End Semester Exam

DSA (Winter 2021)

MM: 100 Marks Time: 90 Mins

Instructions:

1. All questions are mandatory.
2. The paper has been divided into two sections:
    - Section A: Objective Questions [30 Marks]
    - Section B: Subjective Questions [70 Marks]
3. There is no negative marking.
4. Any cases of plagiarism will be strictly dealt with.

Section A [30 Marks]

Objective Questions

Instructions:

- Each question may or may not have **multiple correct answers**.
- Each question carries 2 marks.
- Full credits are awarded for marking all the correct options, zero otherwise.

---

1. Which of the following comparison-based sorting algorithms is/are the best in terms of efficiency? Consider only the worst-case time complexity analysis is used for measuring efficiency.
    a. Heap Sort
    b. Bubble Sort
    c. Merge Sort
    d. Quick Sort

    Ans: a, c

2. You are working with an ed-tech company. You have been given the contact details (raw data) of 10 million users (approx). You have to store this data efficiently, such that the PR team can use this data to reach out to potential customers. The PR team often searches people by name and tries to find their contact details. How would you store this information, such that the PR team can do its work efficiently?
    a. Store the raw data. Even the best sorting algorithm will take too much time with such huge data. A cache-based system could help in reducing the access time for frequently accessed contacts. (Caching: Storing top 100 most frequently accessed contacts separately for faster retrieval of those contacts)
    b. Spend some time to sort the data. It would take some time to sort, but searching the data becomes efficient.
    c. Arrange the data in a binary tree structure. It would ensure related data are connected to each other, making search faster.
    d. Arranging the data is of no significant help. Even binary search will take a lot of time to process a query with such huge data. Thus, divide the data into multiple buckets and then sort and store. Whenever you get a query, decide which bucket this data belongs to, and perform a search.

    Ans: b

3. Pick the correct inequalities. Assume n is sufficiently large.
    a. $O(\log n) < O(n) < O(n \log n) < O(n^2) < O(2^n) < O(n!)$
    b. $O(10^3) < O(2^{\log n}) < O(n^2) < O(2^n)$
    c. $O(1) < O(n^2 \log n) < O(\log(n^n)) < O(n^2) < O(n!)$
    d. $O(\log n) < O(n \log(n^n)) < O(n \log n) < O(n^2)$

    Ans: a, b

4. Which of the following statements is/are true?
    a. Queues implemented using an array are usually a little bit faster than a linked list. However, the size is fixed in the case of an array whereas the size of a linked list can be easily modified.
    b. Linked lists use static memory.
    c. Appending a node in a linked list can be done in a constant time.
    d. Removing a node from a linked list can be done much faster than removing an element from an array.

    Ans: a, c, d

5. How many iterations would it take to search 5 in the array {1, 3, 5, 7, 11, 17, 20} using binary search?
    a. 3
    b. 4
    c. 5
    d. 6

    Ans: a

6. The following method should multiply the two positive integers (using the fact that multiplication is repeated addition) -- so multiply(3, 5) should become 15 (3+3+3+3+3). But the blocks have been mixed up and include **one extra block** that is not needed in a correct solution. Which of them is/are the correct ordering(s) for the block?

    A.
    ```
        } //end else
    } //end method
    ```
    B.
    ```
    return multiply (a, b-1) + a;
    ```
    C.
    ```
    public static int multiply (int a, int b) {
    ```
    D.
    ```
    return 1;
    ```
    E.
    ```
    } //end if
    ```

F. else {
G. if (b==0) {
H. return 0;

a. C G H F E B A
b. C G D F E B A
c. C G H E F B A
d. C G D E F B A

Ans. c

7. You have implemented a stack using two queues. The stack should support push operation in O(1) time. Which of the following statements are true to carry out the following operations? **[2 Marks]**

```
push(5);
push(6);
pop();
push(3);
push(4);
pop();
```

a. The number of enqueue operations is 7.
b. The number of enqueue operations is 5.
c. The number of dequeue operations is 7.
d. The number of dequeue operations is 5.

