Given a set \( S \) of \( n \) points in \( \mathbb{R}^3 \), design an efficient algorithm (preferably linear but no more than \( O(n \log n) \)) to find the smallest enclosing sphere of \( S \).

Your algorithm can be randomized and you must provide detailed proof of correctness and running time.

Answer in the space provided below.

Use RIC where let \( \text{Minball}(S, f) \) be defined as the smallest ball containing a set of \( S \) points of which \( f \) are known points \( 0 \leq f \leq 4 \). The initial algorithm is called as \( \text{Minball}(N, 0) \) since all the (maximum) 4 points on the boundary must be determined. Note that there can be 2, 3 or 4 points on the surface of the smallest ball containing all the points. If there are 2 or 3 points on the boundary then the ball can be determined by the points in dimensions 2 or 3 by considering the line (plane) passing through these points.

In the algorithm described below, we call \( \text{Minball}(N, \phi) \).

Let \( T(i, u) \) denote the expected running time of the algorithm \( \text{Minball} \) with \( n \) points of which \( u \) are to be determined. Note that \( 0 \leq u \leq 4 \).

Using backward analysis we can show that

\[
T(i, u) \leq T(i - 1, u - 1) + \frac{u}{i} \cdot T(i - 1, u - 1) + O(1)
\]

Using \( T(i, 0) = O(1) \), we can show that \( T(i, u) = O((u)!i) \) similar to the RIC based LP algorithm.
Procedure Compute the smallest enclosing ball \((S, D)\)

1. **Input** \(S\) is a set of points in 3D where \(S\) is given as a random sequence;
2. Let \(C_j\) denote the smallest ball containing the first \(j\) points defined by \(k\) points \(k \leq 4\) \(C_0 = D\);
3. **while** \(j \leq n\) **do**
4.     **if** \(p_{j+1} \notin C_j\) **then**
5.         **if** \(D < 4\) **then**
6.             \(C_{j+1}\) is defined by \(D \cup p_{j+1}\);
7.             \(\text{Minball}(S - p_{j+1} , D \cup p_{j+1})\)
8.         **else**
9.             Replace the last added point in \(D\) by \(p_{j+1}\);
10.     \(j \leftarrow j + 1\);
11. **Output** \((C_n)\);