S262 - Even the Rhythm

Isaac Wong {WongChun1234}

2026-02-14

Table of Contents

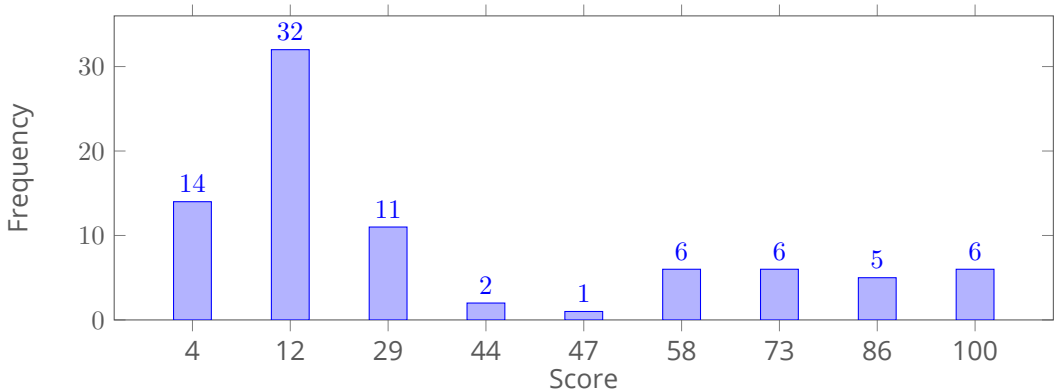
Background

Problem Idea by `happypotato`
Preparation by `happypotato`, `QwertyPi`, `snowysecret`
Presentation by `WongChun1234`

Problem Restatement

- You are given N integers $A_1, A_2, \ldots, A_N$ and integer K.
- Find integers $X_1, X_2, \ldots, X_N$ satisfying the following:
  - X_i is a factor of A_i for all $1 \leq i \leq N$
  - *Note count* $\text{sum}(A_i \div X_i) \leq K$
  - Under the above conditions, **minimise** the *instability* $\max X_i - \min X_i$.
- If there are multiple constructions, output any.

Statistics



First solved by **WYK23F32** (Xu Adam) at **31m 22s**.

Subtasks

For all cases: $3 \leq N \leq 2 \times 10^5, 1 \leq K \leq 10^{11}, 1 \leq A_i \leq 5 \times 10^5$, $N \leq K \leq \text{sum}(A_i)$.

Subtask	Points	Constraints
1	4	$N = 2$, $A_1 = 1$, $A_2 = 2$
2	8	$N = 2$, $A_1 = 1$
3	17	$N \leq 2000$, $A_i \leq 2000$, $A_1 \leq A_2 \leq \cdots \leq A_N$, A_i is prime
4	18	$N \leq 100$, $A_i \leq 100$
5	11	$N \leq 2000$, $A_i \leq 2000$
6	15	$N \leq 5 \times 10^4$, $A_i \leq 10^5$, $A_1 = 1$
7	13	$N \leq 5 \times 10^4$, $A_i \leq 10^5$
8	14	No additional constraints

Table of Contents

Subtask 1 (4%): $N = 2$, $A_1 = 1$, $A_2 = 2$

Since X_i must be a factor of A_i, $X_1 = 1$ and $X_2 = 1$ or 2.

- If $K \geq 3$, we can take $X_2 = 1$ with instability 0.
- If $K = 2$, we have to take $X_2 = 2$ with instability 1.

Expected score: 4

Subtask 2 (8%): $N = 2$, $A_1 = 1$

Now, there can be a lot of options for X_2.

We can brute over the value of X_2, then check if it satisfies the conditions.

Expected score: 12

Subtask 3 (17%): $N \leq 2000$, $A_i \leq 2000$, A_i is sorted, A_i is prime

For each i, $X_i = 1$ or A_i.

Now, we make an observation:

Observation. If there are two distinct sequences X with the same instability, picking the one with less notes is always not worse. (In other words, you will not prefer the one with more notes in any case.)

