Lily Bernard at Learngame-HC College

Context

Ludo Bernard is the principal at Learngame-HC College and wishes the best results for his daughter, Lily Bernard, a final year student at the college. Here, all that matters is the z-score, one of the most common measures in statistics to gauge and compare performance among students. It's the z-score that counts for university admissions.

For a subject, if n is the number of students and $x_1, x_2 \ldots x_n$ are the final grades of the students, the average and standard deviation of the cohort are calculated as follows:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$$

$$= \sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n} - \mu^2}$$

Then the z-score of a student A is calculated as follows:

$$z_A = \frac{x_A - \mu}{\sigma}$$

In the mathematics course, the final grades of all students have been recorded. Lily has a good grade, so her z-score is also very good. However, since she wants to study medicine, her father Ludo is worried and wonders if her z-score will be sufficient.

Being the principal, he decides to manipulate the students' grades to improve Lily's z-score. However, to avoid suspicion, he must not change the grades randomly. He sets 4 rules to avoid being exposed:

1. **Do not change Lily's grade.** Indeed, as she is his daughter (conflict of interest), her grade may be checked by a third party.
2. **Do not lower the grade of any student.** Indeed, a student surprised negatively by his grade is more likely to request a review, which would expose the scheme.
3. **If a student A has a higher grade than a student B before cheating, then student A must have a grade greater than or equal to student B after cheating.** If two friends A and B who talk a lot think that A did better in the evaluations, but B ends up with a better grade, they may suspect foul play.
4. **Grades must remain multiples of 0.01%.** Indeed, the format of the grades does not allow for precision greater than one hundredth of a percentage.

In short, the principal allows himself to increase the grades of any subset of students, as long as the order of the grades is preserved and his daughter's grade remains unchanged.

Your Mission

The file data.txt contains the blank grades of the students. Your mission is to modify the grades while respecting the previous 4 constraints, in order to optimize Lily's z-score.

The answer to the puzzle is a string in the same format as the one in data.txt, but for which the grades have been modified.

Flag

The file 'flag-steps-fake.txt' serves as an example to guide you on how to create the FLAG. Except for one big detail, reproduce exactly the steps in 'flag-steps-fake.txt'.

The big detail is obviously that you have to have to change the grades of the students before applying these steps.

Solutionnaire

Theory

Let's observe the following expression that gives Lily's z-score:

$$z_L = \frac{x_L - \mu}{\sigma}$$

From an intuitive point of view, the simplest way to improve it is to increase x_L, but we don't have the right. Another simple way is to decrease μ, but it's impossible because the rules prevent us from lowering a student's grade. Our last means is therefore to decrease σ. How? By increasing the grades of the weakest students. However, we must not increase the grades of weak students infinitely because it should not increase μ too much at the same time.

From a more technical point of view, there are too many possibilities to attempt a brute force now. We still need to theoretically reduce the number of possibilities.

Basic principle in optimization: if an improvement is possible, it means that we are not in an optimal state. In our case, if it is possible to decrease the standard deviation without modifying the average, then this is an improvement, so the current state is not optimal.

The question is: how to decrease the standard deviation without modifying the average? Here's how:

By bringing two grades closer together. This means taking 2 grades a and b such that $a + 0.02 \leq b$, then changing the grades to $a' = a + 0.01$ and $b' = b - 0.01$. We will call this operation a **convergence**.

It is clear that this operation does not change the average, but the proof that it decreases the standard deviation will be left to the end so as not to break the reading flow.

However, it is not always legal to make a convergence. Indeed, no grade should decrease. However, if we decide to increase two grades, then the difference between the two new grades must be 0 or 0.01, because if the difference exceeds 0.01, it is possible to make a convergence between the two grades, which would improve z_L!

In short, we arrive at the following result: **The difference between the highest and lowest of all increased grades must not exceed** 0.01. In other words, our new set of increased grades should look like this: we set a floor p. All grades below p will be increased to p or $p + 0.01$. Grades that were already higher than $p + 0.01$ will remain unchanged.

