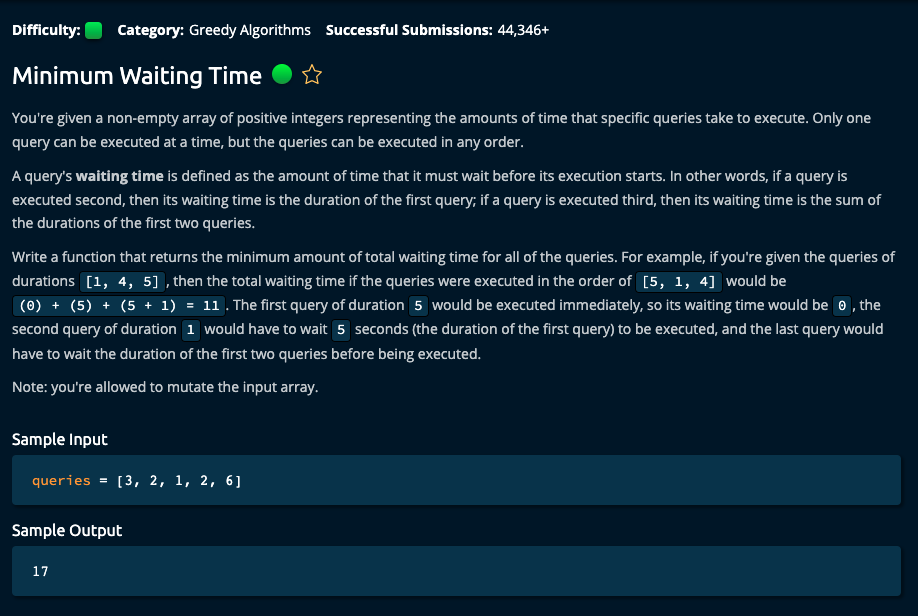
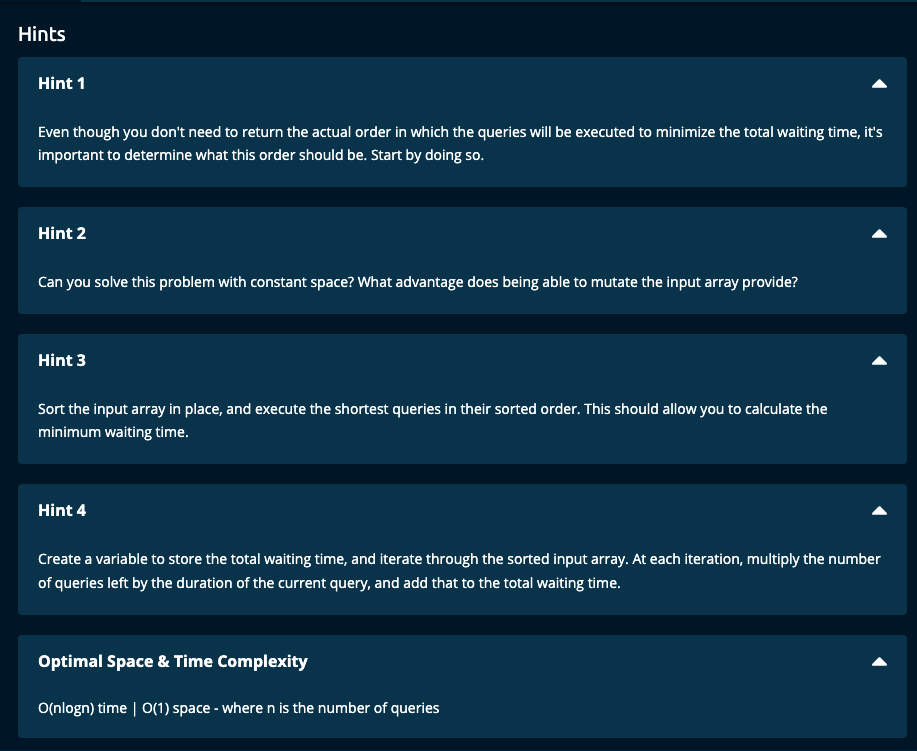
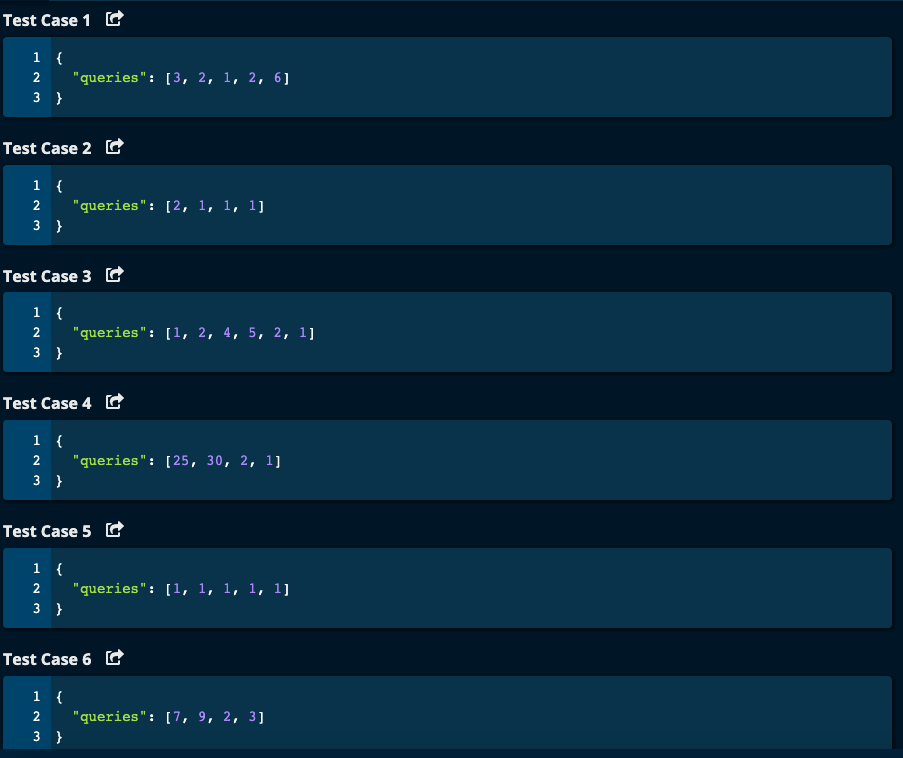