Ans. a d

8. The preorder traversal sequence of a binary search tree is 30, 20, 10, 15, 25, 23, 39, 35, 42. Which one of the following is the postorder traversal sequence of the same tree?
   a. 15,10,23,25,20,35,42,39,30
   b. 10,20,15,23,25,35,42,39,30
   c. 15,10,25,23,20,42,35,39,30
   d. 15,20,10,23,25,42,35,39,30

Ans. a

9. The height of a BST is given as h. Consider the height of the tree as the no. of edges in the longest path from root to the leaf. The maximum no. of nodes possible in the tree is

   a) 2^{h-1} -1
   b) 2^{h+1} -1
   c) 2^{h} +1
   d) 2^{h-1} +1

Ans. b

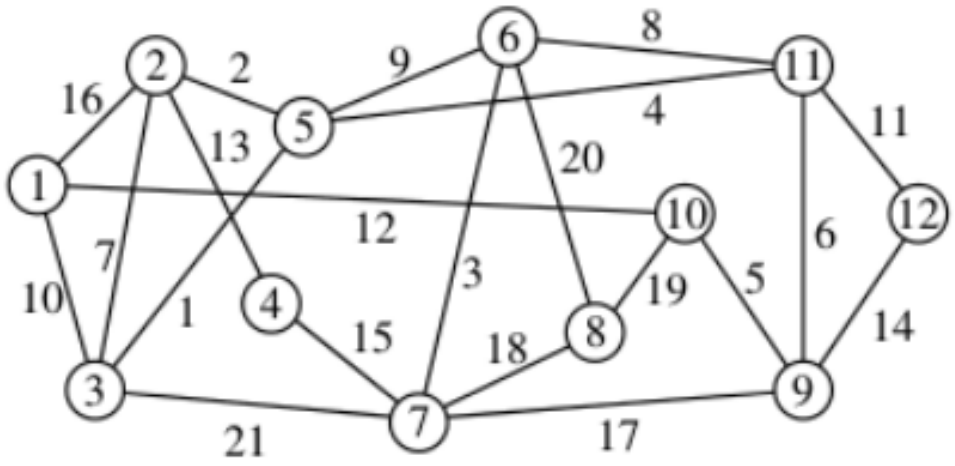
10. Which of the following statement(s) is/are true in regard to the below-given code?

```
public class LinkedList {
    public void function(Node ptr) {
        ptr.val = ptr.next.val;
        ptr.next = ptr.next.next;
    }
}
```

a. The value of the next node is copied to ptr.
b. ptr.val is deleted.
c. ptr can be the tail node.
d. The value of ptr is copied to the next node.

Ans. a, b

Consider the graph given below for answering questions (11-12):



11. If we start with node 10 as the starting node and use Prim's algorithm to construct the minimum spanning tree. The total weight of the Minimum spanning tree thus formed will be:

a) 82
b) 90
c) 81
d) 78

Ans: c) 81

12. If we start with node 10 as the starting node and use Prim's algorithm to construct the minimum spanning tree, give the order in which nodes will be added to the resulting MST:

a) 10, 9, 11, 5, 6, 3, 2, 8, 7, 1, 4, 12
b) 10, 9, 11, 5, 3, 2, 6, 7, 1, 12, 4, 8
c) 10, 1, 3, 5, 2, 11, 9, 12, 4, 8, 6, 7
d) 10, 1, 3, 5, 2, 9, 12, 11, 4, 8, 6, 7

Ans: b) 10, 9, 11, 5, 3, 2, 6, 7, 1, 12, 4, 8.

13. How to resolve the problem of collisions while implementing hashmap?

a) Deterministic hash function
b) Uniformly random hash function
c) Increase the size of array
d) Chaining

Ans: d

14. Consider the following two statements:
P: Two BSTs (could be unbalanced), containing n elements in each, could be merged into a single balanced BST in O(n) time.
Q: BFS takes O(V + E) time irrespective of whether the graph is presented with an adjacency list or with an adjacency matrix.
Which of the following is correct?

a. Both P and Q are true.
b. P is true, but Q is false.
c. P is false, but Q is true
d. Both P and Q are false.

Ans: b. P is true, but Q is false.
Explanation: Q is false because BFS will take O(V^2) time using an adjacency matrix.

15. In an unweighted graph where the distance between any two vertices is at most T, which of the following statement(s) is/are true:

a) Any BFS tree has depth at most T
b) A DFS tree might have depth > T
c) Both DFS and BFS trees will always have depth <= T
d) A BFS tree can have depth > T
e) None of the above

Ans: a, b
Explanation: Since all vertices are connected by a path with at most T edges, and since BFS always finds the path with the fewest edges, the BFS tree will have depth at most T. A DFS tree may have depth up to V-1 (for example, in a complete graph).

