k-dimensional orthogonal range search with tries

University of New Brunswick Faculty of Computer Science Technical Report TR 02-154
April, 2002

Bradford G. Nickerson, Lingke Bu

Faculty of Computer Science
University of New Brunswick
Fredericton, Canada, E3B 5A1
{bgn,g0az5}@unb.ca

Abstract

We present an algorithm using tries to solve the orthogonal range search problem on k dimensions. The algorithm reports all k-d hyper-rectangles intersecting a k-d axis-aligned query hyper-rectangle W, and supports dynamic operations. For the input data set D and W drawn from a uniform, random distribution, we analyse the expected time for orthogonal range search. We show that the storage $S(n,k) = O(nk)$, and the expected preprocessing time $P(n,k) = O(nk)$ for a trie containing n k-d hyper-rectangles.

1 Introduction

Orthogonal range search [Knut98, Bent79] finds and reports all objects falling in a specified query hyper-rectangle W. This report considers the case where the objects to be searched are k-d hyper-rectangles. We investigate the use of k-d tries (e.g. Flajolet and Puech [1986], Shang [2001]) as a data structure to support orthogonal range search.

Given a collection F of records, each containing several attributes or keys, an orthogonal range query asks for all records with key values each inside specified ranges. Range search is the process of reporting the appropriate records falling inside the query range. The range search problem can be interpreted geometrically by considering the record attributes as coordinates and the k values for each record as a point in a k-dimensional coordinate space [Bent79]. Our definition for orthogonal range search is as follows:

Definition 1 *For a data space R^k, for k = number of dimensions, orthogonal range search is defined as finding and reporting the set $HR, (|HR| = A, HR \subseteq D, D$ = set of axis-aligned orthogonal data objects represented as hyper-rectangles, $|D| = n)$ of data intersecting the query hyper-rectangle $W = \{[L1, H1], [L_2, H_2], \cdots, [L_k, H_k]\}$, where $[L_j, H_j]$ represents a range for dimension j of the query hyper-rectangle.*

The classical orthogonal range search of Bentley [Bent79] is generalized to allow each of the n records in the collection F to be defined by a coordinate (or key) range.

1.1 Previous Work

Data structures supporting orthogonal range search on such records have been constructed, with one of the most popular being the k-dimensional variant of the B-tree known as the R-tree introduced by Guttman [1984]. Edelsbrunner [1983] introduced the d-fold rectangle tree to support orthogonal range search on k-d hyper-rectangles with $S(n,k) = O(n \log^{k-1} n)$, $Q(n,k) = O(\log^{2k-1} n + A)$ and $P(n,k) = O(n \log^{k} n)$. $Q(n,k)$ represents the time for orthogonal range search on a data structure containing n k-dimensional hyper-rectangles.

Bentley [Bent79, Bent80] reviews several data structures for k-dimensional point range searching including sequential scan, projection, cells, k-d trees, range trees and k-ranges. The d-fold $BB(\alpha)$ tree [Luek78, 82] has worst-case total time of $O(n \log^{k} n)$ for n operations (where an operation can be to insert or delete a point, or to perform a range search). The k-d tree [Bent75] is a binary tree. The k-d tree requires $O(kn)$ space and a total path length of $O(n \log n)$ for n k-d points inserted in random order. The analysis of range search for balanced k-d trees shows that $Q(n,k) = O(sn^{1-1/k} + A)$ for s of the k coordinates restricted to a subrange [LeeW77], and $(k-s)$ of the coordinates unspecified. Devroye et al [2000] analyzed range search on squarish k-d trees [Devr00] and random k-d trees [Chan01]. Fredman constructed a model for complexity analysis of range search [Fred81-a][Fred81-b]. Lower bounds for range search were studied by Chazelle [1990-a, 1990-b]. Bentley and Maurer [1980] investigated three k-ranges for range search. They showed that one level k-ranges had $Q(n,k) = O(k \log n + A)$, $S(n,k) = P(n,k) = O(n^{2k-1})$, and multi-level k-ranges require linear space, $P(n,k) = O(n \log n)$ and $Q(n,k) = O(n^{\epsilon})$. Merrett et al [1996] introduced the zoom trie for spatial data display, whose upper levels are used for lower resolutions, with the leaf level used for full resolution. Chazelle [1988] gives a comprehensive overview of data structures for k-dimensional searching, including the description of a k-dimensional rectangle reporting algorithm (supporting dynamic operations) with $Q(n,k) = O(A(\log(\frac{2n}{A})^2) + \log^{k-1} n)$.

There are thirteen possible relationships between two intervals [Alle83]. Shang's approach achieves interval containment by superimposing a PR-Trie with a ZoomTrie [Shan01]. In Shang's thesis, one-dimensional interval relationships are represented as queries in two dimensions. In this paper, we consider only one relationship (intersection) for orthogonal range search.

As pointed out in Flajolet and Peuch [1986], 1-d tries tend to be better balanced compared to 1-d search trees. For k-dimensional search, this improved balance can lead to aymptotically smaller search times.

2 k-d Tries for Range Search

Without loss of generality, we assume our search space is defined on the set of positive integers in k-dimensional space. We assume the space is finite, limited by the number of bits B used to represent an integer. B is the number of bits used for representing a

coordinate value in binary, B = $\log_2(MAXIMUM - MINIMUM + 1)$, where MINIMUM and MAXIMUM are the whole search space's upper and lower bounds.

2.1 Tries for Spatial Data

Binary tries are data structures which use a binary representation of the key to store keys as a path in the tree. Binary k-d tries use the principle of bit interleaving (also called a shuffle operation by Flajolet and Peuch [1986]). Child nodes in a k-d trie cover $\frac{1}{2}$ the search space volume of their parent. For a point p(x, y) on two dimensions, each coordinate value has B bits, and the bit interleaved value is $b_0^x\, b_0^y\, b_1^x\, b_1^y \cdots b_B^x\, b_B^y$, where b_i^x is the i^{th} bit value for x, and b_i^y is the i^{th} bit value for y. For a line segment with a start point s and end point e, bit interleaving treats the line segment as a 4-dimensional point $p = (x_s, y_s, x_e, y_e)$. For this so-called 4-d trie, the bit interleaving results in a bit string: $b_0^{x_s}\, b_0^{y_s}\, b_0^{x_e}\, b_0^{y_e} \cdots b_B^{x_s}\, b_B^{y_s}\, b_B^{x_e}\, b_B^{y_e}$. For a triangle which has three points on the plane, we treat it as a 6-d point $p = (x_0, y_0, x_1, y_1, x_2, y_2)$ or as three segments which construct the triangle. We represent a 2-dimensional rectangle as four coordinate values$(x^{min}, x^{max}, y^{min}, y^{max})$, which, after bit interleaving gives the bit string: $b_0^{x^{min}}\, b_0^{x^{max}}\, b_0^{y^{min}}\, b_0^{y^{max}}\, b_1^{x^{min}}\, b_1^{x^{max}}\, b_1^{y^{min}}\, b_1^{y^{max}} \cdots b_B^{x^{min}}\, b_B^{x^{max}}\, b_B^{y^{min}}\, b_B^{y^{max}}$. Extending the bit interleaving principle to k dimensions, on every dimension j, $\forall j \in \{1 \cdots k\}$, we represent the k-d hyper-rectangle as $(x_j^{min}, x_j^{max})^k$, so the resultant bit string will be

$$b_0^{x_1^{min}}\, b_0^{x_1^{max}}\, b_0^{x_2^{min}}\, b_0^{x_2^{max}}\, b_0^{x_3^{min}}\, b_0^{x_3^{max}} \cdots b_0^{x_k^{min}}\, b_0^{x_k^{max}}$$
$$\ddots$$
$$b_B^{x_1^{min}}\, b_B^{x_1^{max}}\, b_B^{x_2^{min}}\, b_B^{x_2^{max}}\, b_B^{x_3^{min}}\, b_B^{x_3^{max}} \cdots b_B^{x_k^{min}}\, b_B^{x_k^{max}}.$$

Thus, a k-d hyper-rectangle can be represented as a $2k$-dimensional point in a binary trie of height $2k$B. Figure 1 below is an algorithm for performing bit interleaving of a k-dimensional hyper-rectangle to give a single key.

```
procedure interbit( R:HyerRectangle ):key
begin
int cursor = 1, number = 2kB − 1;
    for(i = B-1; i ≥ 0; i −− )
    { cursor = cursor << i;
      for( int j = 0; j < k; j ++ )
      { key[number−−] = x_j^min & cursor >> i;
        key[number−−] = x_j^max & cursor >> i; }
      cursor = 1; }
    return key;
end;
```

Figure 1: C pseudo code for converting a k-dimensional hyper-rectangle R to a single bit-interleaved key.

