P8. Bulldozers

[[홍원표 21701065]]

문제 분석

There are m tasks, n days to complete all tasks, d days to complete each task, and s_i which indicates which date to start each task. k number of tasks can be done at each day. n, d, m, and s_i are given as inputs, and the problem is to find the minimum k that works with the given inputs. The restriction is as follows: **1 <= n <= 100,000** **1 <= d < n** **1 <= m <= 100,000**

문제 풀이

This problem can be solved using simple **brute-force**, (with one little trick). The meaning of brute-force in this problem means to **start the k as 1 and increase by 1 whenever testing with the following k fails**. The trick is to change the form of si into **'groups of same number**'. For example, 1 2 4 2 1 3 5 6 2 3 6 4 will become 'two 1s, three 2s, two 3s, two 4s, one 5, and two 6s." Then, the testing process can be reduced because we're **not 'counting off' s_i, but rather 'subtracting' days.**

문제 풀이 분석

Each testing for certain k takes **n days of loop** (although it may stop at earlier point). Also, in each day-loop, **d number of loop** exists to shorten the due dates of the remaining tasks by 1. The maximum of d can be n. Therefore, **Time Complexity: $O(n^2)$**

Each indication of s_i, and the number of same s_i are required, so **Space Complexity: O(n)**

Discussion

The problem states that each task must be solved within d days, but this is very ambiguous. To be precise, days represent 24 hours. For example, if d is 1, the work must be completed within that very day, in 24 hours. This should make sense because the input data is stated as 1 <= d < n. However, as I was solving this problem, I realized that this was not the case. What this problem actually wanted was that d being 0 is the same as the d being 1 as I intended. If d = 0, this would mean that the task must be completed before the next day, as in it must be completed this day.

I believe this must be corrected, because otherwise requirements of d should have been 0 <= d < n

```
day 1, k 1
  2 0 0
  1 0 0
  0 1 0
   leftover = 1
day 2, k 1
  3 1 0
  3 0 0
  0 3 0
   leftover = 3
day 3, k 1
  2 3 0
  2 2 0
  0 2 2
   leftover = 4
day 4, k 1
  2 2 2
  2 2 1
fails
day 1, k 2
  2 0 0
  0 0 0
  0 0 0
   leftover = 0
day 2, k 2
  3 0 0
  1 0 0
  0 1 0
   leftover = 1
day 3, k 2
  2 1 0
  1 0 0
  0 1 0
   leftover = 1
day 4, k 2
  2 1 0
  1 0 0
  0 1 0
   leftover = 1
day 5, k 2
  1 1 0
  0 0 0
  0 0 0
   leftover = 0
day 6, k 2
  2 0 0
  0 0 0
  0 0 0
   leftover = 0
day 7, k 2
  0 0 0
  0 0 0
  0 0 0
   leftover = 0
day 8, k 2
  0 0 0
  0 0 0
  0 0 0
   leftover = 0
```

```
8 2 12
1 2 4 2 1 3 5 6 2 3 6 4
```