**Rohan D. Shah**

**A01943549**

**CS 5050 HW5**

**Technical analysis**

**Naïve brute force**

In the graph we can see that the values increase in the **exponential** manner. As it was expected, this is the slowest algorithm and **can handle points only upto 2^14 - 2^15.** After that it takes a considerable amount of time in execution.

**Slow DnC Approach:**

To overcome the shortcomings of brute force we use the DnC where we **first sort the values on the basis of X** and then compute the minimum distance using DnC. It is fairly faster than the brute force approach for all the three distribution and can **handle a high number of points, close to 2^21**. **Its complexity is nLogn**. This algorithm will only work well if the **sorting** is done by an **algorithm** which has the complexity of **nLogn and not n^2.** Otherwise it will behave the same way as the naïve algorithm.

**Fast DnC Approach:**

Even thought the Slow DnC is fairly fast, we can further optimize that algorithm by sorting the points on the basis of Y where we **first sort the values on the basis of X** and then compute the minimum distance **using DnC and sort on Y.** It is faster than the brute force approach and can handle a **high number of points, close to 2^21. Its complexity is nLogn.** This algorithm will only work well if the **sorting** is done by **an algorithm** which has **the complexity of nLogn and not n^2**. Otherwise it will behave the same way as the naïve algorithm. It gives almost the **same performance as the slow DnC for randomly distributed** points but **leads the slow DnC with a huge margin when it comes to the hexagonal and mixed distribution. This might be because the points are evenly distributed in the latter two distributions.**

**Does different distribution make any difference in the performance of the algorithms?**

**Naïve algorithm:**

**No** it does not. Since the algorithm **computes distance for every point, it does not matter where the points are placed.** So if the points are clustered in one place or distributed uniformly, it does not care and the performance remains almost the same. This is evident in the graph in the sheet 4 of the empirical analysis 1 file.

**Slow DnC algorithm:**

Here we see a **considerable difference in the performance between the various distribution patterns.** The algorithm is **faster when the data is randomly distributed than when it is hexagonally distributed.** The hexagonal and mixed distribution performance is almost the same. This might be happening because the points are randomly distributed. This is because **if the points are more in one place and less in the other the dividing becomes faster and the algorithm does not have to calculate distance between more points. Since the points are sorted, more points will be ignored when randomly distributed.** Whereas in the hexagonal distribution the points to take into consideration will be the same or more as compared to random distribution as it will be uniformly arranged.

**Fast DnC algorithm:**

Here we see a **some difference in the performance between the various distribution patterns but not as much as slow DnC.** The algorithm is **faster when the data is randomly distributed than when it is hexagonally distributed.** The hexagonal and mixed distribution performance is almost the same. This is because if the points are more in one place and less in the other the dividing becomes faster and the algorithm does not have to calculate distance between more points. Since the points are sorted, more points will be ignored when randomly distributed. Whereas in the hexagonal distribution the points to take into consideration will be the same or more as compared to random distribution as it will be uniformly arranged. **The reason why the algorithm is less faster because it also has to sort on Y which the slow DnC does not and hence the difference between the performance is not as evident as the slow DnC between the various distributions.**

**Since the algorithms behaved a little faster in random distribution, they were able to handle more number of points when the distribution was random.**

**The performance of fast DnC is almost the same as slow DnC in random distribution.**