**Problem Statement:**

Given 10M datapoints in form of vectors, each vector is of dimension as 100. Return 10 closest vectors to the given query vector in less than 1 sec latency – Query Time. (Parameter: Euclidean Distance).

1. **Pseudo Code for Brute Force Solution:**

(Data for the problem can be generated using **datageneration.cpp** script)

Create an ifstream to load the data into RAM in dataset vector, initially data was stored in file

Generate query point using random function.

Create a distance vector (vector<int> distances),

Start the timer to measure query time, StartTime.

for i in range(0 to datasetsize):

calculate Euclidean distance between dataset[i] and query point and store at distances[i]

create topTen vector to record indices of closest 10 vectors

for i in range(0 to 10):

find minimum distance from distances vector and record that index in topTen

change value at minimum to INT\_MAX

Stop the timer, StopTime

QueryTime = StopTime – StartTime

Do this for 100 queries and find median

**Conclusion**: The median time found in Bruteforce method is around 17 sec.

1. **Pseudo Code for Priority Queue implementation:**

Create a ifstream to load the data into RAM in dataset vector, initially data was stored in file

Generate query point using random function.

Start timer, StartTime

Create the Priority Queue **pq** using first 10 elements of the dataset where each element of queue is a Pair<<int>, <int>> with first entry as distance between the query point and current data point and second entry is the index

For i in range(10 to datasetSize):

Find the distance of current point with query

If distance is less than the top of priority queue, Push the element in Queue

If Queue size is greater than 10, Pop the top of Queue

At last K smallest elements are in Priority Queue

Create topTen array to hold indices of closest 10 vectors

For i in range(0 to 10):

Save pq.top().second in topTen[i]

Pop the top element from priority queue

Stop the timer, StopTime

QueryTime = StopTime – StartTime

Do this for 100 queries and find median

**Conclusion:** The median time for this implementation is 16.01 sec, which is not much improvement from bruteforce.

* Both the algorithms are not fast, improvements needs to be done in the solution.