CSE204 : Offline 8 Report

Submitted by :

ID : 1805012

Complexity Analysis:

```cpp
distance_index_pair calculate_distance2(point *points_x, point* points_y,int st,int en){
    if( st > en )
        return {inf,-1,-1,inf,-1,-1};
    if( st == en ){
        points_y[st] = points_x[st];
        return {inf,-1,-1,inf,-1,-1};
    }
    if( en-st == 1 ){
        for(int i=st;i<=en;i++)
            points_y[i] = points_x[i];
        merge_sort(points_y+st,2,cmp_ymajor);
        return {distance(points_x[st],points_x[en]),points_x[st].index,points_x[en].index,inf,-1,-1};
    }
    if( en-st == 2){
        for(int i=st;i<=en;i++)
            points_y[i] = points_x[i];
        merge_sort(points_y+st,3,cmp_ymajor);

        distance_index dist_ind[3] ;
        int id = 0;
        for(int i=st;i<=en;i++)
            for(int j=i+1;j<=en;j++)
                dist_ind[id++] = {distance(points_x[i],points_x[j]),points_x[i].index,points_x[j].index};
        merge_sort(dist_ind,3,cmp_distance_index);
        return {dist_ind[0],dist_ind[1]};
    }

    int mid = (st+en)/2;
    distance_index_pair left = calculate_distance2(points_x,points_y, st, mid);
    distance_index_pair right = calculate_distance2(points_x,points_y, mid+1, en);
    distance_index dist_ind[4] = {left.di[0],left.di[1],right.di[0],right.di[1]};

    merge_sort(dist_ind,4,cmp_distance_index);
    double k = dist_ind[1].dist;
```

Let’s say the time complexity of the function is T(n).

The first 4 if blocks take Θ(1) time.

There are two recursive calls to the function, which costs T(n/2) each.

```
113
114     int I = mid;
115     int J = en;
116     int i = st,j=mid+1;
117     int id = st;
118     int n = 0;
119     while( i<=I || j<=J ){
120         if( i>I )
121             points_y_aux[id++] = points_y[j++];
122         else if( j>J )
123             points_y_aux[id++] = points_y[i++];
124         else if( cmp_ymajor(points_y[i],points_y[j]))
125             points_y_aux[id++] = points_y[i++];
126         else points_y_aux[id++] = points_y[j++];
127     }
128     for(i=st;i<=en;i++)
129         points_y[i] = points_y_aux[i];
130
131     distance_index strip_min = {inf,-1,-1};
132     i = st;
133     while( i<=en ){
134         if( abs(points_y[i].x-points_x[mid].x) < k ){
135             strip[n] = points_y[i];
136             side[n] = (points_y[i].x<=points_x[mid].x) ;
137             n++;
138         }
139         i++;
140     }
141
142     for(i=0;i<n;i++){
143         for(j=i+1;j<n && (strip[j].y-strip[i].y) < k; j++ ){
144             if( (side[i]^side[j]) == 1 ){
145                 double temp_distance = distance(strip[i],strip[j]);
146                 if( strip_min.dist > temp_distance){
147                     strip_min.dist = temp_distance;
148                     strip_min.i = strip[i].index;
149                     strip_min.j = strip[j].index;
150                 }
151             }
152         }
153     }
154
155     if( strip_min.dist < dist_ind[3].dist ){
156         dist_ind[3] = strip_min;
157     }
158     merge_sort(dist_ind,4,cmp_distance_index);
159     return {dist_ind[0],dist_ind[1]};
160 }
```

Lines 114 – 129 merges the ‘points_y’ variables, which is Θ(n)

Copying them to form valid strip takes Θ(n).

Lines 142 – 153 calculates the minimum distance in the strip. It takes Θ(n), though it has a nested loop. The value of ‘j’-’i’ is at most 7, if the points are distinct. So it is Θ(n).

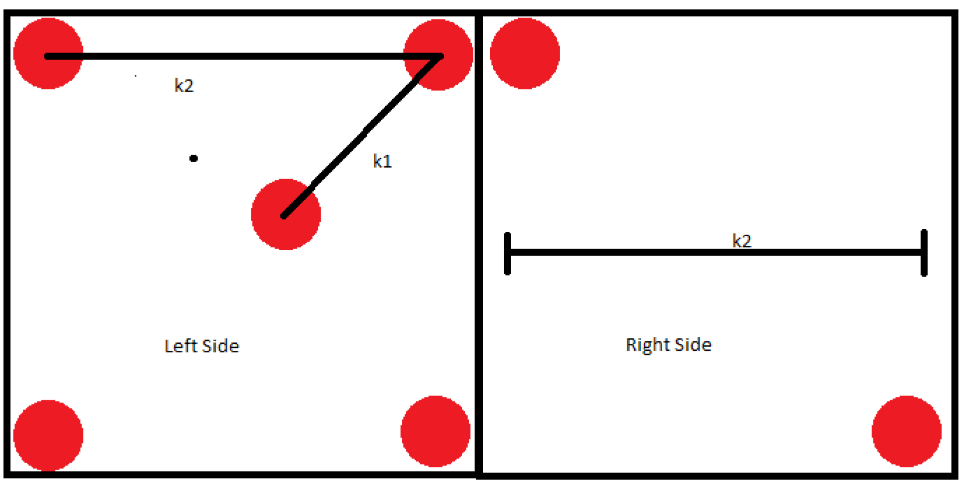
Let’s say, the distances returned by left side is k1,k2, where k1<k2 and the distances returned by right side are k3,k4, where k3<k4.

