BLG336-E PROJECT REPORT

Project No	2
Student	Alperen KANTARCI – 150140140
Instructor	Zehra Çataltepe
Date	20.04.2018

1. Introduction

As stated in description of project, we are asked to solve closes pair problem in 3 dimensional space.

2. Development and Environment

The application is written on latest Arch Linux operating system with using Atom text editor and Clion IDE. Program compiled at Arch Linux and ITU SSH server successfully. The important note that in order to compile following command should run. **g++ -std=c++11 150140140.cpp**

3. Questions

3.1-) In the cases where we solve sub problems and combine them to solve one big problem we use generally divide and conquer methods and master theorem help us to calculate running time of these kind of algorithms. a is the number of the sub problems where each subproblem is b times smaller than original problem. So dividing to what gives us b and how many sub problem do we get after diving gives us a.

3.2-) In order to create better data structer i created a struck which is called Ball. This struct holds x,y,z values of the each balls. First i sort the balls according to the x values then divide them equal sizes. Solving for each sub part gives us the correct result at the end. So for each sub part we continue to divide and calculate. After getting two minimum distance from each side we are looking to the closest pairs across the mid line since we didn't consider them when dividing. So after this calculation we find the closes pair distance.

3.3-) The main idea of the algorithm that divide the 3D space from half and calculate each half recursively. Then at the and create an calculation space at the middle and check if this new space contains closer pairs than either right or left sides. It uses same algorithm that we learned in lecture and only difference is the ecludian calculation and space creation on the middle since the algorithm that we learned was on 2D space.

For calculating the recurrence, we divide to half in every recursive call and after dividing we calculate the plane in O(n) , since i sort the y values at each call of minOfPlane function it takes O(nlogn). closestPair implementation finds in O(n) so recurrence is:

$T(n) = 2T(n/2) + O(n) + O(n\log n) + O(n)$

$T(n) = 2T(n/2) + O(n\log n)$

$T(n) = T(n\ log^2 n)$

So time complexity of my algorithm is $O(n\ log^2 n)$

```
main:
GET data text name
READ data and create object array to store
SORT the array according to the x
PRINT the running time and number of calculations
FINISH
```

```
closestPairRec:
IF there is only 3 or less ball
   CALL closesPair for these balls
ENDIF
FIND the ball in the middle according to the sorted array
CALL closesPairRec for left side of the middle
CALL closesPairRec for right side of the middle
GET minimum distance from the called functions.
CREATE new plane in the middle according to the minimum distance
RETURN result of the minOfPlane function for newly created plane.
```

```
closestPair:
SET min as infinity
FOR every ball in the list that given
   FOR every ball that is ahead of the current ball
      IF distance between two ball is smaller than current min
         SET min as new distance
      ENDIF
   ENDFOR
ENDFOR
```

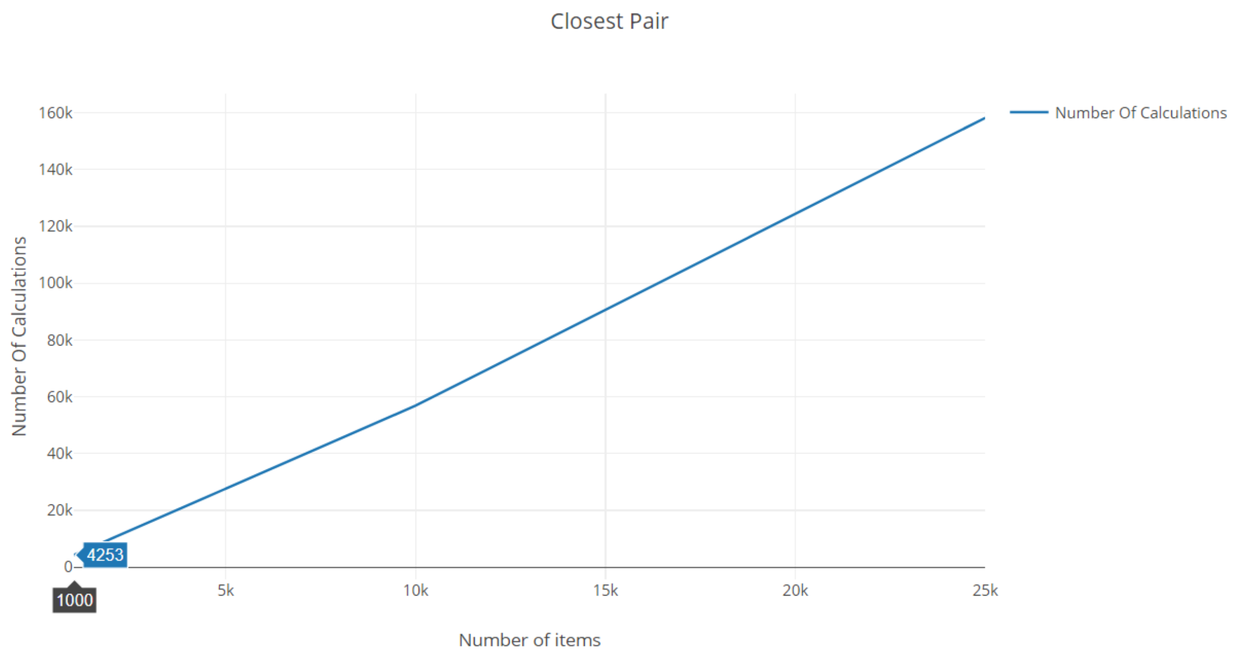
```
minOfPlane:
SORT current ball list according to the y axis
SET minimum as minimum distance within two sides
FOR every ball within the threshold set by minimum (plane)
    CALCULATE distance between the balls
    IF distance between two ball is smaller than current min
         SET min as new distance
    ENDIF
ENDFOR
```

