BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI, DUBAI CAMPUS
DUBAI INTERNATIONAL ACADEMIC CITY, DUBAI
SECOND SEMESTER 2019 – 2020
DEPARTMENT OF COMPUTER SCIENCE

Questions: 15
Pages: 5

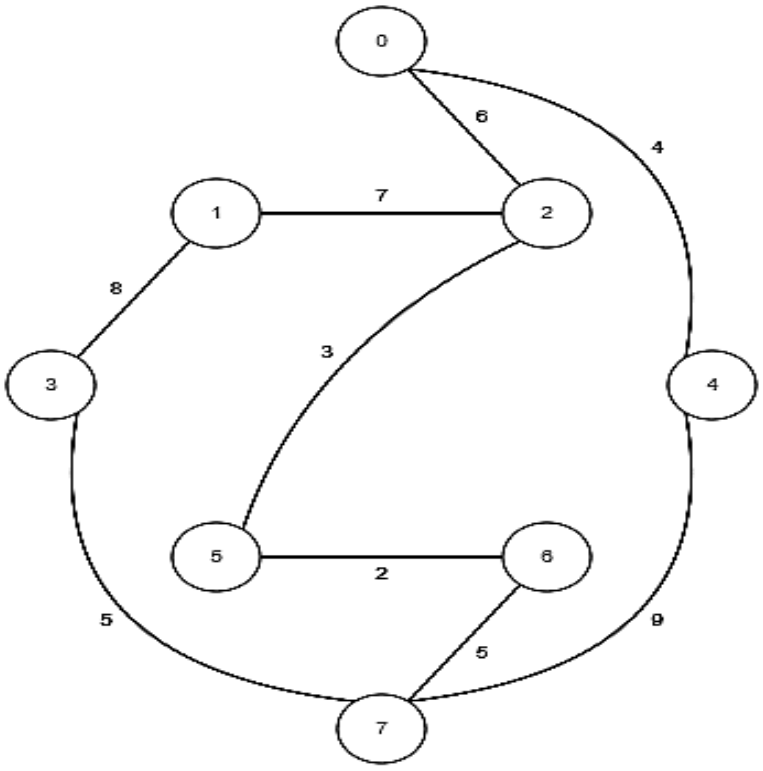
COURSE : Data Structures and Algorithms (CS F211)
COMPONENT : Compre (Closed Book) **WEIGHTAGE :** 35% (70 Marks)
DATE/SESSION : 09-Jun-2020, AN **DURATION :** 2 hours 50 mins

Instructions:
Please write down any assumptions that you make.
← represents the assignment operator
Show all the steps to score full marks

Q1 Apply bucket sort to the numerical values listed below. Assume that the range 1.0 - 2.0 is divided into 10 buckets, each of size 0.1. Explain your work step by step.

1.93, 1.16, 1.04, 1.72, 1.96, 1.44, 1.13, 1.70, 1.20, 1.53 **[3M]**

Q2 Apply the Dijkstra's shortest path algorithm to the following graph with 0 as the initial vertex. Compute the cost of the shortest paths from vertex 0 to all vertices. Compute the shortest path from vertex 0 to all vertices. Show all the steps.

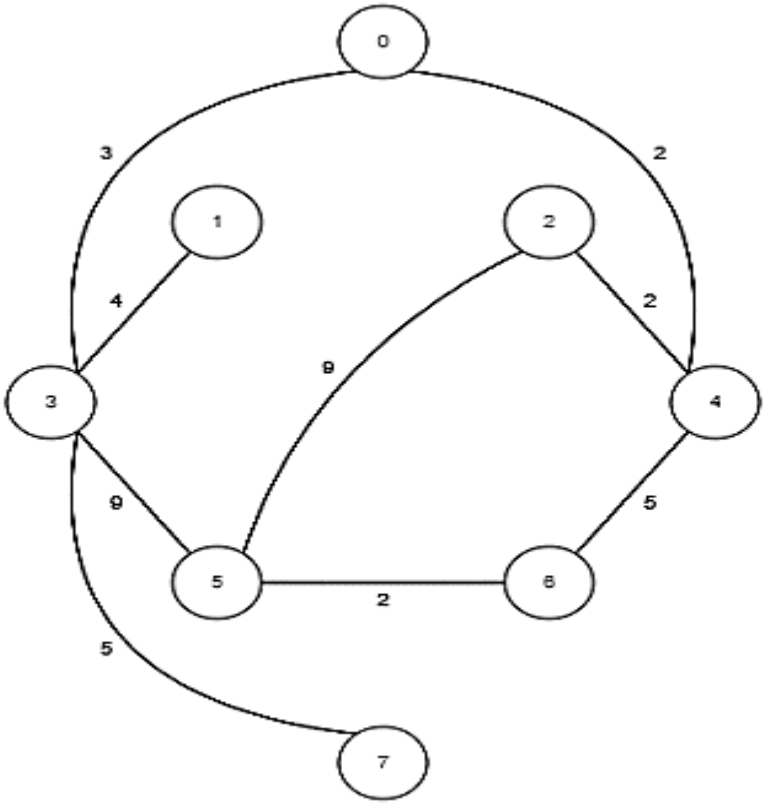


[6M]

