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){
            store two points in to pair_arr
             and 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;
        create left array with n1
        create right array with n2
        use for loop to copy all the numbers 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;
        while(index<size){
            find all the points in middle band
        }
        mergeSort(middle,0,size_middle-1,'y');
        dmin2=cloest_cross_pair(middle,dmin2,size_middle);
        return dmin2;
    }
}
int main(int argc,char **argv){
    read file
    merge sort in x direction
    float dim=closet_pair();
    output information into file
}
```

Analysis

1. We called closet_pair function. In this function, we used divided and conquer. We broke the points into left half and right half. So it would be T(N)=2T(N/2)+?. In order to figure out ?, we have to go into the function cloest_pair.
2. First. We merge sort the array in x direction and it costs Nlog(N) (PS:This is at main function which is outside the cloest_pair function).
3. Then, we go through the array and put them into left half and right half, so it takes N times. After that, according the minimum value between left half and right half, we set up a range to collect all the points in the middle band and put them into a new array ‘Middle’, it takes much

smaller time than N, so we set it to ‘n’. Finally, we used merge sort to sort Middle array in y direction, so It would be nlog(n).

4. We get in to the function cloest_corss_pair. In this function, we go over all the points in the middle band and find their pair, because there is a maximum number of pairs possibilities for each point. It would cost Cn (C is a constant number).
5. Overall, we would have N + nlog(n) + n, according to dominate rules, we only have N since N is much bigger than n
6. So, it is T(N)=2T(N/2) + N, overall, the runtime would be N*log(N)

	10^2	10^3	10^4	10^5
1	173 (Micro)	3835	36280	291137
2	283	3788	36419	346172
3	311	3818	36662	1000000
4	293	3800	36618	340851
5	299	3778	36459	286846
6	370	3743	35948	328990
7	309	3825	36297	337682
8	275	3893	36125	1000000
9	285	3830	36122	356762
10	284	3819	36011	1000000
AVG	288.2	3812.9	36294.1	528844

X-axis unit is sample size

y-axis unit is Microsecond

Intercept:

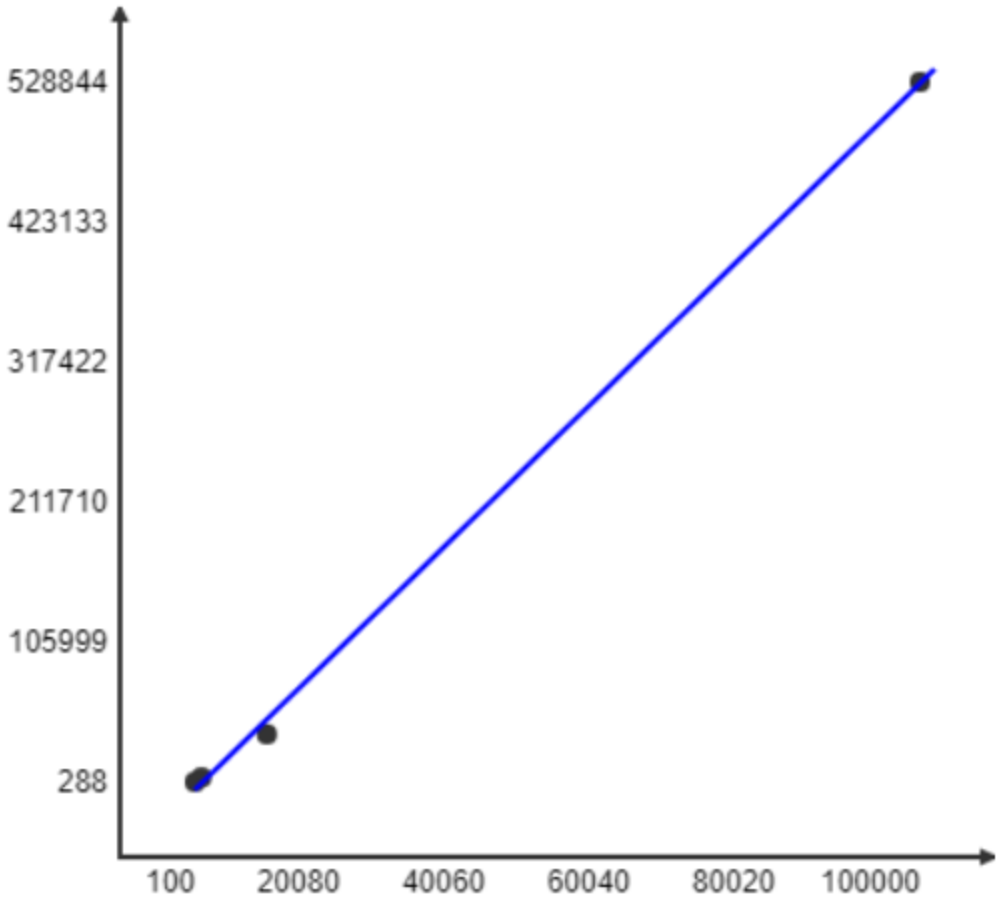
-5927.389051

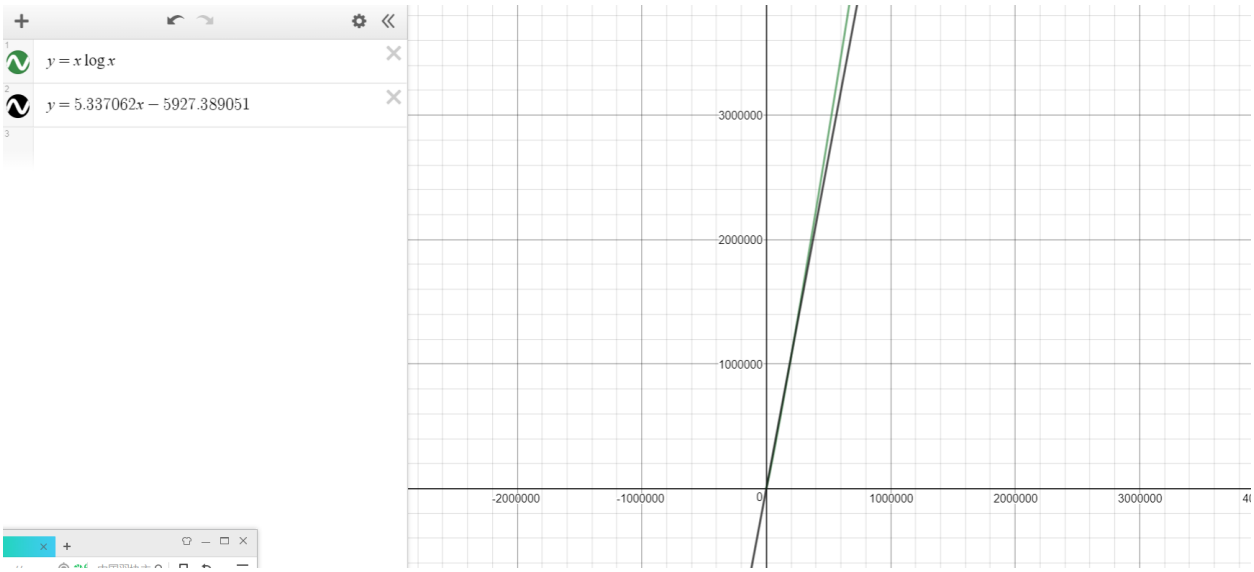
Slope:

5.337062

Line of Best Fit :

y = 5.337062x - 5927.389051




y = x log x
y = 5.337062x − 5927.389051
3000000
2000000
1000000
-2000000
-1000000
0
1000000
2000000
3000000