Given a set D of hyper-rectangles on k-dimensional space, the collection of these hyper-rectangles is denoted by $D = \{R_1, R_2, \cdots, R_n\}$, where n is the number of hyper-rectangles in the set. For the i^{th} hyper-rectangle $R_i \in D$, let $(x_{ij}^{min}, x_{ij}^{max})$ denote the j^{th} side of hyper-rectangle R_i, $1 \leq j \leq k$ and $1 \leq i \leq n$. We denote by T the $2k$-d trie constructed by inserting all the hyper-rectangles in D into an initially empty trie. Given a node u in T, we denote by T_u the subtree of T rooted at u. There are altogether n leaves in T. Every leaf is associated with one hyper-rectangle. Figure 2 is an algorithm to insert one k-d hyper-rectangle into trie T. The height of the trie (i.e. the length of the key) is $2k$B. In the insertion loop from $2k$B-1 to 0, when a bit value equals 0, we go to the left branch and when the bit value equals 1, we go to the right branch of the trie. After preprocessing all n hyper-rectangles in D, we obtain the trie T, which allows us to carry out an orthogonal range search.

```
procedure Trie_Insert( R:HyperRectangle, T:TrieNode )
begin
    if( T == NULL )
    { T = new TrieNode(k);}
    Key* keyPoint=interbit(R);/*get the key after bit interleaving on hyper-rectangle R*/
    TrieNode* P = T;/* begin traversing*/
    for( int level = (2kB - 1); level ≥ 0; level -- )
   { if (keyPoint→getKey(level) )/* if the keyPoint bit on this level=1, go right*/
     {    if( P→right == NULL)
          {  p = new TrieNode( P, K );
             P→ Right = p;/*create a trie node and insert into right side of Trie T*/ }
          P = P→ Right /*continuing traversing on the right side*/; }
     else /*go left */
     {     if( P→Left = NULL )
           {  p = new TrieNode( P,K );
              P→ Left = p;/*create a trie node and insert into left side of Trie T;*/}
           P = P→ Left /*continuing traversing on the left side*/; }}
end;
```

Figure 2: C pseudo code for inserting a k-d hyper-rectangle R into a $2k$-d trie T.

Theorem 1 *$P(n,k) = O(nk)$ for a k-dimensional trie containing n hyper-rectangles.*

The proof is straightforward as each of the n inserted hyper-rectangles visits $2k$B nodes. Figures 3 and 4 give one example of building a binary trie from 15 2-d rectangles with number of data bits B= 5.

2.2 k-d Tries for spatial range search

The query hyper-rectangle $W = [L_1, H_1] \times [L_2, H_2] \times \cdots \times [L_k, H_k]$, which we abbreviate as $[L_j, H_j]^k$. For a hyper-rectangle $R_i \in D$, the set of k hyper-rectangle sides is

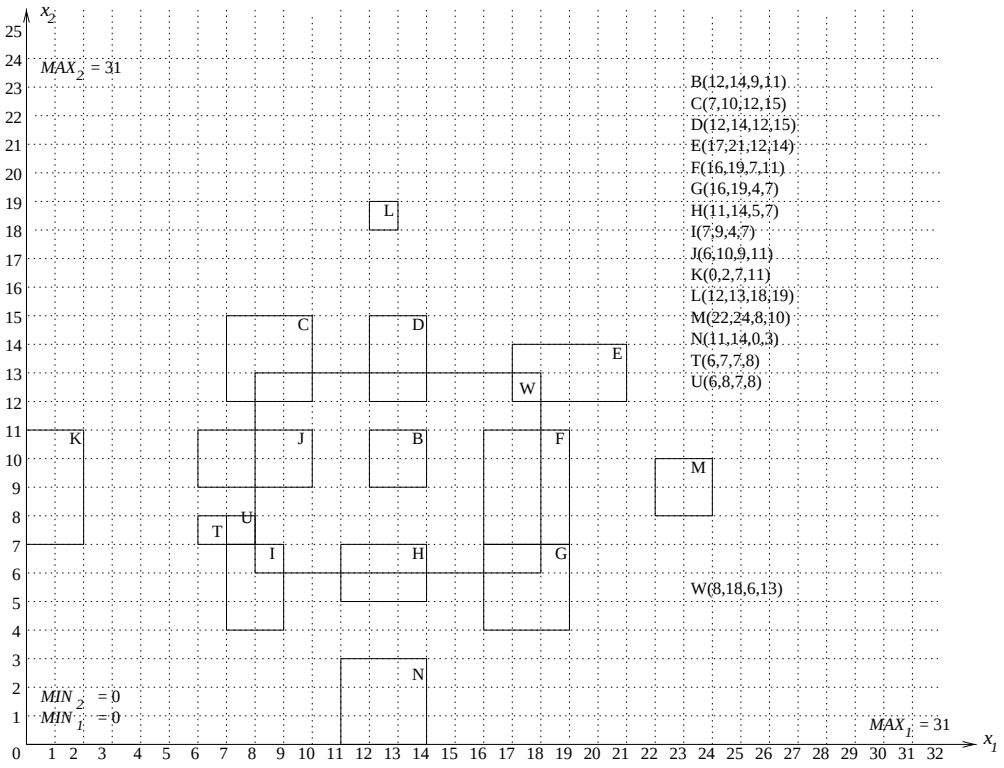

Figure 3: Example of 15 rectangles B, C, D, E, F, G, H, I, J, K, L, M, N, U, and T with a query hyper-rectangle W and number of data bits B= 5.

defined by $\{(x_{ij}^{min}, x_{ij}^{max}), \forall j \in \{1, \cdots, k\}\}$. We define $[MIN_j, MAX_j], \forall j \in \{1, \cdots, k\}$, as the minimum and maximum possible data coordinate values for dimension j. On every dimension, $MIN_j \leq x_{ij}^{min} \leq x_{ij}^{max} \leq MAX_j, \forall j \in \{1 \cdots k\}, i \in \{1 \cdots n\}$.

Definition 2 *Two hyper-rectangles intersect if and only if their sides on every dimension in the data space intersect, i.e.* $R_1 \cap R_2$ *is true, iff* $\forall j \in \{1, \cdots, k\}$, $(x_{1j}^{min}, x_{1j}^{max}) \cap (x_{2j}^{min}, x_{2j}^{max})$ *is true,* $x_{1j}^{min} \in [MIN_j, x_{2j}^{max})$ *and* $x_{1j}^{max} \in (x_{2j}^{min}, MAX_j]$.

This defines intersection strictly as an overlap in the sense of Allen [1983] and Egenhofer [1994]. Based on Definition 2, the hyper-rectangle R_i intersects W iff $x_{ij}^{min} \in [MIN_j, H_j)$ and $x_{ij}^{max} \in (L_j, MAX_j], \forall j \in \{1 \cdots k\}$.

k-dimensional orthogonal range search is performed using our $2k$-d trie for a query W. We use j as the index of the data space, $j \in \{1 \cdots k\}$, and we use p as the index for our problem space, $p \in \{1 \cdots 2k\}$. They are related as $j = p/2$. Each node in the trie T is a $2k$ range state; that is, every node has a cover space defined as $NC^{2k} = [L_p, U_p]^{2k}, 1 \leq p \leq 2k$. For a given query hyper-rectangle $W = [L_j, H_j]^k$, we obtain the query hyper-rectangle's cover space WC^{2k} and define it to be $WC^{2k} = \{([MIN_j, H_j - 1], [L_j + 1, MAX_j])\}^k$. There are three types of relations of WC^{2k} with NC^{2k}, which we call BLACK, GREY, and WHITE. Figure 5 below illustrates the three cases. Dashed lines are used for WC and solid lines for NC. After projecting the edge points to the problem space, we can

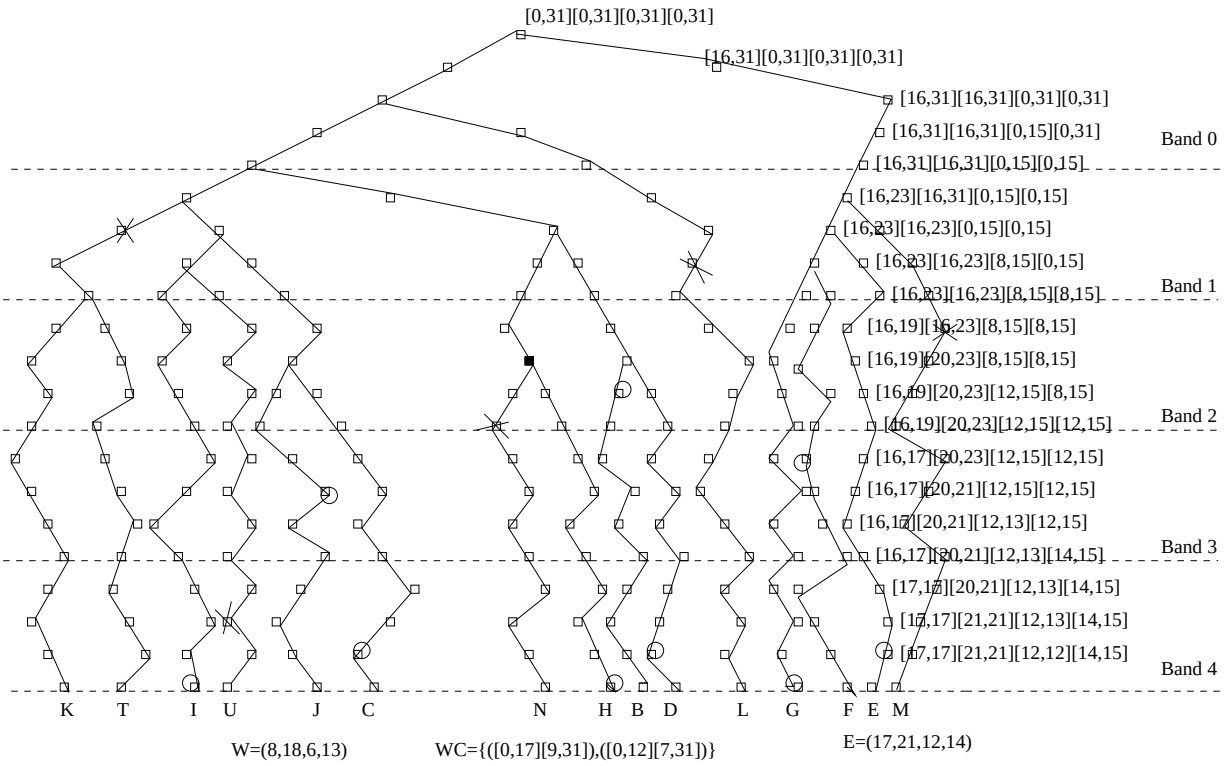

Figure 4: Example of a binary 4-d trie for the 2-d data of Figure 3. The list of 8-tuples near the right hand side is the cover space NC of each node on the trie path representing rectangle E.

distinguish the relation of a node's cover space NC with the query hyper-rectangle W's cover space WC on the p^{th} side.