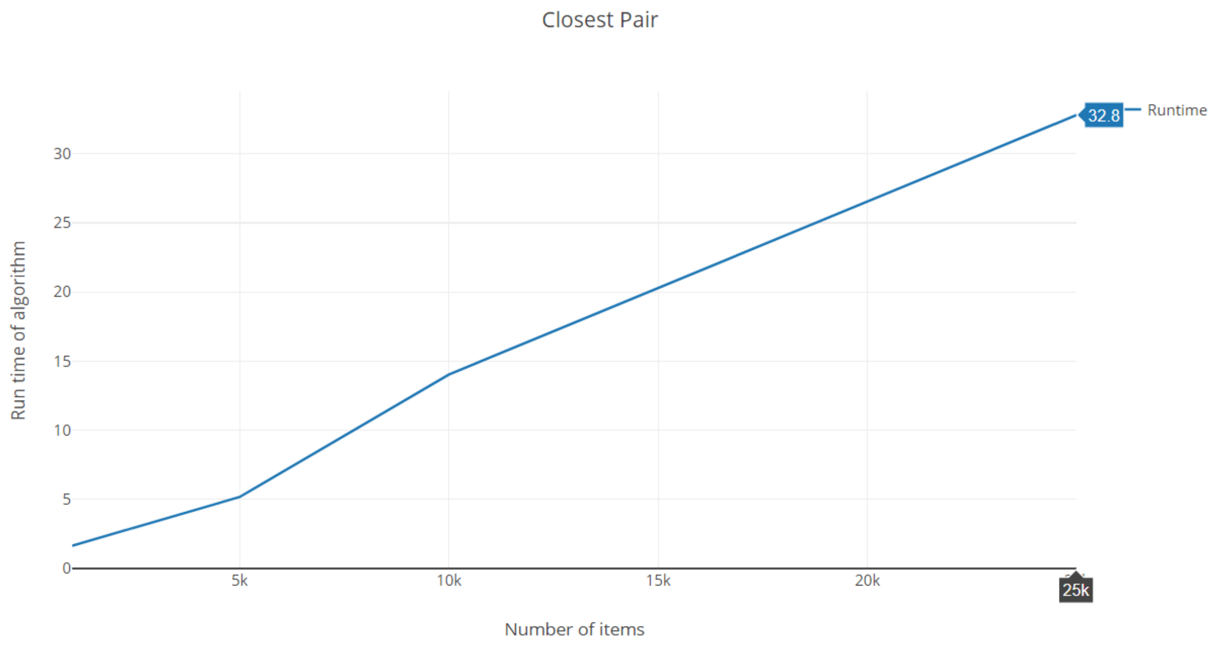
```
ecludianDist:
INCREASE numberOfCalculation
RETURN ecludian distance in 3D space between two balls
```

3.4-)

Number of Points	**Number of Calculations**	**Running Time**
1000	4253	1.661 ms
5000	27661	5.177 ms
10000	56883	14.036 ms
25000	158227	32.800 ms

As you can see the graphs at the below algorithm shows nearly linear time run time and calculations. But it is O(n*logn) as I calculated the algorithm. Both calculations and running time proves it.


Closest Pair
Number Of Calculations
160k
140k
120k
100k
80k
60k
40k
20k
5k
10k
15k
20k
25k
Number of items
Number Of Calculations


Closest Pair
Run time of algorithm
Runtime
5k
10k
15k
20k
25k
Number of items