Assignment 3

Vivek Kumar

(vivekkumar13@iisc.ac.in)

# Code Architecture:

- *spectralDecomp_OneIter* function takes the edge list $E$ as input and first computes the adjacency matrix $A$ and degree matrix $D$ of the graph consisting of edges $E$. After that it computes the eigen vector $v$ corresponding to the second smallest eigen value of $Lx = \lambda Dx$ where $L$ is graph Laplacian i.e., $L = D - A$. Then it divides the vertex set of the graph into two partitions based on the sign of elements of $v$.

- *spectralRecursive* function first calls the *spectralDecomp_OneIter* function to get the two partitions for the given set of edges. Then *spectralRecursive* function is called twice to recursively divide the partitions even further until the base condition is met.

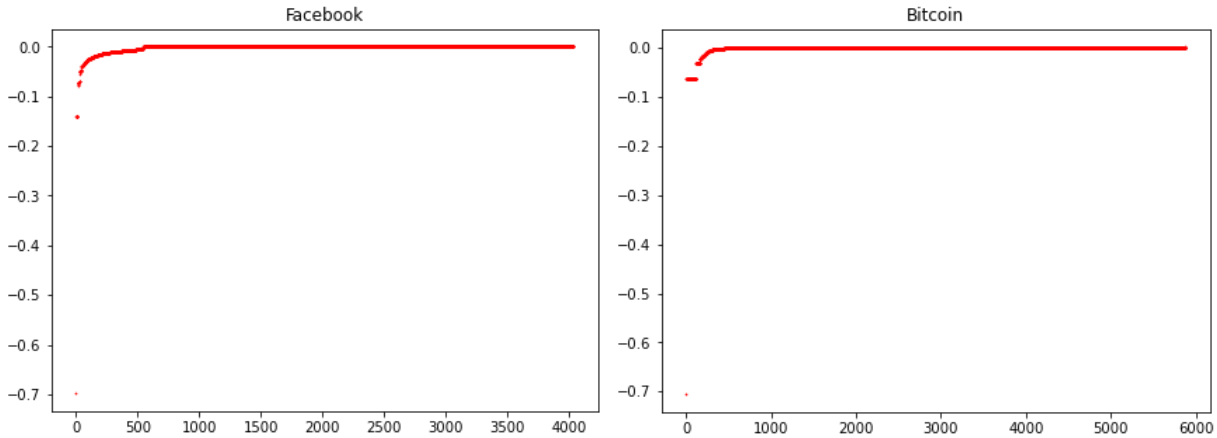
- *SpectralDecomposition* function does some pre-processing before calling *spectralRecursive* and after *spectralRecursive* returns, it does some post-processing to format the partition array.

- *deltaQ* function returns the change in Modularity if node $u$ is moved from partition $x$ to partition $y$ by using method the mentioned in Louvain Algorithm slides.

