**The first page of your homework submission must be a cover sheet answering the following questions.** Do not leave it until the last minute; it's fine to fill out the cover sheet before you have completely finished the assignment. Assignments submitted without a cover sheet, or with a cover sheet obviously dashed off without much thought at the last minute, will not be graded.

- How many hours would you estimate that you spent on this assignment?

- Explain (in one or two sentences) one thing you learned through doing this assignment.

- What is one thing you think you need to review or study more? What do you plan to do about it?

**Question 1.** Suppose you are choosing between the following three algorithms:

1. Algorithm A solves problems by dividing them into five subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time.
2. Algorithm B solves problems of size $n$ by recursively solving two subproblems of size $n-1$ and then combining the solutions in constant time.
3. Algorithm C solves problems of size $n$ by dividing them into nine subproblems of size $n/3$, recursively solving each subproblem, and then combining the solutions in $O(n^2)$ time.

What are the running times of each of these algorithms (in asymptotic notation) and which would you choose?

**Question 2** (K&T 5.1)**.** You are interested in analyzing some hard-to-obtain data from two separate databases. Each database contains $n$ numerical values—so there are $2n$ values total—and you may assume that no two values are the same. You'd like to determine the median of this set of $2n$ values, which we define to be the $n$th smallest value.

Notice that according to this definition, the median of $[1,3,4,7]$ is 3, *not* 3.5.

However, the only way you can access these values is through *queries* to the databases. In a single query, you can specify a value $k$ to one of the two databases, and the chosen database will return the $k$th smallest value that it contains. Since queries are expensive, you would like to compute the median using as few queries as possible.

Give an algorithm that finds the median value using $O(\log n)$ queries. If it makes things easier, you may assume that $n$ is a power of two.

*2*

As always, make sure that you justify the correctness of your algorithm, and analyze its time complexity.

**Question 3.** Recall that the *Fibonacci numbers* $F_n$ are defined recursively by

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_n &= F_{n-1} + F_{n-2}, \end{aligned}$$

with the first few given by $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$

As you may know, directly turning the definition of Fibonacci numbers into a recursive implementation is a terrible idea; the resulting algorithm takes $\Theta(\varphi^n)$ time (where $\varphi = (1+\sqrt{5})/2 \approx 1.618\ldots$).

The usual iterative algorithm to repeatedly calculate the next Fibonacci number from the previous two seems like it would take $\Theta(n)$ time to compute $F_n$: it just loops from 1 to $n$ and does one addition each loop, right? Well... yes, it does one addition each loop, but we can't really assume that these additions take constant time, because the Fibonacci numbers involved can get quite large! Let's analyze the situation more carefully.

(a) Recall from a previous homework that the size of the $n$th Fibonacci number $F_n$ is $\Theta(\varphi^n)$. Given this fact, approximately how many bits (in terms of $\Theta$) are needed to represent $F_n$? Simplify your answer as much as possible.

(b) Now suppose we implement the usual iterative algorithm as follows: initialize an array $F$ of size $n$, which can hold arbitrary-size integers. Initialize $F[0] = 0$ and $F[1] = 1$. Then loop $i$ from 2 to $n$, and at each iteration, set $F[i] \leftarrow F[i-1] + F[i-2]$. Taking into account the time needed to add integers of a given size, what is the running time of this algorithm?

(c) We can do better! In addition to their definition, Fibonacci numbers satisfy the following recurrences (you can just take my word for it):

$$F_{2n+1} = F_n^2 + F_{n+1}^2$$
$$F_{2n} = 2F_nF_{n+1} - F_n^2$$

For example, $F_7 = 13 = F_3^2 + F_4^2 = 2^2 + 3^2$, and $F_8 = 21 = 2F_4F_5 - F_4^2 = 2 \cdot 3 \cdot 5 - 3^2$.

Explain how to turn these recurrences into a recursive algorithm for computing Fibonacci numbers.

*3(c)*

(d) Analyze the running time of this algorithm. Be sure to include the time needed to do any additions or multiplications. Assume we will use Karatsuba's algorithm for multiplication.

**Question 4.** An array $A[1 \ldots n]$ is said to have a *majority element* if *more than half* of its entries are the same. Given an array, the task is to design an efficient algorithm to tell whether the array has a majority element, and, if so, to find that element. The *only* thing you may assume about the elements of the array is that you can test whether two of them are equal (in constant time). In particular, the elements of the array are *not* necessarily from some ordered domain like the integers, and so there can be no comparisons of the form $A[i] > A[j]$. You also may not assume that there is a hash function for the elements, so they cannot be used as the keys of a dictionary/hash table.

*4b*

(a) What is a brute-force algorithm for this problem? How long does it take to run?

(b) Show how to solve this problem in $O(n \log n)$ time. Make sure to prove that your algorithm is correct (via induction) and give a recurrence relation for the running time of your algorithm.

*4c*

(c) (**Extra credit**) Can you give a linear-time algorithm?