Q3 Apply Huffman's algorithm to the following text to find the Huffman encoding of it's symbols:
this is a very easy test

Draw the Huffman Tree for the above symbols. Show all the steps. **[5M]**

Q4 Apply the Prim's algorithm to compute the minimum spanning tree of the following graph with 0 as the initial vertex. Show all the steps.



[6M]

Q5 Consider an array of n integers ranging from 0, 1, ..., n^6-1. What is the time complexity of Radix Sort when using base-5 representation? Justify your answer. **[3M]**

Q6 Design a procedure `PlacePivot(L, temp1, temp2)` that takes a doubly linked list and two end pointers as input and places the pivot element in appropriate place (as in quick sort) in `O(n)` time (`n` is the number of nodes in the list). Can we extend the above procedure to have a variant of quick sort algorithm that works on doubly linked lists in `O(n logn)` time? Justify your answer. **[8M]**

Q7 Suppose that you are given a pointer *temp* to an arbitrary intermediate node in a Binary search tree. Design an *O(h)* algorithm *Next(temp)* that would output the in-order predecessor of *temp*, given *temp* has a left subtree (Note that the inorder traversal algorithm doesn't work). Further, analyze and verify the running time. **[5M]**

Q8 Consider the following binary tree traversal sequences provided with. Design and reconstruct the binary tree corresponding to the following pair of sequences (each alphabet symbol represents a distinct node in the tree).

In-order: G A C D B E U K F
Post-order: G A D C U K F E B **[5M]**

Q9 Given two sorted arrays A[ ] and B[ ] of length *m* and *n* each, write an *O(m+n)* procedure that would merge these two arrays to one sorted array C[ ]. **[5M]**

Q10 Convert the following infix expression to postfix expression using a stack. Show all the steps.

1 + 2 * 3 – 4 * 9 – 7 **[4M]**

Q11 Write a procedure *delete_all(Q, temp, x)* that takes a circular queue *Q* containing *n* elements implemented as a circular doubly linked list, a pointer to an arbitrary node in the queue (*temp*) and a value *x*. The procedure deletes all the nodes in the queue if the value stored in a node is equal to *x*. Note that your algorithm should be effective and you are instructed to mention all assumptions precisely. What is the time complexity of your algorithm? Justify your answer. **[4M]**

Q12 (a) Assume that we have n non decreasing input values. These values are used to construct a binary search tree and a max-heap by inserting the values one at a time into an initially empty binary search tree and a max-heap. Analyze and express the height of the binary search tree and the max-heap using Big-Oh notation. Explain your answer with an example containing at least 6 elements.

(b) What is the worst case and best case running time to find the minimum element in a binary search tree and a min-heap constructed using the same n elements? Explain your answer with an example containing at least 6 elements. **[5M]**

Q13 Insert the following elements into an initially empty hash table using the hash function $h(k) = k \bmod 11$ and double hashing collision resolution strategy. The secondary hash function is $d(k) = (1 + \text{floor}(k/4)) \bmod 11$. Explain and show your work step by step.

Elements: 34,16,45, 53, 6, 29, 37, 78, 1 **[4M]**

Q14 Analyze the running time of the following procedure and express it using Big-Oh notation with proper justification.

```
Algorithm fun1(n)
j ← 0
i ← 0
k ← 0
l ← 0
sum ← 0
while j < n do
    while i < 10 do
        while k < n do
            while l < 100 do
                sum ← sum + 1
                l ← l + 1
            k ← k + 1
        i ← i + 1
    j ← j + 1
```

[2M]

Q15 Illustrate the working of heap sort to sort the following elements of an array A in ascending order. A[] = {40, 90, 20, 80, 100, 12}. Use bottom up heap construction method to build the heap. **[5M]**

********** ALL THE BEST *********

[50 Mins]DSA(CS F211) T2

1. **[1 Mark]** Consider a binary search tree containing *n* nodes where *n* being an odd number with all key values being distinct. Let the root element be the middle one in the sorted sequence of the given set of key values. What could be the maximum height of the tree? *(height of a single node tree is 1)*

- (n/2)+1
- n-1
- log n

2. **[2 Marks]** After a right rotation in a binary search tree

- The in-order traversal always generates the same sequence of nodes as before rotation.
- The post-order traversal may generate the same sequence of nodes as before rotation.
- The height of the tree always increases by 1.

