INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR	
	Stamp / Signature of the Invigilator

EXAMINATION (End Semester)	SEMESTER (Spring)

Roll Number									Section		Name	
Subject Number	C	S	2	1	0	0	3	Subject Name	*Algorithms – I*			
Department / Center of the Student									Additional sheets			

Important Instructions and Guidelines for Students

1. You must occupy your seat as per the Examination Schedule/Sitting Plan.
2. Do not keep mobile phones or any similar electronic gadgets with you even in the switched off mode.
3. Loose papers, class notes, books or any such materials must not be in your possession, even if they are irrelevant to the subject you are taking examination.
4. Data book, codes, graph papers, relevant standard tables/charts or any other materials are allowed only when instructed by the paper-setter.
5. Use of instrument box, pencil box and non-programmable calculator is allowed during the examination. However, exchange of these items or any other papers (including question papers) is not permitted.
6. Write on both sides of the answer script and do not tear off any page. **Use last page(s) of the answer script for rough work.** Report to the invigilator if the answer script has torn or distorted page(s).
7. It is your responsibility to ensure that you have signed the Attendance Sheet. Keep your Admit Card/Identity Card on the desk for checking by the invigilator.
8. You may leave the examination hall for wash room or for drinking water for a very short period. Record your absence from the Examination Hall in the register provided. Smoking and the consumption of any kind of beverages are strictly prohibited inside the Examination Hall.
9. Do not leave the Examination Hall without submitting your answer script to the invigilator. **In any case, you are not allowed to take away the answer script with you.** After the completion of the examination, do not leave the seat until the invigilators collect all the answer scripts.
10. During the examination, either inside or outside the Examination Hall, gathering information from any kind of sources or exchanging information with others or any such attempt will be treated as **'unfair means'**. Do not adopt unfair means and do not indulge in unseemly behavior.

Violation of any of the above instructions may lead to severe punishment.

Signature of the Student

To be filled in by the examiner

Question Number	1	2	3	4	5	6	7	8	9	10	Total
Marks Obtained											

Marks obtained (in words)	Signature of the Examiner	Signature of the Scrutineer

CS21003 Algorithms–I, Spring 2017–2018

End-Semester Test

24–April–2018 V4, NR121/122/221/322/422, 02:00pm–05:00pm Maximum marks: 60

[*Write your answers in the question paper itself. Be brief and precise. Answer all questions.*
If you use any algorithm/result/formula covered in the class, just mention it, do not elaborate.]

1. In a *ternary search tree* T, each node contains two keys key_1 and key_2, and three child pointers $left$, mid, and $right$. Let v be any node in T, k_1 a key stored in any node in the subtree rooted at $left(v)$, k_2 a key stored in any node in the subtree rooted at $mid(v)$, and k_3 a key stored in any node in the subtree rooted at $right(v)$. Then, we must have $k_1 < key_1(v) < k_2 < key_2(v) < k_3$. A ternary search tree is called *admissible* if the three subtrees at any node of the tree differ in height by at most one.

(a) Let $N(h)$ denote the minimum number of nodes in an admissible ternary search tree of height h. Derive a recurrence relation for $N(h)$. Also, supply the required initial condition(s). **(4+1)**

(b) Let h be the height of an admissible ternary search tree with n nodes. Deduce that $h \leqslant \log_2 n$. (This result shows that admissible ternary search trees are height-balanced.) **(5)**

2. The *Manhattan distance* between two points (h,k) and (h',k') in the Euclidean plane is defined as $|h-h'|+|k-k'|$. You are given n points $P_i=(x_i,y_i)$, $i=0,1,2,\ldots,n-1$, randomly chosen in the unit square (that is, the 1×1 square with corners at $(0,0)$, $(0,1)$, $(1,0)$, and $(1,1)$). The problem is to find, given a query point $Q=(x,y)$ in the unit square, a point P_i closest to Q with respect to the Manhattan distance. Your task is to organize the given points P_i in a data structure such that each query can be answered in expected constant time. Propose a suitable data structure for storing the points P_i. Also state the algorithm how each query is answered. Argue that each query can be processed in expected constant time. **(4+4+2)**

3. Let $G = (V, E)$ be a connected undirected graph with n vertices and (exactly) n edges.

(a) What are the minimum and the maximum numbers of spanning trees that G can have? Justify. **(3)**

(b) Let each edge e of G carry a positive cost (or weight) $c(e)$. Propose an $O(n)$-time algorithm to compute an MST (a minimum spanning tree) of G. Comment on what data structures you used in your algorithm. **(5+2)**

4. Let $G = (V, E)$ be a directed acyclic graph with $V = \{0, 1, 2, \ldots, n-1\}$. An edge (i, j) is in E if and only if $0 \leqslant i < j \leqslant n-1$. Suppose that each edge $(i, j) \in E$ carries a cost $c(i, j)$. You should not assume that all $c(i, j)$ are positive (or non-negative). In other words, negative-cost edges are allowed. We want to solve the single-source-shortest-path (SSSP) problem with source $s = 0$.

(a) Deduce that G contains $\Theta(n^2)$ edges. **(3)**

(b) Prove that there are exactly 2^{i-1} directed paths from the source $s = 0$ to the vertex $i \in \{1, 2, 3, \ldots, n-1\}$. **(4)**

(c) Prove/Disprove: Dijkstra's SSSP algorithm works for this G (in the presence of negative-cost edges). **(3)**

(d) Irrespective of whether Dijkstra's SSSP algorithm works for G or not, it takes $O(n^2 \log n)$ running time in this case. Propose an $O(n^2)$-time algorithm to solve the SSSP problem in the given G (with source $s = 0$). **(10)**

5. Let S be a string of lower-case letters *a–z*. A *blanagram* of S is obtained by changing a single letter of S, and then (optionally) permuting the symbols of this changed S. For example, let $S =$ *entrain*. Its blanagram *terrain* is obtained by changing one n to r, and then permuting the letters. Likewise, *trainer* is another blanagram of *entrain* (but not of *terrain*, because *trainer* and *terrain* consist of exactly the same letters).

Suppose that two strings S and T of the same length n and with the symbols in the range *a–z* are given as input (these need not be dictionary words). Your task is to determine whether S and T are blanagrams (of one another). Propose an $O(n)$-time algorithm to solve this problem. **(10)**