This HW has some exercises, three regular problems, one programming problem, and one bonus problem. All problems on written HWs are to be done individually, no collaboration is allowed.

The exercises are to fortify your knowledge of the basics. Please solve them! You can talk to the course staff if you need help. However, you should not hand in your solutions. We will release sample solutions later so that you can check your work.

Solutions to the three written problems should be submitted as a single PDF file using gradescope, with the answer to each problem starting on a new page.

Submission instructions for the programming problem will be posted on the website and Piazza.

At the end of Problem 3 is a bonus problem. It is an interesting fun problem that is not that hard. But it optional, and you won’t get hints or assistance from the course staff. It is worth 1% under the quiz/bonus category if you solve it correctly. No partial credit. No collaboration. We’ll create a separate place on gradescope for submitting the bonus.

0. (Exercises: Recurrences and Probability.)

Solve each recurrence below in Θ notation. As always, prove your answer. For all of these problems $T(1) = 1$.

(a) Solve $T(n) = 3T([n/2]) + n$.

(b) Now solve $T(n) = 3T([n/2]) + n \lg n$.

(c) Finally, solve $T(n) = n^{2/3}T([n^{1/3}]) + n$.

(E.g., we might get this from a divide-and-conquer procedure that uses linear time to break the problem into $n^{2/3}$ pieces of size $n^{1/3}$ each. Hint: write out the recursion tree.)

(25 pts) 1. (A Good Sort) Consider the following problem.

INPUT: an $n \times n$ matrix $M$ containing $n^2$ distinct numbers, where the $n$ numbers in each row are in sorted order. (Such a matrix is called a row-sorted matrix.)

OUTPUT: a sorted list $L$ of the $n^2$ numbers in the matrix $M$.

EXAMPLE: $n = 3$, so $n^2 = 9$. Say the 9 numbers in $M$ are the digits 1, . . . , 9. Possible inputs (row-sorted matrices) include:

\[
\begin{array}{ccc}
1 & 4 & 7 \\
3 & 5 & 8 \\
2 & 6 & 9
\end{array}
\quad
\begin{array}{ccc}
1 & 4 & 5 \\
2 & 7 & 8 \\
3 & 6 & 9
\end{array}
\quad
\begin{array}{ccc}
3 & 4 & 6 \\
2 & 5 & 9 \\
1 & 7 & 8
\end{array}
\quad
\begin{array}{ccc}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{array}
\quad
\begin{array}{cc}
\text{or} & \text{or} & \text{or} & \text{or}
\end{array}
\quad
\begin{array}{cc}
\ldots & \ldots
\end{array}
\]
The output for all these would be the sorted list \( L = 1, 2, 3, 4, 5, 6, 7, 8, 9 \).

It is clear that we can solve this problem using at most \( 2n^2 \lg n \) comparisons by forgetting about the row-sorted structure of the input matrix \( M \), and sorting the \( n^2 \) numbers using, say, MergeSort. (Remember that \( \lg n^2 = 2 \lg n \), where \( \lg x = \log_2 x \).

In this problem you will show how to do better, and then give a lower bound. For simplicity, you can assume \( n \) is a power of 2.

(a) Show how to solve this problem using at most \( n^2 \lg n \) comparisons.

(b) Show that any comparison-based algorithm to solve this problem must use at least \( n^2 \lg n - O(n^2) \) comparisons.

Some hints for part (b): Show that if you could solve this problem using fewer than that many comparisons, then you could use this to violate the \( \lg(m!) \) lower bound for comparisons needed to sort \( m \) elements (which we prove in Lecture #2). You may want to use the fact that \( m! > (m/e)^m \). Also, recall that you can merge two sorted arrays of size \( k \) using at most \( 2k - 1 \) comparisons.

(25 pts) 2. **(Cut and Merge!)

(a) Give an algorithm that takes as input an unsorted array \( A \) with \( n \) distinct elements, and \( k \) distinct integers \( 1 \leq i_1 < i_2 < \ldots < i_k \leq n \), outputs the \( k \) elements in \( A \) having these ranks using \( O(n \log k) \) comparisons.\(^1\)

(b) In the MERGE problem we have to combine or merge two sorted lists into a single sorted list. Suppose we have two sorted lists of length \( m \) and \( n \) respectively (without loss of generality, we assume \( m \leq n \)). Show that any deterministic comparison-based algorithm for MERGE must use \( \Omega \left( m \log \left( \frac{n+m}{m} \right) \right) \) comparisons in the worst case.

(25 pts) 3. **(Preprocessing for Query Day.)** You are given as input a set \( S \) of \( n \) distinct elements. Today you can preprocess it using at most \( P(n) \) comparisons and store the results. Tomorrow you will be given some element \( q \) and you have to answer whether \( q \in S \)? You can use \( S \) and this stored information to compute this answer in \( Q(n) \) time. (We are in a comparison-based model for this problem, so just count the number of comparisons.)

(a) **(Do not submit!)** Show that if \( P(n) = n \lg n \), then you can achieve \( Q(n) = \lg n \), and if \( P(n) = 0 \), then you can get \( Q(n) = n \), both using deterministic algorithms.

(b) Show that if \( P(n) = 0 \), then any deterministic algorithm must have \( Q(n) \geq n \).

(c) Give a randomized algorithm so that \( P(n) = \frac{1}{2} n \lg n \), such that the expected query time \( Q(n) \) for any fixed query \( q \) is at most \( \sqrt{n} + \frac{1}{2} \log_2 n \).

---

\(^1\)The rank of an element \( a \) with respect to an unsorted array \( A \) is the number of elements in \( A \) that no greater than \( a \). If \( A \) has distinct elements, the minimum element has rank 1, and the median has rank \( |A|/2 \).
(Bonus 1 point.) Show a matching lower bound. (We are happy with a lower bound for deterministic algorithms.) Suppose $P(n) = \frac{1}{2}n \lg n - O(n)$. Show that any deterministic algorithm that uses preprocessing time $P(n)$ must spend query time $Q(n) \geq \sqrt{n}$ for some input.

(25 pts) 4. Programming: Multi-Medians

Write a program which takes as input a list of $n$ distinct numbers $a_1, a_2, \ldots, a_n$, and outputs the elements of ranks $1, 2, 4 = 2^2, 8 = 2^3, 2^4, \ldots, 2^k$, where $2^k$ is the greatest power of 2 that is at most $n$.

Your algorithm should have $O(n)$ running time (deterministic worst-case or randomized expected). Please include a comment at the start of your program explaining your algorithm, and why it runs in linear time.

Details on how to submit, grading policy, etc., will be on the course website and piazza soon. Please do not use any built-in functions (or libraries) for sorting or selection.

INPUT: The first line contains $n$, which is at most $10^6$. The second line consists of the numbers $a_1, a_2, \ldots, a_n$ separated by blanks. These numbers satisfy $-10^9 \leq a_i \leq 10^9$.

OUTPUT: The first line of the output is $k$. The second line consists of the required $k + 1$ numbers, separated by spaces.

For example, if the input is:

```
5
8 3 1 2 6
```

then the output is:

```
2
1 2 6
```

Or if the input is:

```
10
12 5 1000000000 6 7 9 10 8 14 23
```

then the output is:

```
3
5 6 8 14
```