Closest pair problem

- Closest-pair problem calls for finding the two closest points in a set of *n* Points.
- Cluster Analysis in statistics deals with closest pair problem
- Euclidian distance is used to calculate the closest pair problem

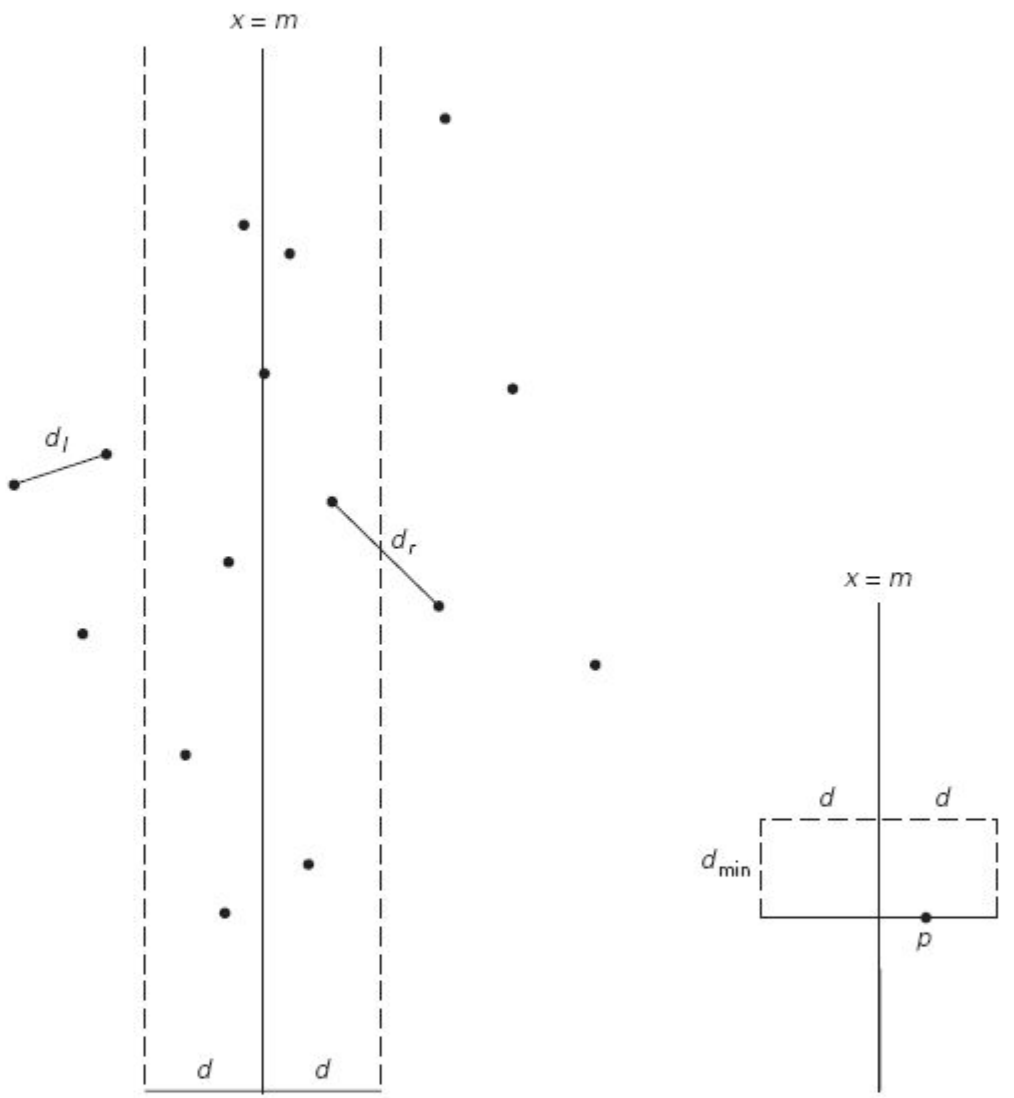
$$d_{xi,xj)} = \sqrt{(xi - xj)^2 + (yi - yj)^2}$$

- Let *P* be a set of *n* > 1 points in the Cartesian plane , we assume points are distinct
- Assume that the points are ordered in nondecreasing order of their *x* coordinate and y coordinate

Closest pair problem

- Closest-pair problem calls for finding the two closest points in a set of *n* Points.
- Divide the points into equal halves
- After dividing the points into two equal halves find the closest point in each equal halves
- Find the distances among each closest point in each halves using Euclidean distance
- Store the distances calculated in a array
- Find the smallest distance in the array an the return the point for which we received minimum distance is closest pair

Closest pair problem -Example



Closest pair problem -Motivation

- When number of points or number of items are available we have to find the closest items or points available
- We find the distance between each pair of distinct points and find a pair with the smallest distance
- A naive algorithm of finding distances between all pairs of points in a space of dimension d and selecting the minimum requires $O(n^2)$ time
- But while we go for calculating distance between two points using Euclidean distance , The total time taken is O(nlogn)
- So Euclidean distance is used in closest pair using divide and conquer mechanism
- Time taken for computation using Brute Force approach is $O(n^2)$

