Algorithms

Assignment: Array Problems

Describe an efficient algorithm that finds, if it exists, an index \( 1 \leq i \leq n \) such that \( A[i] = i \). What is the complexity of your algorithm?

Justify the correctness of your algorithm and the correctness of your complexity claim.

2. Let \( A = [A_1 < A_2 < \cdots < A_n] \) be an array of \( n \) distinct integers sorted in an ascending order and let \( B = [B_1 > B_2 > \cdots > B_n] \) be an array of \( n \) distinct integers sorted in a descending order.

Describe an efficient algorithm that finds, if exists, an index \( 1 \leq i \leq n \) such that \( A_i = B_i \). What is the worst-case number of comparisons made by your algorithm?

Justify the correctness and complexity of your algorithm.

3. For \( n \geq 1 \), let \( A \) be an array of size \( n \) for which the first \( k \) entries contain positive integers and the rest of the array is all zeros. The value of \( n \) is known but the value of \( k \), which can be any number between 0 and \( n \), is unknown.

**Examples:**
- \([34, 13, 21, 0, 0, 0, 0, 0]\): \( k = 3 \) in this array of length 8.
- \([0, 0, 0, 0, 0, 0, 0, 0]\): \( k = 0 \) in this array of length 7.
- \([55, 8, 34, 13, 21, 89]\): \( k = 5 \) in this array of length 6.

Describe an efficient algorithm that determines the value of \( k \) which is the number of positive integers in \( A \). What is the complexity of your algorithm?

Justify the correctness of your algorithm and the correctness of your complexity claim.


Describe an efficient algorithm that finds the number of times \( k \) appears in the array. What is the complexity of your algorithm?

Justify the correctness of your algorithm and the correctness of your complexity claim.

5. For \( n \geq 2 \), let \( A = A[1] < A[2] < \cdots < A[n] \) be a sorted array with \( n \) distinct positive integers from the range \( 1, 2, \ldots, n + 1 \). That is, exactly one of the integers from this range is missing in the array \( A \).

**Examples:** The missing integer in the array \([1, 2, 3, 4, 5, 7, 8, 9]\) is 6, the missing integer in the array \([1, 2, 4, 5]\) is 3, the missing integer in the array \([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15]\) is 12, the missing integer in the array \([2, 3, 4, 5, 6, 7]\) is 1, and the missing integer in the array \([1, 2, 3, 4, 5, 6, 7, 8, 9]\) is 10.

Describe an efficient algorithm that finds the missing integer. The only questions about the integers in the arrays that your algorithm may ask are of the type “is \( A[i] = 1? \)” for some integer \( 1 \leq i \leq n \).

What is the worst-case complexity (number of questions asked) of the algorithm?

Justify the correctness of your algorithm and the correctness of your complexity claim.

6. Assume \( n \geq 1 \) is a power of 2. Let \( A = [A_1 < A_2 < \cdots < A_n] \) be a sorted array of \( n \) distinct positive integers. Let \( x \leq y \) be two positive integers.

Describe an efficient algorithm that determines if one of the integers \( x, x + 1, \ldots, y \) appears in the array. What is the worst-case number of comparisons made by your algorithm?

Justify the correctness and complexity of your algorithm.

7. Assume \( n \geq 1 \) is a power of 2. Let \( A = [A_1 < A_2 < \cdots < A_n] \) be a sorted array of \( n \) distinct positive integers. Let \( x \leq y \) be two positive integers.

Describe an efficient algorithm that determines if all the integers \( x, x + 1, \ldots, y \) appear in the array. What is the worst-case number of comparisons made by your algorithm?

Justify the correctness and complexity of your algorithm.
8. Let \( A = [A_1, A_2, \ldots, A_n] \) be an unsorted array of \( n \geq 1 \) positive integers. Design an efficient algorithm that finds the maximum difference between any two integers in the array. In other words, compute \( M = \max_{1 \leq i, j \leq n} \{A_i - A_j\} \).

What is the exact worst-case number of comparisons made by your algorithm? If you don’t know the exact complexity you may use the “\( O, \Omega, \Theta \)” notation.

Justify the correctness and complexity of your algorithm.

9. 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.

10. Let \( A = [A_1, A_2, \ldots, A_n] \) be an unsorted array of \( n \geq 4 \) distinct integers. Design an efficient algorithm that finds the first, second, and third largest integers in \( A \). What is the worst-case number of comparisons made by your algorithm?

Use words to describe the algorithm. Try to find the exact or almost exact number of comparisons. Ignore floors and ceilings.

Justify the correctness and complexity of your algorithm.

11. For an odd \( n \geq 1 \), let \( A = [A_1, A_2, \ldots, A_n] \) be an unsorted array of \( n \) positive integers that are not necessarily distinct. A majority is an integer that appears at least \((n + 1)/2\) times in the array.

Design a linear-time algorithm that finds a majority in \( A \) if exists.

Hint: Note that there can be at most one majority.

Justify the correctness and complexity of your algorithm.

12. Let \( A = [A_1 < A_2 < \cdots < A_n] \) be a sorted array of \( n \geq 1 \) distinct positive integers and let \( k \) be a positive integer. Design a linear time algorithm that finds, if exist, two indices \( 1 \leq i, j \leq n \) such that \( A_i + A_j = k \).

Justify the correctness and complexity of your algorithm.

13. 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.

14. 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.

15. 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.
16. Design an algorithm that finds the median of 5 distinct keys with at most 6 comparisons.
   Justify the correctness of your algorithm.

17. Design an algorithm that sorts 5 distinct keys with at most 7 comparisons.
   Justify the correctness of your algorithm.

18. **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.

   (a) 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.

   (b) 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.

   (c) 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.