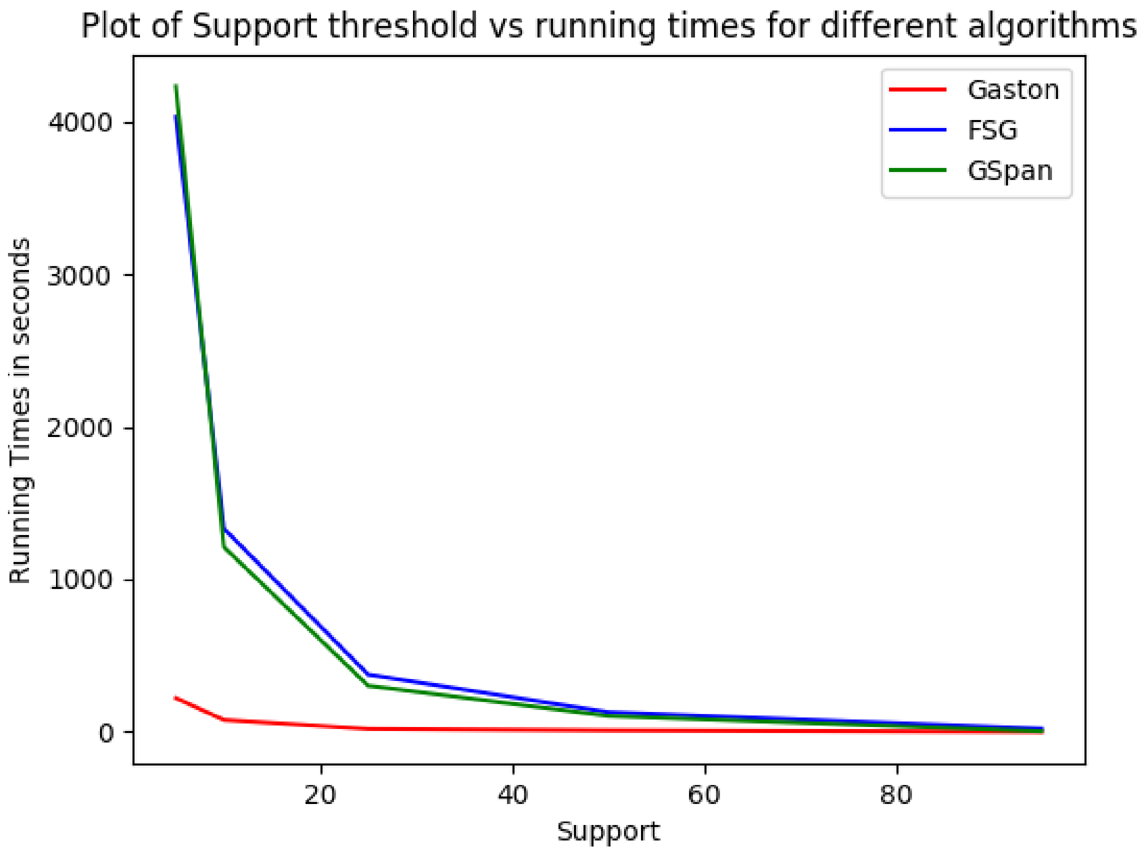
Explanation Q1:

Here we have plotted the running times against the support threshold of three frequent subgraph mining packages names gSpan, FSG (also known as PAFI), and Gaston.

Observations:

- From the graph we can see that running time is decreasing with increasing minimum support threshold. It is a common observation for all frequent subgraph mining algorithms. The reason is, when we increase the support threshold many of subgraphs generated becomes infrequent in each iteration and the algorithm don't need to go for any further computation on them reducing the overall computation cost.
- Running time is increasing exponentially with decrement of minimum support threshold for gSpan and FSG but it is not that severe in case of Gaston.
- If we compare Gspan and FSG there are no significant difference in performance, though Gspan is faster than FSG because FSG works on a breath first search strategy to generate frequent subgraphs which involves candidate generation and pruning. Candidate generation joining two subgraphs and pruning is computationally expensive because it involves subgraph isomorphism. GSpan uses a depth first strategy which doesn't require candidate generation. Gspan assigns a minimum DFS code to every generated candidate which is efficient in removal of duplicates. That is why Gspan faster than FSG.
- The Gaston is way faster than gSpan and FSG for the following reasons:
    - Gaston improves the depth first search strategy of the gspan by first searching the frequent paths and trees and followed trees with cycles.
    - Gspan maintains a global order of the cycles closing edges and adds those edges which are bigger in order than the previous one.
    - Gspan exploit the method of hashing to perform a presort. Then it finally does a subgraph isomorphism test to finally detect and prune out the duplicates. This strategy makes Gspan bery fast than others.

