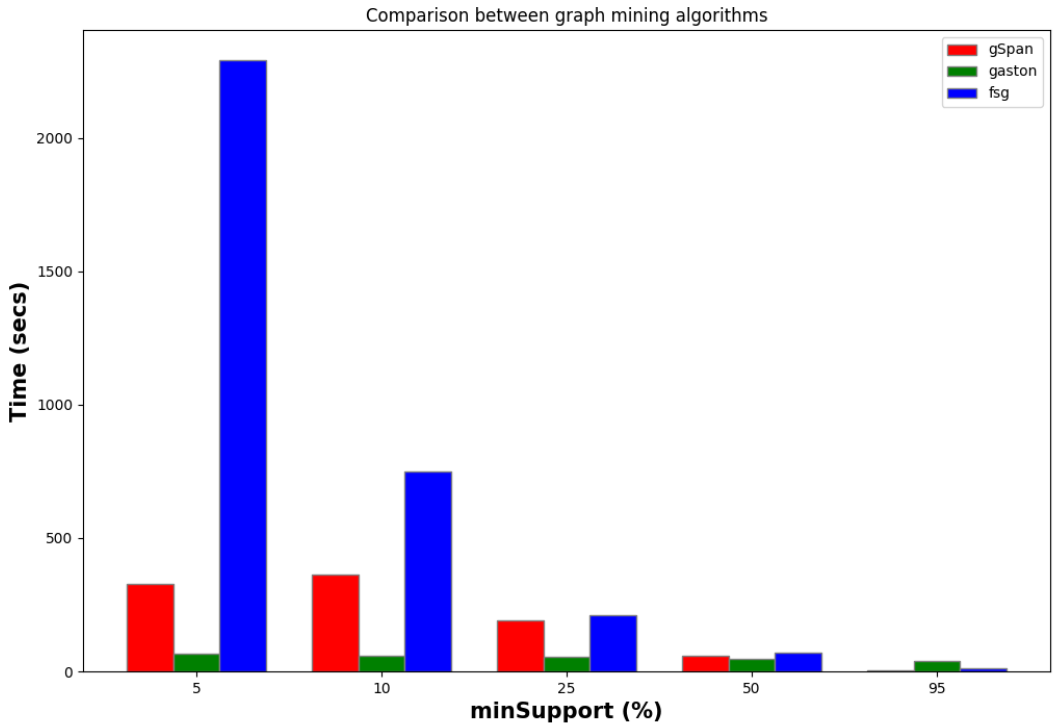
Homework 2
Group Name: Miners@2022

Bikram Mondal (2021MCS2127)
Harshit Verma (2021MCS2133)
Kritika Gupta (2021MCS2136)

Question 1:
Plot for comparison among gSpan, gaston and fsg algorithm.



Explanation:
We tested three subgraph mining tools, gSpan, FSG, and Gaston on the yeast dataset for varying supports and plotted their runtime. Our observation is as follows:
a.) **Gaston is the fastest and optimized even at lower threshold values among all** as it starts by mining paths and then mines subtrees and then proceeds to mine subgraphs with cycles. The expensive subgraphs isomorphism testing occurs only in the subgraph mining phase. Algorithm proceeds by generating cycles that are larger than previous ones.
b.) **FSG is slower than gSpan because gSpan uses DFS approach whereas FSG uses BFS.** The nature of gSpan's DFS-code prunes non-minimal to avoid

useless subgraph isomorphism tests and restricts addition of new nodes to DFS-tree whereas in FSG it goes through huge candidate generation phase and multiple database scans. This leads to highest growth rate in runtime with decreasing support threshold for FSG.

c.) **As the support increases the running time decreases exponentially.** Because low support means we allow more subgraphs to be mined, and this increases exponentially as seen in most distributions.

Question 2:

In this question, we have tried to create an index structure based on gdi-index.

How does it work?

For every graph we compute its canonical labeling and then create a node if it does not exist for the computed canonical labeling and then recursively call for all its subgraphs by removing one vertex each time to create a DAG.

And we also maintain a pointer for subgraphs to its parent graph.

One-vertex graph is linked to the starting null node.

