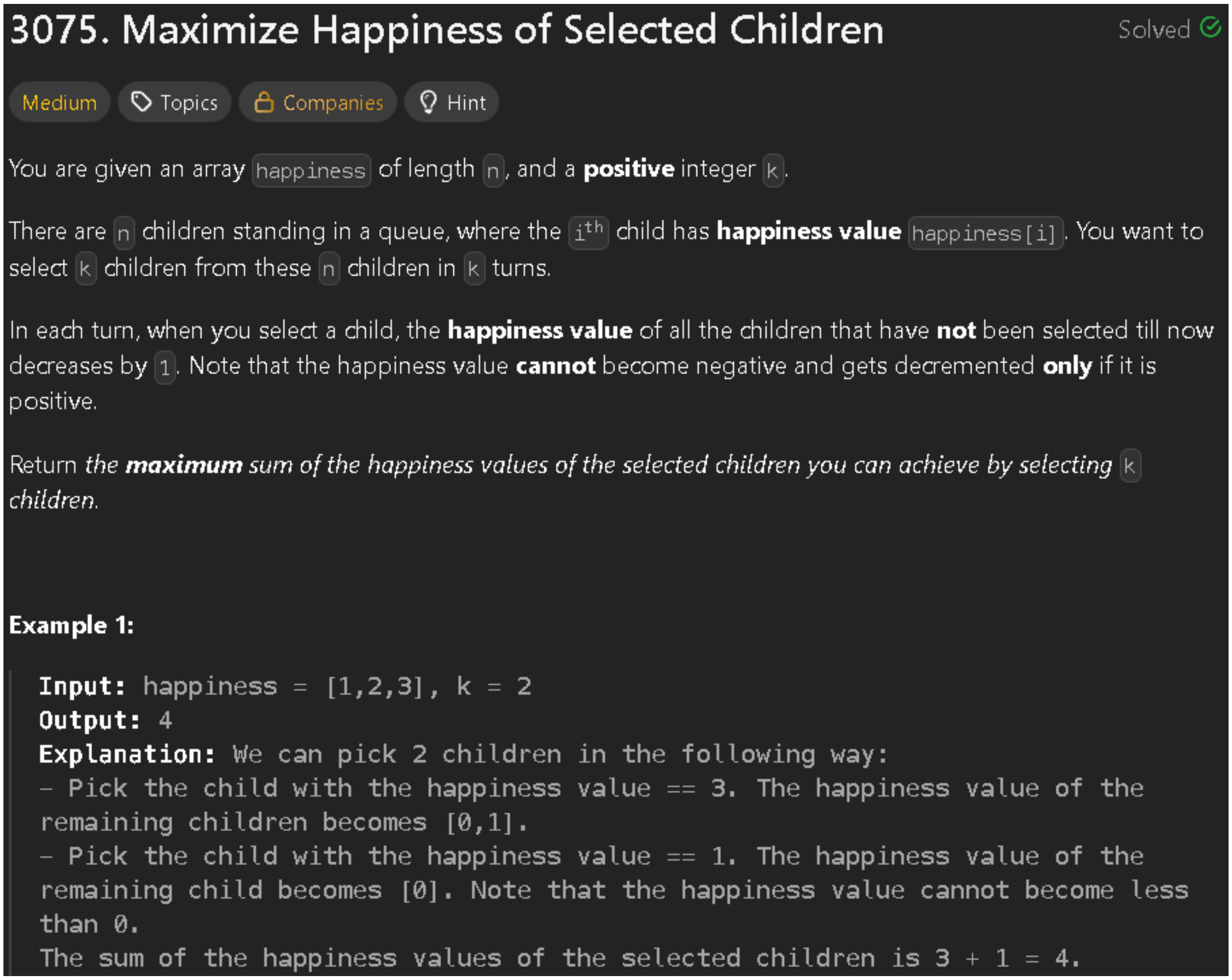


3075. Maximize Happiness of Selected Children

You are given an array `happiness` of length `n`, and a **positive** integer `k`.