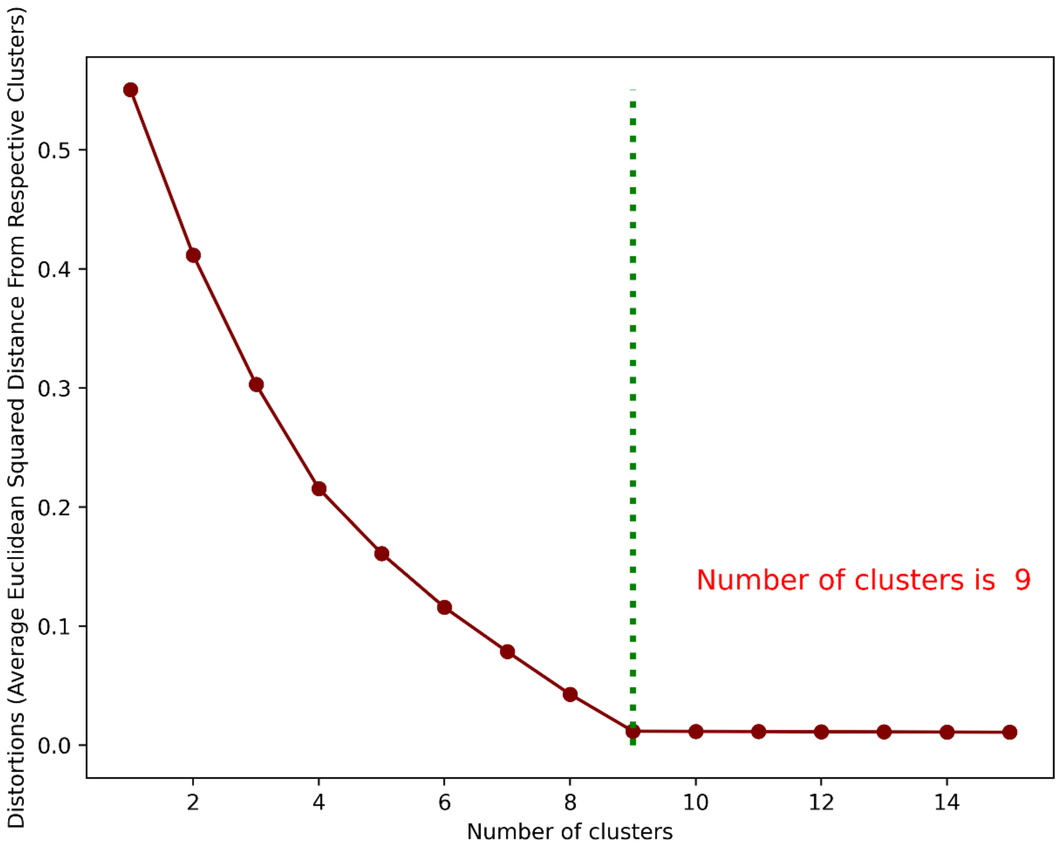

Plot of Support threshold vs running times for different algorithms
Gaston
FSG
GSpan
Running Times in seconds
4000
3000
2000
1000
0
20
40
60
80
Support

Explanation Q3:

In Q3 we have used an elbow plot to determine the number of clusters in the given dataset. Here we have plotted the distortions against the number of clusters in elbow plot. Distortions are the mean of Euclidean squared distances of all the data points from their respective cluster centers.

Observations:

- When we add a new cluster center to the data set, we get slightly less distortion than the previous.
- While adding new clusters to the data , if for a certain number of clusters (k), the decrement of distortion becomes very less than the previous decrement, and we can't see any significant decrement in the distortions in next few iterations we chose that number of clusters k as optimum.
- Here in the above plot we didn't see any further significant decrement in distortions after k = 9, so we decide there are 9 clusters in our dataset.


Distortions (Average Euclidean Squared Distance From Respective Clusters)
Number of clusters is 9
Number of clusters