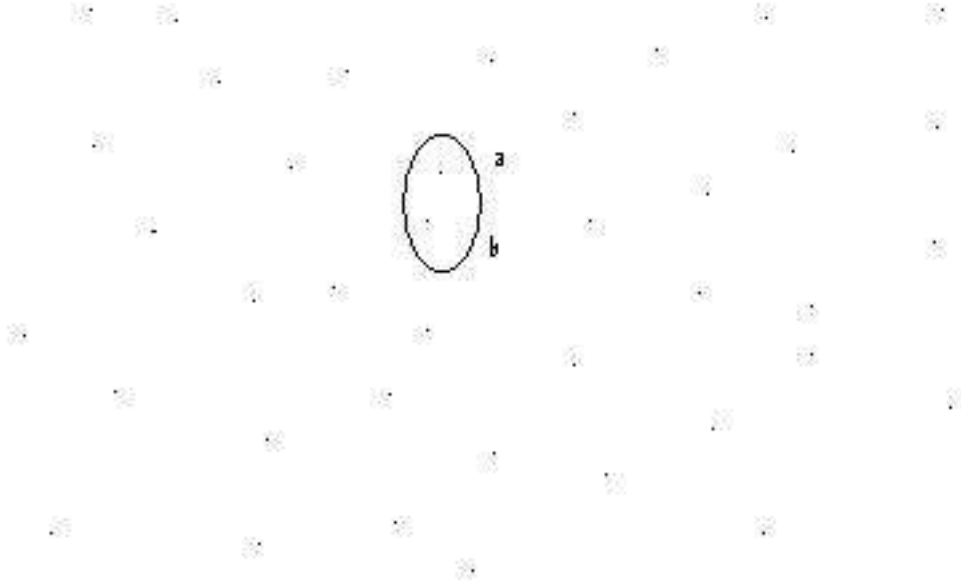


1 Closest Pair Problem

Given a set of n points determine points a, b such that the interpoint distance (Euclidean) is the minimum among all possible pair of points.



Here we are representing this as a 2-D problem but in general it can be an n dimensional problem. Points are represented by their coordinates. For this problem coordinates being integral wont make much of a difference, though we will assume that we have real coordinates.

1.1 Inter Point Distance

Inter point distance between two points $P_i(x_i, y_i)$ and $P_j(x_j, y_j)$ is given as follows :
 $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. This can easily be extended to an n dimensional case.

1.2 Trivial Method

For any dimension d , we can solve this by computing the minimum of $\binom{n}{2}$ pairs. This gives a running time of $\Theta(n^2)$. As we increase the dimension the computation for min distance

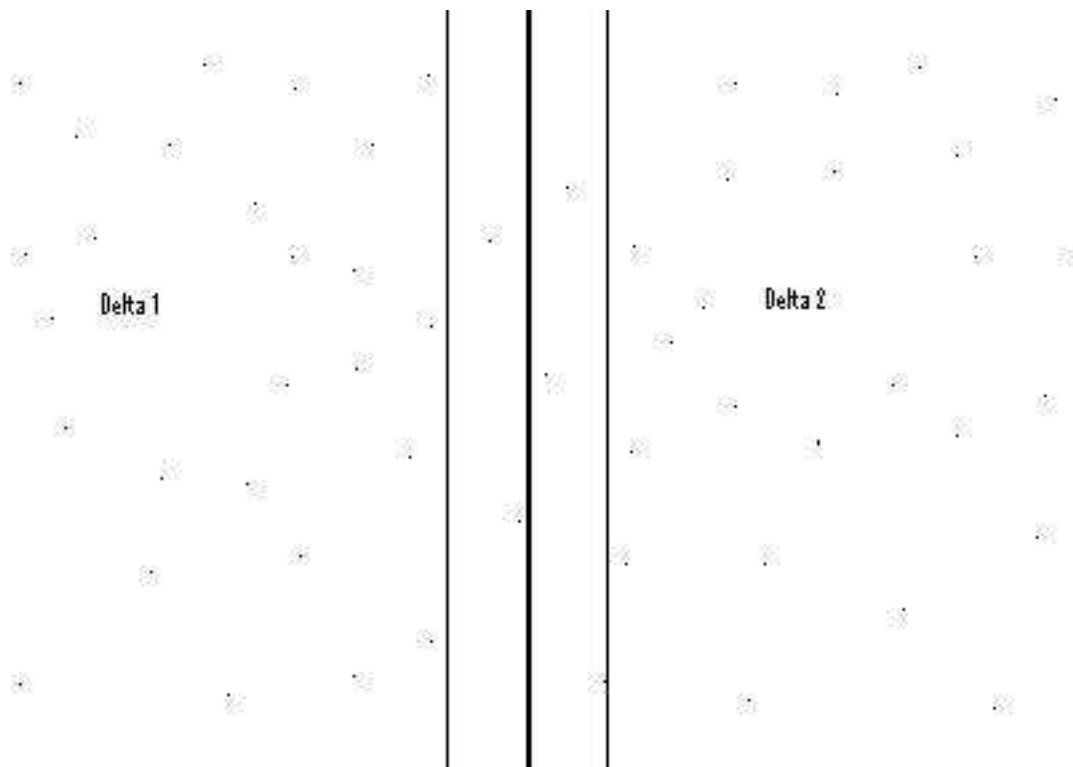
scales up, hence the running time of $\Theta(n^2d)$.