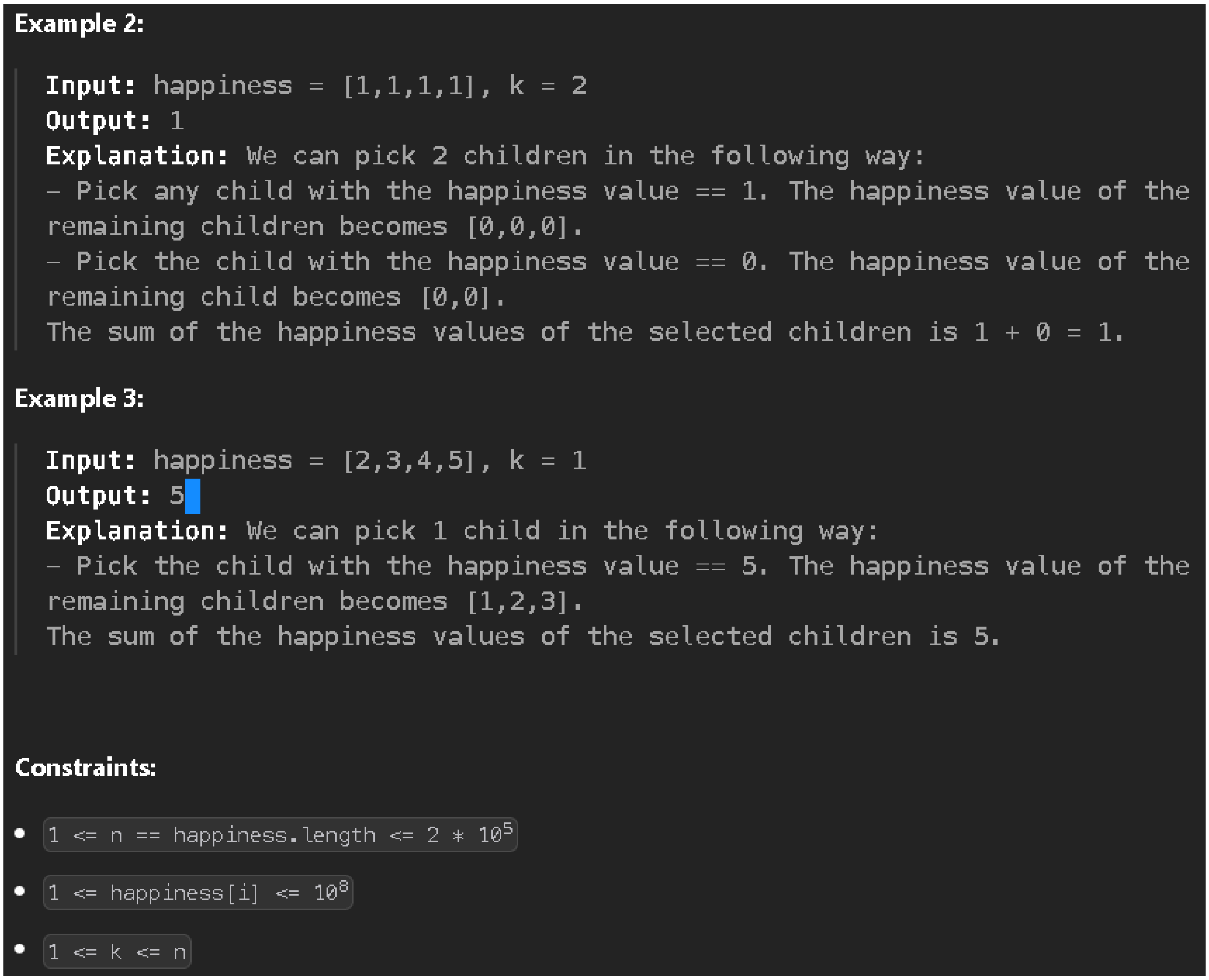
There are `n` children standing in a queue, where the i^{th} child has **happiness value** `happiness[i]`. You want to select `k` children from these `n` children in `k` turns.

In each turn, when you select a child, the **happiness value** of all the children that have **not** been selected till now decreases by `1`. Note that the happiness value **cannot** become negative and gets decremented **only** if it is positive.

Return *the **maximum** sum of the happiness values of the selected children you can achieve by selecting* `k` *children*.

Example 1:

```
Input: happiness = [1,2,3], k = 2
Output: 4
Explanation: We can pick 2 children in the following way:
- Pick the child with the happiness value == 3. The happiness value of the
remaining children becomes [0,1].
- Pick the child with the happiness value == 1. The happiness value of the
remaining child becomes [0]. Note that the happiness value cannot become less
than 0.
The sum of the happiness values of the selected children is 3 + 1 = 4.
```



Example 2:

> **Input:** happiness = [1,1,1,1], k = 2
> **Output:** 1
> **Explanation:** We can pick 2 children in the following way:
> - Pick any child with the happiness value == 1. The happiness value of the remaining children becomes [0,0,0].
> - Pick the child with the happiness value == 0. The happiness value of the remaining child becomes [0,0].
> The sum of the happiness values of the selected children is 1 + 0 = 1.

Example 3:

> **Input:** happiness = [2,3,4,5], k = 1
> **Output:** 5
> **Explanation:** We can pick 1 child in the following way:
> - Pick the child with the happiness value == 5. The happiness value of the remaining children becomes [1,2,3].
> The sum of the happiness values of the selected children is 5.

Constraints:

- `1 <= n == happiness.length <= 2 * `10^5
- `1 <= happiness[i] <= `10^8
- `1 <= k <= n`

Python:

```
class Solution:
    def maximumHappinessSum(self, happiness, k):
        happiness.sort()  # ascending

        ans = 0
        n = len(happiness)

        for i in range(k):
            val = happiness[n - 1 - i] - i
            if val <= 0:
                break
            ans += val
        return ans
```

JavaScript:

```
var maximumHappinessSum = function(happiness, k) {
    happiness.sort((a, b) => a - b); // ascending

    let ans = 0;
    let n = happiness.length;

    for (let i = 0; i < k; i++) {
        let val = happiness[n - 1 - i] - i;
        if (val <= 0) break;
        ans += val;
    }
    return ans;
};
```

Java:

```
import java.util.*;

class Solution {
    public long maximumHappinessSum(int[] happiness, int k) {
        Arrays.sort(happiness); // ascending

        long ans = 0;
        int n = happiness.length;

        for (int i = 0; i < k; i++) {
            int val = happiness[n - 1 - i] - i;
            if (val <= 0) break;
            ans += val;
        }
        return ans;
    }
}
```