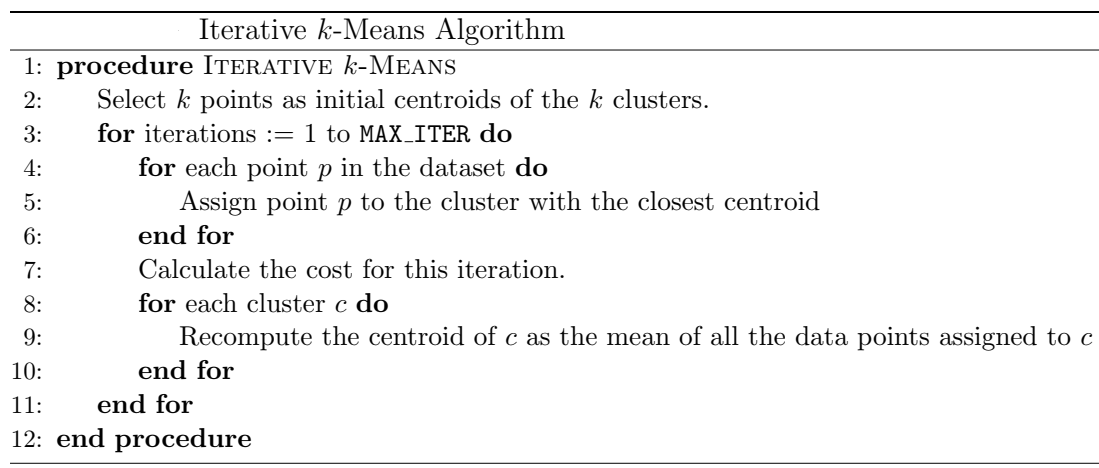
**K-Mean Clustering Algorithms**

Implement iterative k-means using Python. Please use the dataset attached for this problem:

1. data.txt contains the dataset which has 4601 rows and 58 columns. Each row is a document represented as a 58-dimensional vector of features. Each component in the vector represents the importance of a word in the document.
2. c1.txt contains k initial cluster centroids. These centroids were chosen by selecting k = 10 random points from the input data.
3. c2.txt contains initial cluster centroids which are as far apart as possible, using Euclidean distance as the distance metric. (You can do this by choosing 1st centroid c1 randomly, and then finding the point c2 that is farthest from c1, then selecting c3 which is farthest from c1 and c2, and so on).



Set number of iterations (MAX ITER) to 20 and number of clusters k to 10 for all the experiments carried out. Your driver program should ensure that the correct amount of iterations are run

**Problem A**

1. Use the Euclidean distance as the distance measure for clustering, and to compute the cost function SSE for every iteration i.



This means that, for your first iteration, you’ll be computing the cost function using the initial centroids located in one of the two text files. Run the k-means on data.txt using c1.txt and c2.txt. Generate a graph where you plot the cost function SSE(i) as a function of the number of iterations i=1...20 for c1.txt and also for c2.txt. You may  
use a single plot or two different plots, whichever you think best answers the theoretical questions we’re asking you about.

1. What is the percentage change in cost after 10 iterations of the K-Means algorithm when the cluster centroids are initialized using c1.txt vs. c2.txt and the distance metric being used is Euclidean distance? Is random initialization of k-means using c1.txt better than initialization using c2.txt in terms of cost function SSE(i)? Explain your reasoning.

***(Hint: to be clear, the percentage refers to (cost[0]-cost[10])/cost[0].)***

**Problem B**

Repeat the requirements 1 and 2 in problem but using cosine similarity as a measure similarity for clustering and the cost function as

***Note: in cosine similarity when its greater the points are closer (i.e. more similar), and thus when CSSE is bigger then its better.***

Is Euclidean distance K-Means better than cosine similarity K-Mean in terms of cost function? Explain your reasoning.