Python Programming Assignment

Please attempt either problem-1 or problem-2.

Problem-1[10+10+10+10+10]: The goal is to analyze a sequential design (consisting of gates and flip-flops). The gate-level design description is referred to as "netlist" also in the semiconductor industry and VLSI tool flows. A netlist can be considered as a graph where all the gates and flip-flops are represented as nodes and the connections between them are represented as edges. When this graph is computed, a path (with gates falling on them) can be considered as the connection between any 2 flip-flops. Consider a file (named as input.txt), the contents of which are as follows [For logic gate, look at https://www.geeksforgeeks.org/logic-gates/ and for flip-flop, look at: https://www.tutorialspoint.com/digital-electronics/digital-electronics-flip-flops.htm]:

G1 G2 G5 G7
G1 G6 G3 G31 G9
G7 G6 G78 G781 G51

G45 G789 G432 G47

In the above, in any line, the end-points represent the flip-flops ALWAYS. So, as per above lines, flip-flop(s) are G1, G7, G9, G51, G45 and G47 and rest all are gates. To any flip-flop, an incoming path is considered as the linkage from other flip-flop(s). So, in the line "G1 G2 G5 G7" G7 flip-flop has an incoming path from G1. Similarly, "G7 G6 G78 G781 G51" means that G51 has an incoming path from G7. Also, "G45 G789 G432 G47" means that G47 has an incoming path from G45. The same goes on for other lines in input.txt file.

NOTE: In the test inputs provided with this assignment, the names of the flip-flops/gates can start with any alphabetical letter. Therefore, this notation of G1, G2 etc., as shown in the above description are only for representation. Additionally, please note that flip-flops (which are end-points in each line) and the gates (which are present in between the end-points) can start with different alphabetical letters too.

Q1. **[Incoming path-based ranking]** We want to rank the flip-flops of the netlist in the descending order of the total number of incoming path(s) to them.
Therefore, by definition of “incoming path” as described above, we want to obtain ranking of flip-flops as below:
Say, Flip-flops are represented as Fi, Fj, Fk and so on. They respectively have incoming paths as 23, 45, 13 and so on. So, the descending order of ranking would be: Fj, Fi, Fk and so on.
Write a python program to obtain this ranking given input as following 2 different test inputs (which are basically input.txt files mentioned in the above problem description):
a) input_1
b)input_2

Q2. **[Unique Incoming path-based ranking]** If we analyze the provided inputs, we can observe that the same set of flip-flops have different paths between them. i.e., following can be observed between flip-flops G1 and G7:

G1 G2 G5 G7
G1 G21 G51 G678 G7
G1 G24 G53 G781 G7
G1 G22 G55 G76 G890 G7

We want to have a ranking based on only the unique paths between a set of flip-flops. In the above example, even though we have 4 incoming paths to G7 from G1, we would consider only one path from G1 to G7. G7 may have other incoming paths from other flip-flops. But, in this unique incoming path based ranking, we need to take only one incoming path from other flip-flops to G7.

Write a python program to obtain this unique incoming path-based ranking given input same as that of Q1.

Q3.**[Gate-based score ranking]** Another different way of ranking is to consider the gates also in the path from one flip-flop to another. Therefore, we devise a ranking scheme as below:
From any path from flip-flop Fi to Fj, count the number of gates. Say, this number is N. The value “N” is to be used as a score for that incoming path from Fi to Fj.

So, as per the above problem description, “G1 G2 G5 G7” G7 flip-flop has an incoming path from G1 with a score of 2. Similarly, “G7 G6 G78 G781 G51” means that G51 has an incoming path from G7 with a score of 3. Also, “G45 G789 G432 G47” means that G47 has an incoming path from G45 with a score of 2. In the same manner, “G1 G22 G55 G76 G890 G7” has an incoming path from G7 with a score of 4.

Thereafter, to obtain the final ranking of all flip-flops in terms of incoming paths, you just need to add all the individual scores of these paths.

Write a python program to obtain this ranking (the score-based ranking) given input as following 2 different test inputs same as Q1, Q2

Q4. **[Gate counting exercise]** As explained in the problem description, between any 2 flip-flops, there are gates on each path. Write a python program to count the occurrences of the gates and list down top 10 such occurring gates (i.e., top 10 highly ranked gates) in each of the provided input test cases.

Q5. **[Counting occurrence of combination of gates]** Consider the following contents of the input.txt file:

G1 G2 G5 G7
G1 G21 G51 G678 G7
G1 G24 G51 G678 G781 G7
G1 G22 G55 G76 G890 G7

If we observe carefully in the above, “G51 G678” is occurring together (i.e., occurring as a tuple) in more than one path, i.e., it occurs in 2 paths. Write a python program to find the number of highest such occurrence of combined occurrence of gates. This is to say that K-number of gates are occurring together in more than one path. Your program should provide the number of K as output.

Problem 2[20+10+20]: Debugging a design is often considered a bottleneck in the development cycle of integrated circuits (IC’s) for a wide range of systems

from network connected miniaturized devices to large processors powering gigantic servers. The goal of this assignment is to analyze the simulation trace of a design and prepare inputs for a machine learning(ML)-based analysis. In ML paradigm, large amount of data is crunched into smaller numbers called "features". So, to proceed further in debug, we need features of debug data to begin with.

