

# 2178. Maximum Split of Positive Even Integers

[Leetcode Link](#)

You are given an integer `finalSum`. Split it into a sum of a **maximum** number of **unique** positive even integers.

- For example, given `finalSum = 12`, the following splits are **valid** (unique positive even integers summing up to `finalSum`): `(12)`, `(2 + 10)`, `(2 + 4 + 6)`, and `(4 + 8)`. Among them, `(2 + 4 + 6)` contains the maximum number of integers. Note that `finalSum` cannot be split into `(2 + 2 + 4 + 4)` as all the numbers should be unique.

Return *a list of integers that represent a valid split containing a **maximum number** of integers*. If no valid split exists for `finalSum`, return *an **empty** list*. You may return the integers in **any** order.

**Example 1:**

**Input:** `finalSum = 12`

**Output:** `[2,4,6]`

**Explanation:** The following are valid splits: `(12)`, `(2 + 10)`, `(2 + 4 + 6)`, and `(4 + 8)`. `(2 + 4 + 6)` has the maximum number of integers, which is 3. Thus, we return `[2,4,6]`.

Note that `[2,6,4]`, `[6,2,4]`, etc. are also accepted.

**Example 2:**

**Input:** `finalSum = 7`

**Output:** `[]`

**Explanation:** There are no valid splits for the given `finalSum`. Thus, we return an empty array.

**Example 3:**

**Input:** `finalSum = 28`

**Output:** `[6,8,2,12]`

**Explanation:** The following are valid splits: `(2 + 26)`, `(6 + 8 + 2 + 12)`, and `(4 + 24)`. `(6 + 8 + 2 + 12)` has the maximum number of integers, which is 4. Thus, we return `[6,8,2,12]`.

Note that `[10,2,4,12]`, `[6,2,4,16]`, etc. are also accepted.

**Constraints:**

- $1 \leq \text{finalSum} \leq 10^{10}$

## Solution

### Full Solution

First, let's think of how to determine if a sum  $S$  can have a valid split that contains exactly  $K$  integers.

The first case we should consider is whether or not a sum  $S$  can have a valid split of any size. Since our split includes only even integers,  $S$  will only have a split if it's even and we will return an empty list if  $S$  is odd.

One observation we can make is that if some sum  $S$  does have a valid split of  $K$  integers, then a sum  $T$  will also have a valid split of  $K$  integers if  $T$  is even and  $T \geq S$ . Why is this true? Let's denote  $D$  as the difference between  $T$  and  $S$ . From the split of  $K$  integers from the sum  $S$ , incrementing the largest integer in the split by  $D$  results in a valid split of  $K$  integers with a total sum of  $T$ . It can be observed that increasing the greatest integer will always keep the entire list distinct.

### Example

For this example, let's use the split  $12 = 2 + 4 + 6$  with  $S = 12$  and  $K = 3$ . How will we construct a split of size  $K$  with sum  $T = 16$ ?

First, we'll find the difference  $D = T - S = 4$ . Then, we'll add  $D$  to the greatest integer in the split with sum  $S$ , which is 6.

Thus, we obtain the split  $16 = 2 + 4 + 10$  with sum  $T$  and size  $K$ .