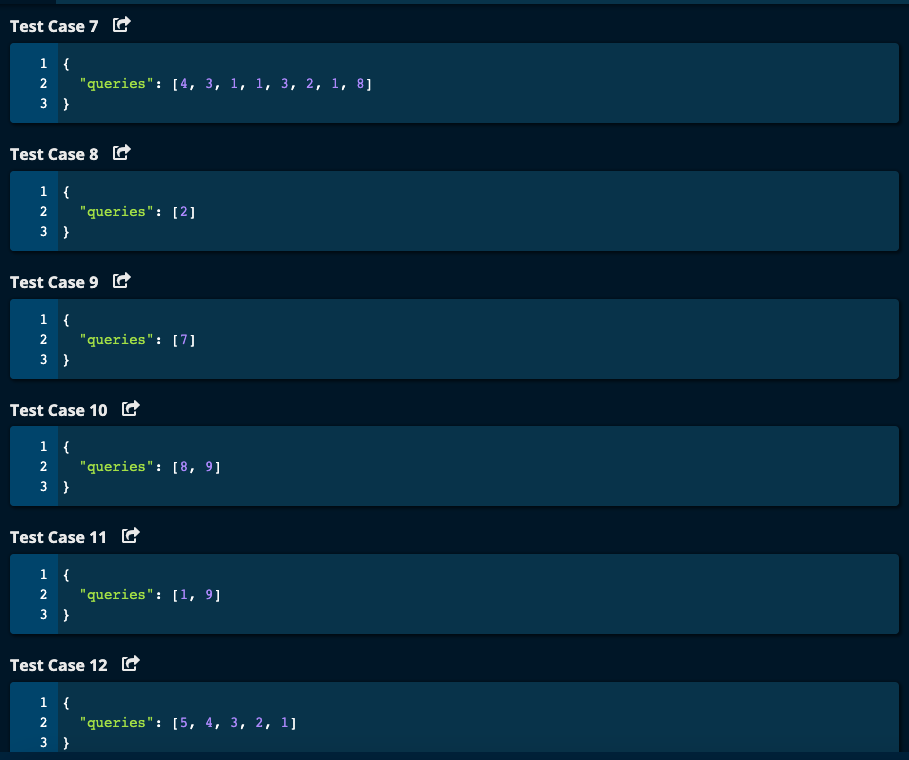
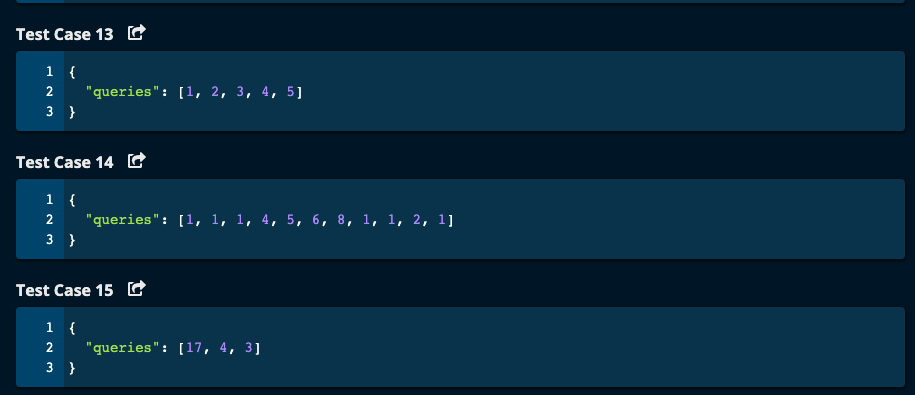
Minimum Waiting Time (Easy)