In this subtask, observe that if we set $X_i = A_i$, then it is best for us to set $X_j = A_j$ for all $1 \leq j \leq i$. (Be reminded that A_i is sorted in the constraints.)

This is because if we pick $X_j = A_j$, the instability doesn't change and the note count decreases.

Subtask 3 (17%): $N \leq 2000$, $A_i \leq 2000$, A_i is sorted, A_i is prime

With that, we can simply brute over the prefix p from 0 to N, then take:

- $X_i = A_i$ if $i \leq p$
- $X_i = 1$ if $i > p$

And then check all $N + 1$ constructions of X naively in $O(N)$ time.

Time complexity: $O(N^2)$

Expected score: 29

Table of Contents

Subtask 4 (18%): $N \leq 100$, $A_i \leq 100$

Consider brute forcing over the value of max X_i. Denote this value as M.

We make another observation:

Observation. After fixing a maximum value M, for each i, you will take the **maximum** $X_i \leq M$ that is a factor of A_i.

Proof. If you can take a higher X_i, $A_i \div X_i$ decreases. Also, max X_i doesn't change (since we defined it to be M), and min X_i won't decrease.

Subtask 4 (18%): $N \leq 100$, $A_i \leq 100$

Then, we have an $O(N \times (\max A_i)^2)$ solution:

- **Brute over** M from 0 to $\max A_i$.
- For each M, for every i, find the **highest** valid $X_i \leq M$ in $O(\max A_i)$ time.
- Then, naively check each construction in $O(N)$ time.

Time complexity: $O(N^3)$

Expected score: 47

Subtask 5 (11%): $N \leq 2000$, $A_i \leq 2000$

Seems like we need a quadratic solution instead of a cubic solution now. Let's revisit our previous solution:

- Brute over M from 0 to max A_i.
- For each M, for every i, find the highest valid $X_i \leq M$ in $O(\max A_i)$ time.
- Then, naively check each construction in $O(N)$ time.

Can we do some precomputation to speed up the process?

Subtask 5 (11%): $N \leq 2000$, $A_i \leq 2000$

Seems like we need a quadratic solution instead of a cubic solution now. Let's revisit our previous solution:

- Brute over M from 0 to max A_i.
- For each M, for every i, find the highest valid $X_i \leq M$ in $O(\max A_i)$ time.
- Then, naively check each construction in $O(N)$ time.

Can we do some precomputation to speed up the process? Yes!

For each i, we can find the largest factor $\leq M$ for all M in $O(\max A_i)$ time. It will be something like `f[i][M] = (A[i] % M == 0 ? M : f[i][M + 1])`.

Alternatively, store all factors of A_i and use binary search every time.

Subtask 5 (11%): $N \leq 2000$, $A_i \leq 2000$

Now, our solution becomes this:

- For every i, **precompute factors / previous factor** in $O(N \times \max A_i)$ time.
- Brute over M from 0 to $\max A_i$.
- For each M, for every i, find the highest valid $X_i \leq M$ in $O(\log A_i)$ / $O(1)$ time.
- Then, naively check each construction in $O(N)$ time.

Time complexity: $O(N \times \max A_i)$ / $O(N \times \max A_i \log \max A_i)$

Expected score: 58

Subtask 6 (15%): $N \leq 5 \times 10^4$, $A_i \leq 10^5$, $A_1 = 1$

We need a sub-quadratic solution now.

Looking at the constraints, we see $A_1 = 1$. What does this do?

Recall from earlier subtasks, $A_1 = 1$ implies $X_1 = 1$, which implies min $X_i = 1$.

Can we use this to speed up our previous solution?

Subtask 6 (15%): $N \leq 5 \times 10^4$, $A_i \leq 10^5$, $A_1 = 1$

Let's look at our previous solution:

- For every i, precompute factors / previous factor in $O(N \times \max A_i)$ time.
- Brute over M from 0 to $\max A_i$.
- For each M, for every i, find the highest valid $X_i \leq M$ in $O(\log A_i)$ / $O(1)$ time.
- Then, naively check each construction in $O(N)$ time.