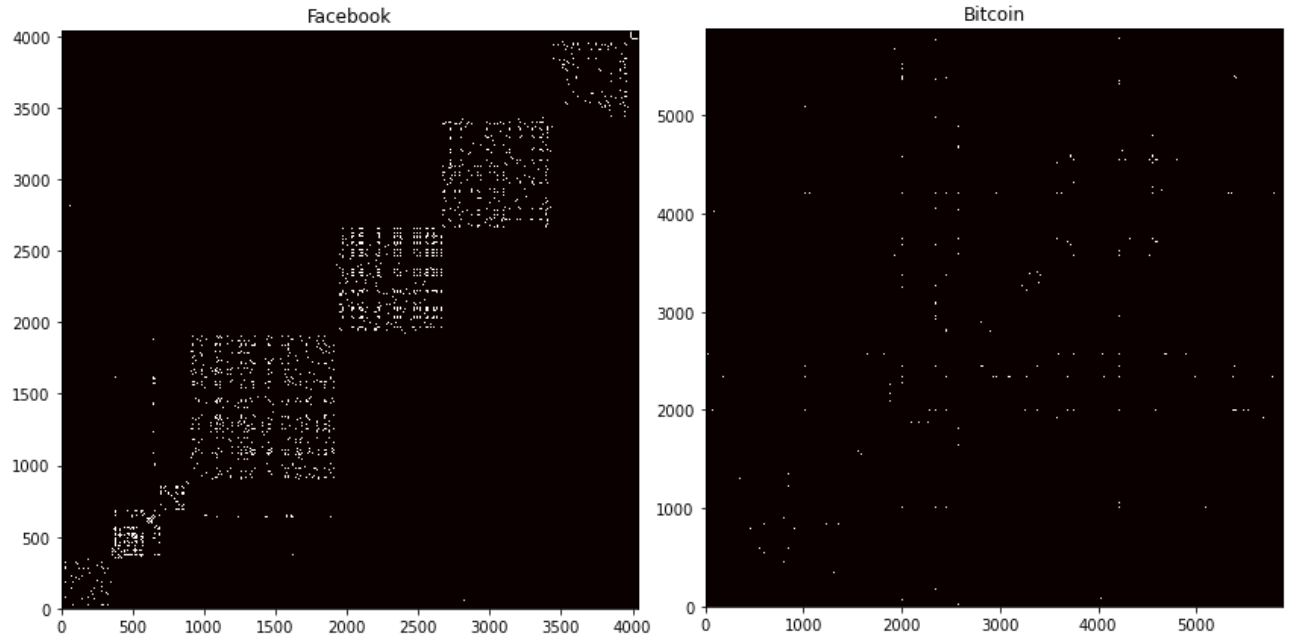
- *louvain_one_iter* function takes edge set as input and return partition array which represents the community Id in which the nodes belong to. Initially this function assigns each node to its own community. For each node $u$ it computes the modularity change using $deltaQ$ function if $u$ moves from current community to other community and assigns $u$ to the community which has largest modularity change. It repeats this process until the modularity is saturated.

# Question 1:

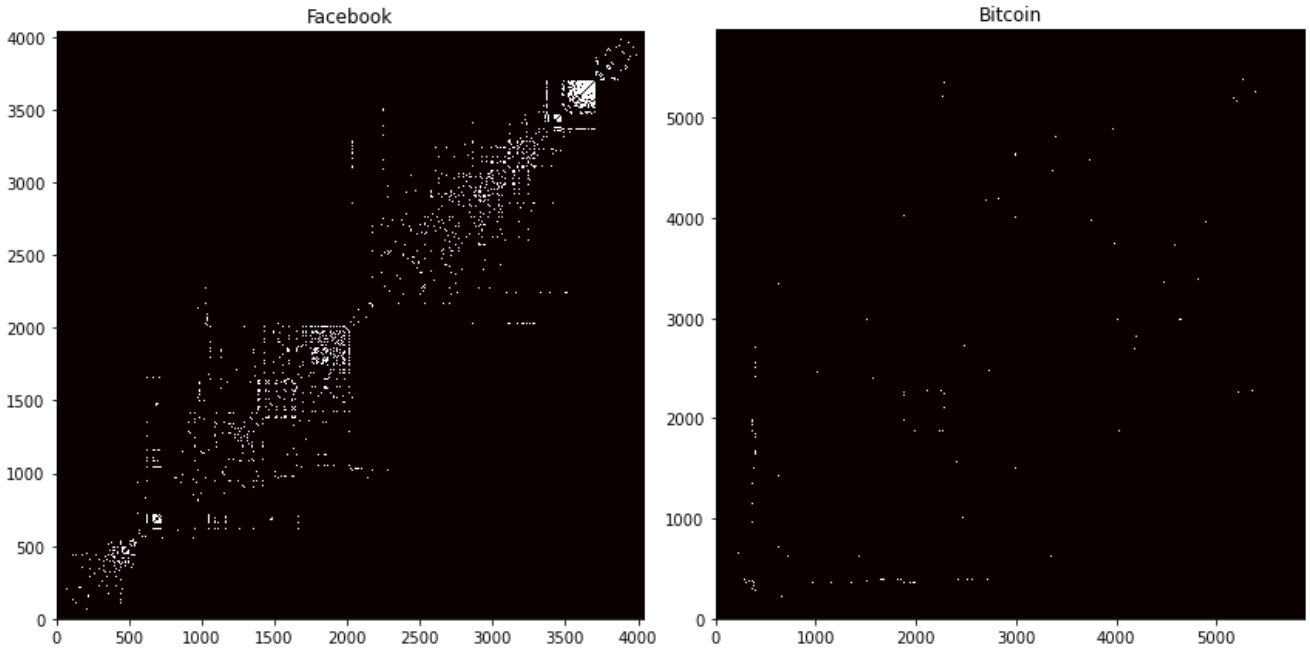
### Fiedler Vector:



**Adjacency Matrix:**




**Graph Partition [Iteration 1]:**

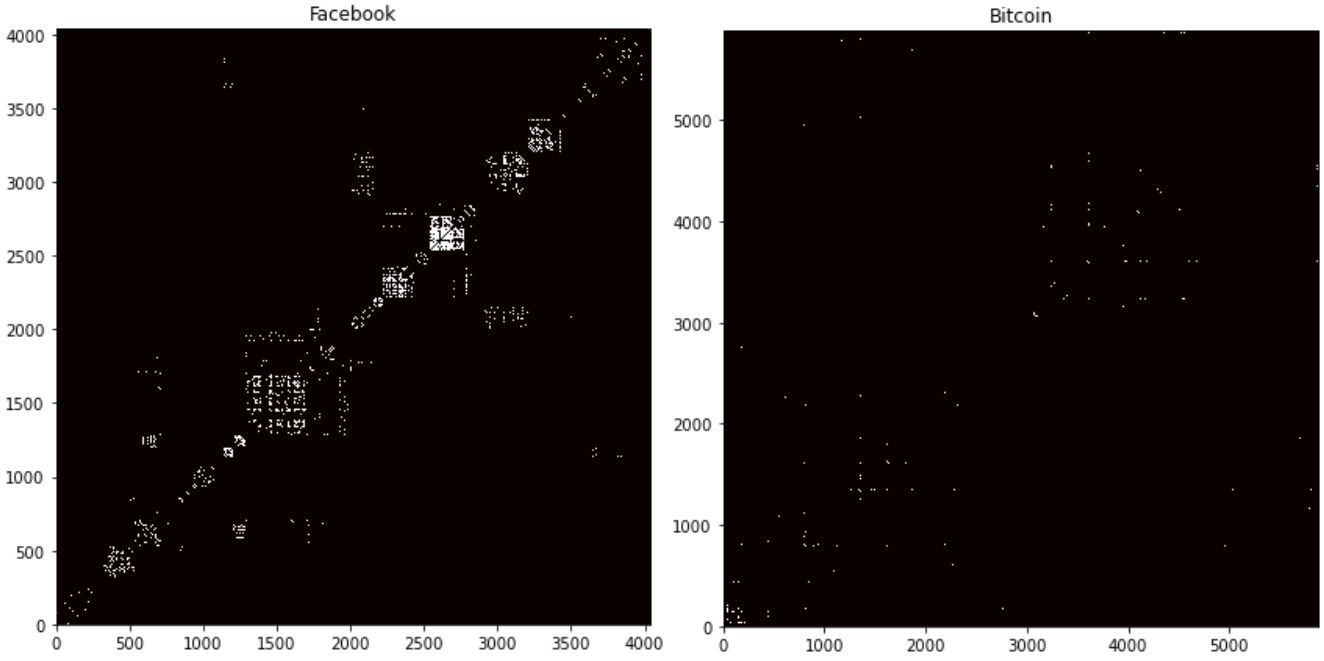



## Question 2:

I used the method as described in the Code Architecture section. To terminate the recursive process, I used normalized cut (from class slides) to determine whether the partition returned by *spectralDecomp_OneIter* function is balanced or not. If normalized cut value goes above 0.6 then I ignore that partition and return. I tried various values for normalized cut. For threshold higher than 0.6 It divide the clusters into more small clusters and vice-versa. Algorithm seems to give good partitions at the threshold of 0.6.

