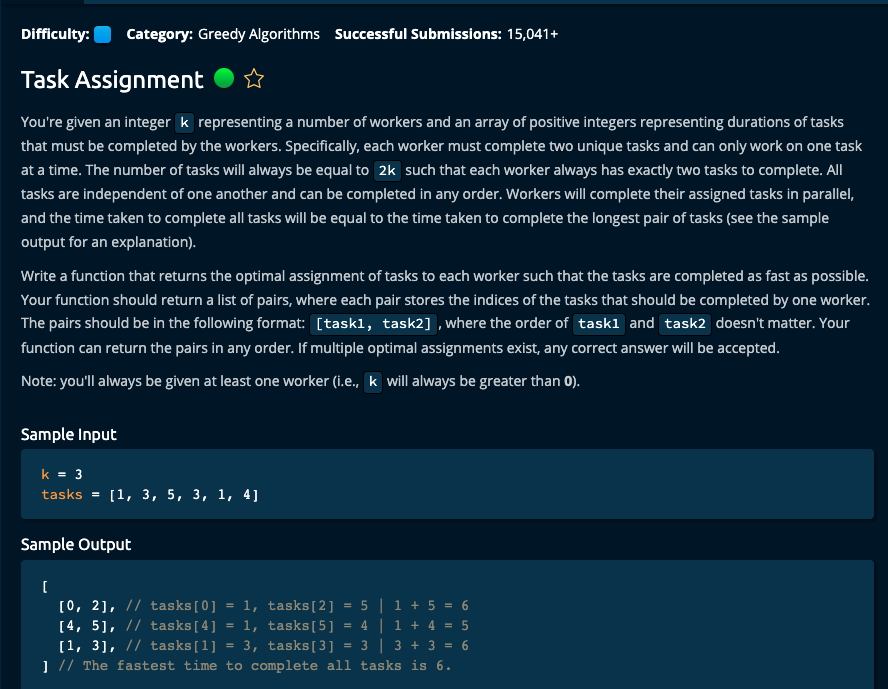
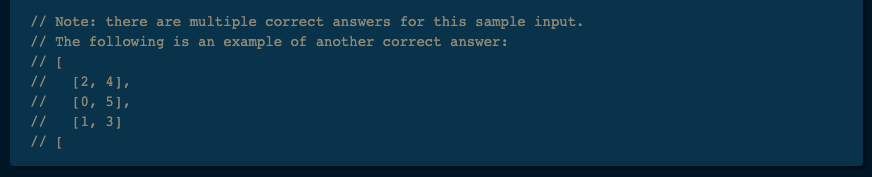
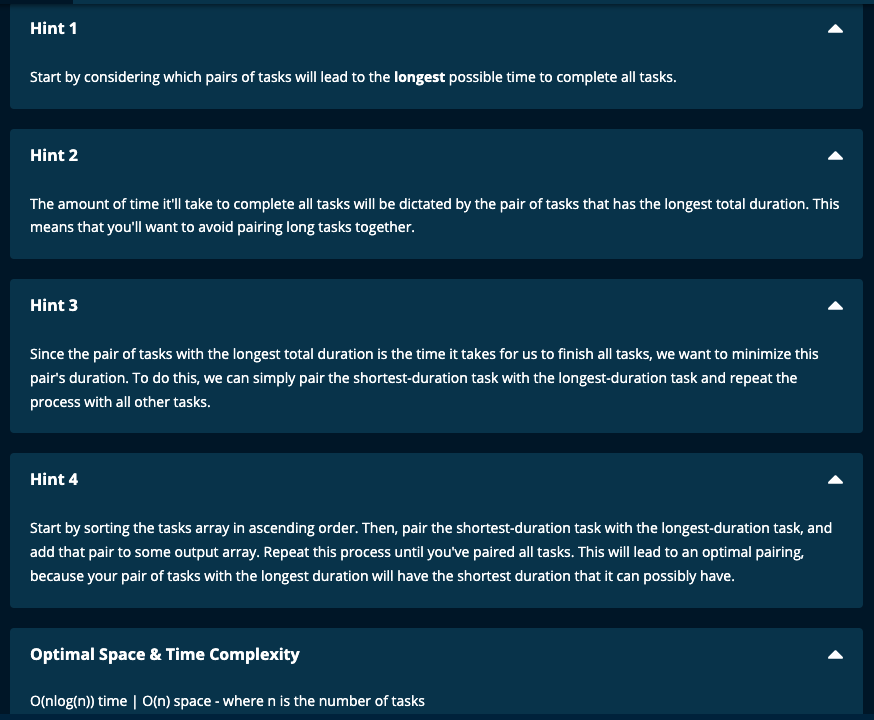