For example, for the data of Figure 3 and Figure 4, $WC^4 = \{([0,17],[9,31]),([0,12],[7,31])\}$ for query hyper-rectangle W, nodes marked with a cross sign in Figure 4 are white, and nodes marked with a circle sign are black.

Definition 3 *If on all $2k$ dimensions, the NC to WC relationship satisfies $WC^p \cap NC^p = BLACK$, $\forall p \in \{1,2,\cdots,2k\}$, the node in the trie is a black node. If the NC to WC relationship satisfies $\exists p \in \{1,2,\cdots,2k\}$, such that $WC^p \cap NC^p = WHITE$, the node in the trie is a white node. All the other nodes are grey nodes, defined as follows: if $S \subseteq \{1,2,\cdots,2k\}$, $p \in S$, $WC^p \cap NC^p = GREY$, $\forall p \notin S$, $p \in \{1,2,\cdots,2k\}$, $WC^p \cap NC^p = BLACK$, the node will be grey.*

We use GN to denote the set of grey nodes in the trie, BN to denote the set of black nodes in the trie, and WN to denote the set of white nodes in the trie. Based on this definition, we can now define our k-d orgthogonal range search algorithm (see Figure 6).

The range search algorithm traverses from the root of trie T down to its leaves. We do a depth first traversal. At the root , level $\ell = 0$. For the root, the cover space NC^{2k} has $L_p = MIN_j$ and $U_p = MAX_j$, $\forall p \in \{1,2,\cdots,2k\}$. The cover space is split on the

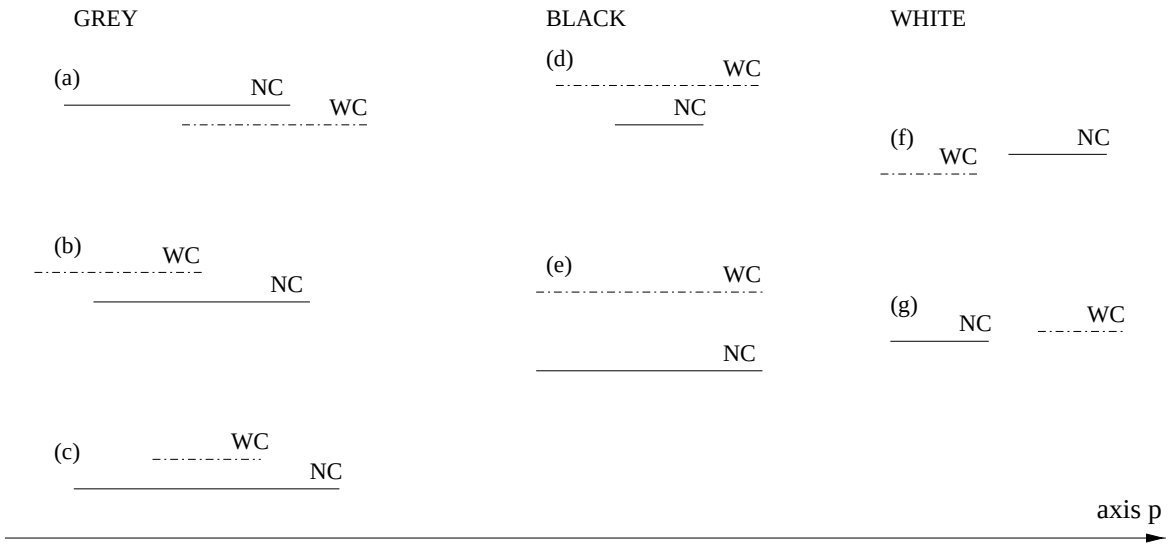

Figure 5: GREY ((a), (b), and (c)), BLACK ((d) and (e)), WHITE ((f) and (g)) relationships of a trie node cover space NC to a query hyper-rectangle cover space WC in dimension p of the $2k$-d problem space.

p^{th} coordinate as we move down, $p = \ell mod 2k$, $\forall \ell \in \{0, 1, \cdots, (2k\text{B}-1)\}$. If on the p^{th} dimension, a parent node T has cover space $[L, U]$, then T's left child's cover space is $[L, (L+U)/2]$ and T's right child's cover space is [(L+U)/2 +1, U]. Comparing a node's cover space NC (stored in L and U) with $[MIN_j, H_j)$ and $(L_j, MAX_j]$, if one of the $2k$ ranges falls outside (as determined by the InRange function), we encounter a white node and the search need not check any subtrees of T. If all the $2k$ ranges fall within NC, we encounter a black node. The nodes in the subtree of a black node are all black nodes. The nodes in the subtree of a white node are all white nodes. When we meet a white node or a black node, we stop traversing down; otherwise we continue splitting and traversing the subtrees. When we reach a black node u, all the leaves associated with a hyper-rectangle inside subtree T_u intersect the query hyper-rectangle W, and we get the range search's results and append them into the reporting List. All the leaves contained in the subtree of a white node do not intersect W. Integer arrays L and U store the lower and upper bounds of node T's cover space on $2k$ dimensions.

Figure 4 is the trie for the data shown in Figure 3. In Figure 4, nodes with a cross mark ($\times$) on them are white nodes. We know no children of the white node can intersect the query hyper-rectangle W, so the subtree attached to the white node is pruned from the search space. The node with a circle sign ($\bigcirc$) on it stands for a black node, which means all hyper-rectangles represented by leaves inside the subtree attached to the black node intersect W. For query hyper-rectangle $W = [8, 18] \times [6, 13]$, $k = 2$, the query hyper-rectangle's cover space $WC^4 = [0, 17] \times [9, 31] \times [0, 12] \times [7, 31]$. The hyper-rectangle denoted as $E = [17, 21] \times [12, 14]$ has its cover space NC^4, the four ranges, listed along the right side of Figure 4. We do half splitting continuously in $2k$ space as we move down the trie. The trie is divided into B bands, each band of height $2k$ (see Figure 4). When the first band $2k$ half-splits are finished, we begin the second band, till the $(\text{B}-1)^{th}$ band.

```
procedure Rangesearch( T:TrieNode, level:int, L:int*, U:int*, int* RI, W:HyperRectangle,
S:SearchSpace, List:HyperRectangleList )
/*S=[MIN_j, MAX_j], ∀j ∈ {1,···,k}, L and U are arrays defining node T's cover space
on k dimensions, RI is an array containing the color of the 2k relationships of NC and
WC at node T*/
begin
    int LT[2*K], UT[2*K], RIT[2*K]; /* internal varibles for recursive call */
    for( int i = 0; i < 2*K; i ++ )
    { LT[i] = L[i]; UT[i] = U[i]; RIT[i] = RI[i]}
    if( (T≠NULL) AND (level ≤ 2kB) )
    {  if( level == 0 ){ p = 0;}
       else{ p = (level - 1) % (2k); }
       /*p is used for cover space's index in L[p], U[p], RIT[p], p=0..2k-1*/
       /*depth first traversal*/
       RIT[p]=InRange(L[p], U[p], level, W, S);/*get the color of T on p^th dimension */
       if( RIT[p] ! = -1 ) /* if not white */
       { int color = NodeColor(RIT); /* determine the color for node T*/
         if( color == 0 ) /*the node is grey node*/
         {   /*continue traversing*/
             level ++;
             p = (level - 1) % (2k);
             if( T→Left ≠ NULL )/*left half splitting*/
             { LT[p] = L[p]; UT[p] = (L[p] + U[p])/2;
               Rangesearch( T→Left, level, LT, UT, RIT, W, S, List );}
             if( T→Right ≠ NULL )/*right half splitting*/
             { LT[p] = ( L[p] + U[p] )/2 +1; UT[p] = U[p];
               Rangesearch( T→Right, level, LT, UT, RIT, W, S, List );}
         }else if( color == 1 )/* the node is black node */
         { Collect(T,List);/*collect all the leaves inside the black node intersecting W*/}
       }
        /* or else if the node is white, unfold the recursion */ }
end;
```

Figure 6: C pseudo code for the k-d orthogonal range search algorithm. InRange(L[p],U[p],level,W,S) and NodeColor(RIT) are functions to decide the color of NC and WC relationships for node T.

For hyper-rectangle E's case, traversal of T during the 2-d range search for rectangles intersecting W stopped at the last $(B-1)^{th}$ band, which is a black node. If we continue one extra step to the leaf level, we will get $NC^4 = [17,17] \times [21,21] \times [12,12] \times [14,14]$, and the lower bound and upper bound of every $2k$ range is equal to the coordinate value of hyper-rectangle E in the data space.

3 Space Analysis

The full trie T's depth is a constant value if the number of dimensions k and coordinate range on every dimension are considered constant. If we assume the coordinate ranges on every dimension are the same, that is $[MIN, MAX]$, B is the number of bits for representing the coordinate value inside the coordinate range. We have T's height $= 2kB = 2k\log_2(MAX - MIN + 1)$, and level $\ell \in \{0, \cdots, 2kB - 1\}$. At the root level $\ell = 0$, and at the leaf level $\ell = 2kB - 1$. n is the size of input data set D. We divide the trie into B bands, each of height $2k$.

Given any coordinate value $u \in [MIN, MAX]$, the bit distance between u and another value $v \in [MIN, MAX]$ is denoted as $bdis(u, v)$. $bdis(u, v)$ is defined as the path length from P (the common ancestor of u and v) down to the leaf level of u or v in the bit distance tree V. For example, $bdis(12, 12) = 0$, $bdis(12, 13) = 1$, $bdis(12, 11) = 3$, $bdis(31, 32) = 6$, $bdis(31, 19) = 4$, $bdis(7, 19) = 5$ (see Figure 7).

