A. V. Gerbessiotis  
CS 610-102  
February 2015  
Spring 2015  
Exam 1 (200 points)*  
60 mins

Name: ..........................................................  
ID Number: .......................  
Exam Type: ...A...  
Exam Number: ............  
Grade: 1: ... 2: ... 3: ... 4: ... 5: ... 6: ... Total: .......

Solve all the problems in the space provided

There are 4 (FOUR) 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 $f(n) = O(n^{\log_b a - \epsilon})$ 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 $T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$, 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 $T(n) = \Theta(f(n))$.

*: Read Syllabus for a possible quiz in April and how this could affect Exam 1.
Problem 1. (6 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) Name 2 divide-and-conquer algorithms introduced in class. (2 points)
(d) Name an incremental algorithm introduced in class. (2 points)

Problem 2. (50 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) MergeSort is stable.
(2) SelectionSort does not sort in-place.
(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 $O(n^3)$.
(5) The asymptotic solution of $T(n) = T(n/2) + n + T(2n/4)$ is $T(n) = \Theta(n \lg n)$.

Problem 3. (45 points)
Rank the following functions by order of growth; that is, find an arrangement $g_1, g_2, g_3, \ldots, g_6, g_7$ 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: $n^2, \{n, 2^{\lg n}\}, \lg n, 2^n, n, n!, 2^{3\lg n}, n \lg n, \lg (n!), (n - 1)!$. 
Problem 4. (30 points)
Show that $n^4 - n^3 - n^2 + 1 = \Theta(n^4)$ by providing the $n_0, c_1, c_2$ of the definition of $\Theta$.

Problem 5. (35 points)
You are given an array $A$ of $m$ keys $A[0..m-1]$. We want to determine whether the keys are distinct (and thus answer YES) or are not-distinct (and thus answer NO). Distinct means that no key appears twice or more; not-distinct means there is at least one key appearing twice or more. Give an algorithm that is time efficient that solves this problem. Analyze its worst-case running time and justify your arguments. (Time efficient means it cannot be solved asymptotically faster than other methods.)
Problem 6. (34 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 $O(n)$ or $\Theta(n)$, 1 to denote running time of $\Theta(1)$, 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.
- (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).

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