CATEGORICAL RANGE REPORTING IN 2D USING WAVELET TREES

by

SWATHI KANTHAREDDY SUMITHRA
Bachelor of Engineering
Visvesvaraya Technological University

A thesis submitted in partial fulfillment of the requirements
for the Degree of Master of Science
in the Department of Electrical and Computer Engineering
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2018

Major Professor
Sharma Thankachan

ABSTRACT

The research involved optimizing the space and bounding the output time by the output size in categorical range reporting of points within the given rectangle query Q in two dimension using wavelet trees and range counting. The time taken to report those points and space to tore n points in set S can be done using wavelet tree and range counting. Consider set S consisting of n points in two-dimension. An orthogonal range reporting query rectangle Q = [*a,b*] x [*c,d*] on set S is sent to report the set of points in S which interacts with the query rectangle[Q]. The time taken to report these points is dependent on the output size. The categorical range reporting is an extension of orthogonal range reporting, where each point (xi, yi) in S is associated with a category $c[i]$ belongs to $[\sigma]$ and the query is to report the set of distinct categories within the query region [*a,b*] x [*c,d*] once. In this paper, we present a new solution for this problem using wavelet trees. The points in S associated with categories are stored in a wavelet tree structure. Wavelet tree structure consists of bit map for these categories. To report the categories in the given rectangle query Q, rank and select operations on the wavelet tree is applied. It was observed that the space taken by the structure was O(n logσ) space and query time is O(k log n logσ). Notice that the new result is more efficient in space when logσ = O(log n). The study provides a new and efficient way of storing large dataset and also bounds the time complexity by the output size k.

TABLE OF CONTENTS

LIST OF FIGURES

1 INTRODUCTION

The wavelet tree data structure is considered as a space efficient data structure and is used to illustrate a sequence and to provide solutions for the queries on the sequences. The wavelet tree was invented by Grossi Gupta and Vitter in 2003[16]. If one looks around for wavelet trees", one is likely to end up reviewing signal processing bibliography[2, 5, 11].The wavelet tree data structure illustrates set of points on a two-dimensional grid (2D grid). Here successive restructuring of the points is carried out by sorting the points start using one coordinate and followed by the other coordinate which results in segregating high and low coordinate values in the final dimension, i.e. wavelet tree uses hierarchical partitioning of the alphabets. Wavelet trees can be considered in three methods: a) as a representation of a sequence b) as a representation of a reordering of elements c) as a representation of a grid of points.Wavelet tree has many applications in indexed pattern matching and data compression as mentioned in[20]

Range searching can be considered as database application. Grid consists of many points of different colors. These points can be viewed as information stored in the databased. A sample query on the grid can be considered. Find out number of departments in which the employees salary is between 60K-75K and age is between 25-30. This query can be solved by sorting all the employees according to their age with salary being the key for the query. This is nothing but a range query applied on the grid. The query rectangle is parallel to both x-axis and y-axis and hence known as orthogonal range reporting.

The categorical range reporting can be defined as expansion of range reporting, where the query reports back set of different colors only once within the query region $[a, b]$ x $[c, d]$ and each point (xi, yi) in S is associated with a color $c[i] \in \sigma$. In the categorical or (colored) range-searching problem,the set of input points is partitioned into categories; [17] the output to a query is a section of data that is related to the categories of points within the query range.

In section II the data structure of wavelet trees and range counting on the grids is discussed. The related work and proposed approach is discussed in section III and IV respectively.

2 PRELIMINARIES

2.1 Wavelet trees

Consider sequence set S[1,n] = [$s_1, s_2, \ldots .s_n$] which consists of series of symbols $s_i \in \Sigma$, where Σ is distinct letters of size σ. We can consider $\Sigma = [1 \ldots .\sigma]$. Below fig1. is an example of wavelet tree. It can be shown that set S can be represented using n[log σ] = n log σ +O(n) bits[16]. The root node consists of the bitmap $B_{v_{root}}$ [1,n]. The bitmap $B_{v_{root}}$ [1,n] is defined as-

$$B_{v_{root}}\ [1,n] = \begin{cases} 0 & \text{for S[i]} < (a+b)/2 \\ 1 & \text{for S[i]} > (a+b)/2 \end{cases}$$

The symbols c$\leq$ (a+b)/2 are defined by S_0 [1, n_0] which are the described as subsequence of S[1, n] and the symbols c> (a+b)/2 are defined by S_1[1, n_1] which are the described as subsequence of S[1,n]. Then, the left child of v_{root} is a wavelet tree for S_0[1, n_0] over alphabet [a..[(a+b)/2] and the right child of v_{root} is a wavelet tree for S_1[1, n_1] over alphabet [1+[(a+b)/2]..b] [16].

