Problems v Algorithms:

Explanation:

Problem_1: Calculate the floored square root of a number

Time Complexity: O(logn)
Converted the number to int while diving it by 2. I used a counter variable to keep track of how many times I am diving the number and returned the counter

Problem_2: Find the index by searching in a rotated sorted array

Time Complexity: O(logn)
Used Binary search as data structure since its the most efficient search algorithm.

Problem_3: Rearrange Array Elements so as to form two number such that their sum is maximum.

Time Complexity: O(n)
I decremented the numbers from 9 until 0 since I can assume that all numbers are between that and checked if they are in the list. I inserted the integers from highest to lowest consecutively into 2 strings and later converted them into an array and returned them.
All of this was done in a single traversal

Problem_4: Given an input array consisting on only 0, 1, and 2, sort the array in a single traversal.
Time complexity: O(n)

I ran through the entire list once and the logic for inserting is shown below,
Inserting 0 logic:
Always inserted 0 at the '0'th position of the result list and kept track of the most recent '0'th index so that I can insert 1 in the position after the most recent '0'th index
Inserting 1 logic:
Since I kept track of most recent '0'th index I inserted 1 after that
Inserting 2 login:
Always inserted 2 at the 'len(array)'th position.

Problem_5: Trie Implementation:
Used the same concepts thought in class and implemented them.

Problem_6: Return a tuple(min, max) out of list of unsorted integers.
Time complexity: O(n)

I ran through the list once keeping track of the max and min number but updating the max and min variable each time I see a number greater than max and less than min respectively.

Problem_7: Router Trie implementation

Took problem_5 as a reference and followed along.
Handled the trailing slash by splitting the path using a separator operator.