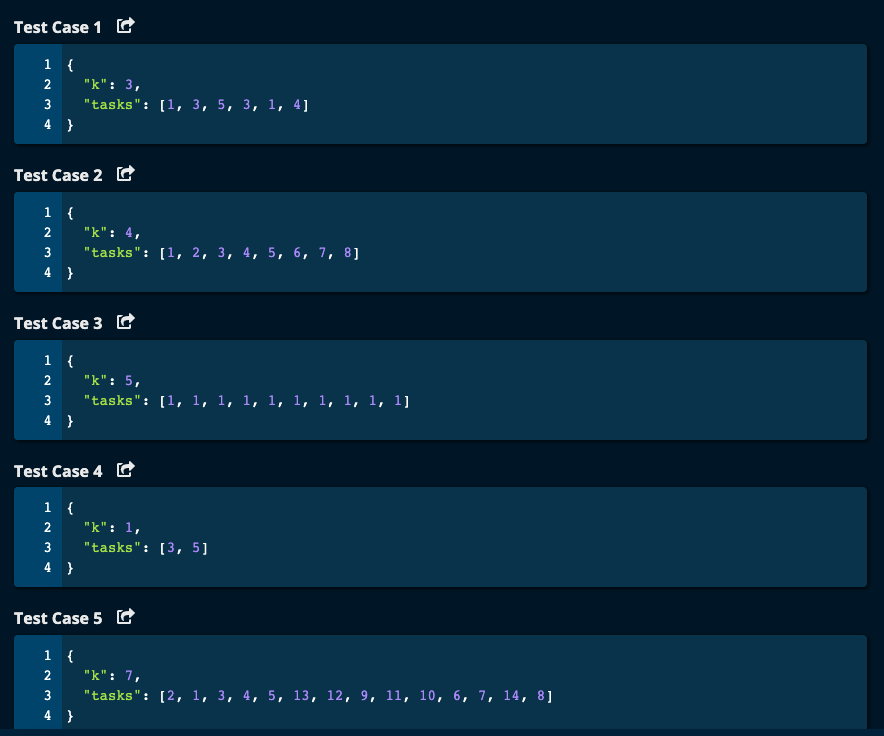
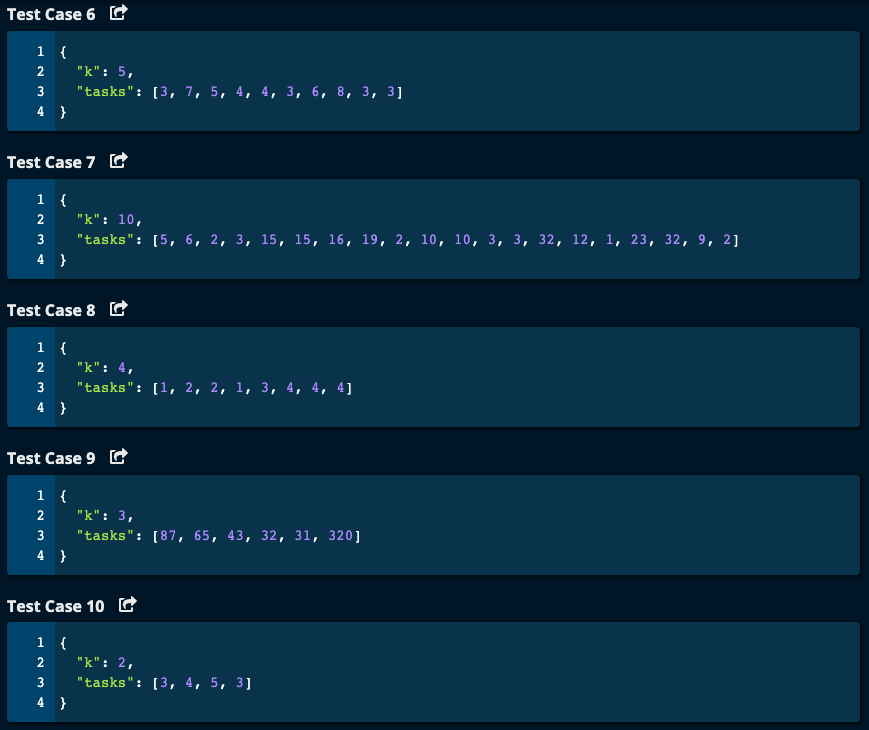
Task Assignment. (Medium)

