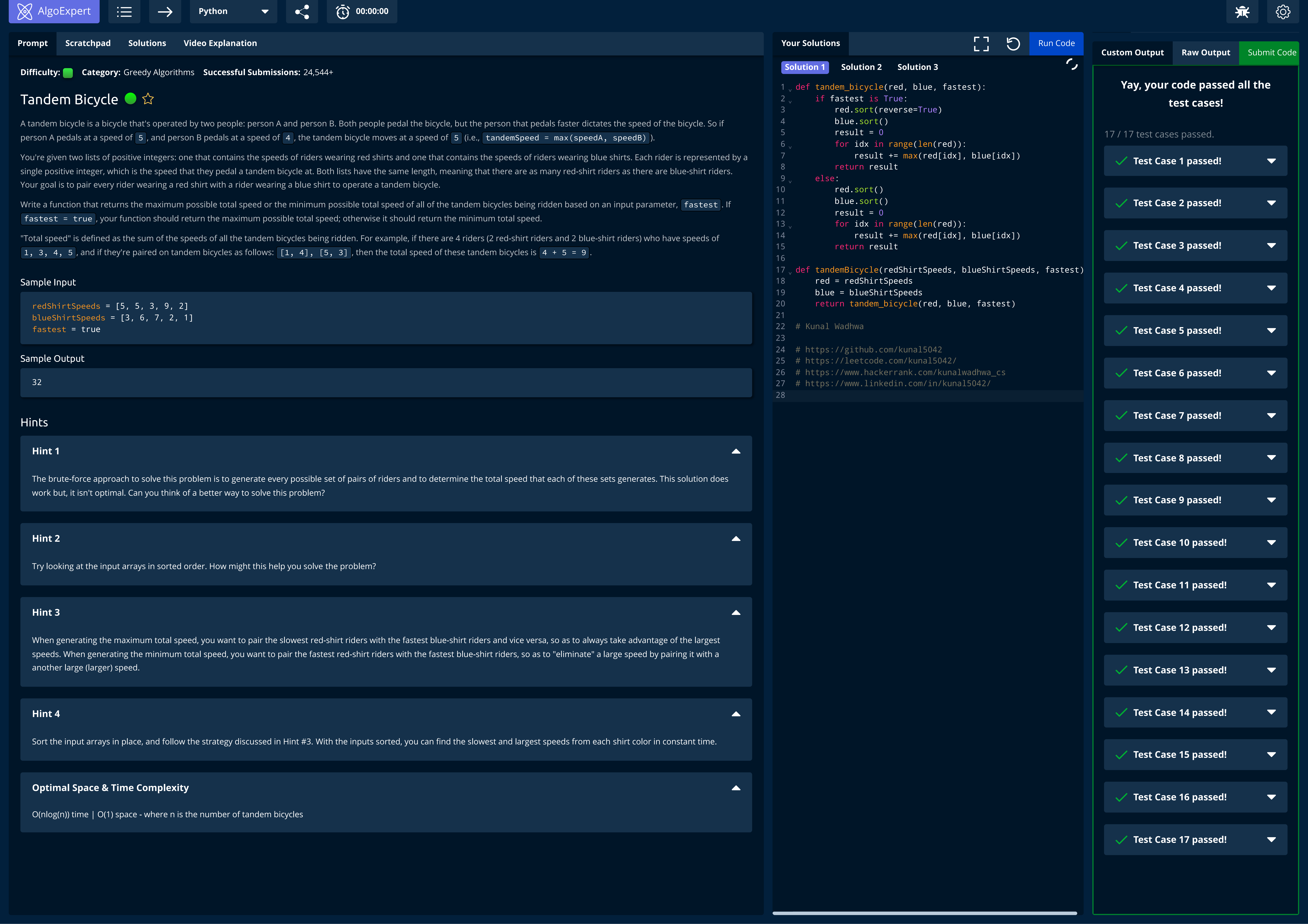

Difficulty: 🟩 **Category:** Greedy Algorithms **Successful Submissions:** 24,544+

Tandem Bicycle

A tandem bicycle is a bicycle that's operated by two people: person A and person B. Both people pedal the bicycle, but the person that pedals faster dictates the speed of the bicycle. So if person A pedals at a speed of `5`, and person B pedals at a speed of `4`, the tandem bicycle moves at a speed of `5` (i.e., `tandemSpeed = max(speedA, speedB)`).

You're given two lists of positive integers: one that contains the speeds of riders wearing red shirts and one that contains the speeds of riders wearing blue shirts. Each rider is represented by a single positive integer, which is the speed that they pedal a tandem bicycle at. Both lists have the same length, meaning that there are as many red-shirt riders as there are blue-shirt riders. Your goal is to pair every rider wearing a red shirt with a rider wearing a blue shirt to operate a tandem bicycle.

Write a function that returns the maximum possible total speed or the minimum possible total speed of all of the tandem bicycles being ridden based on an input parameter, `fastest`. If `fastest = true`, your function should return the maximum possible total speed; otherwise it should return the minimum total speed.

"Total speed" is defined as the sum of the speeds of all the tandem bicycles being ridden. For example, if there are 4 riders (2 red-shirt riders and 2 blue-shirt riders) who have speeds of `1, 3, 4, 5`, and if they're paired on tandem bicycles as follows: `[1, 4], [5, 3]`, then the total speed of these tandem bicycles is `4 + 5 = 9`.

Sample Input

```
redShirtSpeeds = [5, 5, 3, 9, 2]
blueShirtSpeeds = [3, 6, 7, 2, 1]
fastest = true
```

Sample Output

```
32
```

Hints

Hint 1

The brute-force approach to solve this problem is to generate every possible set of pairs of riders and to determine the total speed that each of these sets generates. This solution does work but, it isn't optimal. Can you think of a better way to solve this problem?

Hint 2

Try looking at the input arrays in sorted order. How might this help you solve the problem?

Hint 3

When generating the maximum total speed, you want to pair the slowest red-shirt riders with the fastest blue-shirt riders and vice versa, so as to always take advantage of the largest speeds. When generating the minimum total speed, you want to pair the fastest red-shirt riders with the fastest blue-shirt riders, so as to "eliminate" a large speed by pairing it with a another large (larger) speed.

Hint 4

Sort the input arrays in place, and follow the strategy discussed in Hint #3. With the inputs sorted, you can find the slowest and largest speeds from each shirt color in constant time.

Optimal Space & Time Complexity

O(nlog(n)) time | O(1) space - where n is the number of tandem bicycles

```python
def tandem_bicycle(red, blue, fastest):
    if fastest is True:
        red.sort(reverse=True)
        blue.sort()
        result = 0
        for idx in range(len(red)):
            result += max(red[idx], blue[idx])
        return result
    else:
        red.sort()
        blue.sort()
        result = 0
        for idx in range(len(red)):
            result += max(red[idx], blue[idx])
        return result

def tandemBicycle(redShirtSpeeds, blueShirtSpeeds, fastest)
    red = redShirtSpeeds
    blue = blueShirtSpeeds
    return tandem_bicycle(red, blue, fastest)

# Kunal Wadhwa

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

Yay, your code passed all the test cases!

17 / 17 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!
✓ Test Case 16 passed!
✓ Test Case 17 passed!