Let's try to understand how we can develop some useful features from the simulation trace of a design. We want to analyze the cycle-accurate behavior of the design in terms of the contents of all flip-flops in the design [For flip-flop, look at: https://www.tutorialspoint.com/digital-electronics/digital-electronics-flip-flops.htm]. As flip-flops are the sequential elements in the design, it makes some sense to analyze the value stored in these flip-flops in each clock cycle (obviously, not infinitely! But over a certain number of clock cycles only). Consider that the design contains **M flip-flops** and the **simulation trace is N clock cycles** long. Note that owing to several factors ranging from limitations of on-chip debug data acquisition infrastructure to the subsequent difficulty in root cause discovery, N is either kept close to M or much less than that of M in industrial designs.

So, we have input in the form of a table with N rows and M columns. Let's say that the flip-flops are named as a1, a2,.....b6,p9 and so on.

a1	**a2**	**..**	**..**	**..**	**..**	**..**	**b3**	**b9**	**p1**	**p2**	**p3**	**p4**	**..**	**q7**
1	0	1	1	0	1	1	0	1	1	0	1	0	1	1
0	..	..	..	..	..	..	..	..	..	..	..	..	..	0
1	..	..	..	..	..	..	..	..	..	..	..	..	..	1
1	0	..	..	..	..	..	..	..	..	..	..	..	..	1
1	..	..	..	..	..	..	..	..	..	..	..	..	..	0
0	..	..	..	..	..	..	..	..	..	..	..	..	..	0
0	1	..	..	..	..	..	..	..	..	..	..	..	..	1

We define the below 5 features that need to be extracted from the above data **for each of the flip-flop of the design:**

A. number of 1's across all the cycles
B. first cycle in which 1 is observed
C. cycle from which consecutive run of 1 is observed
D. number of occurrences of k-length consecutive run of 1 (take k=5)
E. maximum length of consecutive run of 1

IMPORTANT1: If you observe that for any flip-flop, some/all of above feature(s) can not be computed, take "-1" as feature values instead.

Q1. **[Feature Computation 1] Write a python program (necessarily involving numpy) to compute the above 5 features (of each flip-flop) given few test cases.** (These test cases are taken from representative benchmark industrial designs and can have different values of M and N). Note that in the input test case, the first row has the name of flip-flops. **Rank the flip-flops in the descending order of averaged value of the sum of all above 5 features.**

NOTE: The instructor agrees that the above featuring scheme has some limitations and may seem like arbitrarily decided. In fact, how to construct "important features' remains a somewhat open research problem to date. [You are encouraged to devise your own features and compute them as a part of additional learning]

Q2. **[Feature Computation 2]** Let's now modify the feature computation scheme slightly. We now have following five features: (please keep the suggestion given in IMPORTANT1: in mind)

A1. number of occurrences of "111000"
B1. number of occurrences of k-length consecutive run of 1 (take k=6)
C1. number of occurrences of k-length consecutive run of 0 (take k=4)
D1. number of occurrences of k-length consecutive run of 00 (take k=4)
E1. number of occurrences of "1100"

IMPORTANT2: To make things simple, try to compute A1 and E1 in a **block-wise manner**. Let's say you get 111000 in any flip-flop from cycle 2 to cycle 8.

Next time, for this flip-flop, you should look for that tuple from cycle 9 onwards ONLY.

Write a python program (necessarily involving numpy) to compute the above 5 features given few test cases. Rank the flip-flops in the descending order of averaged value of the sum of all above 5 features. Do you observe any change in ranking of flip-flops obtained in Q1 and Q2? Which feature is creating the difference in ranking of above 2 schemes?

Q3. **[Tuple-based Distance Computation between Neighbors]**

[Illustration] Consider that in flip-flop "F0", you can observe a tuple of "0011" beginning at cycle P. We want to compute the cycle distance of its immediate neighboring flip-flop "F1" for the same tuple. That is to say, flip-flop "F1" has the same tuple "0011" occurring at cycle Q. We need to calculate the cycle distance: |P-Q| (the absolute value of subtraction of P and Q).

Let's call this distance as cycle distance (CD).

Therefore, for tuple 0011, CD(F0,F1)=|P-Q|

Now, F0 can have this tuple many times as we move from cycle 1 to cycle N. Let's say, **F0 has this tuple 5 times and F1 has this tuple 4 times.** We then need to compute the following distances for F0 and F1:

i) CD for first tuple of F0 & first tuple of F1

ii) CD for second tuple of F0 & second tuple of F1

iii) CD for third tuple of F0 & third tuple of F1

iv) CD for fourth tuple of F0 & fourth tuple of F1

We then compute **Sum(CD(F0,F1)) = i + ii + iii + iv**

{Similar logic applies in vice-versa i.e., when F0 has 4 tuples and F1 has 5 tuples}

Similarly we need to continue doing for F1 and F2, F2 and F3, …….. and so on, till we reach the last flip-flop.

Write a Python program to compute Sum(CD) for neighboring flip-flop combinations from first to the last flip-flop when tuple is "0011", "000111", "1001" and "00001111".

NOTE: It is absolutely fine if any/all of these tuples are not there, take CD=0

----------x----------