2300. Successful Pairs of Spells and Potions

Medium | Topics | Companies | Hint

You are given two positive integer arrays `spells` and `potions`, of length `n` and `m` respectively, where `spells[i]` represents the strength of the `i^th` spell and `potions[j]` represents the strength of the `j^th` potion.

You are also given an integer `success`. A spell and potion pair is considered **successful** if the **product** of their strengths is **at least** `success`.

Return *an integer array* `pairs` *of length* `n` *where* `pairs[i]` *is the number of* ***potions*** *that will form a successful pair with the* `i^th` *spell.*

Example 1:

> **Input:** spells = [5,1,3], potions = [1,2,3,4,5], success = 7
> **Output:** [4,0,3]
> **Explanation:**
> - 0th spell: 5 * [1,2,3,4,5] = [5,**10**,**15**,**20**,**25**]. 4 pairs are successful.
> - 1st spell: 1 * [1,2,3,4,5] = [1,2,3,4,5]. 0 pairs are successful.
> - 2nd spell: 3 * [1,2,3,4,5] = [3,6,**9**,**12**,**15**]. 3 pairs are successful.
> Thus, [4,0,3] is returned.

Example 2:

> **Input:** spells = [3,1,2], potions = [8,5,8], success = 16
> **Output:** [2,0,2]
> **Explanation:**
> - 0th spell: 3 * [8,5,8] = [**24**,15,**24**]. 2 pairs are successful.
> - 1st spell: 1 * [8,5,8] = [8,5,8]. 0 pairs are successful.
> - 2nd spell: 2 * [8,5,8] = [**16**,10,**16**]. 2 pairs are successful.
> Thus, [2,0,2] is returned.

Constraints:

- `n == spells.length`
- `m == potions.length`
- `1 <= n, m <= 10^5`
- `1 <= spells[i], potions[i] <= 10^5`
- `1 <= success <= 10^10`

Approach :- Brute force

Approach 1: Brute force

```
for (i : spells)
{
    count = 0
    for (j : Potions)
    {
        if (i*j ≥ success)
            count ++
    }
    Push count to ans vector
}
```

$T(n) : O(SP)$
$S(n) : O(1)$

Approach 2:

spells : [5 1 3] potions : [1 2 3 4 5]

lets say potions is in sorted order.
now if

i * j ≥ success then
it is obvious that **for all k > j** the equation **i * k ≥ success** holds.

Thus for each spell we need to find the potion with the least strength that will form a successful pair.

Sort potions array

```
for( i : spells )
{
    x = do binary search for least j that
        satisfies i*j ≥ success

    ans.push_back( potions size - x)
}

binary search
{
    l = 0    r = potions size - 1

    while( l ≤ r )
    {
        m = (l+r)/2

        p = i * potions[m]

        if( p ≥ success)        i.e. there might be even
            r = m - 1           lesser valued potions[m]
        else                    that satisfy p ≥ success
            l = m+1
    }

    return l
}
```

$$T(n) : O(P \log P) + O(S \log P)$$

S(n): sorting space