Solve all 7 (SEVEN) problems in the space provided
Read the Problems CAREFULLY!

There are 6 (SIX) pages this page included; the last page is left blank

State of New Jersey Institute of Technology
March 8, 2007

Exam 2 (333 points) in 2 hours

Name: ...........................................

ID Number: ............................ Exam Type: ...A...

Exam Number: ...........

Grade: 1: ... 2: ... 3: ... 4: ... 5: ... 6: ... Total: .......

Solve all 7 (SEVEN) problems in the space provided

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 ...........................................

Any algorithm you present must be given in concise and complete form. Argue about its correctness. Analyze its (worst-case) running time and express it in asymptotic notation. Random ramblings or sketches will not be given any points. You may use algorithms presented in class as black-boxes without further description. For example, instead of repeating the code of MergeSort you can just write MergeSort(B,m) to indicate that you sort an array B of m keys.

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

\[ \sum_{i=0}^{n} x^i = \frac{x^{n+1} - 1}{x - 1}, \quad \sum_{i=0}^{n} ix^i = \frac{(n-1)x^{n+1} - nx^n + x}{(1-x)^2}, \]

B1. \( f(n) = \Theta(g(n)) \) iff \( \exists \) positive constants \( c_1, c_2, n_0 : 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \) \( \forall n \geq n_0. \)

B2. \( f(n) = \Omega(g(n)) \) iff \( \exists \) positive constants \( c_1, n_0 : 0 \leq c_1 g(n) \leq f(n) \) \( \forall n \geq n_0. \)

B3. \( f(n) = O(g(n)) \) iff \( \exists \) positive constants \( c_2, n_0 : 0 \leq f(n) \leq c_2 g(n) \) \( \forall n \geq n_0. \)

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

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

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

M3 If \( f(n) = \Omega(n^{\log_b a + \epsilon}) \) for some constant \( \epsilon > 0 \), and if \( af(n/b) \leq cf(n) \) for some constant \( c < 1 \) and for large \( n \), then \( T(n) = \Theta(f(n)) \).
Problem 1. (50 POINTS)
We have 27 coins all of the same weight except one that is fake and weighs LESS than the other coins. We also have a balance scale; any number of coins can be put on one or the other side of the scale at any one time and the scale will tell us whether the two sides weigh the same or which side weighs more (or less). Can you find the fake coin with only 3 weighings? Explain (and justify your answer).

Problem 2. (50 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.

\[
\begin{align*}
T(n) &= 8T(n/2) + n^4. \\
T(n) &= 4T(n/2) + n^2. \\
T(n) &= T(n-1) + n
\end{align*}
\]
Problem 3. (50 POINTS)
Solve exactly the following recurrence. You may assume that \( n \) is a power of two.

\[
T(n) = 2T(n/2) + 8n, \quad T(4) = 9.
\]

Problem 4. (50 POINTS)
Mark clearly as **TRUE** or **FALSE** each one of the following statements. No justification required. Algorithms are the ones of the textbook/notes. All input instances below consist of \( n \) keys. Any answer other than a fully written TRUE or FALSE will be considered wrong.

1. QuickSort is stable.
2. CountSort is stable.
3. RadixSort sorts in-place.
4. On the increasing input sequence of integers \( \langle 1, 2, \ldots, n \rangle \), InsertionSort is asymptotically faster than CountSort.
5. On the increasing input sequence of integers \( \langle 1, 2, \ldots, n \rangle \), MergeSort is asymptotically faster than CountSort.
6. On the decreasing input sequence of integers \( \langle n, n-1, \ldots, 1 \rangle \), RadixSort is asymptotically faster than HeapSort.
7. The solution of \( T(n) = T(n/5) + T(7n/10) + n \), is \( T(n) = \Theta(n \lg n) \).
8. \( \lg (n!) = O(n \lg n) \).
9. Sorting \( n \) keys in the range \( 0, \ldots, n^2 - 1 \), RadixSort is asymptotically faster than HeapSort.
10. Sorting \( n \) keys in the range \( 0, \ldots, 2^n - 1 \), RadixSort is asymptotically faster than MergeSort.
Problem 5. (50 points)

We perform an inorder and preorder traversal on an arbitrary binary tree. For each one of the traversals we get as output:

Inorder: 1, 2, 4, 3, 7, 6.
Preorder: 4, 2, 1, 7, 3, 6.

Show the binary tree that is consistent with these two traversals.

Problem 6. (50 points)

You are given $n$ numbers in the form of array $A[0..n−1]$. Each number can be an integer in an arbitrary range or similarly, a real; no generic assumptions thus can be made about the range and type of these numbers. You are asked to give a linear-time algorithm that finds the number $x$ that minimizes $\sum_{i=0}^{n-1} |A[i] − x|$ where $|y|$ is the absolute value of $y$. The output $x$ can be one of the input values, or can be computed from a subset of the input values. Justify the correctness of your algorithm and analyze its worst-case running time to verify the linear-time claim.
Problem 7. (33 points)
You are given two sets $A$ and $B$ of $n$ keys each. Each set is represented by an array; the elements of $A$ are all distinct and so are the elements of $B$. Give an efficient algorithm that finds the intersection $C = A \cap B$. If an element $z$ appears in both $A$ and $B$, it should appear only once in $C$. Make sure the running time of your algorithm is asymptotically faster than quadratic. “Efficient algorithm” means an algorithm whose worst-case performance can not be improved upon. So if you suggest an algorithm whose running time is $O(n)$ but there exists an algorithm whose running time is $o(n)$ your algorithm is not efficient and you will get 0 points. Justify your answer and analyze your algorithm.

This is the end of the exam.
Left blank intentionally