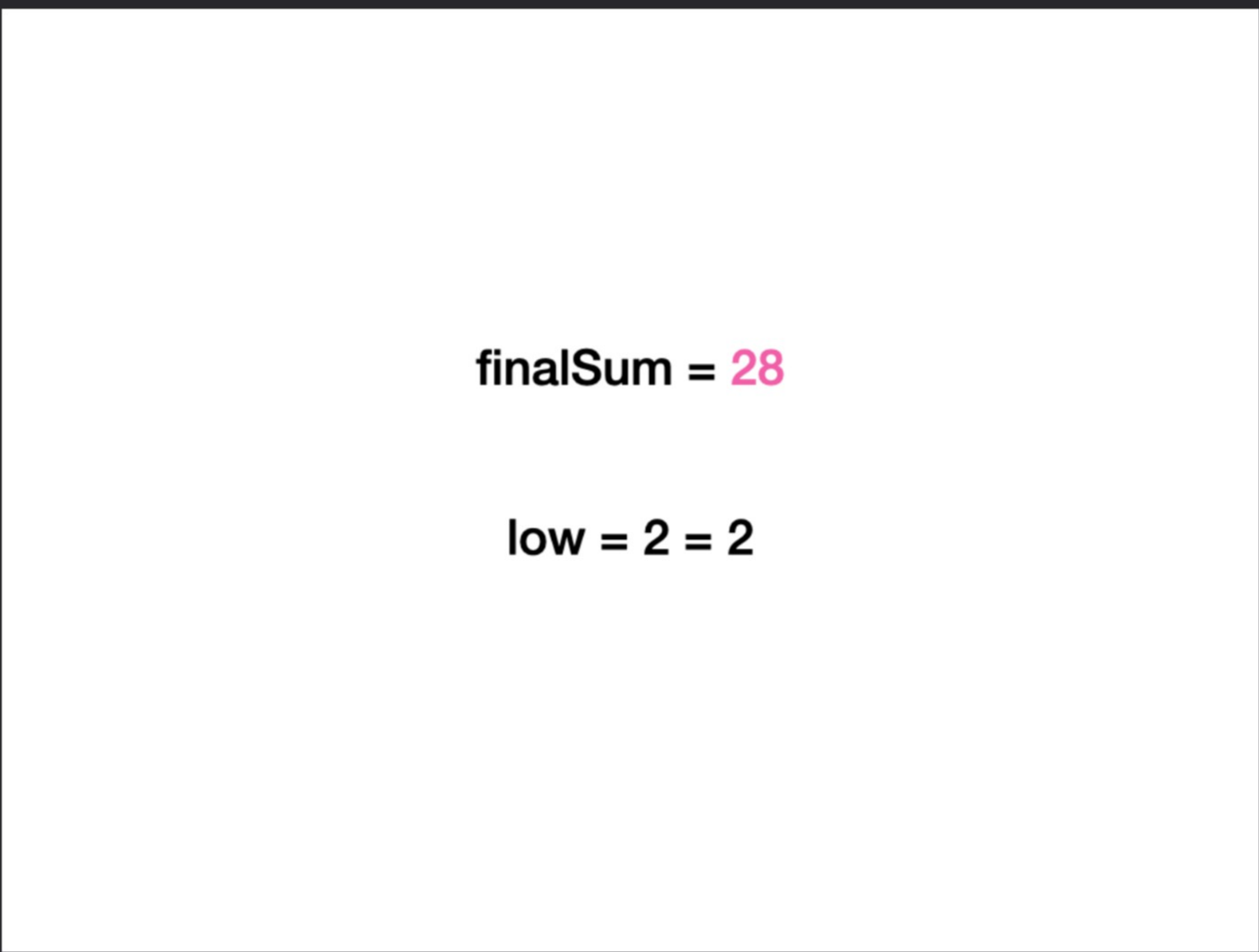
### Back to the Original Problem

We are given  $S$  and asked to find the **maximum** possible  $K$  and construct a split of size  $K$ .

If we take the sum of the smallest  $K$  positive even integers  $(2 + 4 + 6 + 8 + \dots + 2*K)$ , we'll obtain the **least** possible sum that has a split of size  $K$ . Let's denote this sum as **low**. A sum  $S$  will have a valid split of size  $K$  if  $S \geq \text{low}$ .

To solve the problem, we'll first find the **maximum**  $K$  where  $S \geq \text{low}$ . Starting with the split that sums to **low**, we'll add  $S - \text{low}$  to the largest integer to obtain our final split for  $S$ .

### Simulation



### Time Complexity

For a sum  $S$  with a split of **maximum** size  $K$ ,  $\text{low} = 2 + 4 + 6 + 8 + \dots + 2*k$ . In the sum, there are  $O(K)$  elements and the average element is  $O(K)$ , resulting in  $S = O(K^2)$  and  $K = O(\sqrt{S})$ . Since our algorithm runs in  $O(K)$ , our final time complexity is  $O(\sqrt{S})$ .

**Time Complexity:**  $O(\sqrt{S})$

### Space Complexity

Since we construct a list of size  $K$ , our space complexity is also  $O(\sqrt{S})$ .

**Space Complexity:**  $O(\sqrt{S})$

## C++ Solution

```
1 class Solution {
2     public:
3         vector<long long> maximumEvenSplit(long long finalSum) {
4             vector<long long> ans; // integers in our split
5             if (finalSum % 2 == 1) { // odd sum provides no solution
6                 return ans;
7             }
8             long long currentSum = 0; // keep track of the value of low
9             int i = 1;
10            while (currentSum + 2 * i <=
11                    finalSum) { // keep increasing size of split until maximum
12                currentSum += 2 * i;
13                ans.push_back(2 * i);
14                i++;
15            }
16            ans[ans.size() - 1] +=
17                finalSum % 2 == 1 ? 1 : 0; // add S - low to largest element
18            return ans;
19        }
20    };
```

## Java Solution

```
1 class Solution {
2     public List<Long> maximumEvenSplit(long finalSum) {
3         List<Long> ans = new ArrayList<Long>(); // integers in our split
4         if (finalSum % 2 == 1) { // odd sum provides no solution
5             return ans;
6         }
7         long currentSum = 0; // keep track of the value of low
8         int i = 1;
9         while (currentSum + 2 * i
10                <= finalSum) { // keep increasing size of split until maximum
11             currentSum += 2 * i;
12             ans.add((long) 2 * i);
13             i++;
14         }
15         int idx = ans.size() - 1;
16         ans.set(idx,
17                 ans.get(idx) + finalSum
18                     - currentSum); // add S - low to largest element
19         return ans;
20     }
21 }
```

## Python Solution

```
1 class Solution:
2     def maximumEvenSplit(self, finalSum: int) -> List[int]:
3         ans = [] # integers in our split
4         if finalSum % 2 == 1: # odd sum provides no solution
5             return ans
6         currentSum = 0
7         i = 1
8         while (
9             currentSum + 2 * i <= finalSum
10        ): # keep increasing size of split until maximum
11            currentSum += 2 * i
12            ans.append(2 * i)
13            i += 1
14        ans[len(ans) - 1] += finalSum - currentSum # add S - low to largest element
15        return ans
16
```

## Javascript Solution

```
1 /**
2  * @param {number} finalSum
3  * @return {number[]}
4  */
5 var maximumEvenSplit = function (finalSum) {
6     let ans = []; // integers in our split
7     if (finalSum % 2 === 1) {
8         // odd sum provides no solution
9         return ans;
10    }
11    let currentSum = 0; // keep track of the value of low
12    let i = 1;
13    while (currentSum + 2 * i <= finalSum) {
14        // keep increasing size of split until maximum
15        currentSum += 2 * i;
16        ans.push(2 * i);
17        i++;
18    }
19    ans[ans.length - 1] += finalSum - currentSum; // add S - low to largest element
20    return ans;
21 };
```