3. **[2 Marks]** Consider the array *A[1...n] = {120, 30, 100, 20, 130, 50, 60, 45}* with *n = 8,* that would get sorted in ascending order using insertion sort procedure given below.

```
01: Insertion-Sort(A[], n)
02: for j <- 2 to n
03:     key <- A[j]
04:     i<- j - 1
05:     while i> 0 and A[i] > key do
06:          A[i+1] <- A[i]
07:          i<- i – 1
08:     A[i+1] <- key
```

The number of *for loop* iterations over which, the body of the *while loop* will be skipped as a whole, is:

- 1
- 3
- 4
- 2

4. **[2 Marks]** Consider an array *A[]* to be sorted in ascending order using quick sort and insertion sort. Let the elements of the array be distinct and be in descending *order* to begin with. The time complexity of insertion sort and quick sort to sort the elements of the array A[] are:

Note: Choose all correct options

- Insertion sort takes O(n^2) time
- Quick sort takes O(n^2) time
- Insertion sort takes O(n) time
- Quick sort takes O(n) time
- Insertion sort takes O(n log n) time
- Quick sort takes O(n log n) time

5. **[3 Marks]** Let the height of a binary search tree *T* having distinct key values, be *h*. Let x be the root of *T* and *y* be the left child of *x*. Let all those values between *x* and *y* be in a subtree rooted at *z*.

Let theheight of the tree be the same even after a right rotation of *T* with respect to *x*. What would be the height of the subtree rooted at *z*?

- h-2
- at most h-2
- one less than the height of the subtree rooted at y.

6. **[4 Marks]** Let the address of the root node of a non-empty binary search tree of height *h* be stored in a variable *head* (*head* is a pointer to *root*). Identify the action performed by the following segment of code:

```
if(head->left==NULL)
{
     head=head->right;
}
else
{
     t=head->right;
     temp=head->left;
     while(temp->right!=NULL)
     {
        temp=temp->right;
     }
     temp->right=t;
}
```

- Deletes the root node of the binary tree
- The height of the tree may become 2h-2 after the execution of the above segment of code
- Performs right rotation with respect to the root node
- The height of the tree may even reduce by 1 after the execution of the above segment of code

7. **[4 Marks]** Which of the following statement(s) regarding merge sort are true?

- Merge sort is based on divide and conquer algorithm design paradigm
- For the merge sort implementation (discussed in class) O(n^2) extra space is sufficient where n is the number of inputs
- The merge procedure of the merge sort algorithm takes O(n) time where n is the number of inputs
- For the merge sort implementation (discussed in class) O(n) extra space is sufficient where n is the number of inputs
- The best case running time of merge sort is O (log n) where n is the number of inputs
- Merge sort algorithm consumes O (log n) stack space where n is the number of inputs
- The merge procedure of the merge sort algorithm takes O (log n) time where n is the number of inputs

8. **[8 Marks]** Consider the array *A[1...n] = {120, 30, 100, 20, 130, 50, 60, 45}* with *n = 8* to be sorted in ascending order using recursive insertion sort procedure given below.

```
Algorithm InsertionSortRec(A[], n)
if n = 1 then
   return
InsertionSortRec(A[],n-1)
key <- A[n]
i<- n - 1
while i> 0 and A[i] > key do
     A[i+1] <- A[i]
i<- i – 1
A[i+1] <- key
```

The contents of the array after the completion of *7th*, *5th*, *3rd*, and initial invocations to InsertionSortRec() are listed below in 4 different options. Select all the 4 correct options.

Note: The initial invocation is InsertionSortRec(A[], 8), the second invocation is InsertionSortRec(A[], 7) and so on.

