Approach 1: Dynamic Programming

**Intuition**

For each rod x, we can write +x, -x, or 0. Our goal is to write 0 using the largest sum of positive terms.

对于每一个数，我们可以写成其正、负或0，目标是用最大的正数和写出0。

For writings that have a sum of 0, let's call the sum of the positive terms written the *score*. For example, +1 +2 +3 -6 has a score of 6.

Since sum(rods) is bounded, it suggests to us to use that fact it in some way. Indeed, if we already wrote some sum in the first few terms, it doesn't matter how we got it. For example, with rods = [1,2,2,3], we could arrive at a sum of 3 in 3 different ways, but the effective score is 3. This upper-bounds the number of states we have to consider to 10001, as there are only this many possible sums in the interval [-5000, 5000].

**Algorithm**

Let dp[i][s] be the largest score we can get using rods[j] (j >= i), after previously writing a sum of s(that isn't included in the score).

当我们使用rods[j](j>=i)时，在此前和为s，让dp[i][s]为最大分数

For example, with rods = [1,2,3,6], we might have dp[1][1] = 5, as after writing 1, we could write +2 +3 -6 with the remaining rods[i:] for a score of 5.

In the base case, dp[rods.length][s] is 0 when s == 0, and -infinity everywhere else. The recursion is dp[i][s] = max(dp[i+1][s], dp[i+1][s-rods[i]], rods[i] + dp[i+1][s+rods[i]]).

from functools import lru\_cache

class Solution:

def tallestBillboard(self, rods):

@lru\_cache(None)

def dp(i, s):

if i == len(rods):

return 0 if s == 0 else float('-inf')

return max(dp(i + 1, s),

dp(i + 1, s - rods[i]),

dp(i + 1, s + rods[i]) + rods[i])

return dp(0, 0)

#### Approach 2: Meet in the Middle

**Intuition**

Typically, the complexity of brute force can be reduced with a "meet in the middle" technique. As applied to this problem, we have 3^N3*N* possible states, from writing either +x, -x, or 0 for each rod x, and we want to make this brute force faster.

What we can do is write the first and last 3^{N/2}3*N*/2 states separately, and attempt to combine them. For example, if we have rods [1, 3, 5, 7], then the first two rods create up to nine states: [0+0, 0+3, 0-3, 1+0, 1+3, 1-3, -1+0, -1+3, -1-3], and the last two rods also create nine states.

1和3对应9种状态，可进行记录

We will store each state as the sum of positive terms, and the sum of absolute values of the negative terms. For example, +1 +2 -3 -4 becomes (4, 7). Let's also call the difference 4 - 7 to be the delta of this state, so this state has a delta of -3.

Our high level goal is to combine states with deltas that sum to 0. The score of a state will be the sum of the positive terms, and we want the highest score. Note that for each delta, we only care about the state that has the highest score.

**Algorithm**

Split the rods into two halves: left and right.

For each half, use brute force to compute the reachable states as defined above. Then, for each state, record the delta and the maximum score.

Now, we have a left and right halves with [(delta, score)] information. We'll find the largest total score, with total delta 0.

class Solution(object):

def tallestBillboard(self, rods):

def make(A):

states = {(0, 0)}

for x in A:

states |= ({(a+x, b) for a, b in states} |

{(a, b+x) for a, b in states})

delta = {}

for a, b in states:

delta[a-b] = max(delta.get(a-b, 0), a)

return delta

N = len(rods)

Ldelta = make(rods[:N/2])

Rdelta = make(rods[N/2:])

ans = 0

for d in Ldelta:

if -d in Rdelta:

ans = max(ans, Ldelta[d] + Rdelta[-d])

return ans