Department of Computer Engineering, SVNIT, Surat

M Tech I - 1st Semester, Mid-Semester Examinations

CO 603: Algorithms and Computational Complexity

10:30 hrs to 12:00 hrs, 18th October 2021

Instructions:

1. Write your Admission number clearly in the answer books along with the other details. Write page numbers on all pages that you use.
2. Assume any necessary data but giving proper justifications.
3. Be brief, precise, clear and to the point in answering the questions. Unnecessary elaboration WILL NOT fetch more marks.
4. Maximum marks = 30. Duration = 1.5 hrs. Each MAIN question carries 10 marks, maximum.
5. All sub-questions carry equal marks unless stated otherwise.

1. (a) If T(n)=2^{10n}, then T(n)= $O(2^n)$? If T(n)=2^{10+n}, then T(n)= $O(2^n)$? Your answer must be supported with a formal proof. [2]

   (b) For non-negative real-valued functions $f(n)$ and $g(n)$ defined on the positive integers, with f(n) and g(n) at least 2 for all n, and also given that f(n)=O(g(n)), and c be a positive constant, then is f(n) * lg $(f(n)^c)$ = O(g(n) $* \, lg(g(n))$)? [4]

   (c) Consider an integer vector X of size n, denoted as X = $x_1, x_2, x_3, \ldots\ldots x_n$, given as input to the Quicksort algorithm. Consider also a sorted instance of this vector denoted as L= $\ell_1, \ell_2, \ell_3, \ldots\ldots \ell_{i-1}, \ell_i, \ell_{i+1}, \ldots\ldots \ell_{j-1}, \ell_j, \ell_{j+1}, \ldots\ldots \ell_n$. Now with respect to the given data, answer the following questions: [4]

      (i) If the number of comparisons made by the algorithm, each comparing the two elements x_i and x_j is denoted by $\sum_{j=i+1}^{n}\sum_{i=1}^{n-1} X_{ij}$, and the expected value of a random variable A is denoted by E(A), then what is the expression for the expected value of the average number of comparisons made ?

      (ii) If P_{ij} is the probability that the two elements ℓ_i and ℓ_j are compared, then in terms of P_{ij}, what is the value of the average number of comparisons made ?

      (iii) In the given sorted instance L, when would the two keys ℓ_i and ℓ_j, be compared ?

      (iv) In terms of 1(c)(iii) above, what is the average number of comparisons made by the quicksort algorithm? How ?

2. (a) Consider the following problem: You are incharge of the MIS department at an institute. The MIS department has a processor that can operate 24 hours a day, every day. People submit requests to run daily jobs on the processor. Each such job comes with a start time and an end time within the 24 hours of one day only. And, if the job is accepted to run on the processor, it must run continuously, every day, for the period between its start and end times. Note that those jobs that are specified with the times that begin before midnight and end after midnight - are not accepted - that is, a job must start and complete within a specific date only - which means that the date can be ignored.
   Given a list of n such jobs, your goal is to accept as many jobs as possible (regardless of their length), subject to the constraint that the processor can run at most one job at any given point in time. Design an algorithm with a running time that is polynomial in n, and write its pseudocode. You may assume for simplicity that no two jobs have the same start or end times. An Example: Consider the following four jobs, specified by (start-time, endtime) pairs. (6 P.M., 11 P.M.), (9 A.M., 4 P.M.), (3 A.M., 1 P.M.), (2 P.M., 6 P.M.). Then, the optimal solution would be to pick the three jobs, which can be scheduled without overlapping. [2]

   (b) Give a formal proof on the correctness of the algorithm that you designed in 2(a). [3]

(c) Given the Mergesort recurrence by expressions viz.
$\rightarrow T(n) \leq 0$ if n=1 and
$\rightarrow T(n) \leq T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n$, otherwise,
that does not assume n to be a power of 2, prove that the solution to the recurrence, still is T(n) = O(n lg n). [5]

3. (a) A small business say, a photocopying service with a single large machine faces the following scheduling problem: each morning, they get a set of jobs from customers. They want to do the jobs on their single machine in an order that keeps their customers happiest. The customer i's job will take t_i time to complete. Given a schedule (i.e. an ordering of the jobs), [6]

- let C_i denote the finishing time of job i. For example, if job j is the first to be done, we would have $C_j = t_j$ and if job j is done right after job i, we would have $C_j = C_i + t_j$.
- Each customer i also has a given weight w_i that represents his or her importance to the business. The happiness of customer i is expected to be dependent on the finishing time of the customer i's job.
- Hence, the company decides that they want to order the jobs to minimize the weighted sum of the completion times, i.e. to minimize $\sum_{i=1}^{n} w_i * t_i$

Design an efficient algorithm to solve this problem and analyse its complexity, giving logical arguments. Consider that, you are given a set of n jobs with processing time t_i, and weight w_i, for each job. You want to order the jobs so as to minimize the weighted sum of the completion times, $\sum_{i=1}^{n} w_i * t_i$.

(b) Consider the following problem: Prof M S Gaur, the Director, IIT Jammu, drives an automobile from Jammu to Delhi along NH1A and then onwards. His car's petrol tank, when full, holds enough petrol to travel n kms and his map gives the distances between petrol stations on his route. The professor wishes to make as few petrol station stops as possible along the way. Assume that an efficient method, based on the Greedy design technique using which Professor Gaur can determine, at which minimal number of petrol stations should he stop is given and is as shown in the box, below.
Then, show using formal notations, how does this solution exhibits **the Optimal substructure and the Greedy choice properties** ? [4]
You can use the following notations:

- m possible gas stations,
- the first stop is the k^{th} gas station, i.e. there are k gas stations beyond the start within n kilometres of the start and the greedy solution chooses the k^{th} station as its first stop.
- an optimal solution has s gas stations, say...

Starting will a full tank of petrol, Professor Gaur should go to the farthest gas station he can get to within n kilometres of Jammu. Fill up there. Then go to the farthest gas station further he can get to, within n kilometres of where he last filled up, and fill up there, and so on. Looked at another way, at each petrol station, Professor Gaur should check whether he can make it to the next petrol station without stopping at this one. If he can, skip this one. If he cannot, then fill up. Professor Gaur doesn't need to know how much petrol he has or how far the next station is to implement this approach, since at each fillup, he can determine which is the next station at which hell need to stop.

***paper ends here**