Padmavati Tutorial

Pre-requisites:

This problem will require concepts of gcd, divisibility and number theory to solve. Make sure your basics of gcd, euclidean algorithm and number theory are clear before attempting this problem. The following links can help you learn the prerequisites:

- https://www.topcoder.com/community/data-science/data-science-tutorials/prime-numbers-factorization-and-euler-function/
- https://www.topcoder.com/community/data-science/data-science-tutorials/mathematics-for-topcoders/
- http://www.geeksforgeeks.org/basic-and-extended-euclidean-algorithms/
- http://www.virtualnerd.com/pre-algebra/factors-fractions-exponents/prime-factorization-greatest-common-factor/greatest-common-factor/greatest-common-factor-two-numbers
- https://www.hackerearth.com/practice/math/number-theory/basic-number-theory-1/tutorial/

Problem Description:

There are n variables, each with a specific initial value ai. The task is to make the GCD of all the variables not equal to 1. This can be achieved by either deleting some variables or changing the values of some variables. Changing the values can only be done by increasing the value one-by-one. Both the tasks: deleting or increasing value by 1 require some specific cost. The cost of deleting a variable is x and that of increasing the value of a variable by 1 is y. We have to ensure that the variables are not co-prime (the gcd of all variables is not 1) at the minimum possible cost.

Difficulty Level:

Medium-Hard

Editorial:

Let us begin by understanding how we got the given sample output from the given sample input. In the example, number 1 must be deleted (with cost 23) and number 16 must increased by 1 (with cost 17) and then the numbers will become [17, 17, 17] with gcd = 17. Hence, our objective is achieved.

Let's define *cost*(*g*): the cost if we want gcd of the array to become *g*. The answer is *min (cost*(*g*)) for $g > 1$. Now, let's see how to calculate *cost*(*g*). For each number like *c*, we can delete it with cost *x* or we can add it till *g* divides it. So, we must pay min(x, y, ([c/g] *g - c)) . Let's iterate on possible values for [c/g] *g ([] is the greatest integer function).

Before entering the main part of the solution, let's define two helper functions:

– *cnt*(*l*, *r*): this function returns the number of *i*'s such that $l \leq ai \leq r$.

– *sum*(*l*, *r*): this function returns $\sum(l <= i <= r)\ a_i$. To implement this function, define an array *ps*, such that *psi* keeps the sum of values less than or equal to *i*. Then *sum*(*l*, *r*) = *psr* - *psl* - 1.

Now for each multiple of *g* like *k*, let's find the cost of numbers in the range (*k* - *g*, *k*], and sum up these values. We must find the best *f* and divide the range into two segments (*k* - *g*, *f*] and (*f*, *k*] and delete the numbers in the first range and add the numbers in second range till they become *k*. Now to find the best value for *f*, (g - f) *y = x/y => f = g - min(g, x/y)

So, the total cost for this range is *cnt*(*k* - *g* + 1, *k* - $\lfloor f \rfloor$) × *x* + (*cnt*(*k* - $\lfloor f \rfloor$ + 1, *k*) × *k* - *sum*(*k* - $\lfloor f \rfloor$ + 1, *k*)) × *y*.

Complexity of solution:

The complexity of this solution is $O(max\ a_i * log\ (max\ a_i)$), where a_i is the initial value assigned to the variable i.