# Manual metric

## Average laptop power:

There are various models, but I can only find detail report of two.

<https://www.delltechnologies.com/asset/en-us/products/laptops-and-2-in-1s/technical-support/full-lca-latitude7300-anniversary-edition.pdf> 4,62 w

<https://esg.asus.com/english/file/PEP_Notebook_C423.pdf> 3.53w

So, I take the average between two, 4.075w

## Mean Time Spent:

I utilize the Codeforces API to retrieve submission data for the first 10,000 participants in a contest. From these participants, I filter out only those who completed the tasks sequentially. I exclude outliers by removing data whose submission times fall outside the range of two standard deviations from the mean to ensure accuracy. Then I get the final mean.

## Background energy consumption:

This refers to using an average laptop to write the code not including the testing and debugging. Background energy consumption = mean time spent \* average laptop power

## Mean Runtime:

Similar to the mean time spent, I get the mean runtime of python. But there is no accepted answer in python for task F and G, so I also get the mean runtime of C++. Then I find the mean ratio between the two and estimate the testing runtime of python of task F and G base on the ratio.

## Estimated number of testing:

<https://www.researchgate.net/publication/261355752_Adoption_of_Software_Testing_in_Open_Source_Projects--A_Preliminary_Study_on_50000_Projects> This shows there is a 0.335 positive correlation between lines of code and the test cases. So, I take 10 participants’ code and take the mean of lines and get the estimated number of testing they do. Alternatively, we can probably use mean number of submissions per user, but I believe most participants test their code locally before submitting.

## Testing Energy Consumption

I think of two ways of how we estimate this, one is estimating the energy consumption by using the submitted code and running the test case on our machine and use an energy profiler. But we do not have access to all the test cases, so another way is to estimate the energy consumption by the runtime. We still run the code on one test case and record the CPU and RAM power. Then we can calculate the testing energy consumption by: Testing Energy Consumption = (CPU power + RAM power \* memory usage percentage) \* mean runtime \* estimated number of testing

## Debugging Energy Consumption:

<https://www.pullrequest.com/blog/cost-of-bad-code/> According to this article, an average developer spends 42% of the time debugging the code. Need to know how much time does the person spend on running the code. <https://faculty.washington.edu/ajko/papers/Ko2006SeekRelateCollect.pdf> This article shows that 10% of time people would run their code during the debugging time. The formular then becomes: Debugging Energy Consumption = mean time spent \* debugging time ratio \* debugging run code ratio \* ( runtime CPU power + runtime RAM power \* mean RAM usage percentage)

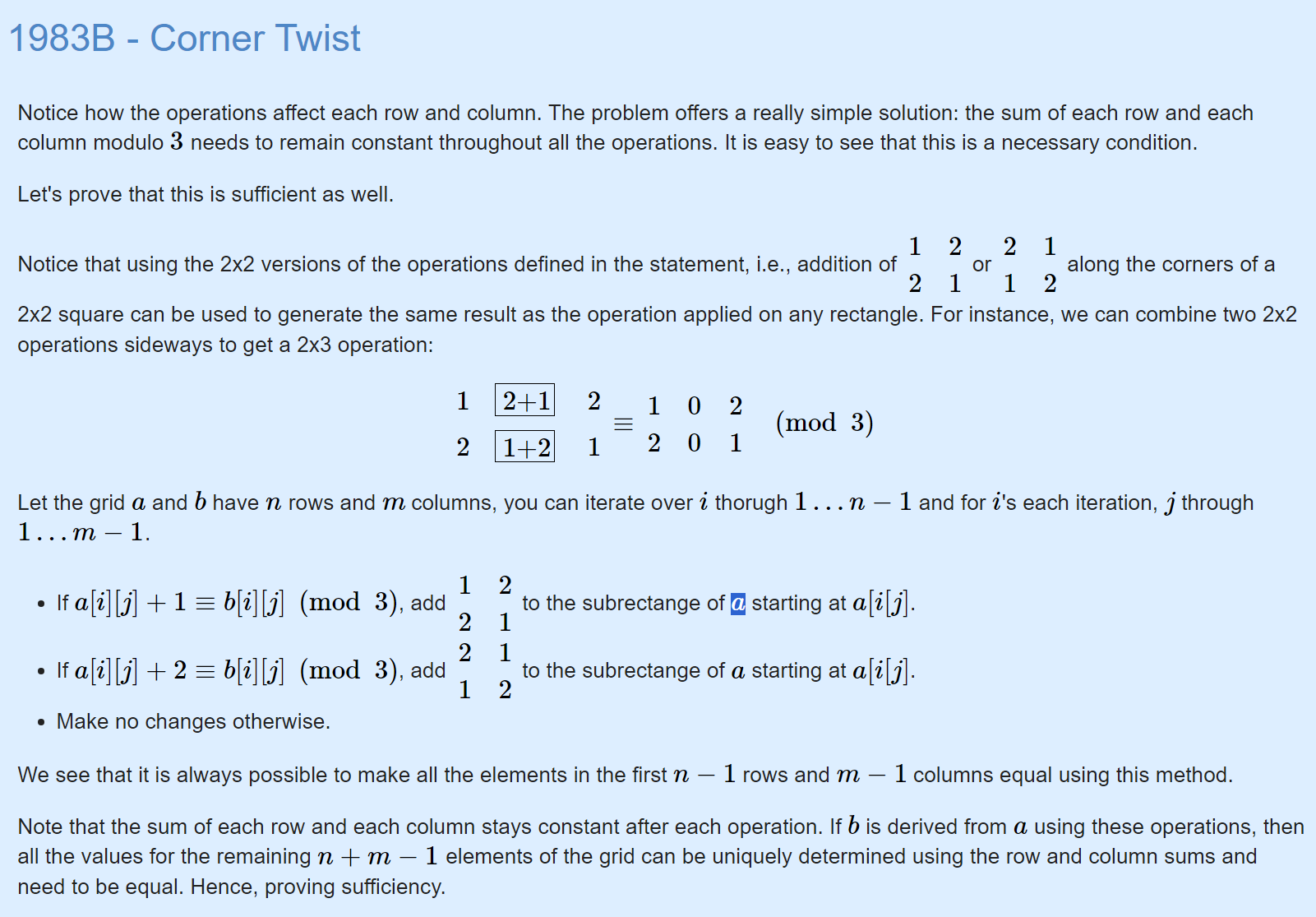
# LLM Metric:

