2071. Maximum Number of Tasks You Can Assign

Solved

Hard | Topics | Companies | Hint

You have `n` tasks and `m` workers. Each task has a strength requirement stored in a **0-indexed** integer array `tasks`, with the `i^th` task requiring `tasks[i]` strength to complete. The strength of each worker is stored in a **0-indexed** integer array `workers`, with the `j^th` worker having `workers[j]` strength. Each worker can only be assigned to a **single** task and must have a strength **greater than or equal** to the task's strength requirement (i.e., `workers[j] >= tasks[i]`).

Additionally, you have `pills` magical pills that will **increase a worker's strength** by `strength`. You can decide which workers receive the magical pills, however, you may only give each worker **at most one** magical pill.

Given the **0-indexed** integer arrays `tasks` and `workers` and the integers `pills` and `strength`, return *the* ***maximum*** *number of tasks that can be completed.*

greedy + deque + binary search

Example 1:

```
Input: tasks = [3,2,1], workers = [0,3,3], pills = 1, strength = 1
Output: 3
Explanation:
We can assign the magical pill and tasks as follows:
- Give the magical pill to worker 0.
- Assign worker 0 to task 2 (0 + 1 >= 1)
- Assign worker 1 to task 1 (3 >= 2)
- Assign worker 2 to task 0 (3 >= 3)
```

```python
'''
class Solution:
    def maxTaskAssign(self, tasks: List[int], workers: List[int], pills: int, strength: int) -> int:
        tasks.sort()
        workers.sort()

        def max_assign(k):
            t = tasks[:k]
            w = deque(workers[-k:])
            p = pills

            for i in reversed(range(k)):
                if w and w[-1] >= t[i]:
                    w.pop()
                elif p > 0:
                    if w and w[0] + strength >= t[i]:
                        w.popleft()
                        p -= 1
                    else:
                        return False
                else:
                    return False
            return True

        l, h = 1, min(len(tasks), len(workers))
        while l < h:
            m = (l + h + 1) // 2
            if max_assign(m):
                l = m
            else:
                h = m - 1

        return l

    '''
```

- Only considers the strongest k workers
- Tries to assign them greedily to hardest tasks first
- Cannot easily find the weakest available worker that can be boosted with a pill
- It assumes that pill + weakest will always work — but that's not true in all configurations
- No efficient way to find the smallest available worker ≥ (task - strength)

This results in missed assignments, especially when weak workers can't be boosted for mid-to-hard tasks.

Could use SortedList from SortedContainer to deal with this problem, but we have a better solution, see below.

```python
class Solution:
    def maxTaskAssign(
        self, tasks: List[int], workers: List[int], pills: int, strength: int
    ) -> int:
        n, m = len(tasks), len(workers)
        tasks.sort()
        workers.sort()

        def check(mid: int) -> bool:
            p = pills
            ws = deque()
            ptr = m - 1
            # Enumerate each task from largest to smallest
            for i in range(mid - 1, -1, -1):
                while ptr >= m - mid and workers[ptr] + strength >= tasks[i]:
                    ws.appendleft(workers[ptr])
                    ptr -= 1

                if not ws:
                    return False
                elif ws[-1] >= tasks[i]:
                    ws.pop()
                else:
                    if p == 0:
                        return False
                    p -= 1
                    ws.popleft()
            return True

        left, right, ans = 1, min(m, n), 0
        while left <= right:
            mid = (left + right) // 2
            if check(mid):
                ans = mid
                left = mid + 1
            else:
                right = mid - 1

        return ans
```

pre-sorts workers and preselects only those who can handle tasks[i] with or without a pill.

Then deque:

- ws[-1]: strongest worker without a pill
- ws[0]: weakest worker who needs a pill