BLG336E - Analysis of Algorithms II

2017-2018 Spring, Project 2 Report

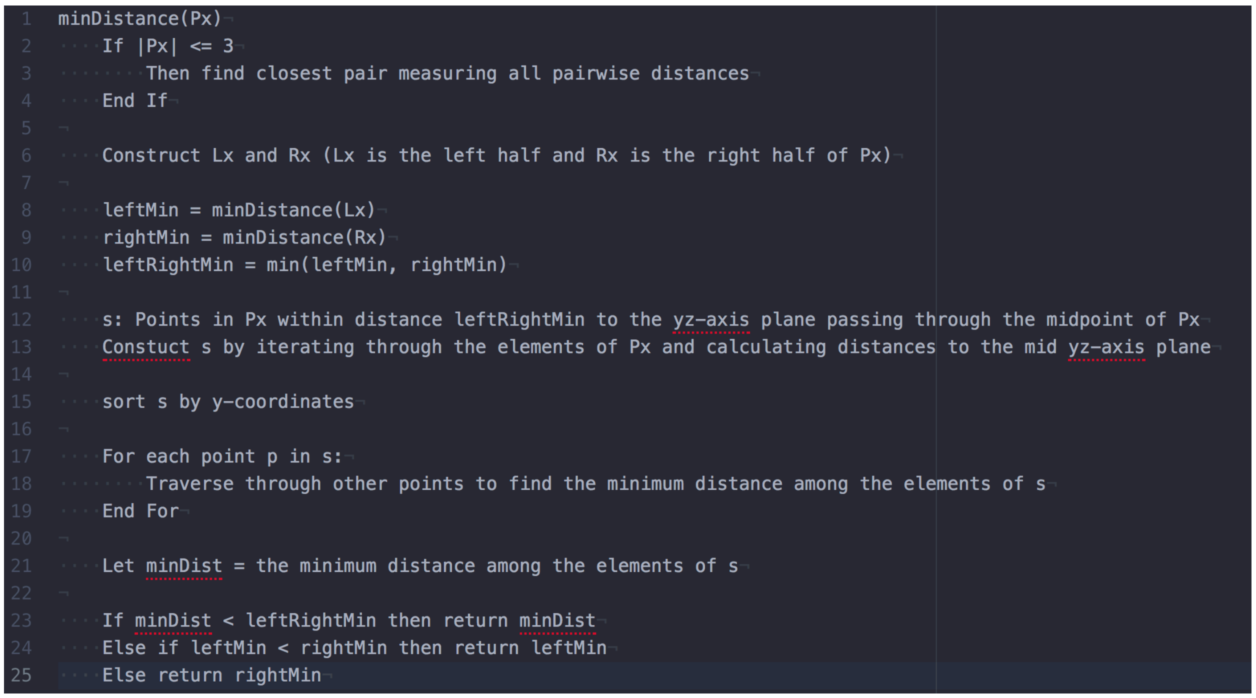
Onat Şahin - 150150129

NOTE: My code uses the function stoi(), which is a C++ 11 function. Therefore, my code needs to be compiled with the command “g++ -std=c++11 150150129.cpp”.

1) Master theorem is an asymptotic analysis method to find the complexity of recurrences, which are generally used for divide and conquer approach. In the given recurrence, (a) means how many recursive calls are made and (b) means 1/b of the input list is sent to recursive calls.

2) First of all, the given list of balls is sorted by their x coordinate. A recursive function is used to implement a divide and conquer approach. This function uses two recursive calls with different halves of the input list. The intuition behind this approach is to find the closest points in different halves of the input. As more recursive calls are made, the input of these calls become smaller. For inputs with a size smaller than 4 elements, the closest pair is found by measuring all pairwise distances. After two recursive calls return the left minimum and the right minimum, minimum of these two values are found. Then, it is needed to check the distances between balls in different halves. For this, balls which are closer to the middle yz-plane of the space than the previously found minimum distance are stored in a different list. By measuring pairwise distances in this list, the middle minimum is found. It is proven that if this list is sorted by the y coordinate, finding the minimum will take O(n) time. Finally, the minimum of left minimum, right minimum and middle minimum is returned as the minimum distance between 2 different balls in the space.

3) Pseudocode of my minimum distance algorithm:



```
minDistance(Px)
    If |Px| <= 3
        Then find closest pair measuring all pairwise distances
    End If

    Construct Lx and Rx (Lx is the left half and Rx is the right half of Px)

    leftMin = minDistance(Lx)
    rightMin = minDistance(Rx)
    leftRightMin = min(leftMin, rightMin)

    s: Points in Px within distance leftRightMin to the yz-axis plane passing through the midpoint of Px
    Constuct s by iterating through the elements of Px and calculating distances to the mid yz-axis plane

    sort s by y-coordinates

    For each point p in s:
        Traverse through other points to find the minimum distance among the elements of s
    End For

    Let minDist = the minimum distance among the elements of s

    If minDist < leftRightMin then return minDist
    Else if leftMin < rightMin then return leftMin
    Else return rightMin
```

Complexity:

- This algorithm makes two recursive calls with an array half the size of the input array.
- In line 6, the whole array is traversed, which is in O(n).
- In line 13, the whole array is traversed and minimum is found. This operation is in O(n).
- In line 15, the list s is sorted. This operation is in O(nlogn)
- It is proven that because of the size of s and s being sorted by y-coordinates, the for loop between the lines 17 and 19 take O(n) time.

Therefore if the complexity of this algorithm is T(n):

T(n) = 2T(n/2) + O(n) + O(n) + O(nlogn) + O(n)

Since O(nlogn) dominates O(n), we can write the recurrence relation of the algorithm as:

T(n) = 2T(n/2) + O(nlogn)

If we use a recursion tree, we can see that the algorithm divides logn times. Therefore the complexity is:

T(n) = $O(n \log n \log n)$

4) The table below shows the runtime and total number of distance calculations for the given input files. Runtime values here are the result of observing the runtime for each files 5 times and taking average. These observations are done in my computer, due to ssh not being precise in measuring time.

Input File	Runtime (miliseconds)	Total number of distance calculations
1000	2.503	5481
5000	15.844	34085
10000	29.346	70086
25000	71.387	192217

Looking at the values, it can be seen that the changes between different files closely represent the given complexity, which is $O(n \log n \log n)$. For example, we expect the runtime and number of calculations when 5000 balls are used to be more than 5 times of the corresponding values when 1000 balls are used(because of the algorithm's complexity). This expectation is fullfilled by the results. This relation can also be observed between the other results.