1.3 Problem in One Dimension

In one dimension all the points are in a line after we sort them in $\Theta(n \log n)$. After sorting we can go through them to find the consecutive points with least separation. Thus the problem in one dimension is solved in $\Theta(n \log n)$ time which is mainly dominated by sorting time.

2 Divide And Conquer

Consider the 2-D problem. We can find the median line (x-coordinate) in $\Theta(n)$ time. Now let δ_1 be the minimum interpoint distance in the left half. Similarly let δ_2 be the minimum interpoint distance in the right half.



We have $\delta = \min(\delta_1, \delta_2)$. The pairs where the left half is on the left side and the right half on the right side form a separate case. We need not worry about all such points, only points lying with the $2 - \delta$ strip are of concern to us. All others will have interpoint distance greater than δ and can be eliminated from consideration. The kind of recurrence we are looking will be of the following type :

$$T(n) = 2 * T(\frac{n}{2}) + \Theta(n) + MiddleCase$$

where $\Theta(n)$ time for finding the median and the Middle Case refers to the number of comparisons needed to be done within the $2 - \delta$ Strip.

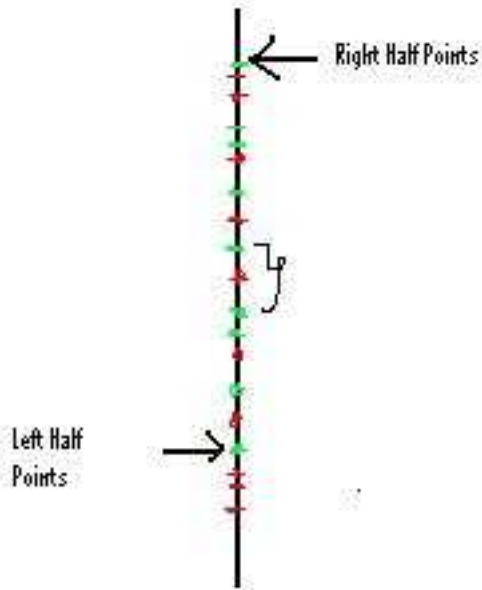
2.1 Number of Comparisons within the Delta Strip

Sparsity Condition: We cannot have too many points within one side of the 2δ Strip as it will make the interpoint distance on that half less than δ Strip itself, which is a contradiction according to the definition of δ Strip.

What is the maximum number of points in the 2δ Strip \times δ square where the inter point Distance is $\geq \delta$?

2.2 Solving Technique for 2D

Project all point on the vertical line. For every projected point P we need to find out how many other points on the vertical line we must compare it with as shown in the illustration below.



The green points are the Projection of the points on the right half of the δ Strip while the red points are For the left half. For every point we need to go δ up and δ down and compare points lying within that interval only .Thus for every point we have to do a constant C number of comparisons. We can sort them on the basis of y coordinate before doing the Divide and Conquer procedure. Thus the overall time will be $\Theta (n \log n) + T(n) \dots$ where $T(n) = 2T(n/2) + O(n) + C_d(n)$. This constant C_d scales up with dimension 'd'.

2.3 3 and Higher Dimensions

For the same problem in 3-D we can project points on both half within delta distance on a plane. Thus we see by projecting we are reducing the problem by one dimension.

Again because of sparsity condition there are no more than a constant number of points say C_d (dependent on 'd') in a $2 - \delta \times 2 - \delta$ Square.

2.4 Lemma:

In the d -dimension it is possible to find a partitioning plane H with the following properties:

- The size of the smaller set $\geq \frac{n}{2d}$ and largest set $\leq (1 - \frac{1}{2d})n$
- If the min interpoint distance is δ^* then the $2\delta^*$ slice does not contain more than $C_d n^{(1 - \frac{1}{d})}$ points. (Note δ is unknown).

We can apply a second recursion to the inner place thus solving the same problem in 2-D now. The recursive equation would look something like this in that case:

$T(n) \leq 2T(n/2) + M(k) + O(n)$ where $M(k)$ is the time to merge k points in the $2 - \delta$ Slab. Here,

- $M(k)$ equals $O(k \log k)$
- If we use the above lemma choose the *special line* to partition $k = O(n^{\frac{2}{3}})$. So $M(n^{\frac{2}{3}})$ is $O(n)$.

2.5 Alternate Method

This method is for the 2D sub-problem reduced from projecting 3D problem onto a plane. The procedure maintain a δ separated grid as shown below:

For every point we have to look at the Eight Adjacent cells only. Sparsity condition guarantees not too many points within a cell. So for each point we need a constant time to search points within eight adjacent cells. Hence for n points we would need $\Theta(n)$.

