1. Sometimes an algorithm doesn’t always need the entire input to produce the correct output. Give an example of such an algorithm.

2. Assume given an alphabetically ordered list consisting of 25000 names. We sequentially search for a certain name \( N \). How many steps are needed in the best-case, worst-case, average-case?


4. Give a worst-case and a best-case input array of length 5 for insertion sort.

5. Adapt the pseudo-code for insertion sort to sort in non-increasing (instead of non-decreasing) order, and show correctness of your algorithm.

6. Give pseudo-code for the algorithm of Euclid for computing the greatest common divisor.

7. Give the time complexity of the following loops in terms of \( \Theta \).

   **Algorithm Loop1\((n)\):**
   
   \[
   s := 0 \\
   \text{for } i := 1 \text{ to } n \text{ do} \\
   \hspace{1em} s := s + i
   \]

   **Algorithm Loop2\((n)\):**
   
   \[
   p := 1 \\
   \text{for } i := 1 \text{ to } 2n \text{ do} \\
   \hspace{1em} p := p \cdot i
   \]

8. Consider the following algorithm, with \( A[1 \ldots n] \) an array with \( n \) integer values:
Algorithm maxSubarray(A, n):

max := 0
for left := 1 to n do
    sum := 0
    for right := left to n do
        sum := sum + A[right]
        if sum > max then
            max := sum
    done
done
return max

(a) What does this algorithm compute?
(b) What is the worst-case time-complexity in terms of Θ?

9. Consider the following definition of the power function, for n ≥ 0:

\[ p(x, n) = \begin{cases} 
1 & \text{if } n = 0 \\
x \cdot p(x, n-1) & \text{if } n > 0 
\end{cases} \]

Give a pseudocode description of an algorithm Power(x, n) to compute the power function according to this definition. As an example, calculate Power(x, 5) with a sequence of equations. Argue (informally) that the number of recursive calls is in Θ(n).

10. Consider the following definition of the power function, for n ≥ 0:

\[ q(x, n) = \begin{cases} 
1 & \text{if } n = 0 \\
x \cdot q(x, \frac{n-1}{2})^2 & \text{if } n > 0 \text{ even} \\
q(x, \frac{n}{2})^2 & \text{if } n > 0 \text{ odd} 
\end{cases} \]

Give a pseudocode description of an algorithm Qower(x, n) to compute the power function according to this definition. As an example, calculate Qower(x, 8) with a sequence of equations. Argue (informally) that the number of recursive calls is in Θ(log n).

11. We compare (sorting) algorithms I with running time function \( I(n) = 8n^2 \) and M with running time function \( M(n) = 64n \log n \). (The logarithm is for base 2.) For what n is I faster than M?

12. We compare algorithms P with running time \( P(n) = 2^5 \cdot n^2 \) and E with running time \( E(n) = 2^n \). For what n is E faster than P?