75 Days of Code
Day 51
Problem no : 2542
Problem Title : Maximum Subsequence Score

.

You are given two 0-indexed integer arrays nums1 and nums2 of equal length n and a positive integer k. You must choose a subsequence of indices from nums1 of length k.

For chosen indices i0, i1, ..., ik - 1, your score is defined as:

The sum of the selected elements from nums1 multiplied with the minimum of the selected elements from nums2.
It can defined simply as: (nums1[i0] + nums1[i1] +...+ nums1[ik - 1]) * min(nums2[i0] , nums2[i1], ... ,nums2[ik - 1]).
Return the maximum possible score.

A subsequence of indices of an array is a set that can be derived from the set {0, 1, ..., n-1} by deleting some or no elements.

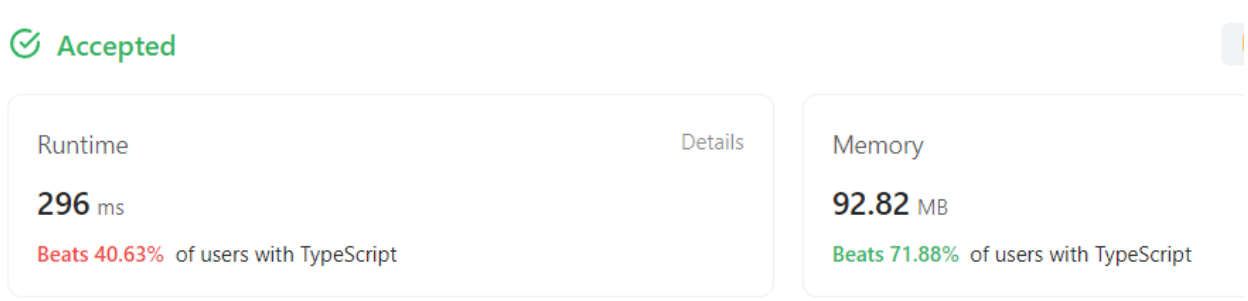
Example 1:

Input: nums1 = [1,3,3,2], nums2 = [2,1,3,4], k = 3
Output: 12
Explanation:
The four possible subsequence scores are:
- We choose the indices 0, 1, and 2 with score = (1+3+3) * min(2,1,3) = 7.
- We choose the indices 0, 1, and 3 with score = (1+3+2) * min(2,1,4) = 6.

- We choose the indices 0, 2, and 3 with score = (1+3+2) * min(2,3,4) = 12.
- We choose the indices 1, 2, and 3 with score = (3+3+2) * min(1,3,4) = 8.
Therefore, we return the max score, which is 12.
Example 2:

Input: nums1 = [4,2,3,1,1], nums2 = [7,5,10,9,6], k = 1
Output: 30
Explanation:
Choosing index 2 is optimal: nums1[2] * nums2[2] = 3 * 10 = 30 is the maximum possible score.

```typescript
interface pair<T, U> {
    first: T;
    second: U;
};
function maxScore(nums1: number[], nums2: number[], k: number): number {
    type number2 = pair<number, number>;
    function comparator(a: number2, b: number2): number {
        if (a.first === b.first) return a.second - b.second;
        else return b.first - a.first;
    }
    const n: number = nums1.length;
    const p: number2[] = new Array(n);
    const pq = new MinPriorityQueue();
    let res: number = 0, sum: number = 0;
    for (let i = 0; i < n; i++) p[i] = { first: nums2[i], second: nums1[i] };
    p.sort(comparator);
    for (let i = 0; i < n; i++) {
        const { first: a, second: b } = p[i];
        sum += b;
        pq.enqueue(b);
        if (i < k - 1) continue;
        if (pq.size() > k) sum -= pq.dequeue().element;
        res = Math.max(res, sum * a);
    }
    return res;
};
```



Shubham Agrahari

Shubham Agrahari