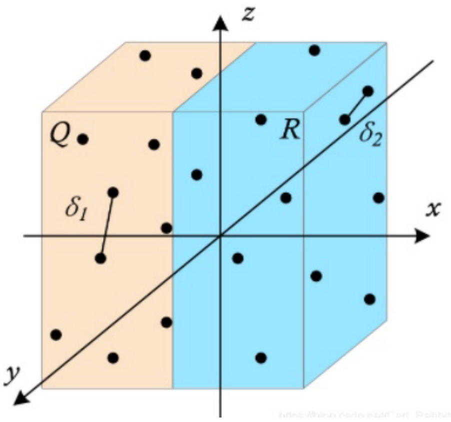
Main idea for this question is Divide and Conquer.

For a 3-D point, we know the Euclidean distance of any two points is $\sqrt{\sum_{k=1}^{3}(x_{ik}-x_{jk})^2}$

Algorithm description:

① For any out of order point set P, we need to sort x and z respectively and get P_x, P_z.

② Divide.



For any point of set P, the numbers of middle $|P|=n$, we mark $\lfloor n/2 \rfloor$ as m_i, the coordinates of m_i denoted as x^*, divid Q and R zone, If coordinates less than x^*, put Q, otherwise, put R zone.

③ maintain a sorted Aarray.

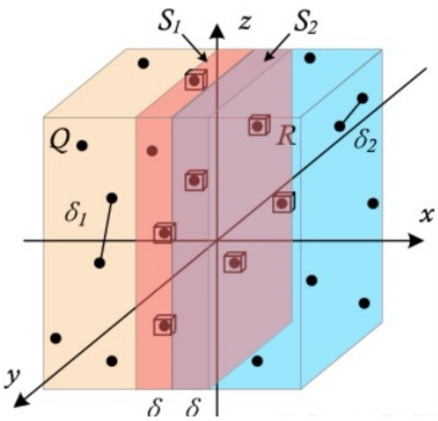
For Q and R zone, we want to keep 4 sorted arrays, Q_x, Q_z, R_x, R_z.

③ Merge:

We suppose the closet points $\{q_1, q_2\}$ in R zone, the distance denoted as δ_1, likewise, $\{r_1, r_2\}$ in R zone, distance δ_2.

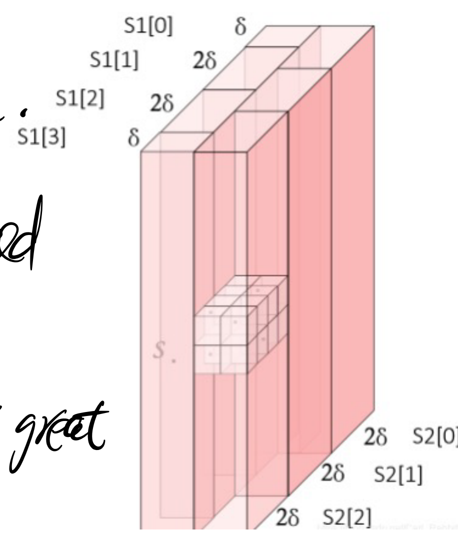
For all points $\{p_i, q_i\}$ and $p_i \in Q \wedge p_j \in R$, which has a closest distance δ_3.

④ Build 2 S zones.



let $\delta = \min(\delta_1, \delta_2)$, the area left and right denoted δ as S zone, left is S_1 zone, right is S_2 zone.

⑤ Further divided for y zone.



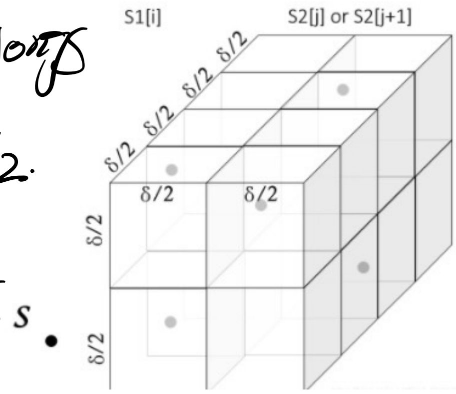
For any point in $S_1[i]$, we only need to compare $S_2[i-1]$ and $S_2[i]$

If it exceeds, only in y direction will great than δ.

then, we can use the Pigeonhole principle to reduce the number of checks in the z dimension.

We can suppose the check point s belongs to S_1. each box with a length of $\delta/2$.

For each box, the largest distance is $\sqrt{3}\delta/2 < \delta$.



To sumarize this idea:

First determine the minimum value y for divide.

traverse the ordered P_z from small to large,

if the distance $x^* > \delta$, the it doesn't belong S.

otherwise,

if this point s in S_1 zone, add it to S_1 array, figure out that it should belong to ith of S_1 and correspond to $S_2[i-1]$ and $S_2[i]$, then we need to record $len(S_2[i-1])$ and $len(S_1[i])$, and all points in this area less than s.

if s point in S_2, then add it to corresponded $S_2[i]$.

Then, we traverse the whole S_1, and check 16 points according to divide and reference position of S.

⑥. Algorithms analysis.

The splitting and maintenance of Divide step is $O(n)$,
the step of merge is $O(n)$ as well, then we know

$$f(n) = \begin{cases} 2f(n/2) + O(n) & n > 3. \\ O(1) & n \leq 3. \end{cases}$$

Same as 2D,
the total Algorithmas is $O(n\log n)$.
and I'll show this using a sample code.