + B * C\hat{}(D/E)$. Convert the expression to the equivalent postfix expression. Show the step by step working of the algorithm by filling in the following table, assuming standard operator precedence relations.

Input Read	Action Performed	Stack Content
A	Operand: Print	
+	Operator: Push(+)	+
......	......	......
......	......	......

[6M]

7. Assume that you have been provided with a doubly linked list holding integer values in the range $0, 1, \ldots, 1000$ where the node structure is as follows.

```
struct node
{
    int data;
    struct node*prev;
    struct node*next;
}
```

(a) Design a function/algorithm/pseudocode *traverseList(struct node*temp)* that takes a pointer to the last node in the list as input/parameter and prints the elements in the list starting from the last node until first. Further, extend your algorithm to count the number of unique elements in the array. Your algorithm should return this count. You may use an auxiliary data structure for the same. **[3M+3M]**

(b) Express the running time of your algorithm using big-$\mathcal{O}$ notation, assuming the length of the list to be n. Justify your answer. **[2M]**

********END********

DSA Quiz-I

The respondent's email address (**rahulcsbn@gmail.com**) was recorded on submission of this form.

The intent of the procedure below is to delete the last node of the list, assuming 'first' to be a reference to the first node in the list. * 1 point

```
removeLast(Node first)
{
     Node p, q;
     p = first;
     q = p.next;
     while (q.next != null)
     {
          p = q;
          q = q.next;
     }
     p.next = null;
}
```

Which of the following describes the class of linked lists for which this method works correctly?

- All singly linked lists
- All non-empty singly linked lists
- All circular linked lists
- [✓ All singly linked lists with more than one node]

DSA Quiz-I

Assuming ‘first’ to be a reference to the head node in a list, which of the following procedures correctly inserts a value x at the front of the list and returns a reference to the head of the updated list? * 1 point

```
I.          insertFront(Node first, int x)
            {
                 first = new Node();
                 first.datum = x;
                 first.next = first;
                 return first;
            }


II.         insertFront(Node first, int x)
            {
                 Node n = new Node();
                 n.datum = x;
                 n.next = first;
                 return n;
            }

III.        insertFront(Node first, int x)
            {
                 Node n = new Node();
                 first = n;
                 n.datum = x;
            `    n.next = first;
                 return first;
            }
```

[✓ II Only]

I only

III only

II and III only

DSA Quiz-I

Assuming N to be the length of the input list, the extra space consumed by the given standard merge sort procedure is proportional to: * 1 point

- [✓ N]
- log N
- N log N
- a constant

DSA Quiz-I

Consider the procedure Search(A,low,high,x) you are being provided with. What would be an appropriate upper bound on the running time of this procedure assuming the array A to be of size n ? * 1 point

- [✓ O(N)]
- O(log N)
- O(N log N)
- O(1)

Consider the procedure Search(A,low,high,x) you are being provided with. Let array A[] be of size n and be sorted in ascending order. Let x be present in A and be the smallest in the list. What would be an appropriate upper bound on the running time under the given scenario? * 1 point

- O(N)
- O(1)
- [✓ O(log N)]
- O(N log N)

DSA Quiz-I

Consider the standard merge procedure you are being provided with. Recall that the procedure is all about a sequence of pairwise comparisons between the elements in the array A[]. Let there be n distinct values in the input array and let x be the smallest element in the list. What would be an appropriate upper bound on the number of pairwise comparisons that x would participate in? * 1 point

- O(N)
- [✓ O(log N)]
- O(N log N)
- O(1)

Let there be n distinct values in the input array and let x be an arbitrary element in the list. What would be an appropriate upper bound on the number of pairwise comparisons that x would participate in? * 1 point

- O(log N)
- [✓ O(N)]
- O(N log N)
- O(1)

DSA Quiz-I

Suppose that we are implementing a queue having n values using a circular doubly linked list. Assuming that we keep track of front pointer but not rear, can we perform both enqueue and dequeue operations in O(1) time? * 1 point

- We can perform enqueue operation in O(1) time, and dequeue in O(n time)
- [✓ Yes, we can perform both enqueue and dequeue operations in O(1) time]
- Neither enqueue nor dequeue operation can be performed in O(1) time
- We can perform dequeue operation in O(1) time, and enqueue in O(n time)

DSA Quiz-I

Consider the standard merge procedure you are being provided with. Recall that the procedure is all about a sequence of pairwise comparisons between the elements in the array A[]. What will be the maximum number of pairwise comparisons needed to merge the sublists A[p...q] and A[q+1...r]? * 1 point

- r-p+1
- max(q-p+1,r-q)
- [✓ r-p]
- min(q-p+1,r-q)

What will be the minimum number of pairwise comparisons (between the elements in the array A[]) needed to merge the sublists A[p...q] and A[q+1...r]? * 1 point

- [✓ min(q-p+1,r-q)]
- r-p
- min(q-p,r-q)
- max(q-p+1,r-q)