1. (10 pts) Suppose $A$ and $B$ are two sorted arrays, each with $n$ numbers. Design an $O(\log n)$
time algorithm to find the median of the set $A \cup B$. (You can assume that there are no
duplicates in the set $A \cup B$.)

2. (20 pts) Solve the following recurrences. Assume in each case that $T(c) = 1$, for an
appropriate constant $c$. Provide sufficient details to show how you got the answer. (For
example, show how you applied the Master method, or details of any other method you
used.)

(a) $T(n) = 3T(n/2) + n^2$
(b) $T(n) = 4T(n/2) + n^2$
(c) $T(n) = 5T(n/2) + n^2$
(d) Solve the following recurrence both using the Master Method and the basic recur-
rence expansion method.

$$T(n) = \sqrt{n}T(\sqrt{n}) + n$$

Why do you think you get different answers? Which answer is the correct one?

3. (20 pts) Suppose we have a large database that includes many copies of each data point.
In particular, suppose the total number of entries in the database is $N$ but there are only
$m$ distinct entries, and $N$ is much much larger than $m$. As an example, the database


has 32 data points but only 6 distinct values. This compact representation of the database
takes only $O(m)$ memory, instead of $\Theta(N)$.

Rather than storing the database as above, we can store it as a list of tuples of the form
(value, frequency), with unsorted order of values. Our example database can be stored
as $\{(D, 2), (B, 8), (F, 1), (A, 16), (C, 4), (E, 1)\}$.

Given such a representation, design an $O(m)$ worst-case time algorithm for computing
the $k$th smallest data value in the original database. Your algorithm should take time
$O(m)$, which is the size of the compressed database, and report the item that has the
$k$th smallest value in the uncompressed database. (For instance, assuming the dictionary
order $A < B < C < D < E < F$ in our example, the answer for $k = 20$ is $B$, and answer
for $k = 25$ is $C$.)

Describe your algorithm concisely but clearly, prove its correctness, and analyze its run-
ing time.
The following problem will not be graded but is a good problem to practice your analysis and proof skills.

- The linear-time Selection algorithm described in class divides the input elements into groups of 5, and then recursively solves the problem.
  - Re-analyze the algorithm when we divide the elements into groups of 3. Specifically, determine the sizes of the subproblems on which recursive calls will be made. Write down the final recurrence, and solve it. Does the algorithm still run in $O(n)$ time?
  - Re-analyze the algorithm if the elements are divided into groups of 7. Again, determine the sizes of the recursive subproblems, derive the final recurrence, and solve it. Does the algorithm still run in $O(n)$ time?

Submission Instructions.

- Submissions and grading will be done through Gradescope: http://gradescope.com
- Sign up for the course using the entry code: MX3B6B
- After signup, you can see the homeworks for the course and due date in your account. Select Homework 2 and just follow the instructions on the website. You can either submit one image per question, or upload one pdf and identify the pages for each question.