1. Describe and analyze an algorithm that answers the following query in $O(\sqrt{n})$ time, assuming the points $P$ are stored in a kd-tree.

$\text{CountAbove}(b)$: Return the number of points in $P$ that lie above the horizontal line $y = b$.

To avoid some boundary cases, assume that $n = 2^k - 1$ for some integer $k$, that all points in $P$ have distinct $x$- and $y$-coordinates, and that no point in $P$ lies directly on the line $y = b$. [Hint: How many boxes does the query line intersect?]

**Solution:** The main idea is to recursively calculate the total number of points above the line in both subtrees, but we don’t recurse if all points in a subtree lie above the line, or all points in a subtree lie below the line.

```plaintext
((Return the number of points above \(y = b\) in a kd-tree rooted at \(v\).))

\[\text{CountAbove}(v, b):\]
if \(v = \text{Null}\)
    return 0
else if \(v.\text{dir} = \text{Vertical}\)
    return \(\text{CountAbove}(v.\text{left}, b) + [v.y > b] + \text{CountAbove}(v.\text{right}, b)\)
else \((v.\text{dir} = \text{Horizontal})\)
    if \(v.y > b\)
        return \(\text{size}(v.\text{up}) + 1 + \text{CountAbove}(v.\text{down}, b)\)
    else
        return \(\text{CountAbove}(v.\text{up}, b)\)
```

(The expression “\([v.y > b]\)” evaluates to 1 if \(v.y > b\) and 0 otherwise; this notation is called the \textit{Iverson bracket}.)

We can prove by induction that this algorithm is correct. The base case \(v = \text{Null}\) is trivial. If \(v\) is a vertical node, we correctly count \(v\)’s point if it lies above the line, and by the induction hypothesis, we correctly count the points above the line on either side of the vertical cut. If \(v\) is a horizontal node and \(v.y > b\), we correctly count \(v\)’s point and all points in \(v.\text{up}\), and the induction hypothesis implies that the points below the cut are counted correctly. Finally, if \(v\) is a horizontal node and \(v.y < b\), all the points above the line \(y = b\) are in the upper subtree, and the induction hypothesis implies that these points are counted correctly.

Let \(H(n)\) and \(V(n)\) denote the running times when the top-level cut is horizontal or vertical, respectively. Ignoring floors and ceilings, which don’t matter asymptotically, we have mutual recurrences

\[V(n) = 2H(n/2) + O(1) \quad \text{and} \quad H(n) = V(n/2) + O(1),\]

with the usual base cases \(H(n) = V(n) = O(1)\) when \(n = O(1)\). Substituting \(H(n/2) = V(n/4) + O(1)\) in the first recurrence simplifies it to

\[V(n) = 2V(n/4) + O(1).\]
We can solve this simpler recurrence using the recursion tree method, as follows.

The root of the recursion tree for $V(n)$ has value 1 and two children, each of which is the root of a recursion tree for $V(n/4)$. So the recursion tree has one root, two nodes at depth 1, four nodes at depth 2, and more generally, $2^d$ nodes at any depth $d$. Thus, the level sums form an increasing geometric series $T(n) = 1 + 2 + 4 + \cdots + 2^D = O(2^D)$, where $D$ is the maximum depth of the tree. The overall depth of the recursion tree is at most $L = \log_4 n$. Thus, $T(n) = O(2^{\log_4 n}) = O(n^{\log_4 2}) = O(n^{1/2}) = O(\sqrt{n})$, as required. ■