Pseudo-code

```
float cloest_cross_pair(point* middle,float dmin,int size){
    float fd=dmin;
    for(int i=0;i<size;i++){
        int j=i+1;
        while((middle[j].y-middle[i].y)<=dmin and j<size){
            float d=get_distance(middle[i],middle[j]);
            if(d<=fd){
                update fd and add points into pair_arr
            }
            j++;
        }
    }
    return fd;
}

float closet_pair(point *p,int size){
    if(size<=3){
        float d1,d2,d3,dmin1;
        d1=get_distance(p[0],p[1]);
        if(size==2){
            pair_arr[pair_size].a=p[0];
            pair_arr[pair_size].b=p[1];
            pair_size++;
            return d1;
        }
        else{
            d1=get_distance(p[0],p[1]);
            d2=get_distance(p[1],p[2]);
            d3=get_distance(p[0],p[2]);
            dmin1=min(d1,d2);
            dmin1=min(dmin1,d3);
            if(dmin1==d1){
                store the first and second point
            }
            if(dmin1==d2){
                store the second and third point
            }
            if(dmin1==d3){
                    store the first and third point
            }
```

```
            return dmin1;
        }
    }
    else
    {
        int medin=size/2;
        int n1=medin,n2=size-medin;
        point *left=new point[n1];
        point *right=new point[n2];
        use for loop to copy all number into new array
        float dL=closet_pair(left,n1);
        float dR=closet_pair(right,n2);
        float dmin2=min(dL,dR);

        float low=p[medin].x-dmin2;
        float high=p[medin].x+dmin2;
        int index=0,size_middle=0;
        point *middle=new point[200000];
        while(index<size){
            find all points in middle band
        }
        dmin2=cloest_cross_pair(middle,dmin2,size_middle);
        return dmin2;
    }
}
int main(int argc,char **argv){
    read file
    merge sort in x direction
    merge sort in y direction
    float dim=closet_pair()
    output information into file
}
```

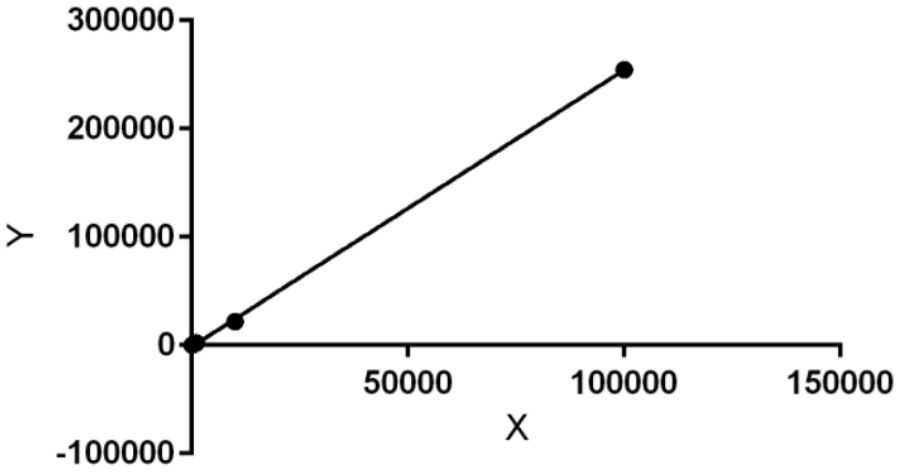
Analysis

1. We merge sort in x direction and y direction in main function. It would take 2*NlogN.
2. In closet_pair function, we copy all the array, so it would take N. Then, we find points in middle band, It would take n, and n is

smaller than N. Then we call the function cloest_crorss_pair, it would take C*n times.

3. Overall, we have N+n+C*n, according to dominate rules. We only take N.
4. So it would be T(N)=T(N/2)+O(N), as a result, I would take O(NlogN)

	10^2	10^3	10^4	10^5
1	237(Micro)	1978	22999	173952
2	230	1953	28002	144044
3	253	2038	33260	1000000
4	158	2595	17206	144834
5	175	2633	22398	187518
6	160	2010	22763	241877
7	184	2433	16742	160356
8	194	1976	19463	148918
9	160	1937	18236	143568
10	249	2547	17223	200501
AVG	200	2210	21829.2	254562.8



Best-fit values	
Slope	2.556 ± 0.02452
Y-intercept	-1301 ± 1232
X-intercept	509.0
1/Slope	0.3912

95% Confidence Intervals	
Slope	2.451 to 2.662
Y-intercept	-6604 to 4001
X-intercept	-1603 to 2527