Algorithm 1 Find

Input: Point
Output: Node

```
1: curr ← root
2: while !curr.isLeaf do
3:     next ← 0
4:     if Point.x ≥ curr.center.x then
5:         next ← next + 4
6:     if Point.y ≥ curr.center.y then
7:         next ← next + 2
8:     if Point.z ≥ curr.center.z then
9:         next ← next + 1
10:    curr ← curr.children[next]
11: return curr
```

Algorithm 2 Insert

Input: Point
Output: Boolean

```
1: node ← find(Point)
2: if node.contains(Point) then
3:     return False
4: node.points.add(Point)
5: if node.points.size ≥ limit then
6:     node.subdivide
7: return True
```

Algorithm 3 Node Subdivision

```
1: isLeaf ← False
2: newHalfSize ← halfSize ÷ 2
3: for i ∈ {0, 1} do
4:     for j ∈ {0, 1} do
5:         for k ∈ {0, 1} do
6:             X ← center.x - newHalfSize + i × halfSize
7:             Y ← center.y - newHalfSize + j × halfSize
8:             Z ← center.z - newHalfSize + k × halfSize
9:             children[i × 4 + j × 2 + k] ← Node({X, Y, Z}, newHalfSize)
10: for all Points in Node do
11:     remove(Point)
12:     next ← 0
13:     if Point.x ≥ center.x then
14:         next ← next + 4
15:     if Point.y ≥ center.y then
16:         next ← next + 2
17:     if Point.z ≥ center.z then
18:         next ← next + 1
19:     children[next].points.add(Point)
```

Algorithm 4 Resize

```
    Input: Point
1:  while Point is out of bounds do
2:      if Point.x < root.center.x then
3:          direction.x ← -1
4:      if Point.y < root.center.x then
5:          direction.y ← -1
6:      if Point.z < root.center.x then
7:          direction.z ← -1
8:      newHalfSize ← root.halfSize × 2;
9:      newCenter.x ← root.center.x + direction.x × root.halfsize
10:     newCenter.y ← root.center.y + direction.y × root.halfsize
11:     newCenter.z ← root.center.z + direction.z × root.halfsize
12:     newRoot ← Node(NULL, newCenter, newHalfSize)
13:     newRoot.subdivide
14:     rootCode ← (direction.x + 1) × 2
15:     rootCode ← rootCode + (direction.y + 1)
16:     rootCode ← rootCode + (direction.z + 1) ÷ 2
17:     newRoot.children[rootCode] ← root
18:     root ← newRoot
```