Prompt | Scratchpad | Solutions | Video Explanation

Difficulty: Green **Category:** Greedy Algorithms **Successful Submissions:** 34,640+

Minimum Waiting Time

You're given a non-empty array of positive integers representing the amounts of time that specific queries take to execute. Only one query can be executed at a time, but the queries can be executed in any order.

A query's **waiting time** is defined as the amount of time that it must wait before its execution starts. In other words, if a query is executed second, then its waiting time is the duration of the first query; if a query is executed third, then its waiting time is the sum of the durations of the first two queries.

Write a function that returns the minimum amount of total waiting time for all of the queries. For example, if you're given the queries of durations `[1, 4, 5]`, then the total waiting time if the queries were executed in the order of `[5, 1, 4]` would be `(0) + (5) + (5 + 1) = 11`. The first query of duration `5` would be executed immediately, so its waiting time would be `0`, the second query of duration `1` would have to wait `5` seconds (the duration of the first query) to be executed, and the last query would have to wait the duration of the first two queries before being executed.

Note: you're allowed to mutate the input array.

Sample Input

```
queries = [3, 2, 1, 2, 6]
```

Sample Output

```
17
```

Hints

Hint 1

Even though you don't need to return the actual order in which the queries will be executed to minimize the total waiting time, it's important to determine what this order should be. Start by doing so.

Hint 2

Can you solve this problem with constant space? What advantage does being able to mutate the input array provide?

Hint 3

Sort the input array in place, and execute the shortest queries in their sorted order. This should allow you to calculate the minimum waiting time.

Hint 4

Create a variable to store the total waiting time, and iterate through the sorted input array. At each iteration, multiply the number of queries left by the duration of the current query, and add that to the total waiting time.

Optimal Space & Time Complexity

O(nlogn) time | O(1) space - where n is the number of queries

Your Solutions | Run Code

Solution 1 | Solution 2 | Solution 3

```python
'''O(n log(n)) Time and O(1) Space'''
def minimumWaitingTime(queries):
    current_wt = 0
    total_waiting_time = 0
    queries.sort()

    for idx in range(1, len(queries)):
        current_wt += queries[idx-1]
        total_waiting_time += current_wt

    return total_waiting_time

# Kunal Wadhwa
# https://github.com/kunal5042
# https://leetcode.com/kunal5042/
# https://www.hackerrank.com/kunalwadhwa_cs
# https://www.linkedin.com/in/kunal5042/

```

Custom Output | Raw Output | Submit Code

Yay, your code passed all the test cases!

15 / 15 test cases passed.

- Test Case 1 passed!
- Test Case 2 passed!
- Test Case 3 passed!
- Test Case 4 passed!
- Test Case 5 passed!
- Test Case 6 passed!
- Test Case 7 passed!
- Test Case 8 passed!
- Test Case 9 passed!
- Test Case 10 passed!
- Test Case 11 passed!
- Test Case 12 passed!
- Test Case 13 passed!
- Test Case 14 passed!
- Test Case 15 passed!