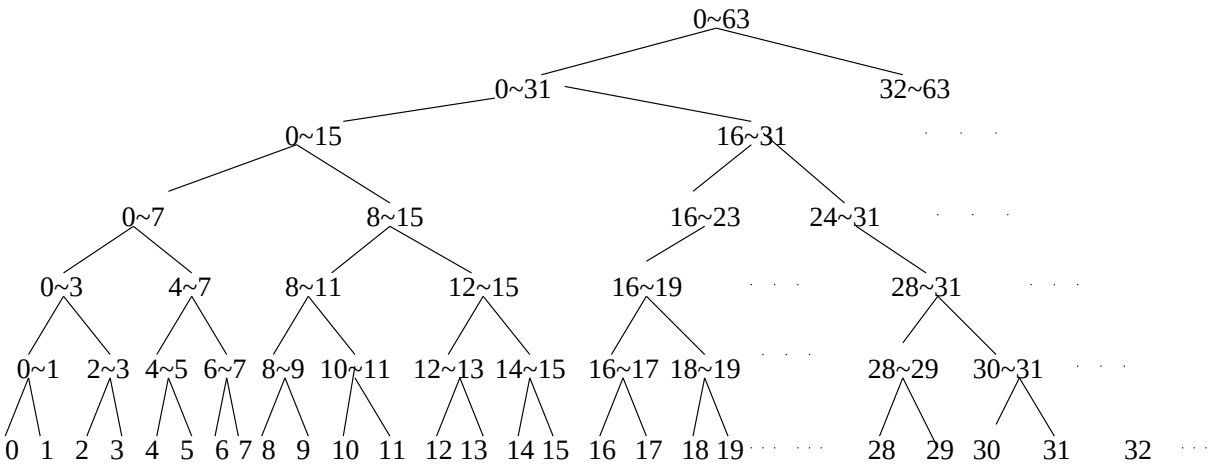

Figure 7: Example of bit distance tree V for B = 6.

We use $max\{\}^k$ to denote the maximum value of the set inside braces on k dimensions, and we have the following lemma:

Lemma 1 *For any two hyper-rectangles $R_1 = (x_{1j}^{min}, x_{1j}^{max})^k$ and $R_2 = (x_{2j}^{min}, x_{2j}^{max})^k$, their nearest common ancestor lies in the band $B - max\{bdis(x_{1j}^{min}, x_{2j}^{min}), bdis(x_{1j}^{max}, x_{2j}^{max})\}^k, j \in \{1, 2, \cdots, k\}$.*

Proof. Bit distance tree V with maximum bit distance B is only for coordinate values one one dimension in $2k$ dimensions. The height of bit distance tree V is B. Each node's cover space is one-half of it's parents' cover space in V. From the root whose domain is [0,2^B-1] down to u and v's nearest common ancester node, u and v share the same path. The path length from leaf level to the nearest common ancestor node P is bdis(u, v). The level of P is B - bdis(u, v). From root level 0 to level B - bdis(u, v), u and v's first B - bdis(u, v) bits are identical. From level B - bdis(u, v) down to leaf level B, the bit

sequence for u and v is different. Our trie is a $2k$-d trie, and each of the $2k$ ranges will half split alternatively. From the root, whose cover is $[0, 2^B - 1]^{2k}$, u_1 will separate with v_1 on the band of B - bdis(u_1, v_1), u_2 will separate with v_2 on the band of B - bdis(u_2, v_2), $\cdots$, u_{2k} will separate with v_{2k} on the band of B - bdis(u_{2k}, v_{2k}). That is, u and v's nearest common ancestor node lies in the band of B - $\max\{bdis(u_p, v_p)\}^{2k}$. ∎

For example, as shown in Figure 3 and Figure 4 where $k = 2$, rectangle $N(11, 14, 0, 3)$ and rectangle $H(11, 14, 5, 7)$, we compare $2k$ pairs of bit distance: $bdis(11, 11) = 0$, $bdis(14, 14) = 0$, $bdis(0, 5) = 3$, $bdis(3, 7) = 3$. The maximum bit distance among N and H is 3, so N and H's nearest common ancestor lies in the band of B - 3 = 2. This common ancestor node is indicated in Figure 4 by a black filled in square in Band 2.

We use lemma 1 to compute the storage space for our trie.

Theorem 2 *The expected space $S(n, k)$ for a binary trie containing n k-dimensional hyper-rectangles is $O(kn)$.*

Proof. After the first two hyper-rectangles R_1 and R_2 are inserted, the number of nodes they share will be $2k(\mathrm{B}-E\{max\{bdis(x_{1j}^{min}, x_{2j}^{min}), bdis(x_{1j}^{max}, x_{2j}^{max})\}^k\})$. The separate branches for both will occupy $2kE\{max\{bdis(x_{1j}^{min}, x_{2j}^{min}), bdis(x_{1j}^{max}, x_{2j}^{max})\}^k\}$ nodes. The expected total number of nodes occupied by two hyper-rectangles will be:

$$\begin{aligned}
&2k(\mathrm{B}-E\{max\{bdis(x_{1j}^{min}, x_{2j}^{min}), bdis(x_{1j}^{max}, x_{2j}^{max})\}^k\}) \\
&\qquad +2kE\{max\{bdis(x_{1j}^{min}, x_{2j}^{min}), bdis(x_{1j}^{max}, x_{2j}^{max})\}^k\}2 \\
&= 2k\mathrm{B}+2kE\{max\{bdis(x_{1j}^{min}, x_{2j}^{min}), bdis(x_{1j}^{max}, x_{2j}^{max})\}^k\}.
\end{aligned}$$

Adding a third hyper-rectangle R_3, the nearest common ancestor of the three of them would be in the band:

$$\begin{aligned}
&\mathrm{B}-E\{max\{bdis(x_{1j}^{min}, x_{2j}^{min}), bdis(x_{1j}^{min}, x_{3j}^{min}), bdis(x_{2j}^{min}, x_{3j}^{min}), \\
&\qquad bdis(x_{1j}^{max}, x_{2j}^{max}), bdis(x_{1j}^{max}, x_{3j}^{max}), bdis(x_{2j}^{max}, x_{3j}^{max})\}^k\}
\end{aligned}$$

The expected number of nodes in the trie is now

$$\begin{aligned}
&2k\mathrm{B}+2kE\{max\{bdis(x_{1j}^{min}, x_{2j}^{min}), bdis(x_{1j}^{max}, x_{2j}^{max})\}^k\} \\
&\quad +2kE\{max\{bdis(x_{1j}^{min}, x_{2j}^{min}), bdis(x_{1j}^{min}, x_{3j}^{min}), bdis(x_{2j}^{min}, x_{i3j}^{min}), \\
&\qquad bdis(x_{1j}^{max}, x_{2j}^{max}), bdis(x_{1j}^{max}, x_{3j}^{max}), bdis(x_{2j}^{max}, x_{3j}^{max})\}^k\}.
\end{aligned}$$

No extra storage is needed for common nodes. We use $max_i\{\}^k$ to denote the maximum bit distance of hyper-rectangles, where i is the number of hyper-rectangles. There are $2k\begin{pmatrix} i \\ 2 \end{pmatrix}$ bit distances from which to determine $max_i\{\}^k$. Using induction on i, we obtain the total number of nodes in the trie for n input hyper-rectangles as

$$S(n, k) = 2k\mathrm{B}+2kE\{max_2\{\}^k\}+2kE\{max_3\{\}^k\}+2kE\{max_4\{\}^k\}+\cdots+2kE\{max_n\{\}^k\} \tag{1}$$

For any given value $x_{1p} \in \{0, 1, 2, \cdots, 2^B - 1\}$, the value $x_{2p} \in \{0, 1, 2, \cdots, 2^B - 1\}$ has a bit distance relation with x_{1p} as follows: if $bdis(x_{1p}, x_{2p}) = 0$, the number of possible x_{2p} values $= 1$; if $bdis(x_{1p}, x_{2p}) = 1$, the number of possible x_{2p} values $= 2^0 = 1$; if $bdis(x_{1p}, x_{2p}) = 2$, the number of possible x_{2p} values $= 2^1 = 2$; if $bdis(x_{1p}, x_{2p}) = 3$, the number of possible x_{2p} values $= 2^2 = 4, \cdots$, if $bdis(x_{1p}, x_{2p}) =$B, the number of possible x_{2p} values $= 2^{B-1}$. The bit distance between any two x_{ij} values is one of $\{0, 1, \cdots,$B$\}$.

The probability for a certain bit distance would be $Pr\{bdis(x_{1p}, x_{2p}) = 0\} = \frac{1}{2^B}$, $Pr\{bdis(x_{1p}, x_{2p}) = 1\} = \frac{1}{2^B}$, $Pr\{bdis(x_{1p}, x_{2p}) = 2\} = \frac{2}{2^B}$, $\cdots$, $Pr\{bdis(x_{1p}, x_{2p}) = a\} = \frac{2^{a-1}}{2^B}$, $\cdots$, $Pr\{bdis(x_{1p}, x_{2p}) =B\} = \frac{2^{B-1}}{2^B}$.

