```java
private void findNnNode(Node node, Deque<Node> nodeStack,Nn nn,
                              DistanceCache distanceCache) {
    nodeStack.push(node);
    if (node.isLeaf()) {
        boolean update = false;
        IntArrayList children = node.getChildren();
        for (int i = 0; i < children.size(); i++) {
                /……/遍历叶节点的所有数据点,找到距离最近的数据点
        }//
        if (update) {
            nn.nodeStack.clear();
            nn.nodeStack.addAll(nodeStack);//保存节点路径
        }
    } else {
        int vpDocId = node.getVpDocId();
        float distance = distanceCache.distance(vpDocId);
        ++queryStat.distanceFunctionVpInvocations;
        if (Float.compare(distance, nn.distance) < 0) {
            /……/更新 NN 结果及路径
        }
        FloatArrayList cBounds = node.getCBounds();
        int size = cBounds.size() / 2;
        Node[] cNodes = node.getCNodes();
        float low, high;
        for (int i = 0; i < size; i++) {
            low = cBounds.get(i * 2) - nn.distance;
            high = cBounds.get(i * 2 + 1) + nn.distance;
            //以当前最近数据点的距离作为剪枝容忍距离
            if (Float.compare(distance, low) >= 0 &&
                        Float.compare(distance, high) <= 0) {
                findNnNode(cNodes[i], nodeStack, queryStat, nn,
                        distanceCache);
            }
        }
    }
    nodeStack.pop();
}
```