NOTE: All students should complete exercises 1–5. Students in the honors section should additionally complete exercises 6–8, and hand in the solutions to these problems separately on Wednesday, Oct. 10.

1. Design an algorithm to compute the union of two sets of numbers. Each set is given as an array of distinct elements. The output should be an array consisting of the distinct elements of the union. The algorithm should run in time $O(n \log n)$, where $n$ is the total number of elements in both sets.

2. Given two sets of numbers $S_1$ and $S_2$, and a number $x$, find whether there exists an element of $S_1$ and an element of $S_2$ whose sum is exactly $x$. Each set is given as an array. The algorithm should run in time $O(n \log n)$, where $n$ is the total number of elements in both sets. How would you modify your algorithm if one set were substantially smaller than the other (e.g., $|S_1| = O(\log n)$)?

3. The input is a list of $n$ points $(x, y)$. You are to sort these points so that the $x$-coordinates are nondecreasing: if the sorted list is $(x_1, y_1), \ldots, (x_n, y_n)$, then $x_i \leq x_{i+1}$ for $i = 1, \ldots, n-1$. Moreover, assume that there are many duplicates among the $x$-coordinates, such that the number of distinct $x$-coordinates is only $O(\log n)$. Design an algorithm for this problem that runs in time $O(n \log \log n)$.

4. In our analysis of Quick Select in class, we showed that if $N_j$ is the problem size at level $j$, then $E[N_j] \leq \alpha^j n$, where $\alpha = \sqrt{2/3}$. Show that this bound in fact holds for $\alpha = 3/4$.

Hint: the key is to analyze $E[\max(S, T)]$, where $S$ and $T$ is uniformly distributed over $\{0, \ldots, n-1\}$, and $T = n-1 - S$. Using the defining formula for expectation, it should be easy to show that $E[\max(S, T)] \leq (3/4)n$, just by drawing the right picture.

5. Design and analyze an algorithm that takes as input $k$ sorted lists, and merges the lists into one sorted list. If the total number of elements in all the lists is $n$, your algorithm should run in time $O(n \log \log n)$.

6. Honors Exercise. Prove an $\Omega(n \log k)$ lower bound for the previous problem in the comparison model.

Hint: assume the input consists of $k$ sorted lists, each with $n/k$ elements; moreover, assume that the numbers in the lists are integers in the range $1, \ldots, n$, with each integer appearing exactly once; show that there are at least $(k!)^{n/k}$ such inputs.

7. Honors Exercise. The input is a min-heap of size $n$ (with the smallest element at the top) given as an array, a number $x$, and an integer $k$, where $1 \leq k \leq n$. Design an algorithm to determine whether the $k$th smallest element in the heap is at most $x$. The worst case running time of your algorithm should be $O(k)$, independent of $n$. You may use $O(k)$ space.

Hint: you do not have to find the $k$th smallest element; you need only determine its relationship to $x$.

8. Honors Exercise. Consider again the analysis of quick sort presented in class. We calculated an expected recursion depth of $O(\log n)$. If you look closely at the proof, the bound on the expected is actually $c \ln(n) + O(1)$, where $c = 2/\ln(3/2) \approx 4.933$, where “$\ln$” is the natural logarithm. The goal of this exercise is to develop a better bound. The algorithm remains the same, but instead of estimating the expected value of the sum of squares at level $j$, we estimate the sum of the $\beta$th powers, where $\beta$ is a parameter that we can adjust (in the analysis — the
choice of $\beta$ does not affect the algorithm itself). By adapting the proof in class, derive a bound on the expected depth of $F(\beta) \ln(n) + O(1)$, where

$$F(\beta) := \frac{\beta}{\ln\left(\frac{\beta+1}{2}\right)}.$$ 

Now find the value of $\beta$ that minimizes $F(\beta)$. To do this, use a calculator, computer program, or online calculator, such as Wolfram Alpha — for this exercise, simply describe the steps you took to solve this and report your numerical findings, including the optimal value of $\beta$ and the corresponding value $F(\beta)$. You should be able to beat 4.933. As it happens, this approach will yield the best bound possible.

If you feel truly inspired, for extra credit, try repeating the above parameterized analysis for the median-of-three version of quick sort from the last exercise.