A. V. Gerbessiotis  
CS 610-102  
March 2014  
Spring 2014  
Exam 1 (333 points)  
120 mins

Name: .................................................................  
ID Number: .........................  Exam Type: ......  Exam Number: ..............

Grade: 1: ... 2: ... 3: ... 4: ... 5: ... 6: ... 7: ... 8: ... 9: ... 10: ... 11: ... Total: ......

Solve all the problems in the space provided

There are 6 (SIX) 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

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

For \( x \neq 1 \), we have that

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

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. \( T(n) = aT\left(\frac{n}{b}\right) + f(n) \), such that \( a > 0, b > 1, f(n) > 0 \).

M1 If \( f(n) = O\left(n^{\log_{b} a - \epsilon}\right) \) 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 \( T(n) = \Theta(n^{\log_{b} a} \log^{k+1} n) \), 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.** (33 points)
Answer with a TRUE or a FALSE or give a brief answer.

(a) Did you switch off your mobile phone/device? (If NOT, DO IT NOW!)
(b) Did you write your ID on page 1?

(c,d,e) Name three divide-and-conquer algorithms introduced in class.

(f) Name a greedy algorithm introduced in class.

**Problem 2.** (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) InsertionSort is stable.
(2) QuickSort is stable.
(3) On the sorted sequence \(\langle 1, 2, \ldots, n \rangle\), InsertionSort is asymptotically faster than MergeSort.
(4) On the reverse-sorted sequence \(\langle n, n-1, \ldots, 1 \rangle\), InsertionSort has running time that is \(\Theta(n^2)\).
(5) The asymptotic solution of \(T(n) = T(n/2) + \lg n\) is \(T(n) = \Theta(\lg n)\).
(6) There exists a comparison-based sorting algorithm that sorts 4 keys using at most 5 comparisons in the worst-case.

**Problem 3.** (30 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.

\[
\begin{align*}
(1) \quad T(n) & = T(n/2) + n^2. \\
(2) \quad T(n) & = 8T(n/2) + n^3. \\
(3) \quad T(n) & = T(n-1) + 1.
\end{align*}
\]
Problem 4. (30 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 5. (30 points)
You may assume that $\lg n$ and $\sqrt{n}$ are both integer in this problem. You are given two sorted sequences stored in arrays $A$ and $B$ of $\lg n$ and $n^2$ keys respectively. We are interested in forming the intersection $C = A \cap B$, i.e. an array $C$ containing the common keys of $A$ and $B$; common keys appear once in $C$. Give a space and time efficient algorithm for solving this problem. (Time efficient means its running time cannot be asymptotically improved upon; space efficient means likewise for space requirements.) Justify your arguments.

Problem 6. (30 points)
Using Huffman coding, how many bits do you need to compress this text of 18 characters (count only bits of character representation)?

ALGOALGOALGORRRRRM
Problem 7. (30 points)
You are given a positive integer $n$. We want to determine whether $n$ is a perfect square, i.e. whether there exists integer $x > 1$ such that $x^2$ is equal to $n$. Give an $o(\sqrt{n})$ algorithm that determines whether such an $x$ exists and prints it, or says "NO". (Note the little-o in the asymptotic notation; one multiplication costs one.)

Problem 8. (30 points)
A set $A$ of $n$ integers is given. Determine in worst case time $o(n^3)$ (little oh) whether there exist three different integers $a, b, c$ of $A$ so that $a - b + c = 0$. (You may assume all integers are distinct.)
Problem 9. (30 points)


Give an algorithm that outputs a hilly-sorted sequence from an arbitrary input of \( 2n+1 \) distinct keys and whose worst-case running time is \( o(n^2) \). This is a little-oh \( o \), not a big-Oh \( O \). Can you make your hilly-sort algorithm in-place? Justify all your arguments. Example 10 < 90 > 30 < 70 > 50 < 60 > 40 < 80 > 20 is hilly-sorted.

Problem 10. (30 points)

Google top 10-ranked documents (aka page one). You are given \( n \) keys arbitrarily ordered each one representing the rank of a web-page. Any two such keys can be compared to each other and we can determine whether one is less than, equal to, or greater than the other. We want to find the 15 largest of the \( n \) keys and then display them in reverse sorted order, larger to smaller of the 15.

(a) Give an algorithm that solves this problem in no more than \( 15n + o(n) \) comparisons.

(b) Give an algorithm that solves this problem in no more than \( 4n + o(n) \) comparisons.
Problem 11. (30 points)
Fill in the table below by providing the running time for every one of the indicated operations and every one indicated data structure of size $n$. Express the answer in terms of $n$. Fill entries by writing $n$ to denote running time of $O(n)$, 1 to denote running time of $O(1)$, and so on. All keys already in data structure are distinct. We fill in the entry for $\text{Succ}(z)$ into an SDL. Since it is $\Theta(1)$ we write 1.

- (a) $\text{Ins}$ is Insertion of a key $x$ into data structure,
- (b) $\text{Del}$ is Deletion of a record of the data structure pointed by a pointer/reference (for linked structures) or index (for array-based structures) $z$, i.e. the location that will be deleted is known,
- (c) $\text{Sea}$ is Search for key $x$.
- (d) $\text{FindMin}$ is FindMinimum, i.e. determine and return the value of the key with the smallest value among the keys of the data structure; the key remains in it.
- (e) $\text{FindMax}$ is FindMaximum, defined similarly to FindMinimum.
- (f) $\text{ExMin}$ is ExtractMin, that is return and remove the key with smallest value.
- (g) $\text{ExMax}$ is ExtractMax, that is return and remove the key with largest value.
- (h) $\text{Succ}$ is the successor of $z$, i.e. the smallest among all the elements greater than or equal to the value pointed by pointer/reference $z$ (it has nothing to do with pointers/references).

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** Sorted Array (SA) with elements ordered in increasing key order.

**RSL** ReverseSorted Singly Linked List (RSL) with the elements in decreasing order and the largest value key pointed by a head pointer; no tail pointer.

**SDL** Sorted Doubly Linked List (SDL) with the elements in increasing order and the smallest value key pointed by a head pointer; no tail pointer.

**MXH** A max-heap stored as an array $A[0..\text{sizeof}(A) - 1]$, where $\text{sizeof}(A) > n = \text{length}(A)$, i.e. $A$ is big enough for a new key to be inserted. $A[0]$ is holding the MAX key.

<table>
<thead>
<tr>
<th></th>
<th>SA</th>
<th>RSL</th>
<th>SDL</th>
<th>MXH</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Ins}(x)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{Del}(z)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{Sea}(x)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{FindMin}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{ExMax}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{Succ}(z)$</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>