

## 5524. Maximum Profit of Operating a Centennial Wheel

missions (/contest/weekly-contest-208/problems/maximum-profit-of-operating-a-centennial-wheel/submissions/)

o Contest (/contest/weekly-contest-208/)

You are the operator of a Centennial Wheel that has **four gondolas**, and each gondola has room for **up to four people**. You have the ability to rotate the gondolas **counterclockwise**, which costs you `runningCost` dollars.

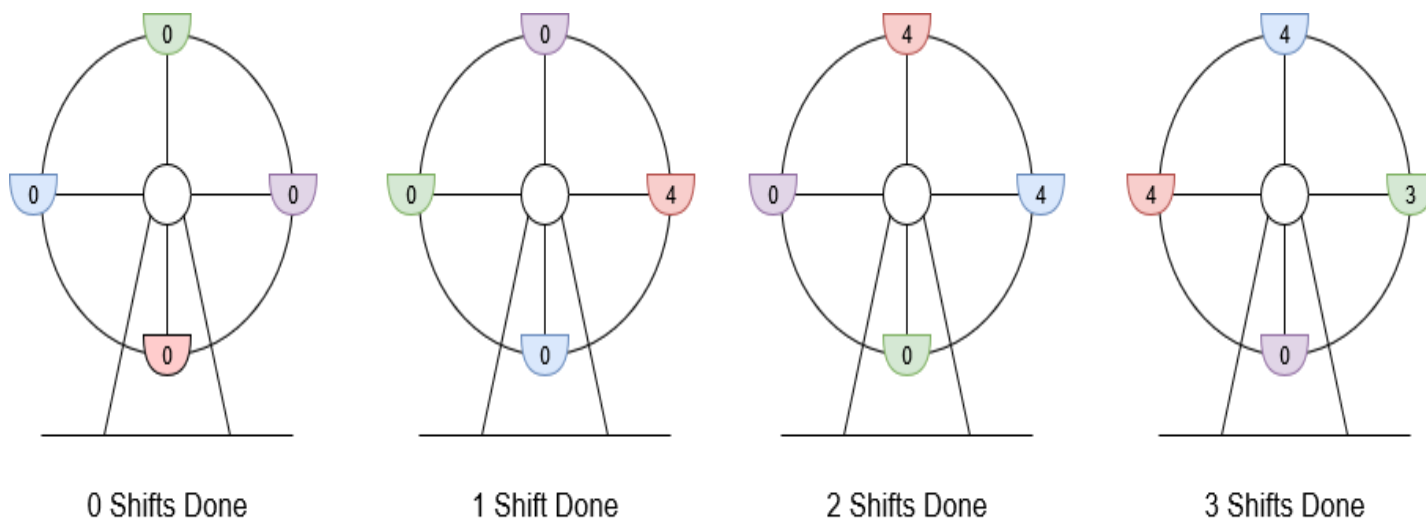
You are given an array `customers` of length `n` where `customers[i]` is the number of new customers arriving just before the  $i^{\text{th}}$  rotation (0-indexed). This means you **must rotate** the wheel  $i$  times before `customers[i]` arrive. Each customer pays `boardingCost` dollars when they board on the gondola closest to the ground and will exit once that gondola reaches the ground again.

You can stop the wheel at any time, including **before serving all customers**. If you decide to stop serving customers, **all subsequent rotations are free** in order to get all the customers down safely. Note that if there are currently more than four customers waiting at the wheel, only four will board the gondola, and the rest will wait **for the next rotation**.

Return the *minimum number of rotations you need to perform to maximize your profit*. If there is **no scenario** where the profit is positive, return `-1`.

User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Medium

### Example 1:



**Input:** customers = [8,3], boardingCost = 5, runningCost = 6

**Output:** 3

**Explanation:** The numbers written on the gondolas are the number of people currently there.  
1. 8 customers arrive, 4 board and 4 wait for the next gondola, the wheel rotates. Current profit is  $4 * \$5 - 4 * \$4 = \$0$ .  
2. 3 customers arrive, the 4 waiting board the wheel and the other 3 wait, the wheel rotates. Current profit is  $7 * \$5 - 4 * \$4 = \$11$ .  
3. The final 3 customers board the gondola, the wheel rotates. Current profit is  $11 * \$5 - 7 * \$4 = \$37$ .  
The highest profit was \$37 after rotating the wheel 3 times.

#### Example 2:

**Input:** customers = [10,9,6], boardingCost = 6, runningCost = 4

**Output:** 7

**Explanation:**

1. 10 customers arrive, 4 board and 6 wait for the next gondola, the wheel rotates. Current profit is  $4 * \$6 - 4 * \$4 = \$8$ .  
2. 9 customers arrive, 4 board and 11 wait (2 originally waiting, 9 newly waiting), the wheel rotates. Current profit is  $8 * \$6 - 8 * \$4 = \$40$ .  
3. The final 6 customers arrive, 4 board and 13 wait, the wheel rotates. Current profit is  $12 * \$6 - 12 * \$4 = \$72$ .  
4. 4 board and 9 wait, the wheel rotates. Current profit is  $16 * \$6 - 16 * \$4 = \$80$ .  
5. 4 board and 5 wait, the wheel rotates. Current profit is  $20 * \$6 - 20 * \$4 = \$100$ .  
6. 4 board and 1 waits, the wheel rotates. Current profit is  $24 * \$6 - 24 * \$4 = \$120$ .  
7. 1 boards, the wheel rotates. Current profit is  $25 * \$6 - 25 * \$4 = \$122$ .  
The highest profit was \$122 after rotating the wheel 7 times.