Section B [70 Marks]

Subjective Questions

Instructions:

1. All questions are compulsory.
2. Avoid writing pseudo-code, unless specified otherwise.
3. Avoid language specific details while writing pseudocode.
4. Whenever writing an algorithm, give a step wise explanation. It should be short and simple to understand.
5. Avoid implementation level details while writing an algorithm.

---

1. After an unsuccessful tech career, Harsh has decided to work as a pizza delivery boy at Pizza Hat. As a part of his daily routine, he has to deliver N orders to different locations (junctions) in the city. There are V different junctions. These V junctions are connected by E straight bidirectional roads. Each junction is connected to at most 4 different roads. Harsh starts from Pizza Hat, and waits for an order. As soon as the order is prepared, he travels to the delivery point on his brand new bicycle. After this, he returns to Pizza Hat through the same route. In this way, his day ends when he delivers N such orders. He gets really tired in this process, and thus wants to minimize the overall distance travelled by him. As he sucks at data structures and algorithms, he is asking for your help.

   [10 Marks]

   a. What data structure would you recommend using here? What algorithm could be used to compute distance travelled by Harsh each day?

      [2 Marks]

      Data structure: Graph
      Algorithm: Dijikstra's Algorithm

   b. Write the pseudo code for finding the minimum distance travelled by him on a given day. **[5 Marks]**

```
function Dijkstra(Graph, source):
        for each vertex v in Graph:
                dist[v] := infinity
        dist[source] := 0
        Q := the set of all nodes in Graph
        while Q is not empty:          // main while loop
                u := node in Q with smallest dist[ ]
                remove u from Q
                for each neighbor v of u:
                        alt := dist[u] + dist_between(u, v)
                        if alt < dist[v]     // Relax (u,v)
```

```
                                        dist[v] := alt
        return dist[ ]

function distanceTravelledByHarsh(Graph, source, orderLocations[]):
        distance[] = Dijkstra(Graph, source)
        ans = 0
        for location in orderLocations:
                ans += distance[location]
        return 2 * ans;
```

Note for TA:
3 marks for Dijkstra
2 marks for distanceTravelledByHarsh

c. Analyze the run time of the given algorithm. **[3 Marks]**

Here, Q can be implemented as a priority queue (min heap). This way all operations, extractMin(), updatePriority() and insert() can be done in log n time in the priority queue.

Thus, the runtime of the Dijkstra algorithm here is (E + V) log V.

Here, E = O(V). So, students may write only in terms of V, that is also correct. O(V log V)

(Refer to this article if you wish to know how all these operations are done in log n time)

After this, we know calculating distance travelled by Harsh can be done in O(N) time, as the shortest distance between source and destination can be done using a distance array.

Total runtime: (E + V) log V + N

Note for TA:
2 marks for Dijkstra runtime. If another approach using an array is written, with $O(V^2)$ runtime, deduct 1 mark.
1 mark for distanceTravelledByHarsh and final runtime

6 marks (for a + b + c), if the BFS approach is used. The graph is weighted, thus BFS doesn't work.
0, if the student has made MST for the corresponding graph. (Reference)

2. Given two polynomial, say P and Q of the general form:

$a_0x^n + a_1x^{n-1} + \ldots + a_{n-1}x + a_n$

Do note that you can skip terms having zero as their coefficients for better efficiency. Your task is to compute:

- the addition of these two polynomials
- the multiplication of these two polynomials

[12 Marks]

Answer the following questions**:**

a. What data structure will you use to represent the polynomials efficiently in the memory and why? **[2 Marks]**

- Linked List [1 Mark]
- Reason: efficient use of memory [1 Mark]

b. Assuming that you'll get these polynomials stored as function parameters in the best way possible, explain how you would compute the addition of the given two polynomials. **[5 Marks]**

- Initialize a new zero polynomial, say result. [1 Mark]
- Check if the head of P has a higher degree or the head of Q has a higher degree.
  - Add the higher degree term to the result, and adjust the head of P or Q to the next pointer. [1.5 Mark]
  - In case both of them have equal degree, add the coefficients and append to the result. [1.5 Mark]
- Do this recursively, until the heads of P and Q both become null. [1 Mark]

c. Similarly, explain how you would compute the multiplication of the given two polynomials. **[5 Marks]**

- Initialize a new zero polynomial, say result. [1 Mark]
- Iterate over the terms of polynomial P. For each term say t1 in polynomial P, do the following:
  - Iterate over the terms of the polynomial Q. For each term, say t2 in the polynomial Q, compute the product of t1 and t2 and add to the result. [2 Marks]
- Merge all terms with the same power. [1 Mark]
- Arrange the list in descending order of their power of the variable.

