Marc's Cakewalk

HackerRank

Marc loves cupcakes, but he also likes to stay fit. Each cupcake has a calorie count, and Marc can walk a distance to expend those calories. If Marc has eaten j cupcakes so far, after eating a cupcake with c calories he must walk *at least* $2^j \times c$ miles to maintain his weight.

Example
$calorie = [5, 10, 7]$

If he eats the cupcakes in the order shown, the miles he will need to walk are $(2^0 \times 5) + (2^1 \times 10) + (2^2 \times 7) = 5 + 20 + 28 = 53$. This is not the minimum, though, so we need to test other orders of consumption. In this case, our minimum miles is calculated as $(2^0 \times 10) + (2^1 \times 7) + (2^2 \times 5) = 10 + 14 + 20 = 44$.

Given the individual calorie counts for each of the cupcakes, determine the minimum number of miles Marc must walk to maintain his weight. Note that he can eat the cupcakes *in any order*.

Function Description

Complete the *marcsCakewalk* function in the editor below.

marcsCakewalk has the following parameter(s):

- *int calorie[n]:* the calorie counts for each cupcake

Returns

- *long:* the minimum miles necessary

Input Format

The first line contains an integer n, the number of cupcakes in $calorie$.
The second line contains n space-separated integers, $calorie[i]$.

Constraints

- $1 \leq n \leq 40$
- $1 \leq c[i] \leq 1000$

Sample Input 0

```
3
1 3 2
```

Sample Output 0

```
11
```

Explanation 0

Let's say the number of miles Marc must walk to maintain his weight is $miles$. He can minimize $miles$ by eating the $n = 3$ cupcakes in the following order:

1. Eat the cupcake with $c_1 = 3$ calories, so $miles = 0 + (3 \cdot 2^0) = 3$.
2. Eat the cupcake with $c_2 = 2$ calories, so $miles = 3 + (2 \cdot 2^1) = 7$.
3. Eat the cupcake with $c_0 = 1$ calories, so $miles = 7 + (1 \cdot 2^2) = 11$.

We then print the final value of $miles$, which is 11, as our answer.

Sample Input 1

```
4
7 4 9 6
```

Sample Output 1

```
79
```

Explanation 1

$$(2^0 * 9) + (2^1 * 7) + (2^2 * 6) + (2^3 * 4) = 9 + 14 + 24 + 32 = 79$$