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. (15 pts) 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?

3. (20 pts)
   (a) Solve the following recurrence both using the Master Method and the basic recurrence expansion method.

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

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

   (b) Show that the following recurrence $T(n) = 2T(n/2) + 2$ for finding both min and max solves to $T(n) = 3n/2 - 2$.

4. (15 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, with $N \gg m$. As an example, the database may be

which has 32 data points but only 6 distinct values.

Rather than storing the database as above, we can store the data as a list of tuples of the form \((value, \text{frequency})\), with unsorted order of values. For instance, our example database can be stored as \{\((D, 2)\), \((B, 8)\), \((F, 1)\), \((A, 16)\), \((C, 4)\), \((E, 1)\)\}. This compact representation of the database takes only \(O(m)\) memory, instead of \(\Theta(N)\).

Given such a representation, design an \(O(m)\) worst-case time algorithm for computing the \(k\)th smallest data value in the original database. That is, your algorithm should take time \(O(m)\), which is the size of the compressed database, but report the item that has the \(k\)th smallest value in the uncompressed database. Describe your algorithm concisely but clearly, prove its correctness, and analyze its running time.