But there can be a large number of different canonical labels. So we get a hash of the canonical label which leads to collision because different labels may hash to the same value. For all collided graphs we run subgraph isomorphism using vf2.

Question 3:

Step 1:

We have generated 3 datasets (For 2D, 3D and 4D) using the *generateDataset_d_dim_hpc_compiled* file by following the mentioned instructions.

Step 2:

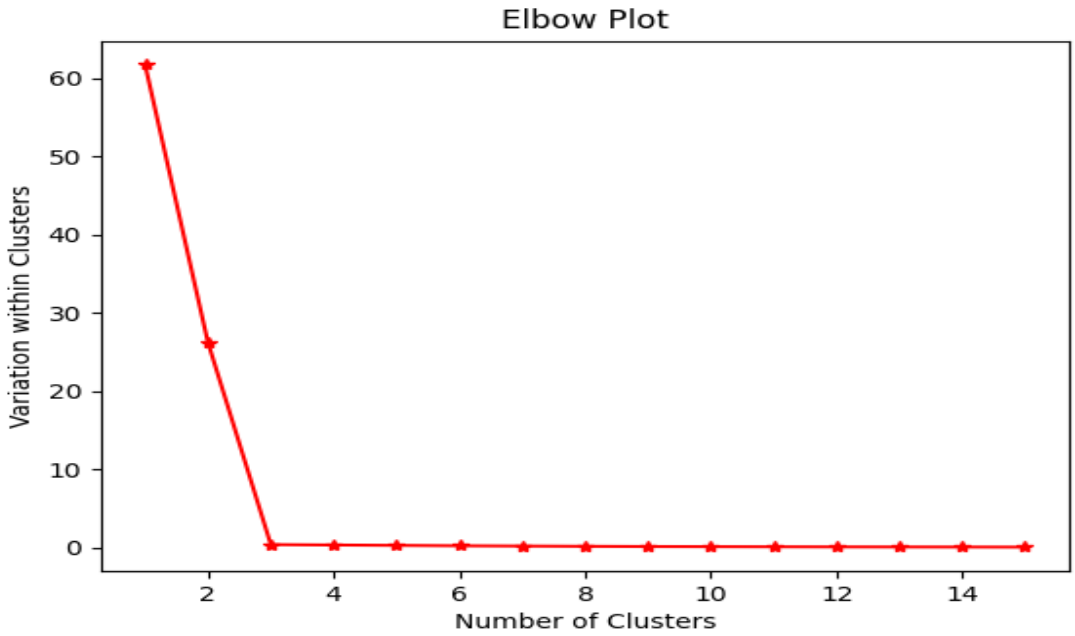
Written a python program (*kmeans.py*) that will take input dataset, dataset dimension and output plot file name as input, apply KMeans of scikit-learn library on the given dataset for k=1,2,….,15 and will generate the Elbow plot.

Step 3:

Written a generic script *elbow_plot.sh* that will run the *kmeans.py* file with the above mentioned 3 arguments and generate the Elbow plot.

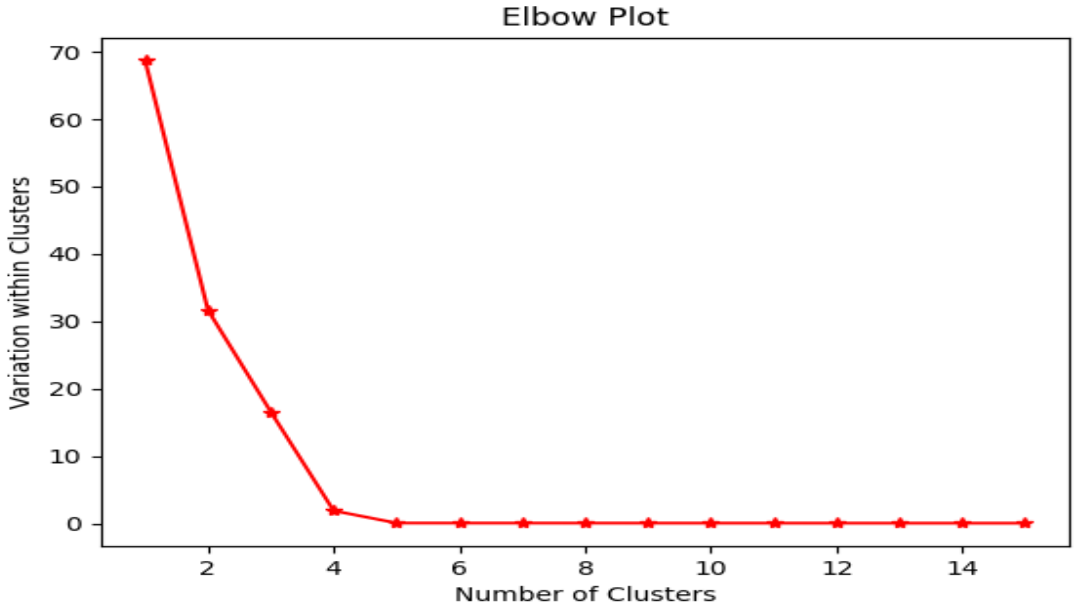
Elbow plots for each dimension and respective value of K:

- **2D Dataset:**



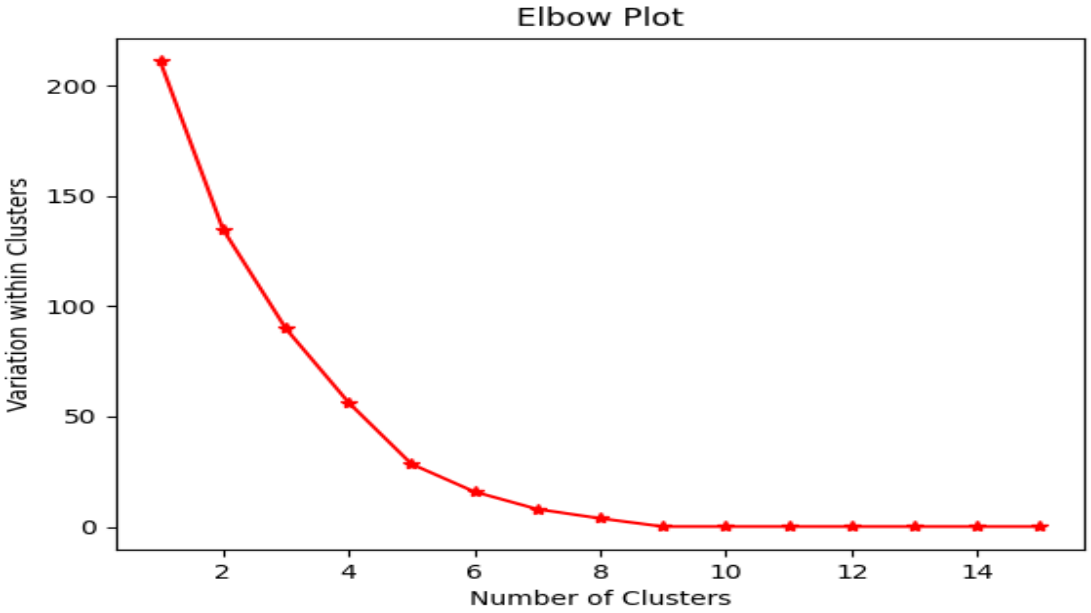
The value of K for the above plot: 3

- **3D Dataset:**



The value of K for the above plot: 4

- **4D Dataset:**



The best value of K for the above plot: 8

Explanation for the plots:

The X-axis denotes the number of clusters and Y-axis denotes the variation within clusters.

The variation within clusters decreases as the number of clusters increases just like we get in the plot. The reason is that in k-means the points are assigned to the closest centroid and the variation is calculated as the sum of squared differences of all the points to the centroid which they belong to. Now if k increases i.e no of centroid increases, no of points in the clusters decreases and only the closest points are assigned to that cluster. The clusters become dense. So, the sum of squared differences decreases i.e variation within clusters decreases.

To determine the value of K from the plot we may consider the plot as “arm” and the elbow will be the best value of K. Basically if the variation is 0 then that means there is no variation, it happens when every data point is a cluster. So we have to choose the value of K in such a way such that variation is very less (nearly equal to 0, but not 0). We have analyzed the plots in this way and determined the best value for K.