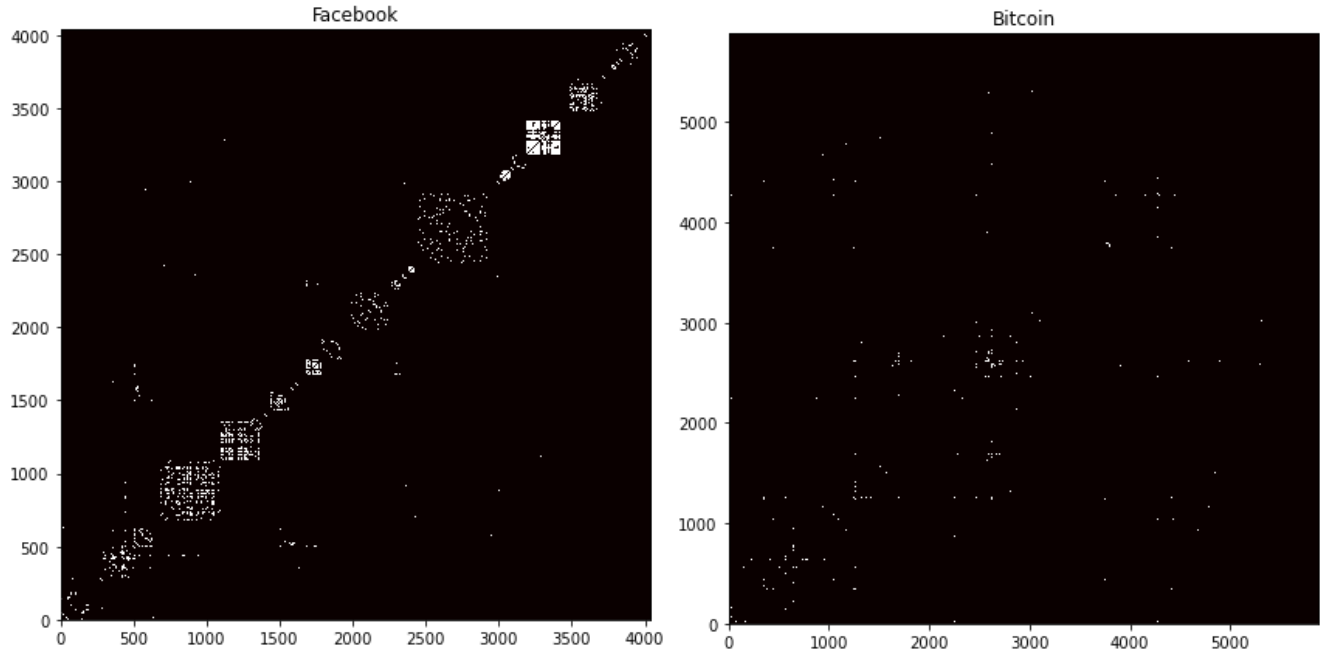
## Question 3 [Spectral Decomposition]:

Adjacency Matrix sorted by increasing order of community Id of partitions.




## Question 4 [Louvain Algorithm]:

Adjacency Matrix sorted by increasing order of community Id of partitions.



## Question 5:

I first assigned each node to its own community and computed the initial modularity for that using the formula mentioned in slides. Then for each node I computed the modularity change if node moved to other community. After that I assigned the node to the community which gives maximum modularity gain. I repeated this process until modularity is increasing. Since modularity is a measure of quality of decomposition into communities, therefore by maximizing modularity quality of decomposition is also improved.

## Question 6:

Spectral Decomposition Function Takes **34.67** seconds on Facebook dataset and **85.47** seconds on Bitcoin dataset.

Louvain Algorithm takes **41.17** seconds on Facebook dataset and **68.30** seconds on Bitcoin dataset.

## Question 7:

We cannot directly compare both algorithms. Spectral Decomposition can be a better choice for community detection when communities are clearly distinguishable i.e., there are very few edges joining any two communities compared to the edges within the community because Spectral Decomposition uses the idea of cut edges to determine the communities.

Louvain Algorithm can work well when communities are linked in some hierarchical fashion i.e., union of two or more communities form the bigger community because in each pass it tries to combine the communities from previous pass to form the bigger community.