More in detail, among the increased grades, the number of grades k to be increased to p rather than $p + 0.01$ remains to be determined, but once this number is determined, the grades to be increased to p are those which were the k lowest before increase. Thus, the only parameters to choose to optimize our z-score are the floor p and this number k. This greatly reduces our search space!

Code

See the Python code in the provided files. This code finds p and k that maximize z_L. Note: p and k are almost always named "floor" and "jump_idx" in the code, respectively.

Finally, after executing the code, it turns out that the final solution is much more elegant than expected: in fact, all the increased grades must be equal, and there is therefore no difference of 0.01 between them. The optimal k is therefore literally the number of increased grades. We did not find a theoretical proof of why this is the case.

In short, the code finds that **increasing all grades below** 84.11% **to this value** maximizes z_L.

Comments about the code

Working with integers only

One of the questions that participants might ask is how to avoid working with floating-point numbers to prevent inaccuracies. Well, here's how.

The code operates with scores multiplied by 100. This doesn't change the z-score since it multiplies Lily's score, the average, and the standard deviation all by 100. Therefore, both the denominator and the numerator will be multiplied by 100 as shown below:

$$z_L = \frac{x_L - \mu}{\sigma}$$

This doesn't change the value of the ratio.

Next, the other source of ugly floating-point numbers is the square root in the calculation of the standard deviation. The trick here is to optimize z_L^2 instead of z_L, which is equivalent. If we denote Q as the sum of the squares of the scores:

$$z_L = \frac{x_L - \mu}{\sqrt{\frac{Q}{n} - \mu^2}}$$
$$z_L^2 = \frac{(x_L - \mu)^2}{\frac{Q}{n} - \mu^2}$$

And there you go, no more irrational numbers. But we still have a non-integer over non-integer form. To address this, we can multiply both the numerator and the denominator by n^2. If we denote P as the sum of the scores:

$$\begin{aligned} z_L^2 &= \frac{(x_L - \mu)^2}{\frac{Q}{n} - \mu^2} \\ &= \frac{(Nx_L - P)^2}{NQ - P^2} \end{aligned}$$

Now z_L^2 is in the integer/integer form. For optimization, it's easy to compare two fractions a and b representing a z-score to find the larger one:

(a.num * b.denom) versus (b.num * a.denom)

The most common mistake

This section was added the day after the CSGames 2024.

Some teams submitted exactly the same flag, which was incorrect. You came close, by also raising the grades below a certain threshold to that threshold without modifying the other grades... except they found a threshold of 84.08% instead of 84.11%.

The problem is that your algorithm was probably trying to make note modifications one by one until no individual note could be improved in terms of the z-score. The issue with this kind of algorithm ("local search") is that it can stop at a local maximum that is not a global maximum. That's what happened to you. From a threshold of 84.08%, attempting to modify an individual note worsens the z-score, so your algorithm refused to move further.

The provided code contains the above algorithm. It's not called by default; you need to change MINISTEPSALGO to True in cst. If you do, you'll see the error on the console.

Answer

To obtain the flag, you must generate the same string as in data.txt, but replacing grades that are lower with 84.11%. Then, calculate sha1(modified-string)

FLAG{70dd5eb985dca345fae297712542e61bf0ee7cfa}

Appendix

We had postponed the proof that a convergence decreases the standard deviation. Here it is:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n} - \mu^2}$$

Taking our two grades a and b such that $a + 0.02 \leq b$, we wondered if bringing them together by changing them to $a' = a + 0.01$ and $b' = b - 0.01$ decreases the standard deviation. Well, in the formula above, the contribution to the sum of the old grades is $a^2 + b^2$, while it is $a'^2 + b'^2$ after convergence. The rest of the formula is unchanged. To prove that the standard deviation decreases, it is enough to prove that $a'^2 + b'^2 < a^2 + b^2$.

$$\begin{aligned}
&(a'^2 + b'^2) - (a^2 + b^2) \\
=&(a'^2 - a^2) + (b'^2 - b^2) \\
=&(a' - a)(a' + a) + (b' - b)(b' + b) \\
=&0.01(a' + a) - 0.01(b' + b) \\
=&0.01(a' + a - b' - b) \\
<&0
\end{aligned}$$

QED

Code link

https://github.com/desharnc27/incoming-csg-2024-python