The probability for two hyper-rectangles' maximum bit distance value in the $2k$ problem space would be $Pr\{max_2\{bdis(x_{1p}, x_{2p})\}^{2k} = 0\} = (\frac{1}{2^B})^{2k}$, because the hyper-rectangles are independently distributed in $2k$ dimensions. If we want the maximum bit distance to be 0, then the bit distances on all $2k$ dimensions should be zero. On one dimension p, the probability that $bdis(x_{1p}, x_{2p}) \leq a$ is $\frac{1}{2^B} + \Sigma_{q=1}^{a-1} \frac{2^{q-1}}{2^B}$. We obtain the following probabilities:

$$Pr\{max_2\{bdis(x_{1p}, x_{2p})\}^{2k} = 1\} = \Sigma_{j=1}^{2k} \begin{pmatrix} 2k \\ j \end{pmatrix} (\tfrac{1}{2^B})^j (\tfrac{1}{2^B})^{2k-j},$$

$$\vdots$$

$$Pr\{max_2\{bdis(x_{1p}, x_{2p})\}^{2k} = a\} = \Sigma_{j=1}^{2k} \begin{pmatrix} 2k \\ j \end{pmatrix} (\tfrac{2^{a-1}}{2^B})^j (\tfrac{1}{2^B} + \Sigma_{q=1}^{a-2} \tfrac{2^{q-1}}{2^B})^{2k-j},$$

$$\vdots$$

$$Pr\{max_2\{bdis(x_{1p}, x_{2p})\}^{2k} =\text{B}\} = \Sigma_{j=1}^{2k} \begin{pmatrix} 2k \\ j \end{pmatrix} (\tfrac{2^{B-1}}{2^B})^j (\tfrac{1}{2^B} + \Sigma_{q=1}^{a-2} \tfrac{2^{q-1}}{2^B})^{2k-j}.$$

When the maximum bit distance equals to a, we want one bit distance among 2k values to be a, and all the others should be equal or less than a. This problem can be calculated as follows: there exist j pairs in $2k$ pairs with bit distance equals to a, $(2k - j)$ pairs would have bit distance values $< a$ or $\leq (a - 1)$. This leads to

$$\begin{aligned} &Pr\{max_2\{bdis(x_{1p}, x_{2p})\}^{2k} = a\} \\ &= \Sigma_{j=1}^{2k} \begin{pmatrix} 2k \\ j \end{pmatrix} (\tfrac{2^{a-1}}{2^B})^j (\tfrac{1}{2^B} + \Sigma_{q=1}^{a-2} \tfrac{2^{q-1}}{2^B})^{2k-j} \\ &= \Sigma_{j=1}^{2k} \begin{pmatrix} 2k \\ j \end{pmatrix} (\tfrac{2^{a-1}}{2^B})^{2k} \end{aligned}$$

because $\Sigma_{j=1}^{2k} \begin{pmatrix} 2k \\ j \end{pmatrix} = 2^{2k} - 1$

$$Pr\{max_2\{bdis(x_{1p}, x_{2p})\}^{2k} = a\} = (\tfrac{2^{a-1}}{2^B})^{2k}(2^{2k} - 1).$$

For three hyper-rectangles' maximum bit distance value in $2k$ problem space, we compare $\begin{pmatrix} 3 \\ 2 \end{pmatrix} 2k$ pairs of coordinate values. The function for the probability of three hyper-rectangles with maximum bit distance value equal to a is:

$$Pr\{max_3\{\}^{2k} = a\} = \Sigma_{j=1}^{3\cdot 2k} \left(\begin{array}{c} 3\cdot 2k \\ j \end{array} \right) (\frac{2^{a-1}}{2^B})^j (\frac{1}{2^B} + \Sigma_{q=1}^{a-2} \frac{2^q}{2^B})^{3\cdot 2k-j}$$

$$= \Sigma_{j=1}^{3\cdot 2k} \left(\begin{array}{c} 3\cdot 2k \\ j \end{array} \right) (\frac{2^{a-1}}{2^B})^{3\cdot 2k}$$

$$= (\frac{2^{a-1}}{2^B})^{3\cdot 2k} (2^{3\cdot 2k} - 1)$$

The general function for the probability of i random hyper-rectangles in $2k$ space is as follows:

$$Pr\{max_i\{\}^{2k} = a\} = \Sigma_{j=1}^{\left(\begin{array}{c} i \\ 2 \end{array} \right) 2k} \left(\begin{array}{c} \left(\begin{array}{c} i \\ 2 \end{array} \right) 2k \\ j \end{array} \right) (\frac{2^{a-1}}{2^B})^j [(\frac{1}{2^B}) + \Sigma_{q=1}^{a-2} (\frac{2^{q-1}}{2^B})] \left(\begin{array}{c} i \\ 2 \end{array} \right)^{2k-j}$$

$$= \Sigma_{j=1}^{i(i-1)k} \left(\begin{array}{c} i(i-1)k \\ j \end{array} \right) (\frac{2^{a-1}}{2^B})^{i(i-1)k}$$

$$= (\frac{2^{a-1}}{2^B})^{i(i-1)k} (2^{i(i-1)k} - 1) \qquad (2)$$

Using equation (2) in equation (1) gives the expected number of nodes in the trie as

$S(n,k) = 2k\mathrm{B} + 2k\Sigma_{i=2}^{n} (\Sigma_{a=1}^{B} a (\frac{2^{a-1}}{2^B})^{i(i-1)k} (2^{i(i-1)k} - 1))$

Letting $u = 2^{i(i-1)k}$, we have

$S(n,k) = 2k\mathrm{B} + 2k\Sigma_{i=2}^{n} \Sigma_{a=1}^{B} a u^a \frac{u-1}{u \cdot u^B}$

$I = \Sigma_{a=1}^{B} a u^a$

$I = 1u + 2u^2 + 3u^3 + \cdots + \mathrm{B}u^B$

$\frac{I}{u} = 1 + 2u + 3u^2 + \cdots + \mathrm{B}u^{B-1}$

$\frac{I}{u} - I = 1 + u + u^2 + \cdots + u^{B-1} - \mathrm{B}\,u^B$

$I(\frac{1}{u} - 1) = \frac{1-u^B}{1-u} - \mathrm{B}u^B$

$I = (\frac{1-u^B}{1-u} - \mathrm{B}u^B) \frac{u}{1-u}$, and

$S(n,k) = 2k\mathrm{B} + 2k\Sigma_{i=2}^{n} \frac{u-1}{u} \frac{1}{u^B} \frac{u}{1-u} (\frac{1-u^B}{1-u} - \mathrm{B}u^B)$

$= 2k\mathrm{B} + 2k\Sigma_{i=2}^{n} \frac{1}{u^B} (\mathrm{B}u^B - \frac{1-u^B}{1-u})$

$= 2k\mathrm{B} + 2k\Sigma_{i=2}^{n} (\mathrm{B} - \frac{\frac{1}{u^B} - 1}{1-u})$

$= 2k\mathrm{B} + 2k\Sigma_{i=2}^{n} \mathrm{B} - 2k\Sigma_{i=2}^{n} \frac{1}{u-1} + 2k\Sigma_{i=2}^{n} \frac{1}{u^B(u-1)}$

$= 2k\mathrm{B} + 2k\Sigma_{i=2}^{n} \mathrm{B} - o(1) + 2ko(1)$

The second item dominates, and as we consider B a constant, then

$S(n,k) = O(kn)$ ∎

Theorem 3 *The space for a k-d trie containing n k-d hyper-rectangles is* $S(n,k) = O(kn)$.

Proof. The worst case space requirement occurs if every input hyper-rectangle, after bit interleaving, will occupy as many different paths as possible; that is, their maximum bit distance value would be as large as possible. In this case, the trie will occupy the largest number of nodes. As $n \leq 2^{2kB-1}$, then from the root to level $r = \lceil \log_2 n \rceil - 1$, the binary trie will be complete. At level $\lceil \log_2 n \rceil$, the full binary tree stops and the remaining paths will be single branches dangling from the last full binary tree's level, as shown in Figure 8. From root level to level $\lceil \log_2 n \rceil - 1$, the worst case trie storage will be the same as a full binary tree. From level $\lceil \log_2 n \rceil$ to the leaf level, the trie is slimmer than the triangle of a full binary tree (drawn in dashed lines in Figure 8). The number of nodes in the trie is thus

$$\begin{aligned} S(n,k) &= 1 + 2 + 2^2 + \cdots + 2^{\lceil \log_2 n \rceil - 1} + (2k\mathrm{B} - \lceil \log_2 n \rceil)n \\ &= 2^{\lceil \log_2 n \rceil} - 1 + n(2k\mathrm{B} - \lceil \log_2 n \rceil) \\ &= (2k\mathrm{B}+1)n - n \log n = O(kn), \text{ as } \log_2 n \leq 2k\mathrm{B}. \quad \blacksquare \end{aligned}$$

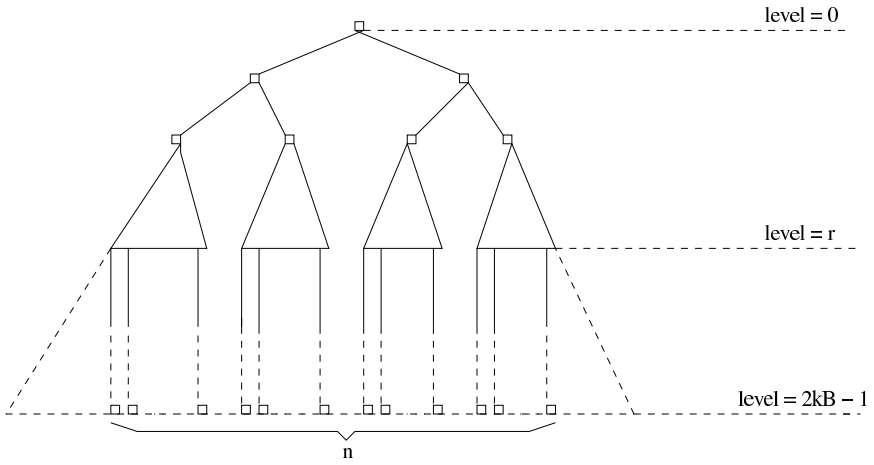

Figure 8: Worst case for storage in a $2k$-d trie.