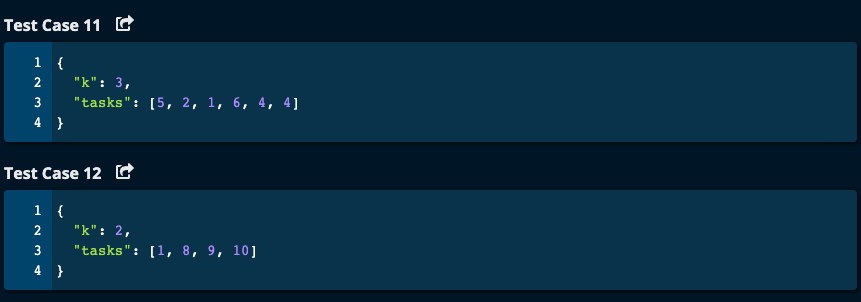












My Solutions:

Soltuion 1:

# My Solution -- O(nlog(n)) Time | O(n) Space

def taskAssignment(k, tasks):

mydict = {} # key is index and value is task duration

for idx, task in enumerate(tasks):

mydict[idx] = task

# Sort the dictionary based on values

#sortedTasks = sorted(mydict, key = lambda x : mydict[x])

sortedTasks = sorted(mydict, key = mydict.get)

i = 0

j = len(tasks) - 1

result = []

while i < j:

result.append([sortedTasks[i], sortedTasks[j]])

i += 1

j -= 1

return result

Solution2:

# My Solution -- O(nlog(n)) Time | O(n) Space

def taskAssignment(k, tasks):

mydict = {} # key is index and value is task duration

for idx, task in enumerate(tasks):

mydict[idx] = task

# Sort the dictionary based on values

sortedTasks = sorted(mydict, key = lambda x : mydict[x])

#sortedTasks = sorted(mydict, key = mydict.get)

result = []

for i in range(k):

result.append([sortedTasks[i], sortedTasks[2\*k - 1 - i]])

return result

Solution 3:

# My Solution -- O(nlog(n)) Time | O(n) Space

def taskAssignment(k, tasks):

mydict = {} # key is index and value is task duration

for idx, task in enumerate(tasks):

mydict[idx] = task

# Sort the dictionary based on values

sortedTasks = sorted(mydict, key = lambda x : mydict[x])

#sortedTasks = sorted(mydict, key = mydict.get)

result = [[sortedTasks[i], sortedTasks[2\*k - 1 - i]] for i in range(k)]

return result

JJ Notes:

1. Create a dictionary with key as the index and value as the task duration called mydict.
2. Sort the dictionary based on values,ie. task duration.
3. Create a result list with pairs of tasks such that the first task is paired with the last and indexes of the first and second tasks move towards the center.

Algoexpert Solution :

NOT Good! Does not sort the dictionary by value!

1. To hold the results, create a list called pairedTasks
2. Create a dictionary with key = task duration and value = a list of indicies called taskDurationToIndices. This is done to look up the indexes of the tasks .
3. Sort the tasks into an array called sortedTasks.
4. Iterate through the first half of the array (array is of size 2\*k since each of the k workers get 2 tasks). The task1Duration is the first value in the sortedTasks and task2Duration is the last value in the sortedTasks. We get the corresponding indexes of these two tasks from looking up the value in the dictionary and then popping the last value in the list.

Then for the next pair, the first task will move one step towards the center from the beginning and the second task will move towards the center from the end.

First task index goes from 0, 1, 2….k – 1.

Second task index goes from 2k – 1, 2k – 2, …, k.

Create a list with the index of the first and second task for each pair.

1. Return the paired tasks list.

