ANALYSIS OF ALGORITHMS HOME WORK2

NAME: ABDULSALAM YAZID
STUDENT NUMER:150160927

NOTE: Even though my code works well on my computer, the code might give an incorrect result for 25k on the itu server, so PLEASE TRY RUNNING the code for 25k on your computer.

Question1.

Masters theorem is a method or technique use to analyze, examine recursive type relations. Intuitively it is used to compute the running time of an algorithm that calls itself.
"a"is the number of the sub problems, simple when a problem is divided into smaller sub problem, "a" represents the how many of the sub problems we have. On the other hand, "b" represent how much the size of the current problem is compared to the original problem

Question2.

Step1:I define a point class which has property x,y,z.
Step2: I start by sorting the points. These points are first sorted base on their x coordinate (Px), then I sorted the points again but this time base on their y coordinate(Px), lastly I sorted the points base on their z coordinates(Pz).
Step3: I then define a point(midPoint) which is the middle point of Px.
Step4: I then define two point arrays. I called them left and right sub arrays.
Step5: I check every point every point in Px, if the point is less than midpoint, I put it in the left sub array, else I put it in the right sub array.
Step6: I recursively find the minimum distance in both the left and right sub arrays and find the min of them, lets call it "delta".
Step7: *Due to the fact that a minimum distance can actually be between two points one in a left array and the other in the right sub array.* I create an area around the sub arrays midpoint, and this area(subArrayMinArea) just has a distance less than delta to the midpoint line from either the left or right sub array.
Step7: I find the minimum point in subArrayMinArea. If this minimum is less than delta, I return this minimum else I return delta as the minimum.

Question3.

```
Closest-Pair(P)
        Construct Px,Py,Pz
        Sort Px,Py,Pz                                          O(nlogn) Time
        (p0,p1)= ClosestPoints (Px,Py,n)

ClosestPoints(Px,Py,n)
 If |P|<=3
       Brute force and find the nearest two distances.
Endif

Construct midpoint
Construct   leftSubArray,RightSubArray                    O(n)   Space
```

```
For I ←0 to n                                                    O(n)   Time
   If Point in Py is less than midpoint
       Put the point to leftSubArray
   Else
       Put the point to RightSubArray

(q0,q1)= ClosestPoints (Px, leftSubArray,mid)
(r0,r1)= ClosestPoints (Px+mid, RightSubArray,n/2)               2T(n/2)  Time


δ=min(d(q0,q1),d(r0,r1))
x*=max x-coordinate of a point in set Q
L={(x,y): x=x}
S: Points in P within distance δ   of L
Construct Sy                                                     O(n) Time ,O(n) Space
For each point s in Sy, compute distance from s
                        to each of next 15 points in Sy.
Let s,s'   pair achieve the min distance

If d(s,s')< δ   then return d(s,s')      //d is the distance between them
Else if d(q0,q1)<d(r0,r1) then return d(q0,q1)
Else return d(r0,r1)
```

Time complexity = O(nlogn)+O(n)+ O(n)+ O(n)= O(nlogn) // because O(nlogn) is the dominant factor

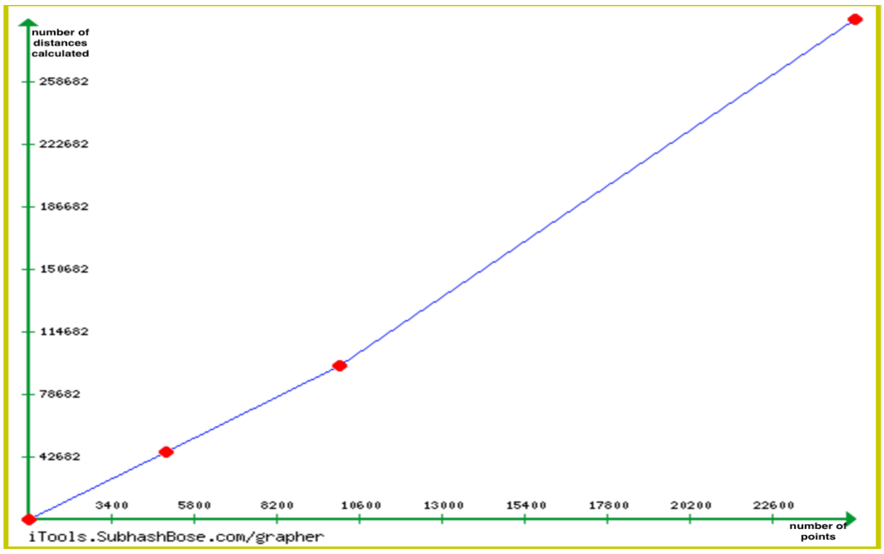
Space complexity = O(n) + O(n) = O(n)

Recurrence relation

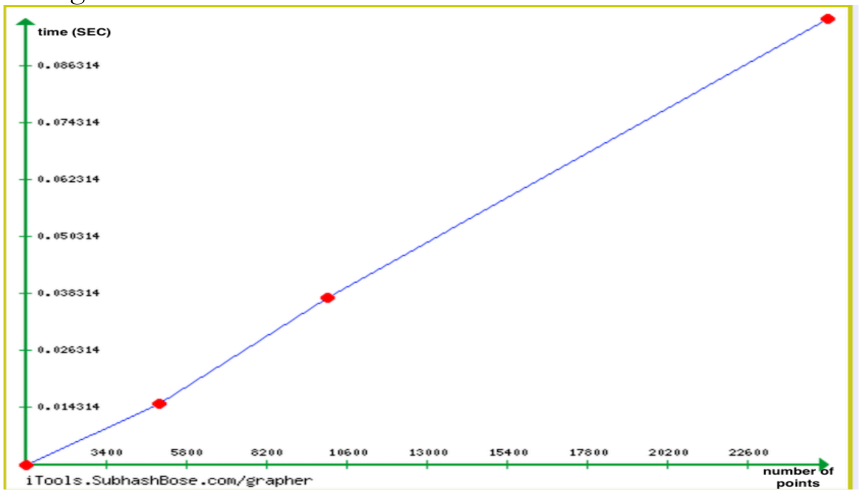
T(n) = 2T(n/2) + n   n>3   // because we brute force when n is <=3
T(n) = n   n<=3

Question4

a. After running the program and obtaining the number of distances for each Input file. I plot every number of coordinates to number of distances calculated, the result was quite astonishing because the growth of number of distances calculates was linear, this is not surprising because we use a divide and conquer approach which saves time from a lot more distances calculations if we had used a brute approach. The below graph is the plot of distances to their corresponding number of distances calculated



b. The analysis in timing resembles the analysis in this part. Here for every number of points the running time was gotten, I realized that the timings don't actually increase much, here also I decided to plot every number of points to the running time just to understand and have a deep insight in what's really happening. The result was not surprising that I got a linear growth, just as in first part due to the fact that a divide and conquer approach was used. The table below gives a better insight in comparing the number of points to the running time.


time (SEC)
number of points
iTools.SubhashBose.com/grapher