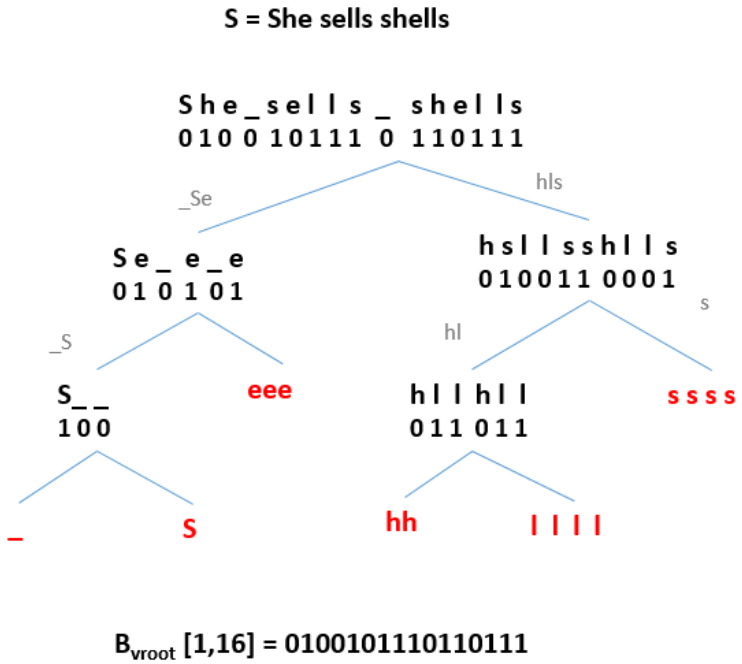

Figure 1: Wavelet tree for the string = "She sells shells". The underscores are used for spaces. Only the bitmaps and the topology is stored in the tree. For understanding purposes, the sequences and the alphabets that belongs to Σ are illustrated.

Above figure illustrates the wavelet tree for the string S= "She sells shell", where Σ ={_, S,

e, h, l ,s} , σ =6, and n= 16. '_'(underscore) is used to represent space. Here, Σ = {_, S, e} is assigned 0 and Σ ={h, l, s} is assigned 1 at the root level. In the next level (level 1) all '0' bits are sent to the left child and all the '1' bits are sent to the right child. Therefore the root bitvector has two children. For the left child of level 1, Σ = {_, S} is assigned 0 and Σ =e is assigned 1. Here again all 0 bits are divided as left child (level 2) and all 1 bits are divided as right child(level 2) for the left child of level 1. For the right child of level 1, Σ =h, l is assigned 0 and Σ =s is assigned 1. Here again all 0 bits are divided as left child(level 2) and all 1 bits are divided as right child(level 2) of the right child of level 1. Now level 1 has 4 children. This hierarchical partitioning is continued until each node consists of distinct alphabet. The wavelet tree has σ leaves and σ-1 internal node and its height is $l$og σ. Wavelet tree stores exactly n bits in each level and at most n bits in the last one [16]. Therefore n [log σ] is the total number of bits the wavelet tree stores.

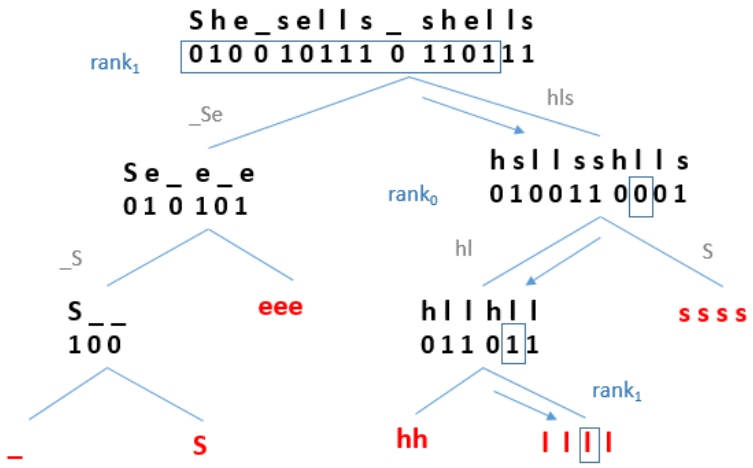

Figure 2: In the above figure, symbol at position S[14] at the root position is tracked down to leaf labeled third frequency of 'l'. Thus the alphabet frequency at position S[14] in the root level is three.

