Algorithms

Assignment: Sorting

Name: .................................................................

Id: .................................................................

<table>
<thead>
<tr>
<th>Grade</th>
</tr>
</thead>
</table>

Good Luck!
1. Let $A = [A_1, A_2, \ldots, A_n]$ be an unsorted array of $n \geq 1$ positive integers.

Design an efficient comparison-based algorithm that finds the minimum positive difference between any two integers in the array. In other words, compute $m = \min_{1 \leq i, j \leq n} \{|A_i - A_j|\}$.

What is the worst-case number of comparisons made by your algorithm?

Justify the correctness and complexity of your algorithm.
2. Design an algorithm that sorts 5 distinct keys with at most 7 comparisons. Justify the correctness of your algorithm.
Let $A = [A_1, A_2, \ldots, A_n]$ be an array of $n \geq 1$ distinct positive integers. An inversion is a pair of indices $1 \leq i, j \leq n$ such that $i < j$ but $A_i > A_j$.

Example: In the array $[30, 80, 20, 40, 10]$, the pair $i = 1$ and $j = 3$ is an inversion because $A_1 = 30$ is greater than $A_3 = 20$. On the other hand, the pair $i = 1$ and $j = 2$ is not an inversion because $A_1 = 30$ is smaller than $A_2 = 80$. In this array there are 7 inversions and 3 non-inversions.

Design an efficient algorithm that counts the number of inversions in $A$.

What is the worst-case number of comparisons made by your algorithm?

Justify the correctness and complexity of your algorithm.
4. For \( n \geq 1 \), let \( A = [A_1, A_2, \ldots, A_n] \) and \( B = [B_1, B_2, \ldots, B_n] \) be arbitrary (not necessarily sorted) arrays containing \( 2n \) distinct positive integers. Array \( A \) dominates array \( B \) if it is possible to rearrange the arrays in a way such that \( A_i > B_i \) for all \( 1 \leq i \leq n \).

Design an efficient algorithm that decides if array \( A \) dominates array \( B \).

What is the worst-case number of comparisons made by your algorithm?

Justify the correctness and complexity of your algorithm.
5. An array $A$ of $n \geq 1$ distinct positive integers is a *hill array* if there exists an index $1 \leq i \leq n$, called the *hilltop*, such that: $A_1 < A_2 < \cdots < A_{i-1} < A_i > A_{i+1} > \cdots > A_{n-1} > A_n$. Note that when the hilltop is $i = 1$ then $A$ is sorted in a descending order and when the hilltop is $i = n$ then $A$ is sorted in an ascending order.

Design a comparison-based algorithm that sorts hill arrays in an ascending order with a linear number ($O(n)$) of comparisons.

Remark: The input is guaranteed to be a hill array but the hilltop index of the array is not known. Justify the correctness and complexity of your algorithm.
6. **Story:** There are \( n \) pancakes all of different sizes that are stacked on top of each other. It is allowed to slip a flipper under one of the pancakes and flip over the whole sack above the flipper. The goal is to arrange pancakes according to their size with the biggest at the bottom.

**Model:** Let \( A \) be an array of size \( n \geq 1 \) containing the numbers 1, \ldots, \( n \) in any order (\( A \) represents an arbitrary permutation). For any \( 2 \leq i \leq n \), the \( F_i \) operation (flip) is to reverse the prefix of size \( i \) of the array.

**Example I:** \( F_5([1, 2, 3, 4, 5, 6, 7, 8]) = [5, 4, 3, 2, 1, 6, 7, 8] \).

**Example II:** \( F_3([3, 2, 1, 4, 5, 6, 7, 8]) = [1, 2, 3, 4, 5, 6, 7, 8] \).

**Problem:** Sort the array in an ascending order with as few as possible flips.

**Remark:** In the comparison model, array operations were free and the optimization goal is to minimize the number of comparisons. In the pancake model, comparisons are free since the final location of each pancake is known. The optimization goal is to minimize the number of array (flip) operations.

Sort the array \([8, 7, 6, 5, 1, 2, 3, 4]\) with as few as possible flips. Specify the list of flips that would sort the array.

Sort the array \([8, 6, 4, 2, 1, 3, 5, 7]\) with as few as possible flips. Specify the list of flips that would sort the array.
Describe an efficient algorithm that sorts any permutation-array of size $n \geq 1$ with flips. How many flips, as a function of $n$, are made by your algorithm in the worst-case? Use words to describe the algorithm and be as accurate as you can with your worst-case complexity. Justify the correctness and complexity of your algorithm.