1. The below algorithm receives numbers in an array, and returns the distance between the two closest ones:

   procedure minDist(A[1..n] of float) returns float:
   
   \[ \begin{align*}
   &d := \infty; \\
   &\text{for } i := 1 \text{ to } n \text{ do} \\
   &\quad \text{for } j := 1 \text{ to } n \text{ do} \\
   &\quad \quad \text{if } i \neq j \text{ and } |A[i] - A[j]| < d \text{ then} \\
   &\quad \quad \quad d := |A[i] - A[j]|; \\
   &\text{return } d; \\
   \end{align*} \]

   What is the time complexity of the algorithm? Modify the algorithm so that it works at least 50% faster. Justify the complexity of the new algorithm.

2. (a) Simulate insertion sort by ordering the letters of the word "ESIMERKKI" in alphabetical order.

   (b) A sorting algorithm is stable if it keeps equal keys in their original mutual order. Is insertion sort stable? If it is, explain why, or give a counter-example if it is not.

3. Implement insertion sort of integer arrays using some programming language which allows to measure the time taken by different phases of execution. In Java one can measure time usage for example as follows:

   ```java
   long t0 = System.currentTimeMillis();
   myInsertionSort(A, 0, n-1);
   long t1 = System.currentTimeMillis();
   System.out.print(t1-t0 + " ms\n");
   ```

   (a) Experiment to find out which is the largest number \( n_a \) of items in increasing order that your implementation is able to process in 100 ms. (If needed, you can allocate more memory (say, 128 MB) for the Java virtual machine by invoking it with command `java -Xmx128m`.)
(b) Find out similarly the maximum size of an array in decreasing order that can be processed within the same time.

(c) Insertion sort of \( n \) items that are in increasing order (the easiest case) takes roughly \( an \) milliseconds, and insertion sort of items in decreasing order (the worst case) takes roughly \( bn^2 \) milliseconds. Solve the constants \( a \) and \( b \) based on your measurements.

(d) Based on the preceding, estimate how long would it take for your implementation to sort \( n_a \) items that are in decreasing order.

4. Explain why the following basic relations between asymptotic estimates hold:

   (a) When \( c > 0, cf(n) = O(f(n)), cf(n) = \Omega(f(n)), \) and \( cf(n) = \Theta(f(n)) \).

   (b) If \( f(n) = O(g(n)) \) and \( g(n) = O(h(n)) \), then \( f(n) = O(h(n)) \).

   (c) If \( p(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_0 \) is a polynomial of degree \( k \), then \( p(n) = \Theta(n^k) \).

5. Order the following expressions in increasing asymptotic order:

   \[
   n, \log_2 n, (1,1)^n, n^{11}, n \log n, 1000n, n^2, 2^n
   \]

   Justify the order. (L’ Hospital’s rule helps here:
   If \( f(n)/g(n) \xrightarrow[n \to \infty]{} \infty/\infty \), then the limit of \( f(n)/g(n) \) is the same as the limit of the derivatives \( \lim_{n \to \infty} f'(n)/g'(n) \).