1049. Last Stone Weight II

You are given an array of integers `stones` where `stones[i]` is the weight of the `i<sup>th</sup>` stone.

We are playing a game with the stones. On each turn, we choose any two stones and smash them together. Suppose the stones have weights `x` and `y` with `x <= y`. The result of this smash is:

- If `x == y`, both stones are destroyed, and
- If `x != y`, the stone of weight `x` is destroyed, and the stone of weight `y` has new weight `y - x`.

At the end of the game, there is **at most one** stone left.

Return *the smallest possible weight of the left stone*. If there are no stones left, return `0`.

Example 1:

```
Input: stones = [2,7,4,1,8,1]
Output: 1
Explanation:
We can combine 2 and 4 to get 2, so the array converts to [2,7,1,8,1]
then,
we can combine 7 and 8 to get 1, so the array converts to [2,1,1,1] then,
we can combine 2 and 1 to get 1, so the array converts to [1,1,1] then,
we can combine 1 and 1 to get 0, so the array converts to [1], then that's
the optimal value.
```

Example 2:

```
Input: stones = [31,26,33,21,40]
Output: 5
```

Example 3:

```
Input: stones = [1,2]
Output: 1
```

```
import sys
class Solution:
    def lastStoneWeightII(self, stones: List[int]) -> int:
        m = sum(stones)
```

```python
        n = len(stones)
        dp = [[False]*(m+1) for i in range(n+1)]

        for i in range(n+1):
            for j in range(m+1):
                if i==0 and j==0:
                    dp[i][j] = True
                elif i==0 and j!=0:
                    dp[i][j]=False
                elif j==0:
                    dp[i][j]=True
                else:
                    tar = stones[i-1]
                    if j-tar>=0:
                        dp[i][j] = dp[i-1][j] or dp[i-1][j-tar]
                    else:
                        dp[i][j] = dp[i-1][j]
        stone = sys.maxsize
        for j in range((m)//2+1):
            if dp[-1][j]==True:
                stone = min(stone,m-2*j)
        return stone
        # print(dp)
```