Closest pair problem -Algorithm

```
if n ≤ 3
        return the minimal distance found by the brute-force algorithm
else
        copy the first _n/2_ points of P to array Pl
        copy the same _n/2_ points from Q to array Ql
        copy the remaining _n/2_ points of P to array Pr
        copy the same _n/2_ points from Q to array Qr
        dl ←EfficientClosestPair(Pl, Ql)
        dr ←EfficientClosestPair(Pr, Qr)
        d ←min{dl, dr}
        m←P[_n/2_ − 1].x
        copy all the points of Q for which |x − m| < d into array S[0..num − 1]
        dminsq ←d2
for i ←0 to num − 2 do
        k←i + 1
while k ≤ num − 1 and (S[k].y − S[i].y)2 < dminsq
        dminsq ←min((S[k].x − S[i].x)2+ (S[k].y − S[i].y)2, dminsq)
k←k + 1
return sqrt(dminsq)
```

Closest pair problem -Analysis

- Running time of the algorithm (without sorting) is:

$$T(n) = 2T(n/2) + M(n), \text{ where } M(n) \in \Theta(n)$$

- By the Master Theorem (with $a = 2$, $b = 2$, $d = 1$)

$$T(n) \in \Theta(n \log n)$$

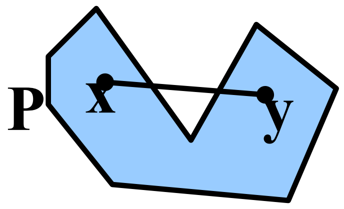
- So the total time is $\Theta(n \log n)$.

Real time applications

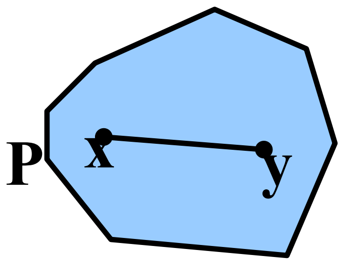
- Air/land/water traffic control system
- Collision avoidance

Convex vs. Concave

- A polygon P is convex if for every pair of points x and y in P, the line xy is also in P; otherwise, it is called concave.



concave



convex

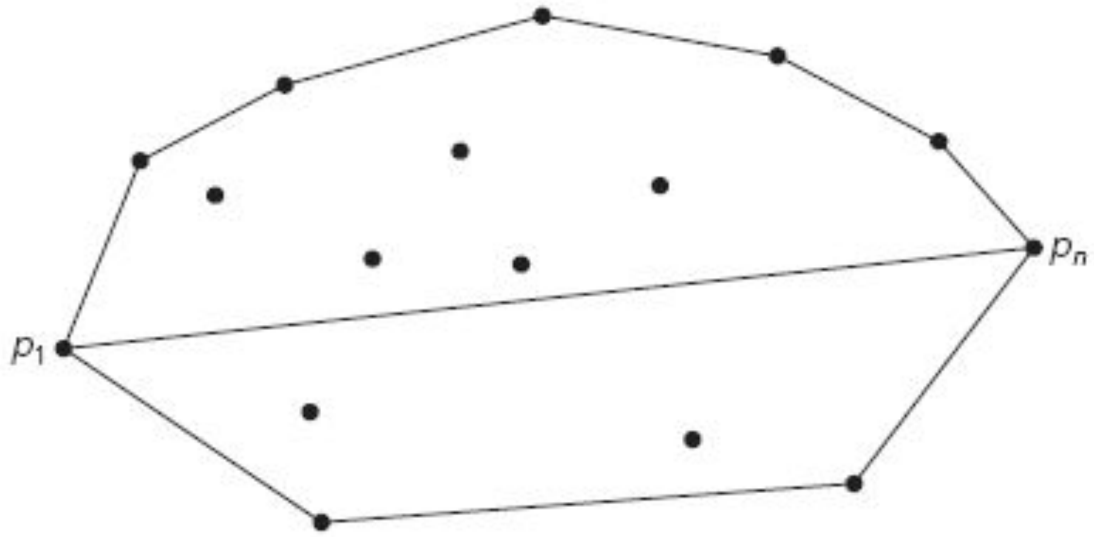
Convex hull Problem

- Convex hull or convex envelope or convex closure of a shape is the smallest convex set that contains it

- Convex Hull is the line completely enclosing a set of points in a plane so that there are no concavities in the line

Convex hull Problem

- Let S be a set of $n>1$ points $p1(x1, y1), \ldots, pn(xn, yn)$ in the Cartesian plane



- Splitted in upper and lower half

Convex hull Problem -Algorithm

```
vector<pair<int, int>> divide(vector<pair<int, int>> a)
{
     if (a.size() <= 5)
        return bruteHull(a);
    vector<pair<int, int>>left, right;
    for (int i=0; i<a.size()/2; i++)
        left.push_back(a[i]);
    for (int i=a.size()/2; i<a.size(); i++)
        right.push_back(a[i]);
     vector<pair<int, int>>left_hull = divide(left);
     vector<pair<int, int>>right_hull = divide(right);
     return merger(left_hull, right_hull);
}
```

Convex Hull - Time Complexity

- Time complexity If points are not initially sorted $O(n \log n)$
- Time efficiency: $T(n) = T(n-1) + O(n)$

  $T(n) = T(x) + T(y) + T(z) + T(v) + O(n)$, where $x + y + z + v \leq n$.

  worst case: $\Theta(n^2)$

  average case: $\Theta(n)$

Real time applications

- Collision avoidance

Assignment

- Implementation of Closest pair problem
- Implementation of Convex hull problem