4 Range Search Cost

Without loss of generality, we consider our analysis in real $[0,1]^k$ space.

4.1 Partial Match Retrieval Using Tries

The analysis of partial match retrieval using k-d tries was eloquently addressed by Flajolet and Peuch [12]. Adapting this analysis to k-d hyper-rectangle in $2k$-d tries,

we ask for all hyper-rectangles in data set D satisfying query hyper-rectangle q, where q= $(q_1, q_2, \cdots, q_{2k})$, $S \subset \{1, 2, \cdots, 2k\}$, s = $|S|$, s of the $2k$ query key values are specified, and $2k - s$ query values are left unspecified; we denote the unspecified queries as $q_p = *$, $p \notin S$ as wild cards. Hyper-rectangle R_i satisfies a partial match query q if and only if $q_p = R_{ip}, \forall p \in S$.

Our analysis is based on the uniform probabilistic model where we assume the input data D and the query hyper-rectangle W are drawn from a uniform random distribution, and that the keys of D are independent. Here, we assume that the keys of D are $2k$-dimensional points representing a k-dimensional hyper-rectangle. A k-d random variable **x** exists for the left side x_{ij}^{min} of each hyper-rectangle R_i, and a second k-d random variable **y** exists for the right side x_{ij}^{max} of each R_i. The $2k$-d keys of D are thus formed from two independent random variables, and the joint density function **t** of each key is the product of the density functions of **x** and **y**. The product of two uniform random density functions gives a uniform random density function **t**, which allows us to use the following theorem:

Theorem 4 *(Flajolet and Peuch [1986]) Given a binary $2k$-d trie T containing a set of k-dimensional input hyper-rectangles $D = \{R_1, \cdots, R_n\}$, assuming data set D and query hyper-rectangle q satisfy the uniform probabilistic model, q= $(q_1, q_2, \cdots, q_{2k})$, $S \subset \{1, 2, \cdots, 2k\}$, the average cost of patial match retrieval $Q_S(n, k)$ measured by the number of nodes traversed in trie T is*

$$Q_S(n, k) = c(\tfrac{1}{2k} \log_2 n) n^{1 - \frac{s}{2k}} + O(1),$$

where c is a constant determined by the the indices of S, $s = |S|$.

Proposition 1 *Given a binary trie T containing a set of k-dimensional input hyper-rectangles $D = \{R_1, \cdots, R_n\}$, assuming input data set D and query hyper-rectangle q satisfying the uniform probabilistic model, q= $(q_1, q_2, \cdots, q_{2k})$, $S \subset \{1, 2, \cdots, 2k\}$, the cost of patial match retrieval $Q_S(n, k)$ measured by the number of nodes traversed in trie T is*

$$Q_S(n, k) = E\{\Sigma_{t=1}^{number\ of\ nodes\ in\ trie} \prod_{p \in S} |NC_t^p|\}$$

Proof. If a node is visited, $q_p \in NC^p, \forall p \in S$. The probability that a node in trie T will be visited is determined by the volume of every node's cover space in the space $[0, 1]$. ∎

4.2 Analysis of Orthogonal Range Search Using Tries

For a given query hyper-rectangle $W = [L_j, H_j]^k$, and for any hyper-rectangle $R_i = [x_{ij}^{min}, x_{ij}^{max}]^k$, if R_i satisfies Lemma 2, the range search for a single result traverses until the leaf level is reached. We use $min\{\}^k$ to denote the minimum value of a set inside braces on k dimensions.

Lemma 2 *If the minimum bit distance of R_i with the query cover space $([MIN_j, H_j - 1] \times [L_j + 1, MAX_j])^k$ is equal to zero or one, we will search R_i till reaching the last band B-1 of R_i . That is, $min\{bdis(x_{ij}^{min}, H_j - 1), bdis(x_{ij}^{max}, L_j + 1)\}^k = 0$ or $1, \forall j \in \{1, 2, \cdots, k\}$.*

Proof. If anyone of the k dimensions has $bdis = 0$ or 1, then the worst case for a single hyper-rectangle happens. For any two hyper-rectangles, their nearest common ancester will lie in the band of B $-max\{bdis(x_{ij}^{min}, H_j - 1), bdis(x_{ij}^{max}, L_j + 1)\}$, $\forall j \in \{1, 2, \cdots, k\}$. If all n input hyper-rectangles R_i are aligned such that the minimum bit distance of R_i with W is 0 or 1, then the worst case for orthogonal range search occurs. ∎

Figure 9 shows an example of the worst case for 2-d orthogonal range search. Searching will go down to all hyper-rectangles' leaf level in the trie T. The hyper-rectangle with dashed lines is the query hyper-rectangle W. For the example shown in Figure 3 and Figure 4, rectangles C, D, E, G, H, I, U correspond to the worst case which needs traversal to the last band.

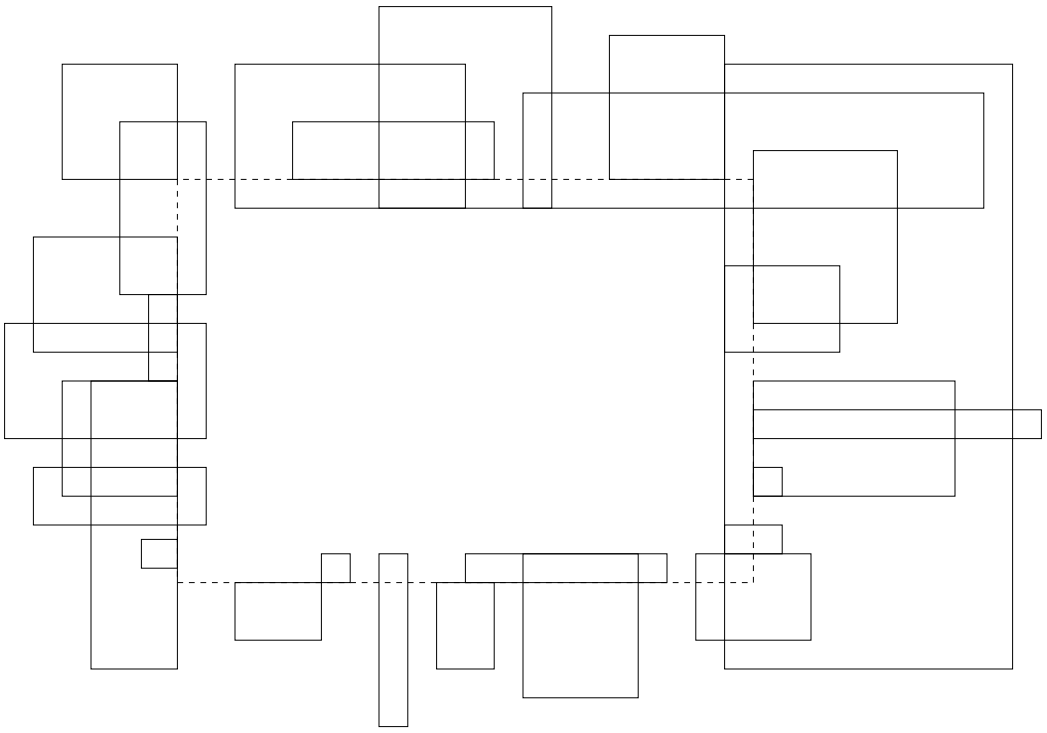

Figure 9: Example of worst case for 2-d orthogonal range search in a 4-d trie.

To get the color types for a node in the trie, we compare all the $2k$ ranges of WC^{2k} with NC^{2k}. In our algorithm, on each level, we do one comparison of the $2k$ ranges and store the color as the node's state. If all $2k$ ranges are black, the node is black; if one range is white, the node is white, and all the other conditions indicate the node is a grey node. Traversing from the root down through the first $2k$ levels, we finish the comparison of $2k$ ranges of WC^{2k} and NC^{2k} on $2k$ dimensions and get the first batch of white and black nodes. On a certain level ℓ in the trie T, after half splitting the cover space from the root (which is the full search space), there are altogether $2^{\frac{\ell}{2k}+1}$ possiblities of the ranges for NC^p. They are $[MIN, MIN+2^{B-\frac{\ell}{2k}-1}], [MIN+2^{B-\frac{\ell}{2k}-1}, MIN+22^{B-\frac{\ell}{2k}-1}], \cdots, [MIN+(2^{\frac{\ell}{2k}+1}-1)2^{B-\frac{\ell}{2k}-1}, MAX]$. Each range's length equals to $2^{B-\frac{\ell}{2k}-1}$, $|NC^p| = 2^{B-\frac{\ell}{2k}-1}$, and the exact NC^p we want to compare with WC^p on this node is determined by previous paths. On level ℓ, the query cover space WC^p we want to compare with is on the j^{th}

dimension in data space, $j = (\ell \, mod(2k))$. If $j \, mod 2 = 0$, we select $WC^p = [MIN, H_j)$; if $j \, mod 2 = 1$, $WC^p = (L_j, MAX]$.

Traversing stops on paths when we meet with black or white nodes and continues when grey nodes are encountered and continues collecting black nodes in the subtree of the black nodes we first met. The time complexity can be determined by computing the number of grey and black nodes in the trie built from input data D. We have the following equation:

$$Q(n,k) = \Sigma_{t=1}^{the\ number\ of\ nodes\ in\ the\ trie} 1_{[node_t \in GN \cup BN]}$$

where we use $1_{[A]}$ as the characteristic function of the event A. The formula counts the number of grey nodes, which, apart from the black nodes traversed to report the in-range hyper-rectangles, represents the time complexity of the orthogonal range search algorithm (see Figure 10).

