Ref. No.: Ex/UG/BE/ CSE/PC/B/T/325 /2021

B.C.S.E 3RD YEAR 2ND SEMESTER EXAM 2021
(non-contact online mode)

Subject-**Design and Analysis of Algorithms**

Time: 3 hours
Full Marks: 70
Read carefully the instructions written on the body of the email through which the question paper is sent**.** Answer scripts as single file must be submitted via Email: **jukamal2015@gmail.com**

*Answer Question No. 1(**compulsory**) and question no. (**2 or 3**) and question no.(**4 or 5**) and question no. (**6 or 7**) and question no.**(8 or 9)***

1. State whether the following statements is TRUE or FALSE **with proper justifications**.

a) Insertion sort is a fast sorting algorithm for small n, but not at all, for large n.

b) Big-Oh notation provides an upper bound on the function of input size *n*, denoting the time complexity of an algorithm, but it does not say anything about how good or tight this bound is.

c) Divide and Conquer Algorithm design considers sub-problems to be non-uniform and overlapped

d) Dynamic programming approach achieves better performance by repeatedly evaluating the function calls to the similar type of sub-problems.

e) If a problem is NP-hard, it may not be NP-complete. But if the problem is NP-complete, it is NP-hard

f) Prim's Algorithm is a greedy algorithm. So it may not give correct output if applied on an undirected connected weighted graph.

g) $\sqrt{n} = \Omega(logn)$

h) $P \subset NP$

i) Shortest path tree produced by Dijktra's algorithm can never be similar to minimum spanning tree produced by Prim' s algorithm.

j) The total running time for the following code segment is O(logn)

```
sum=0;
for (i=1;i<n; i=i+2)
  {for (j=n;  j>n/2; j=j-2)
     {for(k=1; k<n/2; k=k*2)
       { sum++;
        }
      }
    }
```

3 x 10 = **30 marks**

2. a)Suppose you are choosing between the following two algorithms

I) Algorithm A solves problems of size *n* by recursively solving two sub-problems of size *n-1* and then combining the solutions in constant time

II) Algorithm B solves problem of size *n* by dividing them into nine sub-problems of size n/3, recursively solving each sub-problem , and then combining the solutions in O(n^2) time

Compute the running time for the algorithms A & B. Then show which one is better in terms of running time.

b) State Master Theorem. Can we solve the following recurrence equation using Master Theorem ?-explain. T(n)=2T(n/2) + 1 where T(1)=c(constant).

5 +5=10 marks

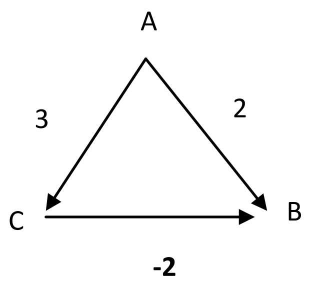
OR

3. Consider the problem of powering a number, that is, compute a^n, where n>0. Write a non-recursive and recursive algorithm for it and compare time complexity. Consider that the divide and conquer algorithm for the problem can be designed in the following way.

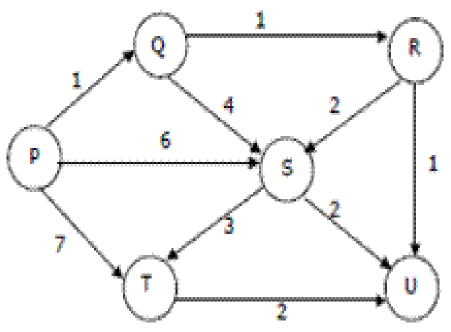
$$a^n = \begin{cases} a^{\frac{n}{2}}.\, a^{\frac{n}{2}} \;\; if\ n\ is\ even \\ a^{\frac{n-1}{2}}.a^{\frac{n-1}{2}}.a \quad if\ n\ is\ odd \end{cases}$$

Now write a function in C style for implementing the divide and conquer algorithm for the problem and find out its running time. (5+5) =10 marks

4. a) Consider the following directed graph and source node A, and explain whether Dijkstra's algorithm will give the correct solution or not.



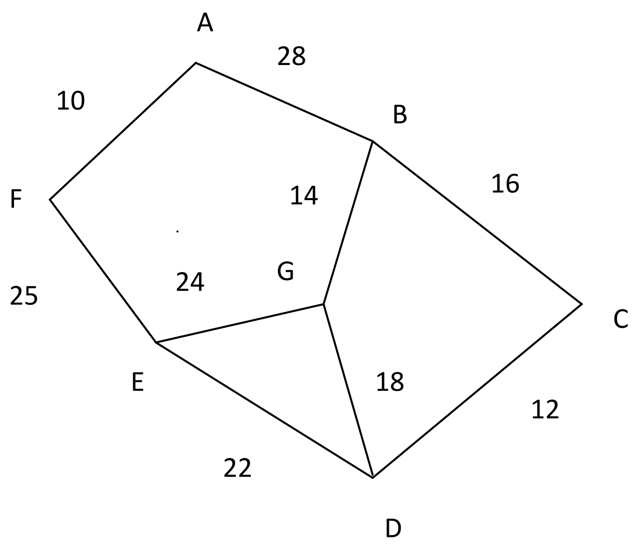
b) Consider the directed graph given below and use Dijkstra's single source shortest-path algorithm on this graph starting with **node p** (show step by step progress of the algorithm with the updates in the priority queue and node values).



3+7=10 marks

OR

5. Why Dijkstra's algorithm is a greedy algorithm?-explain. Find out minimum spanning tree for the following graph using Prim's algorithm.



3+7=10 marks

6. a) In the longest increasing subsequence problem, the input is a sequence of numbers: a_1, a_2..a_n. A subsequence is any subset of these numbers taken in order, of the form $a_{i_1}, a_{i_2} \dots a_{i_k}$, where $1 \le i_1 < i_2 < \cdots i_k \le n$, and an increasing sequence is one in which numbers are getting strictly larger. The task is to find the increasing subsequence of greatest length. For instance, the longest increasing subsequence of 5, 2, 8, 6, 3, 6, 9, 7 is 2, 3, 6, 9.

What would be the running time of the dynamic programming algorithm for the problem?-Explain with DAG based representation of all permissible solutions.

b) Use the dynamic programming approach to find minimum edit distance and the optimal alignment between the strings "D R I N K" and "D R U N K E N". Show only the table of computations, min edit distance and final optimal alignment.

5+5=10 marks

OR

7. Describe recursive solution and dynamic programming solution to finding n-th Fibonacci number. Compare time complexity for both the versions.

What is time complexity of the brute force method and dynamic programming method for solving string alignment problem. 5+5=10 marks

8. a) A problem P is reducible to another problem Q using exponential number of steps. If Q is solvable in $O(n^k)$, then P is also solvable in $O(n^\beta)$, where k and β are constant. Is the above statement true?-explain with justifications.

b) Does Halting problem belongs to NP class?-Explain.

c) Explain the easier method to check whether a given problem is NP-complete or not
4+ 3 +3=10 marks

OR

9. a) Let S be an NP-complete problem and Q and R be two other problems not known to be in NP. Q is polynomial time reducible to S and S is polynomial-time reducible to R. Which one of the following statements is true? Give reasons in support of your correct answer. Explain also why other options are incorrect.

i. R is NP-complete

ii. R is NP-hard

iii. Q is NP-complete

iv. Q is NP-hard.

. b) State the Cook's theorem.

c) Justify this statement " if any of the NP-complete problems can be solvable in polynomial time, then P=NP". 4+3+3=10 marks