Problem 1599: Maximum Profit of Operating a Centennial Wheel

Problem Information

Difficulty: Medium
Acceptance Rate: 44.33%
Paid Only: No
Tags: Array, Simulation

Problem Description

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 `ith` rotation (0-indexed). This means you **must rotate the wheel**`i`**times before the**`customers[i]`**customers arrive**. **You cannot make customers wait if there is room in the gondola**. 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`.

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 - 1 * $6 = $14. 2. 3 customers arrive, the 4 waiting board the wheel and the other 3 wait, the wheel rotates. Current profit is 8 * $5 - 2 * $6 = $28. 3. The final 3 customers board the gondola, the wheel rotates. Current profit is 11 * $5 - 3 * $6 = $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 - 1 * $4 = $20. 2. 9 customers arrive, 4 board and 11 wait (2 originally waiting, 9 newly waiting), the wheel rotates. Current profit is 8 * $6 - 2 * $4 = $40. 3. The final 6 customers arrive, 4 board and 13 wait, the wheel rotates. Current profit is 12 * $6 - 3 * $4 = $60. 4. 4 board and 9 wait, the wheel rotates. Current profit is 16 * $6 - 4 * $4 = $80. 5. 4 board and 5 wait, the wheel rotates. Current profit is 20 * $6 - 5 * $4 = $100. 6. 4 board and 1 waits, the wheel rotates. Current profit is 24 * $6 - 6 * $4 = $120. 7. 1 boards, the wheel rotates. Current profit is 25 * $6 - 7 * $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 - 1 * $92 = -$89. 2. 4 customers arrive, 4 board and 0 wait, the wheel rotates. Current profit is 7 * $1 - 2 * $92 = -$177. 3. 0 customers arrive, 0 board and 0 wait, the wheel rotates. Current profit is 7 * $1 - 3 * $92 = -$269. 4. 5 customers arrive, 4 board and 1 waits, the wheel rotates. Current profit is 11 * $1 - 4 * $92 = -$357. 5. 1 customer arrives, 2 board and 0 wait, the wheel rotates. Current profit is 13 * $1 - 5 * $92 = -$447. The profit was never positive, so return -1.

Constraints:

* `n == customers.length` * `1 <= n <= 105` * `0 <= customers[i] <= 50` * `1 <= boardingCost, runningCost <= 100`

Code Snippets

C++:

```
class Solution {
public:
int minOperationsMaxProfit(vector<int>& customers, int boardingCost, int
runningCost) {

}
};
```

Java:

```
class Solution {
public int minOperationsMaxProfit(int[] customers, int boardingCost, int
runningCost) {

}
}
```

Python3:

```
class Solution:
def minOperationsMaxProfit(self, customers: List[int], boardingCost: int,
runningCost: int) -> int:
```