#### Example 3:

**Input:** customers = [3,4,0,5,1], boardingCost = 1, runningCost = 92

**Output:** -1

**Explanation:**

1. 3 customers arrive, 3 board and 0 wait, the wheel rotates. Current profit is  $3 * \$1 - 3 * \$92 = -273$ .  
2. 4 customers arrive, 4 board and 0 wait, the wheel rotates. Current profit is  $7 * \$1 - 4 * \$92 = -365$ .  
3. 0 customers arrive, 0 board and 0 wait, the wheel rotates. Current profit is  $7 * \$1 - 5 * \$92 = -459$ .  
4. 5 customers arrive, 4 board and 1 waits, the wheel rotates. Current profit is  $12 * \$1 - 4 * \$92 = -368$ .  
5. 1 customer arrives, 2 board and 0 wait, the wheel rotates. Current profit is  $13 * \$1 - 6 * \$92 = -549$ .  
The profit was never positive, so return -1.

#### Example 4:

**Input:** customers = [10,10,6,4,7], boardingCost = 3, runningCost = 8

**Output:** 9

**Explanation:**

1. 10 customers arrive, 4 board and 6 wait, the wheel rotates. Current profit is  $4 * \$3 - 6 * \$8 = \$24$ .
  2. 10 customers arrive, 4 board and 12 wait, the wheel rotates. Current profit is  $8 * \$3 - 12 * \$8 = \$24$ .
  3. 6 customers arrive, 4 board and 14 wait, the wheel rotates. Current profit is  $12 * \$3 - 14 * \$8 = \$24$ .
  4. 4 customers arrive, 4 board and 14 wait, the wheel rotates. Current profit is  $16 * \$3 - 14 * \$8 = \$24$ .
  5. 7 customers arrive, 4 board and 17 wait, the wheel rotates. Current profit is  $20 * \$3 - 17 * \$8 = \$24$ .
  6. 4 board and 13 wait, the wheel rotates. Current profit is  $24 * \$3 - 13 * \$8 = \$24$ .
  7. 4 board and 9 wait, the wheel rotates. Current profit is  $28 * \$3 - 9 * \$8 = \$28$ .
  8. 4 board and 5 wait, the wheel rotates. Current profit is  $32 * \$3 - 5 * \$8 = \$32$ .
  9. 4 board and 1 waits, the wheel rotates. Current profit is  $36 * \$3 - 1 * \$8 = \$36$ .
  10. 1 board and 0 wait, the wheel rotates. Current profit is  $37 * \$3 - 0 * \$8 = \$31$ .
- The highest profit was \$36 after rotating the wheel 9 times.

### Constraints:

- $n == \text{customers.length}$
- $1 \leq n \leq 10^5$
- $0 \leq \text{customers}[i] \leq 50$
- $1 \leq \text{boardingCost}, \text{runningCost} \leq 100$

JavaScript



```

1  /**
2   * @param {number[]} customers
3   * @param {number} boardingCost
4   * @param {number} runningCost
5   * @return {number}
6   */
7  const minOperationsMaxProfit = (customers, boardingCost, runningCost) => {
8      let rotate = 0;
9      let wait = 0;
10     let board = 0;
11     let record = [];
12     let n = customers.length;
13     while (true) {
14         if (rotate > n && record.length == 0) return -1;
15         let l = record[record.length - 1];
16         let sl = record[record.length - 2];
17         if (l <= sl) break;
18         rotate++;
19         if (customers.length != 0) {
20             wait += customers[0];
21             customers.shift();
22         }
23         if (wait > 4) {
24             board += 4;

```

```
25         wait -= 4;
26     } else {
27         let tmp = wait;
28         wait -= tmp;
29         board += tmp;
30     }
31     let profit = board * boardingCost - rotate * runningCost;
32     if(profit > 0) record.push(profit);
33 }
34 return rotate - 1;
35 };
```

☐ Custom Testcase[Use Example Testcases](#)[Run](#)[Submit](#)

Submission Result: **Accepted** (/submissions/detail/401226990/) ?

[More Details > \(/submissions/detail/401226990/\)](#)

Share your acceptance!

◀ 1

Copyright © 2020 LeetCode

[Help Center \(/support/\)](#) | [Students \(/students\)](#) | [Terms \(/terms/\)](#) | [Privacy Policy \(/privacy/\)](#)

 [United States \(/region/\)](#)