Case 1: if k1 is the minimum distance and k2 is the second minimum value then

There is only one pair in left side whose distance is less than k2.

All the pair distances in the right side are greater than k2.

So, for a point common in left side and the strip,



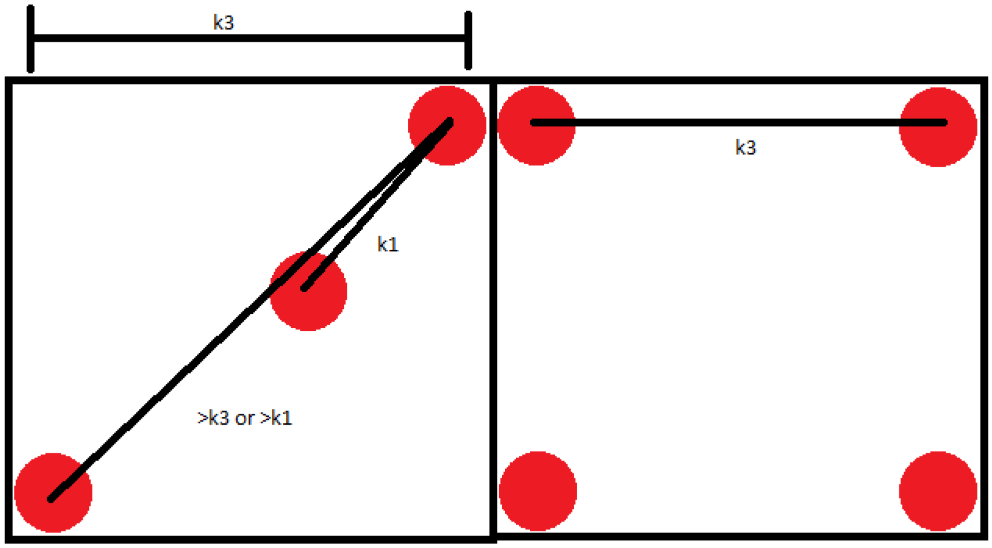
At most 7 points, 5 points on the left, and 2 on the right.

The same is valid for 2 smallest distances on the right side.

Case 2: if k1 is the minimum and k3 is the second minimum.

There is only one pair in left side whose distance is k1.

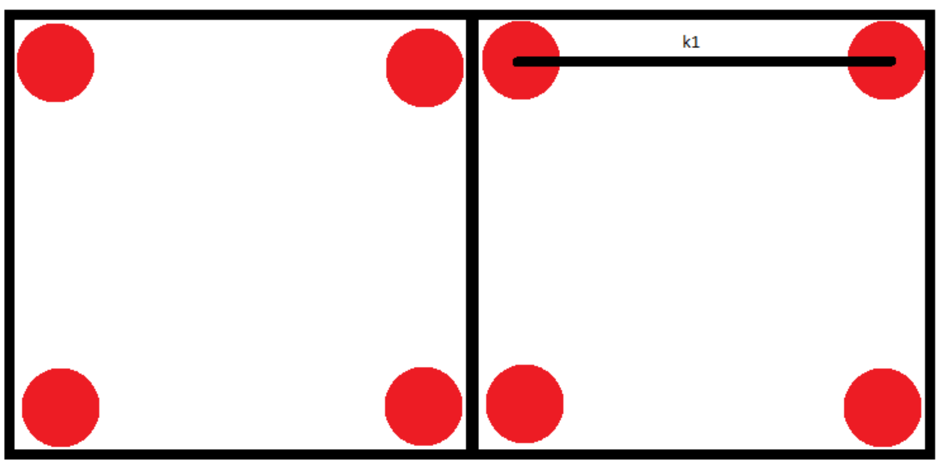
No pair distance of k3 exist in the left side.



At most 7 points, 3 points on the left and 4 on the right.

The same also valid if k1,k2 is on the right and k3,k4 on the left.

Case 3: if k1 and k3 are equal.



Also at most 7 points.

Lines 155-159 takes Θ(1).

So, Total time complexity , $T(n) = \Theta(1)+ 2T(n/2)+\Theta(n)+\Theta(n)+\Theta(n)+\Theta(1)$

or, $T(n) = 2T(n/2)+\Theta(n)$

Using Master method, a = 2,b = 2, f(n) = Θ(n), $n^{\log_2 2}$ = n

So, T(n) = Θ(nlogn)