My Solution:

My Solution -- O(nlog(n)) time | O(1) space

def minimumWaitingTime(queries):

if len(queries) == 0:

return 0

if len(queries) == 1: # if there is only one query, it need not wait.

return 0

sortedQueries = sorted(queries)

print(sortedQueries)

totalWait = 0

leftSum = 0

for i in range(len(sortedQueries)):

if i == 0:

leftSum = 0

else:

leftSum += sortedQueries[i - 1]

totalWait += leftSum

return totalWait

JJ Notes:

1. If length of queries is 0, then return 0. If length of queries is 1, then return 0 since the first query need not wait and can execute right away.
2. Sort queries and call it sortedQueries. This determines the execution order so that the total waiting time is minimized. For example [6, 1] has waiting time of 0 + 6 = 6, whereas [1, 6] has waiting time 0 + 1 = 1. So executing the shorter jobs first saves waiting time.
3. Initialize totalWait time to 0 and leftSum to 0.
4. Iterate through sortedQueries array. If the index is 0, then leftSum = 0 as there is no element to the left of index 0.

Otherwise, add sortedQueries at the previous index to existing leftSum. The totalWait time is the leftSum added to the existing waitTime.

1. Return totalWait.

Algoexpert Solution:

1. First sort the queries array.
2. Initialize totalWaitingTime to 0.
3. Iterate through the array. When the first query is executing, all the remaining queries have to wait for the first query to finish executing. So calculate remaining queries ie. queriesLeft to be len(queries) minus (current index + 1) since index is 0 based.

For example, we have 5 queries and their execution times are [2, 3, 4, 5, 8].

When the first query is executing the remaining 4 queries have to wait for the first job to finish which 2. Total waiting time = duration of the first job \* remaining number of jobs, which is 2 \* 4 = 8.

1. Return totalWaitingTime.

# Algoexpert Solution -- O(nlog(n)) time | O(1) space

def minimumWaitingTime(queries):

queries.sort() # sort the array in place

totalWaitingTime = 0

for idx, duration in enumerate(queries):

queriesLeft = len(queries) - (idx + 1) # since idx begins at 0

totalWaitingTime += duration \* queriesLeft

return totalWaitingTime