

Name:

ID Number:

Exam Type:

...A...

Exam Number:

.....

Grade: 1: ... 2: ... 3: ... 4: ... 5: ... Total:

SOLVE all the problems IN THE SPACE PROVIDED

THERE ARE 3 (THREE) 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. (35 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) InsertionSort does not sort in-place.
- (2) SelectionSort is stable.
- (3) On the sorted sequence $\langle 1, 2, \dots, n \rangle$, SelectionSort is asymptotically faster than MergeSort.
- (4) On the sorted sequence $\langle 1, 2, \dots, n \rangle$, InsertionSort has running time that is $O(n \lg n)$.
- (5) The asymptotic solution of $T(n) = 2T(n/2) + n$ is $T(n) = \Theta(n \lg n)$.
- (6) The asymptotic solution of $T(2n) = 2T(n) + n$ is $T(n) = \Theta(n \lg n)$.
- (7) The asymptotic solution of $T(n) = T(n/2) + n^2$ is $T(n) = \Theta(n^2 \lg n)$.

Problem 2. (50 POINTS)

Rank the following functions by order of growth; that is, find an arrangement $g_1, g_2, g_3, \dots, g_6, g_7$ of the functions satisfying $g_1 = \Omega(g_2)$, $g_2 = \Omega(g_3)$, $g_3 = \Omega(g_4)$, $g_4 = \Omega(g_5)$, $g_5 = \Omega(g_6)$, $g_6 = \Omega(g_7)$. Partition your list in equivalence classes such that $f(n)$ and $h(n)$ are in the same class if and only if $f(n) = \Theta(h(n))$. For example for functions $\lg n, n, n^2$, and $2^{\lg n}$ you could write: $n^2, \{n, 2^{\lg n}\}, \lg n$.

$$3^n, \quad n^3, \quad n!, \quad 2^{3 \lg n}, \quad n / \lg n, \quad 10000, \quad (n - 3)!.$$

Problem 3. (40 POINTS)

Show that $n^3 - 2n^2 = \Theta(n^3)$ by providing the n_0, c_1, c_2 of the definition of Θ .

Problem 4. (41 POINTS)

No asymptotics! You are given the following sorted sequences of a total (combined) $4n$ keys: A with n sorted keys, B with $2n$ sorted keys, C with $n/2$ sorted keys and D with $n/2$ keys. How many comparisons does it take to merge (optimally) those sequences into a sorted sequence of $4n$ keys? Explain and BE precise; no asymptotics. (You may assume n is a multiple of 2.)

Problem 5. (34 POINTS)

Fill in the table below by providing the running time for every one of the indicated operations and indicated data structure (DaSt) of **size** n containing distinct keys. *Fill entries by writing n to denote for example running time of $O(n)$ or $\Theta(n)$, 1 to denote $\Theta(1)$, and so on. We fill in the entry for Succ(z) into an SDL. Since it is $\Theta(1)$ we write 1.*

(a) **Ins**, **Sea** is for Insertion and Search of a key x . (b) **FindMin** or **FindMax** determine and return the value of the key with the smallest or largest value among the keys of the DaSt; the key remains in it. (c) **ExMin** or **Exmax** perform FindMin or FindMax respectively and in addition remove the corresponding key from DaSt. (d) **Del** deletes the record referenced/pointed/indexed by z i.e. the location that will be deleted is known. (e) **Succ** is the successor of z , i.e. the smallest among all the elements greater than or equal to the value pointed by reference/pointer/index z . For removal operations the data structures must be maintained in likewise form after the operation (i.e. there are no gaps or locations with no keys).

SA, RA Sorted Array (SA) and Reverse-Sorted Array with elements ordered appropriately in increasing or decreasing order.

SLL, RLL Sorted SinglyLinked List (SLL) and Reverse-Sorted SinglyLinked List (RLL) and the first value (smallest or largest) key pointed by a head pointer; no tail pointer.

SDL, RDL Sorted DoublyLinked List (SDL) and Reverse-Sort DoublyLinkedList defined similarly as before; no tail pointer again.

	RA	RLL	SDL
Ins(x)			
Del(z)			
Sea(x)			
FindMin			
ExMax			
Succ(z)			1