**Problem 1:** You are given two sorted lists of size $m$ and $n$. Give a $O(\log k)$ time algorithm for computing the $k$-th smallest element in the union of the two lists. **Note:** Observe that the $k$-th smallest element in the union of the arrays $a[a_1, \cdots, a_m]$ and $b[b_1, \cdots, b_n]$ has to be contained in $a[a_1, \cdots, a_k]$ or $b[b_1, \cdots, b_k]$. For simplicity you can assume $k \leq \min(m, n)$ and all the elements of both sets are distinct, and the elements across the two sets are also distinct.

**Problem 2:** Karatsuba algorithm that was introduced during the lecture described an algorithm that multiplies two $n$-bit binary integers $x$ and $y$ in time $n^a$, where $a = \log_2 3$. Call this procedure `fastmultiply(x, y)`.

(a) We want to convert the decimal integer $10^n$ (a 1 followed by $n$ zeros) into binary. Here is the algorithm (assume $n$ is power of 2):

```python
function pwr2bin(n)
    if (n == 1): return 1010
    else:
        z = ???
        return fastmultiply(z, z)
```

Fill in the missing details. Then give a recurrence relation for the running time of the algorithm, and solve the recurrence.

(b) Next, we want to convert any decimal integer $x$ with $n$ digits (where $n$ is a power of 2) into binary. The algorithm is the following:

```python
function dec2bin(x)
    if (n == 1): return binary[x]
    else:
        split $x$ into two decimal numbers $x_L, x_R$ with $n/2$ digits each
        return ???
```

Here `binary[·]` is a vector that contains the binary representation for all one-digit integers. That is `binary[0] = 0_2, binary[1] = 1_2, up to binary[9] = 1001_2`. Assume that a lookup in `binary` takes $O(1)$ time.

Fill in the missing details. Once again, give a recurrence for the running time of the algorithm, and solve it.

**Problem 3:** The Hadamard matrices $H_0, H_1, H_2, \cdots$ are defined as follows:

- $H_0$ is the $1 \times 1$ matrix $[1]$
- For $k > 0$, $H_k$ is the $2^k \times 2^k$ matrix

\[
H_k = \begin{bmatrix}
H_{k-1} & H_{k-1} \\
H_{k-1} & -H_{k-1}
\end{bmatrix}
\]

Design $O(n \log n)$ divide-and-conquer algorithm that given a column vector $v$ of length $n = 2^k$, computes the matrix-vector product $H_k v$. Analyze the time complexity of your algorithm.
Problem 4: There are a lot of customers in SERVEMEFIRST bank, and only one representative, that can
serve them. The time, needed to serve each customer is known in advance. It is \( t_i \) minutes for a customer
\( i \). The total waiting time is

\[
T = \sum_{i=1}^{n} (\text{waiting time of } i\text{th customer})
\]

For example, there are three customers, \( C_1(t_1 = 5 \text{ minutes}), C_2(t_2 = 15 \text{ minutes}), \) and \( C_3(t_3 = 2 \text{ minutes}) \), that are being served in order \( C_1 - C_2 - C_3 \). Then the waiting time of the 1\text{st} customer is 0
minutes, 2\text{nd} customer - 5 minutes, and 3\text{rd} customer - 5 + 15 = 20 (minutes). The total waiting time is
\( T = 0 + 5 + 20 = 25 \) (minutes). If the order of the customers changed, then the total waiting time may
change as well.

The manager of the bank wants to minimize the total waiting time. Give a greedy (efficient) algorithm
for computing the optimal order in which to process the customer. Prove why your algorithm is correct (i.e.
always returns an optimal solution).