- 30, 120, 100, 20, 130, 50, 60, 45
- 20, 30, 100, 120, 130, 50, 60, 45
- 20, 30, 50, 100, 120, 130, 60, 45
- 20, 30, 45, 50, 60, 100, 120, 130
- 20, 30, 120, 100, 130, 50, 60, 45
- 30, 20, 100, 120, 130, 50, 60, 45
- 20, 30, 45, 100, 120, 130, 50, 60
- 20, 30, 45, 50, 100, 120, 130, 60

10. **[8 Marks]** Identify the action performed by the following segment of code on a binary search tree, when *head* contains the address of the root node, having both its children being nonempty.

```
l=head->left;
r=head->right;
head->left=l->right;
l->right=head;
head->right=r->left;
r->left=head;
l->right=r;
head=l;
```

- Performs left rotation at root node
- Performs right rotation at root node

- Right rotation at the node at root position followed by left rotation at the same node
- Performs right rotation at root position followed by a left rotation at root.
- The node that was being the root of the tree would be at a depth 1 after the execution of the above segment of code
- The node that was being the root of the tree would be at a depth 2 after the execution of the above segment of code
- Given the address of a node, rotation at one of its children cannot be performed in O(1) time.
- The segment of code performs a couple of rotations so that the tree remains the same as in the beginning.

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI, DUBAI CAMPUS
DUBAI INTERNATIONAL ACADEMIC CITY, DUBAI
SECOND SEMESTER 2019 – 2020
DEPARTMENT OF COMPUTER SCIENCE

Questions: 7
Pages: 2

COURSE : Data Structures and Algorithms (CS F211)
COMPONENT : Test 1 (Closed Book) **WEIGHTAGE :** 20% (40 Marks)
DATE/DAY : 27-Feb-2020, Th2 **DURATION :** 50 Minutes

Instructions:
Please write down any assumptions that you make.
← represents the assignment operator

Q1 Let the contents of a stack `S[]` be 25,32,48,55,77 with 25 at the bottom and 77 at top. Consider the following pseudo code.

```
temp ← S.pop()+S.pop()
temp ← temp/11
push(temp)
t1 ← S.pop()
t2 ← S.pop()
```

What would be the values of `t1` and `t2` after the execution of the pseudocode? **[5M]**

Q2 What is the advantage of using circular queue over linear queue, assuming an array implementation? Is there any advantage, if we have a linked list based implementation instead?

[3M]

Q3 Assume that we are implementing a circular queue using a doubly linked list. Do we have to keep track of `FRONT` and `REAR` pointers? Can we manage with one instead of both? Justify your answer.

[5M]

Q4(a) Write a pseudocode for `push()` and `pop()` operations on a Stack assuming singly linked list implementation. Note: You are instructed to mention all assumptions precisely.

[4M]

Q4(b) Analyze your pseudocode written for Q4(a) to determine the time complexity (in terms of Big-oh notation) of the `push()` and `pop()` operations. Justify your answer.

[4M]

Q5 Identify whether the following statements are TRUE/FALSE. Justify your answer.

(a) $T(n) = 2^n + n^2$ is $O(n^2)$

(b) $T(n) = \sqrt{n}$ is $O(2^{\log_2 n})$ **[6M]**

Q6 Write the pseudocode for binary search. Analyze the running time of binary search and express the running time in Big-oh notation.

[7M]

Q7 Analyze the running time of the following pseudocodes and represent it using Big-oh notation. Justify your answer.

(a)

```
sum ← 0
i ← 0
while i < n do
      j ← 0
      while j < n * n do
            sum ← sum + 1
            j ← j + 1          //increment loop variable j
      i ← i + 1                //increment loop variable i
```

(b)

```
for i←1 to n do
    if i < n/2 then
       i ← i*(n/2)
    else
       i ← 2*i
    i← i + 1                   //increment loop variable i
```

(c)

```
f ← 1
for i ← 1 to n do
     f ← f * i
     i← i + 1                 //increment loop variable i
```

[6M]

********** ALL THE BEST *********

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI, DUBAI CAMPUS
DUBAI INTERNATIONAL ACADEMIC CITY, DUBAI
SECOND SEMESTER 2019 – 2020
DEPARTMENT OF COMPUTER SCIENCE

Questions:10
Pages: 3

COURSE : Data Structures and Algorithms (CS F211)
COMPONENT : Quiz 1 (Online OpenBook) **WEIGHTAGE :** 10% (20 Marks)
DATE/DAY : 25-Mar-2020, W1 **DURATION :** 20 Minutes

Q1 Which of the following statements are true?

a) Open addressing does not use extra space outside the hash table (Answer)
b) Separate chaining does not use extra space outside the hash table
c) Separate chaining may result in a situation where we may not find space to insert an item in the hash table
d) Open addressing may result in a situation where we may not find space to insert an item in the hash table **[2M]**

