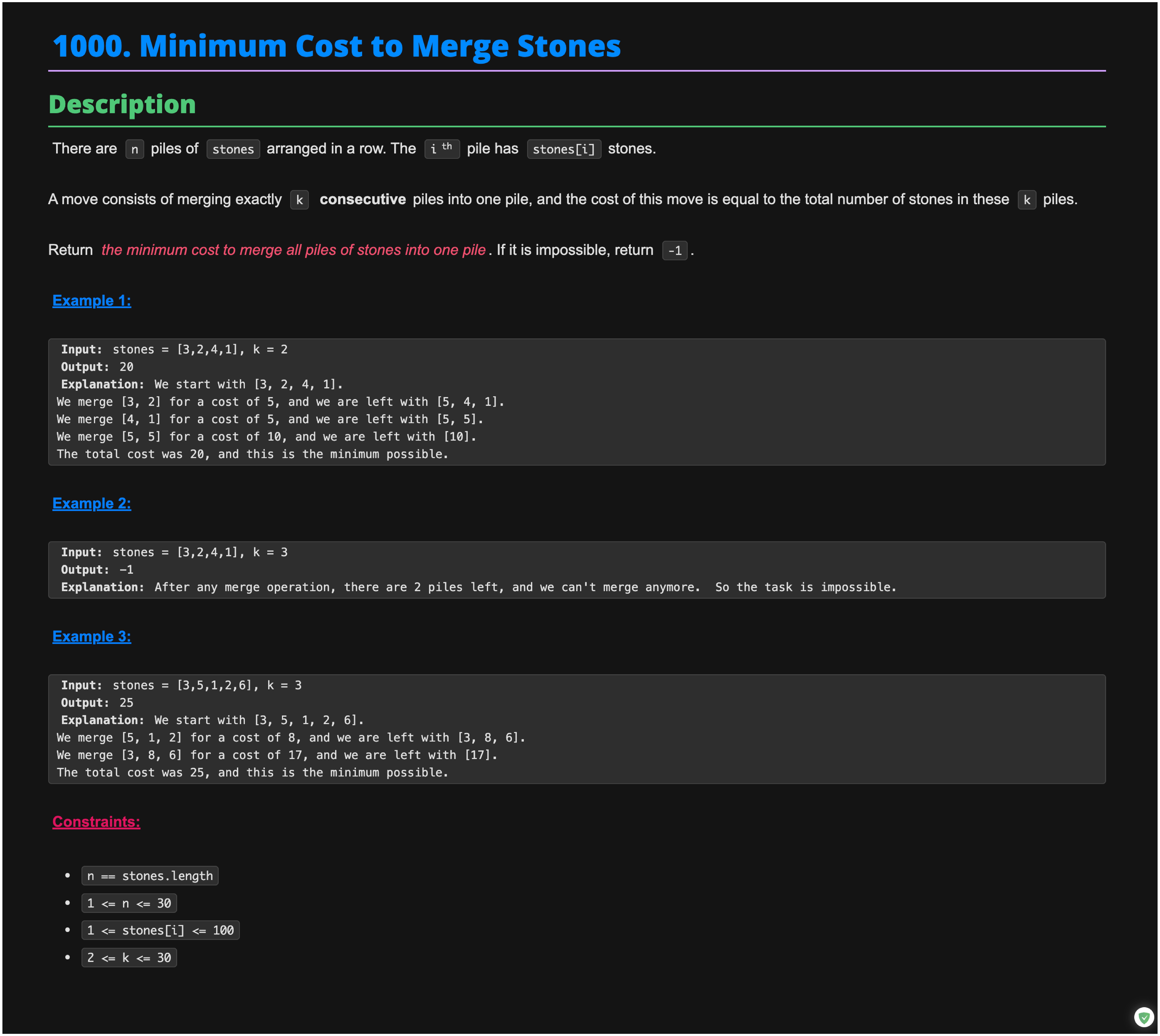

1000. Minimum Cost to Merge Stones

Description

There are `n` piles of `stones` arranged in a row. The `i`$^{\text{th}}$ pile has `stones[i]` stones.

A move consists of merging exactly `k` **consecutive** piles into one pile, and the cost of this move is equal to the total number of stones in these `k` piles.

Return *the minimum cost to merge all piles of stones into one pile*. If it is impossible, return `-1`.

Example 1:

```
 Input: stones = [3,2,4,1], k = 2
 Output: 20
 Explanation: We start with [3, 2, 4, 1].
We merge [3, 2] for a cost of 5, and we are left with [5, 4, 1].
We merge [4, 1] for a cost of 5, and we are left with [5, 5].
We merge [5, 5] for a cost of 10, and we are left with [10].
The total cost was 20, and this is the minimum possible.
```

Example 2:

```
 Input: stones = [3,2,4,1], k = 3
 Output: -1
 Explanation: After any merge operation, there are 2 piles left, and we can't merge anymore.  So the task is impossible.
```

Example 3:

```
 Input: stones = [3,5,1,2,6], k = 3
 Output: 25
 Explanation: We start with [3, 5, 1, 2, 6].
We merge [5, 1, 2] for a cost of 8, and we are left with [3, 8, 6].
We merge [3, 8, 6] for a cost of 17, and we are left with [17].
The total cost was 25, and this is the minimum possible.
```

Constraints:

- `n == stones.length`
- `1 <= n <= 30`
- `1 <= stones[i] <= 100`
- `2 <= k <= 30`