Jaypee Institute ... Technology
Term -1 Examination, March 2022
B.Tech, Semester-IV

Course Title: Algorithm and Problem Solving
Cousre Code: 15B11CI411

Max. Time: 1 hours
Max Marks: 20

COURSE OUTCOMES		COGNITIVE LEVELS
CO1	Analyze the complexity of different algorithms using asymptotic analysis.	Analyze Level (Level 4)
CO2	Select appropriate sorting and searching technique for problem solving	Apply Level (Level 3)
CO3	Apply various algorithm design principles for solving a given problem	Apply Level (Level 3)
CO4	Identify, formulate and design an efficient solution to a given problem using appropriate data structure and algorithm design technique.	Create Level (Level 6)

Ques 1.a. [Marks 3 + 3 = 6] [CO1] Determine the average processing time T(n) of the recursive algorithm:

```
1    int myTest( int n ) {
2     if ( n <= 0 ) return 0;
3     else    {
4     int i = random( n - 1 );
5     return myTest( i ) + myTest( n - 1 - i );
6           }
7    }
```

In the above algorithm random(int n) spends one time unit to return a random integer value uniformly distributed in the range [0, n] whereas all other instructions spend a negligibly small time (e.g., $T(0) = 0$).

1.b. Assume that the array A contains n values, that the method randomValue takes constant number c of computational steps to produce each output value, and that the method goodSort takes n log n computational steps to sort the array. Determine the Big-Oh complexity for the following fragments of code taking into account only the above computational steps:

```
for( i = 0; i < n; i++ ) {
        for( j = 0; j < n; j++ )
                A[ j ] = randomValue( i );
        goodSort(a);
}
```

Ques2. [Marks 4] [CO2] Given an array of n objects, you need to decide if there is an object which is present more than n/2 times. The only operation by which you can access the objects is a function f, which given an array and a key returns the frequency of the element in O(n) time. Give an O(n log n)-time algorithm for this problem.

Ques 3.a. [Marks 3 + 3= 6] [CO3] Let us consider the 2D closest pair algorithm, which divides all points in two sets and recursively calls for two sets. After dividing, it finds the strip in O(n) time, sorts the strip in O(n log n) time and finally finds the closest points in strip in O(n) time. So the time complexity $T(n) = 2T(n/2) + O(n) + O(n \log n) + O(n) = 2T(n/2) + O(n \log n) = T(n \times \log n \times \log n)$ Can we optimize above version of the 2D closest pair algorithm such that the time complexity can be improved to O(n log n) ? If yes then how?

3.b. The subtle part of the 2D closest pair problem is how to efficiently deal with pairs for which one point is in the left half and the other point is in the right half. In the part of the algorithm dealing these pairs, we did not explicitly test that one point was in the left half and one point was in the right half. However for each point p, we merely checked the next 7 points in order of increasing y value, why? Can we extend the same algorithm for 3D closest pair problem? If yes, then for each such p, how many next possible closest points would be required to check?

Ques 4. [CO3] [Marks 4] In the given graph, find the shortest paths from the source to all vertices in the given graph.