Notice that the instability of a constuction is simply $M - 1$ now.

This means we will take the smallest M possible. Then we can binary search on M instead of brute forcing over it!

Let's change our solution accordingly.

Subtask 6 (15%): $N \leq 5 \times 10^4$, $A_i \leq 10^5$, $A_1 = 1$

Let's change our solution accordingly:

- For every i, **precompute factors** in $O(\max A_i \log \max A_i)$ time **with Sieve**.
- **Binary search** on M from 0 to $\max A_i$.
- For each M, for every i, find the highest valid $X_i \leq M$ in $O(\log A_i)$ time.
- Then, naively check each construction in $O(N)$ time.

Note that this solution is not correct in general; it is only correct when $\min X_i = 1$.

Time complexity: $O(N \times \log^2 \max A_i)$

Expected score: 73

Table of Contents

Subtask 7 (13%): $N \leq 5 \times 10^4$, $A_i \leq 10^5$

Again, we need a sub-quadratic solution. This time, the binary search idea doesn't work.

Let's revisit our solution from subtask 5:

- For every i, **precompute factors / previous factor** in $O(N \times \max A_i)$ time.
- Brute over M from 0 to $\max A_i$.
- For each M, for every i, find the highest valid $X_i \leq M$ in $O(\log A_i)$ / $O(1)$ time.
- Then, naively check each construction in $O(N)$ time.

Actually, if we think about it, the distinct possible values of X_i is only $N \times \tau(A_i)$, where $\tau(A_i)$ represents the number of factors of A_i.

Maybe we can do some kind of sliding window algorithm on factors...?

Subtask 7 (13%): $N \leq 5 \times 10^4$, $A_i \leq 10^5$

Maybe we can do some kind of sliding window algorithm on factors...? Let's try it!

- Build a list of pairs (d, i) where d is a factor of A_i.
- Sort all pairs by factor value d.
- Maintain a sliding window $[l, r]$ on this sorted list (two pointers).
  - The window corresponds to choosing min X_i and max X_i from factors in this range.
- Pick the largest X_i inside the window for each i. Dynamically maintain the note count when sliding.

Time complexity: $O(F \log F)$, where $F = \sum \tau(A_i)$. Maximum of $\tau(i)$ for $1 \leq i \leq 10^5$ is 128, so the solution passes this subtask.

Expected score: 86

Subtask 8 (14%): No Additional Constraints ($N \leq 2 \times 10^5$, $A_i \leq 5 \times 10^5$)

Now, the previous solution is too slow. Let's try to optimise it!

Optimisation 1. Sorting is the bottleneck here, can we avoid the sorting?

Since the elements are up to 5×10^5 only, we can do so by performing counting sort.

In practice, we will maintain a vector of is for every d, and do two pointer on the ds.

This optimisation saves a log in our solution.

Subtask 8 (14%): No Additional Constraints ($N \leq 2 \times 10^5$, $A_i \leq 5 \times 10^5$)

Now, the previous solution is too slow. Let's try to optimise it!

Optimisation 2. Notice that there are only a few values of A_i that achieves the maximum $\tau(A_i)$. (Maximum of $\tau(i)$ for $1 \leq i \leq 5 \times 10^5$ is 200.)

What if we de-duplicate elements and consider them in batch? The number of elements will drop from $N \times \tau(A_i)$ to something closer to $N \log A_i$.

If only one of the two optimisations is applied, constant optimisation might be needed to get AC.

Time complexity: $O(N \times \tau(A_i))$ / $O(N \times \log^2 A_i)$ / $O(N \times \log A_i)$

Expected score: 100

Implementation Details

In all the solutions above, it might seem a bit annoying to find a construction. Actually, there is a clean way to do so!

As established before, if we fix some max $X_i = M$, you can easily recover the construction of X.

Therefore, you don't have to store the current X every time you find a better solution. you can simply store max X_i, and recompute it when outputting the answer.