Q2 Given input keys {4371, 1323, 6173, 4199, 4344, 9679, 1989, 4216} and hash function $h(x)=x \ mod$ 17. Find the index into which open addressing with quadratic probing will place the key 4216?

a) 0
b) 1
c) 2
d) 3
e) 4
f) None of the above **[2M]**

Q3 Suppose the following shows the contents of an 11 element hash table A with hash function $h(k)=k \ mod \ 11$, and linear probing.
A[0] =
A[1] = 12
A[2] = 13
A[3] = 36
A[4] = 45
A[5] = 26
A[6] = 6
A[7] = 17
A[8] =
A[9] =
A[10] = 10

Which one of the following choices gives a possible order in which the key values could have been inserted in the table assuming the table was initially empty?

a) 36, 10, 6, 13, 12, 45, 26, 17
b) 10, 6, 12, 13, 45, 36, 26, 17
c) 12, 13, 36, 45, 10, 17, 26, 6 **[2M]**
d) 6, 17, 10, 12, 13, 45, 36, 26
e) None of the above

Q4 Given the running time of a recursive algorithm.
$T(n) = 2T(n-1)$ if $n>1$, $T(1) = 1$
An asymptotic upper bound that depicts the running time of the algorithm is

a) $T(n) = O(2^n)$
b) $T(n) = O(2n)$
c) $T(n) = O(n)$
d) $T(n) = O(2^{(n-1)})$
e) None of the above

[2M]

Q5 Consider the following array to be sorted in ascending order using insertion sort. (Assume we use the insertion sort that inserts the smallest element in the first slot)
A[] = [45, 23, 87, 62, 90, 19, 10, 15, 39, 100]
The contents of the array after the 5th iteration of the insertion sort is

a) 19, 23, 45, 62, 87, 90, 10, 15, 39, 100
b) 23, 45, 62, 87, 90, 19, 10, 15, 39, 100
c) 23, 45, 87, 62, 19, 90, 15, 10, 39, 100
d) 10, 15, 19, 23, 39, 45, 62, 87, 90, 100
e) 10, 15, 19, 23, 39, 45, 62, 90, 87, 100 **[2M]**

Q6 Suppose that we have two sets of numbers A[1...n] (n numbers) and B[1...m] (m numbers) both being sorted in ascending order and all m+n elements being distinct. Let B[m-1] be greater than A[n]. Which is the second largest element in the given list of m+n numbers?

a) A[n]
b) B[m-1]
c) Can't say. **[2M]**

Q7 Suppose that we have two sets of numbers A[1...n] (n numbers) and B[1...m] (m numbers) both being sorted in ascending order and all m+n elements being distinct. Let B[m-1] be greater than A[n]. What is the minimum number of comparisons required to determine the third largest element in the given list of m+n numbers?

a) 1
b) 0
c) 2 **[2M]**
d) Can't say

Q8 Consider the given array of numbers A [1...6] = (25, 45, 36, 08, 07, 30). Let the following PlacePivot(A,1,6) procedure (from slides) be invoked on this given list of numbers. What will be the content of the array after the execution of the procedure?

```
PlacePivot(Inpt[],low,high)
{
  x=Input[low];
  i=low+1;
  j=high;
  while(j>=i)
  {
```

```
        if(Input[i]<=x)
       {
         i++;
        }
       if(Input[j]>=x)
       {
         j--;
       }
       if((Input[i]>x)&&(Input[j]<x)&&(j>i))
       {
          swap(Input[i],Input[j]);
          i++;
          j--;
        }
   }//END while(j>=i)
  swap(Input[j],Input[low]);
}
```

a) A[1...6] =(07, 08, 25, 30, 36, 45)
b) A[1...6] =(08, 07, 25, 45, 36, 30)
c) A[1...6] =(08, 07, 25, 36, 30, 45)
d) A[1...6] =(08, 07, 25, 36, 45, 30)

[2M]

Q9 Suppose that you are given a list of distinct numbers A[1...n] and it is being given that both the lists A[1...n/2] and A[n/2+1...n] are sorted (independently) in ascending order.
What is the minimum time required to compute the number of elements greater than (or less than) A[1] in the given list of n numbers?

a) O(n) time
b) O(1) time
c) O(n log n) time
d) O(log n) time using binary search **[2M]**

Q10 Suppose that you are given a list A[1...n] of integers where A[i]<=i for every 1<=i<=n. Can we have an O(1) algorithm to sort this given list of numbers?

a) Yes
b) We cannot have an O(1) algorithm
c) Yes, if all integers are positive
d) Yes, if all integers are positive and distinct **[2M]**

********** ALL THE BEST *********