[1 Mark]

Notes for TAs:

- The merging and sorting of terms can also be done at the time of adding the terms to the result.
- If a student has given array based implementation, give him/her 50% credits for each of the components. The algorithm might become simpler in that case, but can be very inefficient in large degree polynomials.
- b and c can have alternate implementations. Please check the correctness of code and Mark accordingly.

3. Harsh is afraid of commitment; hence he sucks at the use of arrays. He cannot determine the size of the array he needs. This situation has gotten harsh on him as now he has given up on the use of arrays. However, Shashank will be visiting him to play cards. The game they play is peculiar. Shashank will give Harsh cards one by one, and he has to put them in the correct place. But to Harsh's dismay, he does not know how to sort the cards, given that he cannot use arrays. Assume there are no face cards. **[12 Marks]**

a. What data structure should Harsh use? Explain your choice for the data structure. **[2 Marks]**
b. Which sorting algorithm should he use: Bubble sort, Insertion sort, or Selection sort? **[1 Mark]**
c. Explain the algorithm in which you use the chosen data structure. **[3 Marks]**
d. What is the time complexity of your algorithm? **[2 Mark]**
e. Write the pseudo-code. **[4 Marks]**

Ans.

a. **Linked list [1 Mark]**
   Since Harsh cannot determine the size of the array, we use a linked list as **we do not need to specify its size beforehand**.**[1 Mark]**
b. **Insertion Sort [1 Mark]**
c. (i) We will carry out **insertion sort on the linked list. [1 Mark]**
   (ii) Initially, we start with an empty list. When we get the first card, the list is already sorted.

(iii) As we start receiving the cards, we traverse the linked list, find the position of the current card, and put it there by manipulating the links. Afterward, we change the head of the linked list to the head of the sorted list.

Or any valid and similar explanation [2 Marks]

d. $O(n^2)$ + explanation **[1+1 Marks]**

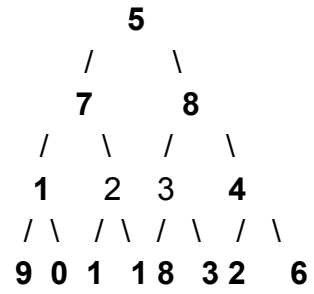
e. **Pseudo-code:**

```
InsertCard (Node node) {
        if (list is empty or head.val>=node.val) {                    [1 Mark; base case]
                node.next = head;
                head = node;
        }
        else {
                Node temp = head;
                while (temp is not null and temp.val<node.val) {    [1.5 Mark; finding correct position]
                        temp = temp.next;
                }
                node.next = temp.next;                                [1.5 Mark; insertion of new node]
                temp.next = node;
        }
}
```

4. Puchki is thrilled about this year's Christmas. She believes that Santa will come and tie gifts on the binary tree she had decorated with utmost belief. However, she is too young and short of reaching the top of the tree and hence seeks your help. She wants you to promise the following: On 25 December 2021, you need to go over the outline, tell her the value of all the gifts and give them to her.

[12 Marks]

For example, if the tree is given below (The nodes in bold consist of the outline of the tree):



The output is: 5 7 1 9 0 1 1 8 3 2 6 4 8

a. Is every binary tree a binary search tree? **[1 Mark]**
b. What is the difference between a full binary tree and a complete binary tree? **[2 Marks]**
c. What is the worst-case time complexity for search, insert and delete operations in a general Binary Search tree? **[1 Mark]**
d. Devise an algorithm to help Puchki. Elaborate briefly. **Pseudo-code is not required.** **[6 Marks]**
e. What is the time complexity of your algorithm? **[1 Mark]**
f. What is the space complexity of your algorithm? **[1 Mark]**

Ans.

a. No/False. **[1 Mark]**
b. In a full binary tree, every node can have either 0 or 2 children. A complete binary tree has all the levels filled except possibly the last level, where all the nodes have to be as left as possible. **[2 Marks]**
c. O(n); n is the number of nodes. **[1 Mark]**
   Note: O(h), is also the correct answer
d. The idea is to do a boundary traversal of the binary tree: left boundary, right boundary, and the tree's base. **[1 Mark]**
   The left boundary of the tree is the path from the root to the **left-most** node. The **right boundary** is defined as the path from the root to the **right-most** node. **[2 Marks]**
   (i) If simple boundary traversal is done, then the root node would be printed twice. It needs to be avoided. To avoid this, we print the root node first and apply boundary traversal on both children.
   (ii) Printing the tree's base would also print the tree's leftmost and rightmost nodes. To avoid printing it, the leaf nodes must be excluded while doing the left and right boundary traversal.
   [1.5 Marks for each of the edge cases] [1.5+1.5 = 3]

e. O(n); n is the number of nodes **[1 Mark][Deduct 0.5 if student does not define n]**
f. O(h); h is the height of the tree **[1 Mark][Deduct 0.5 if student does not define h]**

