EVENING

24 JUN 2022

Please check that this question paper contains 09 questions and two printed pages within first ten minutes.

[Total No. of Questions: 09] [Total No. of Pages: 02]

Uni. Roll No. ....

Program: B.Tech. (Batch 2018 onwards)
Semester: 6th
Name of Subject: Design and Analysis of Algorithm
Subject Code: PCIT-113
Paper ID: 17205

Time Allowed: 03 Hours **Max. Marks: 60**

NOTE:

1) Parts A and B are compulsory.
2) Part-C has Two Questions Q8 and Q9. Both are compulsory, but with internal choice.
3) Any missing data may be assumed appropriately.

Part – A **[Marks: 02 each]**

Q1.

a) Define NP-hard problem.

b) Explain the time complexity of Boyer-Moore Horspool algorithm.

c) Describe ε – approximate algorithms.

d) Illustrate the use of bounding function in backtracking.

e) Examine the shortcomings of the Dijkstra's Algorithm.

f) Evaluate the time complexity of the following fragment of code:-

```
for (i = n; i ≥ 1;  i = i/2)
{
    statement;
}
```

EVENING

24 JUN 2022

Part – B **[Marks: 04 each]**

Q2. Explain how greedy algorithm design technique can be used to solve Travelling Salesperson problem.

Q3. Demonstrate how backtracking can be used to solve n-Queens' problem.

Q4. Illustrate the working of Rabin-Karp algorithm for string matching.

Q5. Examine the best and worst cases of Quicksort algorithm by performing its detailed time complexity analysis.

Q6. Evaluate the efficiency of an algorithm involving dynamic programming to solve all-pairs shortest path problem.

Q7. Appraise the importance of using greedy method and relaxing the condition of $x_i = 0$ or 1 to $0 \leq x_i \leq 1$ while computing optimal solution for 0/1 Knapsack problem using a recursive backtracking algorithm.

Part – C **[Marks: 12 each]**

Q8. Identify the performance of a recursive algorithm that finds the maximum and minimum items in a set of *n* elements against a straightforward method for the same.

OR

Describe the working and performance of Prim's algorithm to compute minimum cost spanning tree.

Q9. Support the statement that an optimization problem cannot be NP-complete whereas a decision problem can be NP-complete.

OR

Design a recursive backtracking algorithm to find all the Hamiltonian cycles in a graph.

****