

Name:

ID Number:

Exam Type:

...A...

Exam Number:

.....

Grade: 1: ... 2: ... 3: ... 4: ... 5: ... 6: ... 7: 8: ... 9: ... Total:

SOLVE all the problems IN THE SPACE PROVIDED

THERE ARE 5 (FIVE) PAGES THIS PAGE INCLUDED
Read and Sign the statement below at the end of the exam
Unsigned exams will be marked with grade 0 (zero).

STATEMENT

On my honor, I pledge that I have not violated the provision of the NJIT Student Honor Code.

Sign below at the end of the exam

Signature

List of useful formulae

$$\sum_{i=1}^n i = \frac{n(n+1)}{2}, \quad \sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6}, \quad \sum_{i=1}^n i^3 = \frac{n^2(n+1)^2}{4}, \quad n! \approx \left(\frac{n}{e}\right)^n, \quad a^{\log_b n} = n^{\log_b a},$$

For $x \neq 1$, we have that

$$\sum_{i=0}^n x^i = \frac{x^{n+1} - 1}{x - 1}, \quad \sum_{i=0}^{n-1} ix^i = \frac{(n-1)x^{n+1} - nx^n + x}{(1-x)^2}, \quad \sum_{i=0}^{\infty} i \cdot (1/2)^i = 2$$

NOTE: Give an algorithm means, present the algorithm, show that it works as claimed, and analyze its running time by providing an asymptotically tight bound.

Master Method. $T(n) = aT(n/b) + f(n)$, such that $a > 0$, $b > 1$, $f(n) > 0$.

M1 If $f(n) = O(n^{\lg_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\lg_b a})$.

M2 If $f(n) = \Theta(n^{\lg_b a} \lg^k n)$, then $T(n) = \Theta(n^{\lg_b a} \lg^{k+1} n)$, where $k \geq 0$ is a non-negative constant.

M3 If $f(n) = \Omega(n^{\lg_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and for large n , then $T(n) = \Theta(f(n))$.

* : Read Syllabus for a possible quiz in April and how this could affect Exam 1.

Problem 1. (30 POINTS)

Give a **TRUE** or **FALSE** for each one of the statements below. Any answer other than a full **TRUE** or **FALSE** will be considered wrong. All algorithms are the book/notes implementations.

- (1) QuickSort is stable.
- (2) On the sorted sequence $\langle 1, 2, \dots, n \rangle$, HeapSort is asymptotically faster than InsertionSort.
- (3) HeapSort is stable.
- (4) The asymptotically fastest so far algorithm that is stable and sorts in-place is MergeSort.
- (5) The in-degree of a vertex of $G = (V, E)$ using an adjacency list representation can be found in $O(n)$ time, $n = |V|$.

Problem 2. (33 POINTS)

Solve the following recurrence relations by providing asymptotically tight bounds. You only need to provide the bound, intermediate derivations are not required. If no boundary case is given, the choice of the constants is yours. You may assume that $T(n)$ is positive and monotonically increasing, if you need to do so.

- (1) $T(n) = T(n/4) + n.$
- (2) $T(n) = 9T(n/3) + n.$
- (3) $T(n) = 4T(n/4) + n \lg n.$

Problem 3. (30 POINTS)

Solve exactly the following recurrence. You may assume that n is a power of two, if it is convenient.

$$T(n) = 2T(n/2) + n - 1, \quad T(2) = 2.$$

Problem 4. (60 POINTS)

For this problem input size is specified separately for each case and thus **Row-1, Column-1** indicates a problem size of n^2 keys. Graphs have n vertices and m edges and represented using an adjacency matrix. Find the worst-case running time of every (efficient) algorithm in **Column-1** by using the time bounds in **Column-2**; use only class-introduced algorithms. If multiple answers are possible, use the asymptotically tightest bound. The same answer in **Column-2** can be reused. Row 7 is filled for you; you should fill the remaining rows. If you see $\lg^t n$, it means $(\lg n)^t$.

Column-1	Answer	Column-2
1: HeapSort of n^2 keys		A. $O(\lg n)$
2: QuickSort of $\sqrt{n}$ keys		B. $O(\sqrt{n})$
3: Binary Search on n^3 sorted keys		C. $O(n)$
4: InsertionSort on $n \lg n$ sorted keys		D. $O(n \lg n)$
5: LinearSearch on $n \lg n$ keys		E. $O(n^2)$
6: Find the indegree of vertex u		F. $O(n^2 \lg n)$
7: MergeSort of n keys	D	G. $O(n^2 \lg^2 n)$
		H. $O(n^4)$

Problem 5. (30 POINTS)

You are given an array $A[0..n]$. The first n positions $A[0..n-1]$ contain all but one of the positive integers $1, \dots, n+1$, in some arbitrary order. Find which integer is missing in $A[0..n-1]$ and return it by using an algorithm that is time and space efficient. Show the correctness of your algorithm and justify your arguments(s). (Efficient means it cannot be improved upon.)

Problem 6. (30 POINTS)

Geeglo search engine. Y. R. NJIT searches for tom and jerry by typing in Geeglo's textbox TOM JERRY. Y.R.NJIT is interested in finding all the documents that contain both TOM and JERRY. The engine returns a sorted sequence T of document numbers (IDs), i.e. all documents containing the word TOM and a sorted sequence J of document numbers (IDs) containing the word JERRY. The length (number of documents) of T is $\sqrt{n}$ and the length of J is n . You need to generate the output C that contains all the document numbers that contain both TOM and JERRY; no duplicates, not necessarily sorted. How much space and time does Y.R.NJIT need to determine the answer C using a space and time efficient algorithm that would determine all the documents containing both keywords TOM, JERRY? Explain.

Problem 7. (30 POINTS)

A set X of n integers, and another set Y of n integers are also given along with integer q . Determine whether there exist $x \in X, y \in Y$ such that $x + y - q = 0$. Use as little extra space as possible (other than the space for X, Y). Your algorithm must also be time efficient. (You may assume that all integers are distinct. 'Determine' means you print ONE pair x, y if such exists or 'NOTFOUND' if no such pair exists.)

Problem 8. (60 POINTS)

(a) The inorder and preorder traversals of a binary tree starting from the root are respectively

Preorder: 10, 30, 50, 60, 40, 70, 80

Inorder: 50, 60, 30, 10, 70, 40, 80

What is the binary tree consistent with those two traversals?

(b) The text below of 25 characters A, C, G, T, W is first encoded in a byte-aligned ASCII code, and then using Huffman coding. (Ignoring spaces that are provided for readability only), what are the prefix codes for each one of the five characters under Huffman coding, and what is the total number of bits used to encode just the text (consisting of A, C, G, T, W and ignoring the spaces or other auxiliary information)? How does this compare to the ASCII encoding?

CCAGT TTCCC WAAGT TCCAA WCAGA

Problem 9. (30 POINTS)

(Time efficient means asymptotic time efficiency cannot be improved upon. Space efficiency is to be interpreted likewise. Comparison efficient means the number of comparisons cannot be improved upon: constants are important for higher-order terms.)

You are given n keys in array A . The keys have no particular order. We would like to determine the 10 smallest of the n keys and display them in the output in sorted order (smaller to larger). Give a comparison efficient and space and time efficient algorithm, explain the algorithm, analyze its worst-case running time, its comparison count, and space complexity. Justify your arguments.