Homework 2:
COL 761

- Due: **20th September 11:59PM**
- You can do the homework in groups of 3
- You are free to use any programming language but it must compile on **HPC**. Please start early
- **30% penalty for each late day**
- **Time limit = 30 min**
- You can do the homework in groups of 3. Only one zip file must be submitted per team
- You should upload all your code to the GitHub repo used for HW1. You are not allowed to make any uploads after the deadline. Otherwise, you will be penalized as per the late submission policy. Your repos will be cloned immediately after the deadline

---

1. This question is about getting yourself familiar with frequent subgraph mining tools. You will use the Yeast Cancer molecule dataset shared on Piazza. The format is the following:
   ```
   #graphID
   # of nodes
   Series of Node Labels
   # of edges
   Series of "Source node, Destination Node, Edge label"
   ```
   Run gSpan, FSG (also known as PAFI), and Gaston (you should be able to find it online) against frequency threshold in the dataset (you may need to write a script to change the format of the dataset) at minSup = **5%, 10%, 25%, 50%** and **95%**. Plot the running times and explain the trend observed in the running times. Specifically comment on the growth rates and why one technique is faster than the others. You are free to consult the respective papers. **(20 points)**

2. Consider the problem of subgraph search. You are given a database of graphs $D = \{G_1, \cdots, G_n\}$ (Ex. Chemical compounds), and a query subgraph Q. Your goal is to identify all data graphs that contain (subgraph isomorphism) the query subgraph. As we know subgraph isomorphism is NP-hard. We need a better method than brute force. So, here is an idea of an index structure. Mine *m* subgraph patterns from D. Convert every data graph G_i into a binary feature vector $F_i = [f_{i,1}, \cdots, f_{i,m}]$ of dimension m such $f_{i,j}$ indicates whether pattern j is contained within G_i or not. Now, given Q, you can construct its feature representation as well. It is now easy to see that Q can be subgraph isomorphic to some data graph G_i only if all the features contained in Q are also contained in G_i. Thus, you can prune out all data graphs that do not satisfy this criteria and perform subgraph isomorphism only on the remaining graphs. Several questions however remain unanswered. What should be the value of m? How do you mine the subgraph patterns? If you mine frequent subgraphs, they are likely to be very high. How do you prune it down to m? What should be the frequency threshold? Your task is to solve these questions so that the time for subgraph isomorphism is as small as possible.

3. You need to submit the following scripts:

    - A script **compile.sh** which should compile all your code
    - A script **plot.sh** which takes graph dataset file and output filename as input and saves the plot for part 1 in png format. [Execution: ./plot.sh <graphs dataset file> <plot output file>]. **(Do not add any extension to the output file name)**
    - A script **index_and_query.sh** that will take a file containing a set of graphs. The script should build the index structure and store it in whatever way. Maximum time allowed to built this structure is 10 minutes. After the completion of indexing your script should output **"Indexing complete"** (**without the quotes and make sure to <u>flush the stdout afterwards</u>**) and then take two space separated strings for file containing query graphs and output filename **(Do not add any extension to the output file name)**
    - For each query, you need to provide the IDs of all graphs containing that query. If there are n graphs in the query file, the output file should have n lines; each line containing the IDs in tab delimited manner
    - Execution: ./index_any_query.sh <file contains Database of graphs>. **Do not output anything else other than "Indexing complete" to stdout.** After your program outputs **"Indexing complete"**, we will provide <query_graphs_file> <output_file> through stdin to your script
    - Assumptions: Both database of graphs and query graphs file will conform to the format stated in part 1. You may assume that all graphs will be chemical compounds. However, we will evaluate your submissions on datasets other than Yeast. The dataset may contain up to 100k graphs

Scoring

- Accuracy: (F-score) * 18 **(18 marks)**
- Querying time efficiency: We will have a competition **only on querying time** among all submitted implementations. The fastest would get full points. If your algorithm is X% slower than the fastest, then you would get X% of full points. You would be in this competition only if you get F-score of 1 in accuracy **(42 marks)**

Compiler Specification:

- GCC version 7.1.0
- Java version 1.8
- Python3 version 3.6.5

Submission instructions

1. All submitted repositories will be cloned immediately following the expiry of the deadline. The timestamp of the last pushed commit will be treated as your submission time.
2. Upload one EntryNo-Assgn2.zip file to your team's Github account. This entry should be of any one of your team's member. Ex. 2016CS10240-Assgn2.zip. On unzipping it should produce one folder. The folder should have the same name as zip file. This folder should contain all the source files and all the bash scripts.

3. The submission zip should also contain a Readme.txt explaining all the files you bundled, explanation for part 1 and your approach for part 2. It should also mention names, entry numbers and individual contribution of each team member to the assignment.
4. In addition to the existing submission files required, you are also required to (separately) submit an **clone.sh** script on Moodle [Do not zip it]. This clone.sh should **only** contain command for cloning your repository into current directory.
5. For loading modules/dependencies/libraries, we will share a sheet on piazza and you are expected to write your name along with full commands that you want to run before running your scripts on HPC.
6. It is absolutely essential to conform to the folder hierarchy and naming conventions.

What will we do ?

1. Firstly we will look at google sheet containing your dependencies and execute those commands. At this stage, all imports are resolved.
2. We will run clone.sh which should clone your remote GitHub Data Mining repository. Inside the cloned (local) repository, we should be able to locate EntryNo-Assgn2.zip which corresponds to your Homework 2 submission.
3. Now, we will unzip EntryNo-Assgn2.zip which should create a folder named EntryNo-Assgn2 containing compile.sh, plot.sh, index_and_query.sh and Readme.txt and all your source files.
4. We will change the working directory to the unzipped directory EntryNo-Assgn2, then inside it, execute **compile.sh** followed by **plot.sh** and **index_and_query.sh** which should generate the required output files in the working directory.

Anti-Plagiarism Policy for Homework 2

Any detected attempts at plagiarism either from parallel/past submissions or the Internet will risk an F-grade in the course.