# 1 Multi-dimensional Range Trees

In $d$-dimensional computational geometry, when one is dealing with sets of points in $d$-space, one could ask a $d$-space range query. Consider a spatial interval $(\langle x_1^1, x_2^1\rangle \ldots \langle x_1^d, x_2^d\rangle)$: this represents an axis-aligned rectilinear solid in $d$-space. A query could ask for the points that occur within this solid. Such problems are solved efficiently in computational geometry using so-called *Range Trees* (See [1], Chapter 5). We draw an analogy between this problem and a multi-attribute database query.

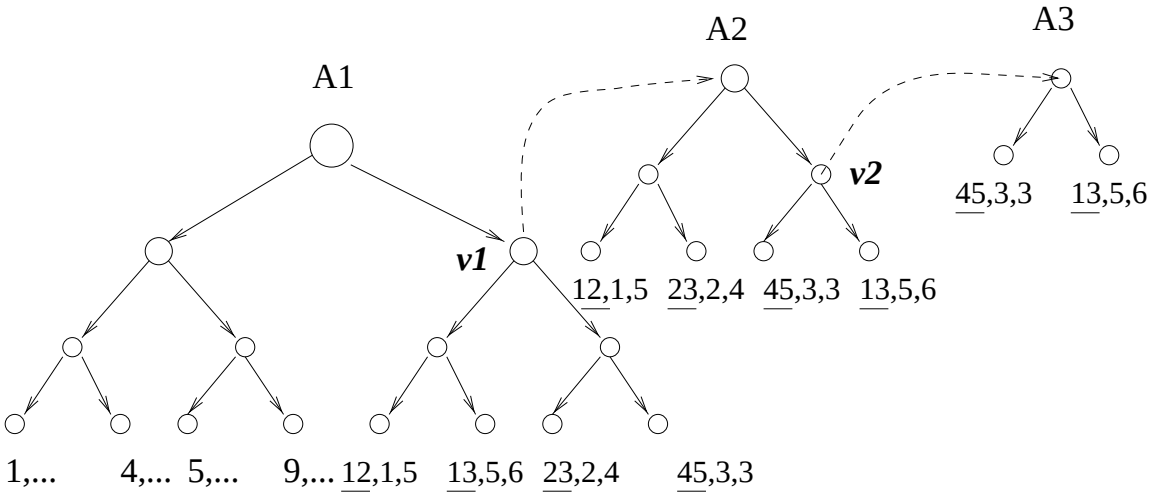


Figure 1: Excerpt of a 3-dimensional range tree, sorted by attributes $A_1$, $A_2$ and $A_3$

Consider the example *mdrt* for points in 3D shown in Figure 1. Consider the first, 3-dimensional *mdrt* (labeled A1). This is simply a tree which sorts the values according to $x$ (first) coordinate ($x$-coordinate values are underlined). Each interior node in tree A1 is the ancestor of a set of values. Consider such an interior node $v1$, which is the ancestor of values with primary key values 12, 13, 23, and 45. An *mdrt* now contains a link from $v1$ to the root of an *associated tree* A2, denoted as $T_{assoc}(v1, A_2)$. This 2-dimensional *mdrt* A2 contains the same set of values with $x - values$ 45, 11, 23, and 13; however, in this tree, they are sorted according to attribute $y$-coordinate. Likewise each interior node $v_i$ in A1 is the ancestor to a set of values, and contains a pointer to an associated 2-dimensional *mdrt* $T_{assoc}(v_i, A_2)$ which sorts the values in the subtree below $v_i$ by $y$-coordinate. In general, each node $v_i^j$ of a $\{d-j+1\}$-dimensional *mdrt* contains a pointer to a $\{d-j\}$-dimensional *mdrt*. The nodes of the final 1-dimensional tree, do not have such pointers.

Given a 2-dimensional range query which asks for all points $(x, y)$ such that $x_1 \leq x \leq x_2$ and $y_1 \leq y \leq y_2$ the structure is used as follows. First, the points $q$ with $x$-values in the range $\langle x_1, x_2$ is identified using tree A1. For simplicity, let us assume (we relax this assumption later) that the values in $q$ form the leaves of a balanced tree with root $LCA(q)$. In an $n$ node tree this range can be found in time $O(\log_2 n)$, the time it takes to find the two end-points of the interval in the first tree. We now follow the link to the associated tree for $y$ this tree sorts just the value set $q$ according to $y$-coordinate So we can find the subset of $q$ satisfying

$y_1 \leq y \leq y_2$ also in $O(\log_2 n)$ time. This gives the intuition behind the efficient processing of conjunctive range queries using *mdrt*s. We now relax the assumption that the result of the first selection $q$ includes just the leaves of a balanced tree.

Let us call the leaves of the subtree rooted at node $v$ the *canonical subset* of $v$, denoted as $P(v)$. If $v$ is a leaf, $P(v) = v$. In [1] (pp 103-107) it is shown that *any* subset of leaves which lies in a range can be expressed as a disjoint union of $O(\log_2 n)$ canonical subsets for the given range query. The roots of these can be found in $O(\log_2 n)$ time in the process of finding the bounding paths for the interval. Given an $x$-range $\langle x, x' \rangle$, we search for them in the tree until we find node $v_{split}$ where the paths split. Now we search for $x$ ($x'$) in the left (right) subtree. At every point in the search for $x$ where the path goes left, the right subtree belongs to the range; the search for $x'$ goes just the opposite way. The result is a quick identification of roots of the canonical subtrees that precisely cover the leaves whose values are in the interval (see Figure 2).

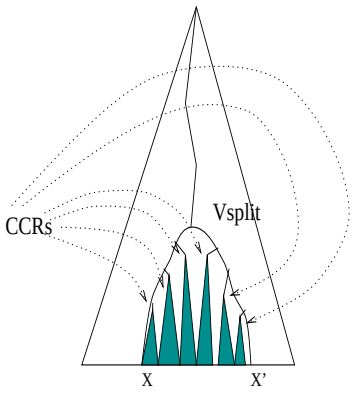


Figure 2: Finding the canonical covering roots (CCRs)

We call these the *covering canonical roots (CCRs)*. Consider, for example, a 2-dimensional range query. First, the CCRs in tree A1 for the given $x$-range are found; there are $O(\log n)$ of these. Then for each of the CCRs, we go to the associated $y$-tree, and find the CCRs in *that* tree for the given $y$-range . This results in $O(\log^2 n)$ CCRs in each $y$-tree examined. The union of all the leaves under these CCRs in $y$-trees constitute the answer. In general, it is shown [1] that $d$-dimensional range queries which have a total of $p$ points satisfying them can be computed in time $O(\log^d(n) + p)$. Range trees require $O(n \log^{d-1} n)$ storage space, and can be constructed in time $O(n \log^{d-1} n)$.

An enhancement of *mdrts* called *fractional cascading* [1] reduces the time to answer a query to $O(\log^{d-1}(n) + p)$ (but does not reduce the space).

# References

[1] M. D. Berg , M. V. Kreveld, M. Overmars and O. Schwarzkopf. *Computational Geometry.* Springer-Verlag, New York.