We decided to change the model to chatgpt4. As we have the data about the training (<https://www.economist.com/technology-quarterly/2024/01/29/data-centres-improved-greatly-in-energy-efficiency-as-they-grew-massively-larger>) and query cost ( <https://towardsdatascience.com/chatgpts-energy-use-per-query-9383b8654487> ).

## Query cost:

0.0017-0.0026 per query so I take the middle 0.0022kwh

## Human Insight:

The written solution for human programmer. Basically, how would a human think about the problem, this will be the input to the LLM if the LLM cannot solve the problem in 5 attempts. Example: 

## Queries before human insight:

I give the problem to the LLM and ask the LLM write the python solution for it. If it fails, I will feed it the test result and ask the LLM to fix the code up to 5 times. After 5 queries if the LLM cannot solve the problem, I will give it the human insight. We record the number of queries and best test cases passed percentage.

## Queries after human insight:

I give the human insight to the LLM and ask LLM to rewrite the code. If it fails, feed again the insight and the test results until the it passes all the test cases up to 3 times. We also record the number of queries and best test cases passed percentage.

## Test cases passed percentage:

Unfortunately, the website will stop running tests if it fails on one. We can only do test cases passed/ total number of test cases. Note one test cases can contain more than one input but still, we only count the test cases not specific test input correctness.

## Query Energy Consumption:

Query Energy Consumption = Total number of queries \* estimated energy per query

## Estimate Time spent on producing the insight:

The insight provided definitely cost a human some time to think and feed it to LLM. So we estimate that by <https://ieeexplore.ieee.org/document/7181430>. This paper tell us a developer would normally spend 38% of the time to understand a task.” The Basic Understanding (BU) is the sum of all the basic moments of program understanding. It is represented by all time intervals between sprees which are longer than the reaction time. Basic understanding can be performed inside development activities (i.e., intra-activities) and across subsequent activities (i.e., inter-activities). • BUintra is the Basic Intra-Activity Understanding Time that is the sum of all the time intervals, longer than RT, between the sprees composing an activity. • BUinter is the Basic Inter-Activity Understanding Time that is the sum of all the time intervals, longer than RT, between subsequent activities.” So I use the intra understanding as an estimation. LLM can also help with understanding the task, for that I say the test cases percentage pass before human insight would be a good indication of how much would the LLM help the human to come up with such insight. The formula then become, time = mean time spent \* 0.38 \* (1 – test passed percentage before human insight)

## Estimated time spent on reading and refactoring:

If the LLM fails even after the human insight is produced, we need to manually fix the code and finish it off. We need to estimate the time spent on both reading and refactoring the code. <https://faculty.washington.edu/ajko/papers/Ko2006SeekRelateCollect.pdf> “Developers spent about a fifth of their noninterrupted time reading code, a fifth of their time editing code” So we can formulate this as: time = mean time spent \* 0.42(debugging time ratio) \* (0.2 +0.2)

## Estimated time spent on adding the missing functionalities:

Time = time spent on reading and refactoring + (1 – test passed percentage after human insight) \* mean time spent - Estimated time spent on producing the insight. Given the human already has produced such human insight, we can estimate that the human does not need to understand the task again and base on the failing test cases we estimate the effort of adding the remaining functionalities.

## Total energy Consumption:

Energy = (Estimated time spent on adding the missing functionalities + Estimate Time spent on producing the insight) \* power of laptop

## Carbon footprint:

270 g Co2e per kwh ( <https://www.nowtricity.com/country/united-kingdom/>)