Part 2 (40% of final grade).

Definition: Given a set of strings (genomes) $S = S_1 \ldots S_n$, a string p (pattern) and an integer k. We say that $S_i, 1 \leq i \leq n$, is a **k-instance** of p if there exists a string Y such that Y is a substring of S_i and Y can be obtained from p by inserting up to k characters to p.

a. Design an efficient algorithm and write a program that does the following: Given a set of COG-spelled genomes S, where each genome is segmented into segments such that each segment could contain one or more operons. Also given parameters q, ℓ, k and an "unknown" COG X: find all strings c (over the COG alphabet) of length ℓ that have $\geq q$ k-instances in S , such that COG X appears at least once in c.

   For simplicity (and speed), you may add the requirement that the pattern appear, as is (without insertions), at least once in one of the input strings.

   Implement the algorithm in python and document your code very clearly. Provide a "high level" description of your method\algorithm in the report.

b. Group your results (all COG strings c complying by the requirements stated in a) by their length (number of COGs in the string) and sort each group by decreasing quorum (the number of genomes in which the string was found). (As was done in Part 1 of the project).

c. **Answer:** How did you enforce the constraint k (in addition to the constraints $q, \ell, and\ X$ that were previously applied and explained in Part 1) on the sought strings c?

d. **Answer:** What is the time and space complexity of your algorithm? Explain clearly and in detail. How did the extension of the problem, in Part 2 of the project, to allow up to k character insertions in instances of the pattern, affect the time and space requirements of your algorithm? Do you see some **practical** time difference in running time as the value of k increases? Measure and report the difference via a graph\chart (running time versus value of k).

e. Consider the "unknown" COGs from the file "COG_INFO_TABLE" that are labeled "uncharacterized", "poorly characterized", or "function uknown". Run your program with various parameter value settings for q, ℓ and k, and several choices for the unknown\uncharacterizedc COGs. Identify a COG X and a corresponding pattern string c containing X, such that the ranking of c greatly increases when allowing k > 0 (versus the ranking the same pattern string resulting from running the program as in Part 1, with k=0). Write a few sentences describing the functional context of X according to your results, and why you think that allowing insertions in the pattern increased its ranking.

f. Your submission should include your documented code (preferably in a notebook), and a report of length 2-4 pages addressing the previous items a-e. The report can be in Hebrew, but you need to run "spellcheck" on it before submitting. Do not send me incremental

versions of your report by email and ask me to give you comments again and again… instead, you are invited to schedule office hour meetings and ask me questions directly.

Code (documented notebook) and report (detailed) due date: 13\12\2021

Behatzlaha!