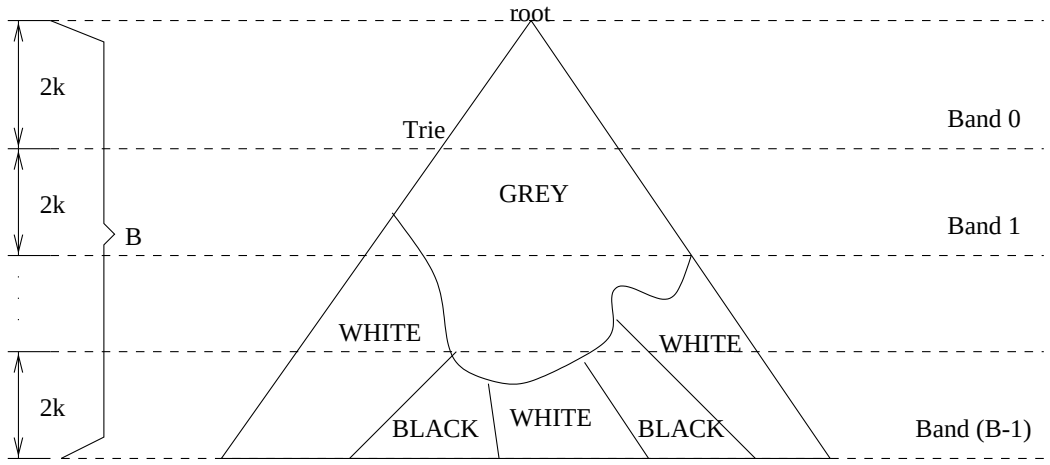

Figure 10: GREY nodes for computing the time complexity.

Lemma 3 $\quad E\{\Sigma_{t=1}^{number\ of\ nodes\ in\ trie} \Pi_{p=1}^{2k} |NC_t^p|\} = O(\log_2 n)$

Proof. Every band has height $2k$. In the b^{th} band of height $2k$ ($h = 1..2k$), we have

$h = 1$, $|NC_{b1}^1| = \frac{1}{2^{b+1}}$, $|NC_{b1}^2| = |NC_{b1}^3| = \cdots = |NC_{b1}^{2k}| = \frac{1}{2^b}$

$h = 2$, $|NC_{b2}^1| = |NC_{b2}^2| = \frac{1}{2^{b+1}}$, $|NC_{b2}^3| = |NC_{b2}^4| = \cdots = |NC_{b2}^{2k}| = \frac{1}{2^b}$

$\vdots$

$h = 2k$, $|NC_{b2k}^1| = |NC_{b2k}^2| = |NC_{b2k}^3| = \cdots = |NC_{b2k}^{2k}| = \frac{1}{2^{b+1}}$

where we have used the notation $|NC_{bh}^p|$ to represent the range for one dimension p at a node of height h within band b. This leads to the volume of one node's data cover $|NC_t|$ on $2k$ dimensions as

$$|NC_t| = \prod_{p=1}^{2k} |NC_t^p| = (\frac{1}{2^{b+1}})^h (\frac{1}{2^b})^{2k-h} = 2^{-2kb-h} \quad (1)$$

for node t lying at height h in band b.

The volume $|NC_t|$ is the same for all nodes on the same level in the trie; as the level ℓ increases, the value of $|NC_t|$ decreases.

We calculate the number of nodes on a certain height h in one band using the worst case for storage (shown in Figure 8). The level $r = \lfloor \log_2 n \rfloor$ (the last level in the trie that concides with a complete binary tree, see Figure 8) lies in the band $d = \lfloor \frac{\lfloor \log_2 n \rfloor}{2k} \rfloor$, which is at height $h = \lfloor \log_2 n \rfloor + 1 - 2kd$ in the d^{th} band. We divide B bands into three parts: (a) from band 0 to band $d-1$, (b) band d, and (c) from band $d+1$ to band B-1. Within part(a), the number of nodes on height h of band b is 2^ℓ, $\ell = 2kb+h-1$. We divide band d into two: one is from height 1 to height $\lceil \log_2 n \rceil + 1 - 2kd$, the other is from height $\lceil \log_2 n \rceil + 2 - 2kd$ to $2k$. Within part(b)'s first part, the number of nodes on height h of band d is $2^{2kd+h-1}$, within band d's second part, the number of nodes on height h of band d is n. The number of nodes on the height h within part(c) is n. Thus, we have

$$\begin{aligned}
&E\{\Sigma_{t=1}^{number\ of\ nodes\ in\ trie} \prod_{p=1}^{2k} |NC_t^p|\} \\
&= E\{\Sigma_{b=0}^{B-1} \Sigma_{h=1}^{2k} \Sigma_{t=1}^{number\ of\ nodes\ on\ height\ h\ in\ band\ b} \prod_{p=1}^{2k} |NC_{bh}^p|\} \\
&= \Sigma_{b=0}^{\lfloor \frac{\lfloor \log_2 n \rfloor}{2k} \rfloor - 1} \Sigma_{h=1}^{2k} \Sigma_{t=1}^{2^{2kb+h}} \prod_{p=1}^{2k} |NC_{bh}^p| && \text{part(a)} \\
&\quad + \Sigma_{b=\lfloor \frac{\lfloor \log_2 n \rfloor}{2k} \rfloor}^{\lfloor \frac{\lfloor \log_2 n \rfloor}{2k} \rfloor} \Sigma_{h=1}^{\lfloor \log_2 n \rfloor + 1 - 2kb} \Sigma_{t=1}^{2^{2kb+h}} \prod_{p=1}^{2k} |NC_{bh}^p| && \text{part(b)1} \\
&\quad\quad + \Sigma_{b=\lfloor \frac{\lfloor \log_2 n \rfloor}{2k} \rfloor}^{\lfloor \frac{\lfloor \log_2 n \rfloor}{2k} \rfloor} \Sigma_{h=\lfloor \log_2 n \rfloor + 2 - 2kb}^{2k} \Sigma_{t=1}^{n} \prod_{p=1}^{2k} |NC_{bh}^p| && \text{part(b)2} \\
&\quad + \Sigma_{b=\lfloor \frac{\lfloor \log_2 n \rfloor}{2k} \rfloor + 1}^{B-1} \Sigma_{h=1}^{2k} \Sigma_{t=1}^{n} \prod_{p=1}^{2k} |NC_{bh}^p| && \text{part(c)}
\end{aligned}$$

Because $\Sigma_{t=1}^{2^{2kb+h}} \prod_{p=1}^{2k} |NC_{bh}^p| = \Sigma_{t=1}^{2^{2kb+h}} 2^{-2kb-h} = \frac{1}{2}$,

$$\begin{aligned}
&E\{\Sigma_{t=1}^{number\ of\ nodes\ in\ trie} \prod_{p=1}^{2k} |NC_t^p|\} \\
&= \Sigma_{b=0}^{\lfloor \frac{\lfloor \log_2 n \rfloor}{2k} \rfloor - 1} \Sigma_{h=1}^{2k} \frac{1}{2} \\
&\quad + \Sigma_{b=\lfloor \frac{\lfloor \log_2 n \rfloor}{2k} \rfloor}^{\lfloor \frac{\lfloor \log_2 n \rfloor}{2k} \rfloor} \Sigma_{h=1}^{\lfloor \log_2 n \rfloor + 1 - 2kb} \frac{1}{2} \\
&\quad\quad + \Sigma_{b=\lfloor \frac{\lfloor \log_2 n \rfloor}{2k} \rfloor}^{\lfloor \frac{\lfloor \log_2 n \rfloor}{2k} \rfloor} \Sigma_{h=\lfloor \log_2 n \rfloor + 2 - 2kb} n 2^{-2kb-h} \\
&\quad + \Sigma_{b=\lfloor \frac{\lfloor \log_2 n \rfloor}{2k} \rfloor + 1}^{B-1} \Sigma_{h=1}^{2k} n 2^{-2kb-h} \\
&= \lfloor \frac{\lfloor \log_2 n \rfloor}{2k} \rfloor \frac{2k-1}{2} \\
&\quad + (\frac{\lfloor \log_2 n \rfloor}{2} - \frac{2k}{2} \lfloor \frac{\lfloor \log_2 n \rfloor}{2k} \rfloor) \\
&\quad\quad + (\frac{n}{2^{\lfloor \log_2 n \rfloor}} - \frac{2n}{2^{2k} 2^{2k \lfloor \frac{\lfloor \log_2 n \rfloor}{2k} \rfloor}})
\end{aligned}$$

$$+(\frac{n}{2^{2k}2^{2k\lfloor\frac{\lceil\log_2 n\rceil}{2k}\rfloor}} - \frac{n}{2^{2kB+4k}})$$

As $n \leq 2^{2kB}$, $\frac{n}{2^{2kB+4k}} \leq 1$, and we obtain the following result:

$E\{\Sigma_{t=1}^{number\ of\ nodes\ in\ trie} \prod_{p=1}^{2k} |NC_t^p|\} = O(\log_2 n)$. ∎

Theorem 5 *Given a binary trie T containing a set of k-dimensional input rectangles $D = \{R_1, R_2, \cdots, R_n\}$, R_i with i.i.d. random variable center c_i on $[0,1]^k$, and with i.i.d. random variable side length d_i distributed on $[0,1]^k$, consider a random orthogonal range search with query hyper-rectangle W with center at Z which is uniformly distributed on $[0,1]^k$, and independent of the centers of D, and with size $\Delta_1 \times \Delta_2 \times \cdots \times \Delta_k$ which are also i.i.d. random variables on $[0,1]^k$. The expected orthogonal range search time*

$$E\{Q(n,k)\} = c_1 ABk + c(\frac{1}{2k}\log_2 n)n^{1-\frac{s}{2k}}\Sigma_{S\subset\{1,\cdots,2k\}}(\prod_{p\notin S}|WC^p|) + O(\log_2 n),$$

where c and c_1 are constant values, and c is determined by s, $s = |S|$.

