Midterm 2

This is a closed book exam. You can only use one two-sided sheet of notes. No book, no other notes, no calculator, no collaboration.

Read the questions carefully, and write your solution clearly and concisely. Partial credit will be given for incomplete solutions.

GOOD LUCK!

Problem 1. Master Theorem

Here is a table of logarithms. In the row $i$ and column $j$ you find the value of $\log_i j$.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$\log_2 1 = 0$</td>
<td>$\log_2 2 = 1$</td>
<td>$\log_2 3 = 1.5849$</td>
<td>$\log_2 4 = 2$</td>
</tr>
<tr>
<td>3</td>
<td>$\log_3 1 = 0$</td>
<td>$\log_3 2 = 0.5849$</td>
<td>$\log_3 3 = 1$</td>
<td>$\log_3 4 = 1.2618$</td>
</tr>
<tr>
<td>4</td>
<td>$\log_4 1 = 0$</td>
<td>$\log_4 2 = 0.6309$</td>
<td>$\log_4 3 = 0.7924$</td>
<td>$\log_4 4 = 1$</td>
</tr>
<tr>
<td>5</td>
<td>$\log_5 1 = 0$</td>
<td>$\log_5 2 = 0.4306$</td>
<td>$\log_5 3 = 0.6826$</td>
<td>$\log_5 4 = 0.8613$</td>
</tr>
</tbody>
</table>

Solve the following recursions ($c$ is always a constant). Give only the final results.

(a) $T(n) = T(n/4) + c$.
(b) $T(n) = 3T(n/2) + cn^3$.
(c) $T(n) = 3T(n/3) + cn$.
(d) $T(n) = 2T(n/3) + c\sqrt{n}$.

Problem 2. Finding Repetitions

You are given a (not necessarily sorted) array $A$ of $n$ (not necessarily distinct) integers. Describe and analyse an $O(n)$ time algorithm that determines if there exists an element $x$ that appears in $A$ more than $n/10$ times.

Problem 3. Analysis of algorithms

Given a vector of integers $X = x_1, \ldots, x_n$ we want to find the $\frac{n}{\log n}$ smallest elements, and we want to output them in sorted order. We can assume that all the elements of $X$ are distinct.

What is the worst-case running time of the following strategies?
Using mergesort. Sort the array using mergesort, and then output the first \( \frac{n}{\log n} \) elements of the sorted array. The pseudo-code is as follows:

Algorithm A \((X[1], \ldots, X[n])\)
begin
  mergesort \((X[1], \ldots, X[n])\)
  allocate vector \(Y[]\) of size \(n/\log n\)
  for \(i := 1\) to \(n/\log n\)
    \(Y[i] := X[i]\)
  return \((Y)\)
end

Using Select. Find the \( \frac{n}{\log n} \)-th element in \(X\), call \(v\) its value, then list all the the \( \frac{n}{\log n} \) elements of \(X\) that are \(\leq v\), finally use mergesort to sort them. The pseudocode is as follows:

Algorithm B \((X[1], \ldots, X[n])\)
begin
  \(s = \text{select}(X[1], \ldots, X[n], n/\log n)\)
  \(v = X[s]\)
  allocate vector \(Y[]\) of size \(n/\log n\)
  \(j = 1\)
  for \(i := 1\) to \(n\)
    if \((X[i] \leq v)\) then
      \(Y[j] := X[i]\)
      \(j := j + 1\)
  mergesort \((Y[1], \ldots, Y[n/\log n])\)
  return \((Y)\)
end

Note: we assume that \(\text{select}(X[i], \ldots, X[n], k)\) returns an index \(s\) such that \(X[s]\) is the \(k\)-th order statistics of \(X[1], \ldots, X[n]\).

Using a Binomial Heap. Create an empty Binomial Heap \(H\), then put all the elements of \(X\) into it using \(n\) \texttt{insert()} operations, then for \(\frac{n}{\log n}\) times extract the minimum of \(H\) and append it to a list. Finally output the list.

Algorithm C \((X[1], \ldots, X[n])\)
begin
  create empty Binomial Heap \(H\)
  allocate vector \(Y[]\) of size \(n/\log n\)
  for \(i := 1\) to \(n\)
    \texttt{insert} \((X[i], H)\)
  for \(j := 1\) to \(n/\log n\)
    \(Y[j] := \text{find-min}(H)\)
    \(\text{delete-min}(H)\)
  return \((Y)\)
end