

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

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

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 $T(n) = \Theta(f(n))$.

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 and thus constants are important for higher-order terms: thus $\Theta(n)$ is worse than $3n$, and $3n$ is worse than $2n + o(n)$.

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 class implementations.

- (1) HeapSort is stable.
- (2) MergeSort sorts in-place.
- (3) On the sorted sequence $\langle 1, 2, \dots, n \rangle$, HeapSort is asymptotically faster than InsertionSort.
- (4) On the decreasing input sequence $\langle n, n-1, \dots, 1 \rangle$, HeapSort is asymptotically faster than QuickSort.
- (5) On a complete binary tree with n nodes, the number of internal nodes is $(n-1)/2$.
- (6) The best case running time of InsertionSort is $\Theta(n)$.

Problem 2. (50 POINTS)

(a) For the following code fragment what is its running time in terms of n ? Use as tight an asymptotic notation as possible.

(b) What is the (exact) value of k returned in line 7 in terms of n ?

```
int codeA(int n)
1.  int i,j,k=0;
2.  for(i=1;i<=n;i=i*2) {
3.      for(j=1;j<=n;j=j*2) {
4.          k++;
5.      }
6.  }
7.  return(k);
```

Problem 3. (40 POINTS)

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

$$T(n) = 2T(n/2) + 12n, \quad T(4) = 6.$$

Problem 4. (60 POINTS)

For this problem input size is specified separately for each case and thus **Row-7, Column-1** indicates a problem size of n keys. 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 (best) 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: BuildMaxHeap on n keys		A. $O(\lg n)$
2: HeapSort of $n/\lg n$ keys		B. $O(\sqrt{n})$
3: Binary Search on 2^n sorted keys		C. $O(n)$
4: InsertionSort on $\sqrt{n}$ keys		D. $O(n \lg n)$
5: LinearSearch on $\sqrt{n}$ keys		E. $O(2^{n-1})$
6: MergeSort of $n \lg n$ keys		F. $O(n \lg^2 n)$
7: MergeSort of n keys	D	G. $O(n^2 \lg^2 n)$
		H. $O(\lg(\lg n))$

Problem 5. (33 POINTS)

The text below of 18 English characters is first encoded in a byte-aligned ASCII code (give number of bits used, 1ASCII=1B) and then using Huffman coding. (Ignoring spaces that are provided for readability only), what are the prefix codes for each one of the characters under Huffman coding, and what is the total number of bits used to encode just the text (and ignoring the spaces or other auxiliary information)? How does this compare to the ASCII encoding (compare bit count)?

ALGO ALGO ALGO RRRR RM

Problem 6. (30 POINTS)

The following recursive algorithm GOOD (lines 0–6) is invoked on an array of n keys according to the invocation also shown below `GOOD(A[0..n-1], 0, n-1)`. What is the operation performed by GOOD ? Reason about it. What is the running time of GOOD on an unordered array of n keys? Justify your answer.

```
GOOD(keys A[0..n-1], int p, int r) {  
0.  int q;  
1.  if (p==r)  
2.    return(TRUE);  
3.  if (A[p] != A[r])  
4.    return(FALSE);  
5.  q= (p+r)/2;  
6.  return (    GOOD(A,p,q) &&    GOOD(A,q+1,r))  
}  
  
1.  GOOD(A[0..n-1], 0, n-1);
```

Problem 7. (30 POINTS)

What is the binary tree T that is consistent with the following two traversals of it?

Inorder traversal of T : 40, 20, 50, 60, 10, 30.

Postorder traversal of T : 40, 60, 50, 20, 30, 10.

Problem 8. (30 POINTS)

What is a comparison-OPTIMAL approach for solving the following problem? Name or briefly describe the algorithm. In addition, describe the comparison-count as accurately as possible; use asymptotic notation as needed for low order terms. You may assume that n is a power of two and all the input keys are distinct. Finding the smallest, second smallest and the third smallest of n keys.

Problem 9. (30 POINTS)

Guugle search engine. I. M. NJIT uses the Guugle engine to search for all the documents that contain both words DATA and STRUCTURES, and thus types in the textbox of the search engine DATA STRUCTURES. The engine is primitive and experimental and returns (a) sorted sequence D of document numbers (IDs), i.e. all documents containing the word DATA and, (b) a separate sorted sequence S of document numbers (IDs) containing the word STRUCTURES. The length (number of documents) of D is n^2 and the length of S is n . I.M. NJIT thus needs to compute output C that contains all the document numbers that contain both DATA and STRUCTURES i.e. the intersection of the two "doclists" D and S . How much space and time does I.M.NJIT need to determine the answer C using a space and time efficient algorithm that would determine all the documents containing both keywords DATA, STRUCTURES for this problem instance? (C need not be sorted; its keys must be unique though, i.e. no duplicates.) Explain.