Problem: Given N points in a rectilinear metric. Find the smallest disc containing these points in O(n)

Solution:

Given a set S of 2D points, find the circle with the smallest radii such that all the other points in S are contained in either C or its circumference

Welzl's Algorithm:

1. It exploits the fact that if the coordinate is not present in the disc constructed so far, then it will be present on the boundary of the new circle/sphere.
2. The following are the observations:
    a. For the size of the input set at most 3, small-case routine is generated. It outputs empty circle, circle with radii 0, circle on diameter constructed by an arc of input coordinates for input sizes 0, 1, and 2, correspondingly.
    b. For input size 3, if the coordinates are vertices of an acute-angled triangle, the circumcircle is outputted. If not, the circle having a diameter of the triangle's lengthiest edge is outputted.
    c. For more substantial scale inputs, the program uses a solution of 1 smaller scale where we choose the outlier point *p* randomly and consistently. The returned circle *D* is verified, and if it encloses *p*, it is returned as a result. Otherwise, we know point *p* is on the border of the result.
3. Algorithm Working:
    a. Let md(P) denote the minimum enclosing disc enclosing the set of points "P."
    b. md(P) for a given set P of n points, is calculated in a Randomized Incremental fashion.
    c. Let P be the set of n points and D = md{p1, p2, ...pi}, for $1 \leq i \leq n$ points seen so far.
    d. Now, if pi+1 ∈ D, then we need not do anything, and now, D = md {p1, p2, ...pi, pi+1}, and we proceed to next point.
    e. Else, we use the fact that pi+1 will lie on the boundary of D 0 = md {p1, p2, ...pi , pi+1}.

Algorithm 1: minidisk (P)

```
if P = ϕ then
    D ← ϕ
else
    choose p ∈ P
    D ← minidisk(P − {p})
    if p ∈/ D then
        D ← b minidisk(P − {p}, p)
    end if
end if
return D
```

The constraint to use the trivial algorithm is |*P*|=0 or |*R*|=3. Equivalent limitation will be |*R*|=3 or |*P*|+|*R*|≤3, and call of the trivial method for *R* in the scenario |*R*|=3 and union of *P* and *R* otherwise. This would prevent some recursive calls on the bottom of the recursion.

```
algorithm welzl is
    Input: Finite sets P and R of coordinates in the plane |R|≤ 3.
    output: Minimum disk circumscribing P with R on the border.
    if P is empty or |R| = 3 then
        return trivial(R)
    choose p in P (arbitrarily and consistently)
    D := welzl(P - { p }, R)
    if p is in D then
        return D
    return welzl(P - { p }, R ∪ { p })
```