Proof. $E\{Q(n,k)\} = E\{\Sigma_{t=1}^{the\ number\ of\ nodes\ in\ the\ trie} 1_{[node_t \in GN \cup BN]}\}$

This calculation includes the reporting time for collection of the subtree of black nodes which arises during the traversal. The probability that a node is black or grey is given as: $Pr(node_i \in GN \cup BN) = \prod_{p=1}^{2k}(|NC^p| + |WC^p|)$. The probability for query hyper-rectangle W's cover space WC to intersect a node's cover space NC is the probability that Z_j, the center of W, is within distance $\frac{\Delta_j}{2}$ of NC^j. This probability is bounded by the volume of NC expanded by Δ_j in the j^{th} dimension, $\forall j \in \{1, \cdots, k\}$. There are two cases. On the left side of the j^{th} dimension, $|WC_j^{min}| = |WC^p| = |\,[0, H_j)\,| = H_j = Z_j + \frac{\Delta_j}{2}$, and $p \bmod 2 = 1$. On the right side of the j^{th} dimension, $|WC_j^{max}| = |WC^p| = |\,(L_j, 1]\,| = 1 - L_j = 1 - (Z_j - \frac{\Delta_j}{2}) = 1 - Z_j + \frac{\Delta_j}{2}$, and $p \bmod 2 = 0$. We have

$$\begin{aligned}
E\{Q(n,k)\} &\leq E\{\Sigma_{t=1}^{number\ of\ nodes\ in\ trie} \prod_{p=1}^{2k}(|WC^p| + |NC_t^p|) \\
&= \Sigma_{S\subseteq\{1,\cdots,2k\}}(\prod_{p\notin S}|WC^p|)E\{\Sigma_{t=1}^{number\ of\ nodes\ in\ trie} \prod_{p\in S}|NC_t^p|\} \\
&= \Sigma_{S\subset\{1,\cdots,2k\}}(\prod_{p\notin S}|WC^p|)E\{\Sigma_{t=1}^{number\ of\ nodes\ in\ trie} \prod_{p\in S}|NC_t^p|\} \\
&\quad + \Sigma_{S=\{1,\cdots,2k\}}(\prod_{p\notin S}|WC^p|)E\{\Sigma_{t=1}^{number\ of\ nodes\ in\ trie} \prod_{p=1}^{2k}|NC_t^p|\} \\
&= \Sigma_{S\subset\{1,\cdots,2k\}}(\prod_{p\notin S}|WC^p|)E\{\Sigma_{t=1}^{number\ of\ nodes\ in\ trie} \prod_{p\in S}|NC_t^p|\} \\
&\quad + E\{\Sigma_{t=1}^{number\ of\ nodes\ in\ trie} \prod_{p=1}^{2k}|NC_t^p|\}
\end{aligned}$$

From Theorem 4 and Proposition 1, we obtain

$$\begin{aligned}
E\{Q(n,k)\} &\leq \Sigma_{S\subset\{1,\cdots,2k\}}(\prod_{p\notin S}|WC^p|)(c(\tfrac{1}{2k}\log_2 n)n^{1-\frac{s}{2k}} + O(1)\,) \\
&\quad + E\{\Sigma_{t=1}^{number\ of\ nodes\ in\ trie} \prod_{p=1}^{2k}|NC_t^p|\},
\end{aligned}$$

and by Lemma 3, we obtain

$$E\{Q(n,k)\} \leq \Sigma_{S\subset\{1,\cdots,2k\}}(\prod_{p\notin S}|WC^p|)(c(\tfrac{1}{2k}\log_2 n)n^{1-\frac{s}{2k}} + O(1)\,) + O(\log_2 n).$$

$E\{Q(n,k)\} \leq c_1 ABk + c(\frac{1}{2k}\log_2 n)n^{1-\frac{s}{2k}}\Sigma_{S\subset\{1,\cdots,2k\}}(\prod_{p\notin S}|WC^p|) + O(\log_2 n)$. ∎

A is the number of data objects found in the orthogonal range search. The number of nodes traversed to collect the in-range hyper-rectangles is thus $c_1 ABk$, where c_1 is a constant value ≤ 2. This traversal for reporting the hyper-rectangles in range is unavoidable. The third item arises from finding the white nodes. When $S = \{1,\cdots,2k\}$, (i.e. s = 2k), $E\{Q(n,k)\} \leq c_1 ABk + c(\frac{1}{2k}\log_2 n) + O(\log_2 n)$. If $S = \emptyset$, $E\{Q(n,k)\} \leq c_1 ABk + \frac{c}{2k} n \log_2 n \prod_{p=1}^{2k}|WC^p| + O(\log_2 n)$. When $0 < s < 2k$, the second term dominates.

5 Conclusion and Open Questions

Using tries, we have shown that the space and preprocessing time for storing n k-d hyper-rectangles is linear in nk. The expected orthogonal range search time $E\{Q(n,k)\} = c_1 ABk + c(\frac{1}{2k}\log_2 n)n^{1-\frac{s}{2k}}\Sigma_{s\subset\{1,\cdots,2k\}}(\prod_{p\notin s}|WC^p|) + O(\log_2 n)$, where c and c_1 are constant values. The algorithm presented supports dynamic operations, and is straightforward to implement.

Improvements in expected storage cost $S(n,k)$ and expected time for orthogonal range search $Q(n,k)$ can be obtained if we compress our binary trie to avoid storing nodes with only one child as is done for Patricia tries. Further investigation of the expected cost for orthogonal range search is warranted to determine the relationship with partial match query expected cost. Under different assumptions about the nature of query window W distribution (e.g. random Gaussian in size), does the expected range search cost improve? Can the $2k$-d trie be adapted for other geometric problems such as half-space range search?

References

[Alle83] J. F. Allen, "Maintaining Knowledge about Temporal Intervals", *Communications of the ACM*, 26(11):832-843, 1983.

[Bent75] J.L. Bentley, Multidimensional binary search trees used for associative searching, *Communications of the ACM vol. 18*, no. 9, pp.509-517, September 1975.

[Bent79] J.L. Bentley and J.H. Friedman, "Data Structures for Range Searching", *Computing Surveys*, vol. 11, no. 4, pp.397-409, 1979.

[Bent80] J.L. Bentley and H.A. Maurer, "Efficient Worst-Case Data Structures for Range Searching",*Acta Informatica*, 13, pp.155-168, 1980.

[Chan01] P. Chanzy, L. Devroye and C. Zamora-Cura,"Analysis of range search for random k-d trees", *Acta Informatica*, Vol.37, Fasc. 4/5, pp.355-383, 2001.

[Chaz88] B. Chazelle, "A Functional Approach for Data Structure and its use in multidimensional searching", *SIAM Journal of Computing*, vol. 17, no. 3, pp.427-462, June 1988.

[Chaz90-a] B. Chazelle, "Lower bounds for orthogonal range searching: I. The reporting case", *Journal of the ACM*, vol. 37, no. 2, pp.200-212, April 1990.

[Chaz90-b] B. Chazelle, "Lower bounds for orthogonal range search: II. The Arithmatic Model", *Journal of the ACM*, vol. 37, no. 3, pp. 439-463, July 1990.

[Devr00] L. Devroye, J. Jabbour and C. Zamora-Cura, "Squarish k-d trees", *SIAM Journal of Computing*, vol.30, no.5, pp.678-700, 2000.

[Edel83] H. Edelsbrunner, "A new approach to rectangle intersection Part I", *Int. J. Computer Mathematics*, vol. 13, pp.209-219, 1983.

[Egen94] M. Egenhofer, "Spatial SQL: A query and presentation language", *IEEE Transaction on Knowledge and Data Engineering*, vol. 6, no. 1, pp. 86-95, 1994.

[FlPu86] P. Flajolet and C. Puech, "Partial Match Retrieval on Multidimensional Data", *Journal of the Association or Computing Machinery*, vol. 33, no. 2, pp.371-407, April 1986.

[Fred81-a] M. L. Fredman, "Lower Bounds On The Complexity Of Some Optimal Data Structures", *SIAM J. Comput.* Vol.10, No.1, February 1981.

[Fred81-b] M. L. Fredman, "A Lower Bound on the Complexity of Orthogonal Range Queries", *Jounal of the Association for Computing Machinery*, Vol.28, No.4, pp.696-705, October 1981.

[Gutt84] A. Guttman, "R-trees: a dynamic index structure for spatial searching", Proc. of the SIGMOD Conference, Boston, pp.47-57, June 1984.

[Knut98] D.E. Knuth, *The art of computer programming*, Volume 3, *Sorting and Searching*, 2^{nd} edition, Addison-Wesley, Boston, pp.564-566, 1998.

[LeeW77] D.T.Lee and C.K. Wong, "Worst-Case Analysis for Region and Partial Region Searches in Multidimensional Binary Search Trees and Balanced Quad Trees", *Acta Informatica*, 9, 23-29, 1977.

[Luek78] G.S. Lueker, "A Data Structure For Orthogonal Range Queries", 19^{th} Symposium on Foundations of Computer Science, pp.28-34, IEEE Computer Society Press, Oct. 1978.

[Luek82] G.S. Lueker and D. E. Willard, "A Data Structure For Dynamic Range Queries", *Information Processing Letters*, Volume 15, Number 5, pp.209-213, 1982.

[Merr96] T.H.Merrett, H.Shang, X.Zhao, "Database Structures, Based on Tries, for Text, Spatial, and General Data", International Symposium on Cooperative Database Systems for Advanced Applications, Kyoto, Japan, v. 2, pp.316-424, 1996.

[Shan01] H.Shang, "Trie Methods for Text and Spatial Data on Secondary Storage", Ph.D. thesis, McGill University, ch6, pp.90-100, January 2001.