5. You are given an array [1.. n] representing a binary min-heap and an integer 'k'. Your friend wants to print all keys in the array that are less than 'k'.
(For example if k = 8 and the array is as shown below, then your algorithm should output keys 3, 5 and 6) **[12 Marks]**

Array Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Keys	3	10	5	13	17	6	11	15	16	21	18	9	8	23

Give an algorithm and pseudocode for your algorithm and discuss its running time. (Let m be the number of keys in A that are smaller than 'k'). Try to come up with the algorithm with the best running time possible.

Solution:
Full Marks for O(m) algorithm with correct pseudo code.
50% Marks for O(mlogN) algorithm.
(Deduct 1 Mark each for any mistakes in runtime calculation or error in pseudo code/algorithm)
O(m) solution — >(5 Marks for Algorithm + 4 Marks for pseudo-code + 3 Marks for runtime)
O(mlogN) solution —> (3 Marks for Algorithm + 2 Marks for pseudo-code + 1 Marks for runtime)
The algorithm can be as follows:

- Start from the root (Since it is the minimum key).
- For each node in consideration, if its value is less than the given 'k', you need to print it and consider its children.
- For each node in consideration, if its value is >= 'k', we should not proceed to consider its children.

Pseudo-code for the same:

```
SmallerThan(A, n, k, i):
        if(A[i] >= k or i>n ) return;
        print(A[i])
        SmallerThan (A , n , k, 2*i )
```

```
        SmallerThan(A, n, k, 2*i +1)
end
```

The function should be called as SmallerThan(A, n, k, 1)

Running time: O(m): Given the number of elements smaller than k in the array are 'm'. In every recursive call before further recursive calls are made, there are a constant number of operations performed. So the total running time of the algorithm is proportional to the total number of recursive calls made. For each element that is smaller than 'k' we make at most 2 recursive calls. Note that we don't make recursive calls when the element is at least k, so the total number of recursive calls would be bounded by 2m. 90-

6. Given an undirected graph in the form of an adjacency matrix with 'n' nodes and 'e' edges, given an edge connecting nodes 'a' and 'b' you have to determine whether this edge is a part of a cycle or not, using BFS traversals only. Give a complete algorithm, pseudocode and analyze the time complexity of the same too.

[12 Marks]

$O(n^2)$ solution — >(5 Marks for Algorithm + 5 Marks for pseudo-code + 2 Marks for runtime)
50% marks for solutions using DFS traversals or solutions with complexities > $O(n^2)$
(For this question complexity < $O(n^2)$ won't be possible since graph is given in the form of adjacency matrix)

(Deduct 1 Mark each for any mistakes in runtime calculation or error in pseudo code/algorithm)

Algorithm:
Firstly disconnect edge a <---> b temporarily.
Run a BFS from vertex a (or b), check if a is reachable from b (or b is reachable from a).
If true, edge a <—> b is a part of a cycle.
Else it is not.
Before returning, reconnect a <---> b.

Pseudocode:

```
check( graph[][], edge e):
        graph[e.vertex1][e.vertex2] = 0          (Assuming 0 denoting not adjacent)
        graph[e.vertex2][e.vertex1] = 0
        Visited[n] -> {false}*n
        bfs(e.vertex1, visited)
```

```
        graph[e.vertex1][e.vertex2] = 1
        graph[e.vertex2][e.vertex1] = 1
        if( visited[e.vertex2] = true) : return true;
        Return false;

bfs(vertex, visited):
        Queue q = {vertex}
        while( q. size() >0):
                S = q.size()
                For s times:
                        Ver = q.pop()
                        if(visited[ver]= true) continue;
                        Visited[ver] = true;
                        For nodes adjacent to ver:
                                q.add(node)
```

Analysis of Complexities:

Since the given graph is in the form of Adjacency Matrix, the complexity of BFS traversal would be $O(n^2)$. Rest all operations are O(n) and hence overall complexity would be $O(n^2)$.