The string S can be recovered from wavelet tree of the S. Now to get S[i], we have to first evaluate $B_{v_{root}}$[i]. If the $B_{v_{root}}$[i] = 0, then s[i] $\leq$ (σ+1)/2 else s[i] > (σ+1)/2. For the first condition, we traverse repeatedly on the left child and for the second condition we traverse repeatedly on the right child. This method helps in finding the position of 'i' that is mapped on to the left or right child of the tree. For left child, $B_{v_{root}}$[i] = 0, is mapped to position i_0 which is the number of 0s in $B_{v_{root}}$ up to position i. For the right child, $B_{v_{root}}$ [i] = 1, is mapped to position i_1, the number of 1s in $B_{v_{root}}$ up to position 'i' [16]. The number of 0s and 1s till the position 'i' in the bitvector B is known as $rank_0$(B,i) and $rank_1$ (B,i) respectively. This method

is repeated till a leaf node is arrived whose label gives us S[i]. Thus the rank operation gives us the symbol frequency till the position of S[i] in the root level. Fig2 illustrates tracking down symbol from root position to the leaf. Rank operation can be computed in O(log σ) time.

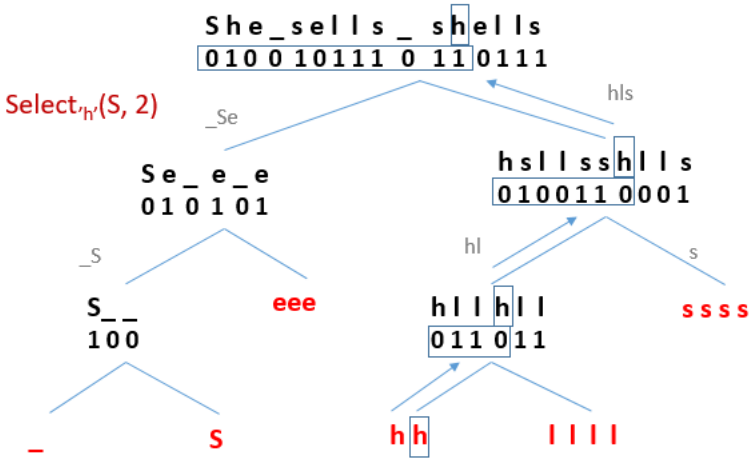

Figure 3: The figure demonstrates the position of symbol 'h' in sequence S at the root level. 'h' is found at S[12] in the bitmap B. Only the bitmaps and the tree topology will be stored.

The position is tracked from root down to the leaf in rank operation, this process can be inversed to find out the position of the symbol at the root level. This method is called select operation. Now consider we start tracking the position of the symbol upward. If the leaf is a right child of the parent node v, then the position *i'* is the i-th occurrence position of '1' in bitmap B. If the leaf is a left child of the parent node v, then the position *i'* is the i-th occurrence position of '0' in bitmap B. This method is continued till we reach root from node v to get the exact position of the symbol in the root bitmap. In fig 3, the second symbol '*h*' at the leaf node is tracked upwards. Since it is the left child of the parent node '*v*', bit '0' occurrence for the second time in the parent node is considered which is at position 4. The parent node is left child, therefore the 4th occurrence of bit '0' is searched in the next parent node *v*. Now the parent node is a right child and has the 4th occurrence of bit 0 at position 7. In the root bit vector the 7th occurrence of bit '1' is searched, which is at the position 12. The tracking is completed as we reached the bitmap B. Therefore, the position of bit '1' occurred for the 7th time is the position of symbol '*h*' in the bitmap B. The operation of finding the $i-th$ '0' or '1' in a bitmap B[1, n] is called $select_0$ (B, i) and $select_1$ (B, i), and it can also be solved in constant time using the n bits of B + O(n) bits [16, 6, 7].

In similar way, the set of data consisting of points can be represented as a wavelet tree. For

example consider set of data describing age and salary of the employees.

P= [x,y]=[age,salary] =(20, 40k) (32, 36K) (50, 60K) (22, 50K (45,80K) (59, 80K) (52, 70K) (45, 70K), where age represents x-coordinate and salary represents y-coordinate

To construct wavelet tree for the above data, the data set has to be sorted according to x-coordinate at the root level and then the tree is sorted according to y-coordinate along the tree nodes.

P_{sorted} =

Age	20	22	32	45	45	50	52	59
Salary	40K	50K	36K	70K	80K	60K	70K	80K

Here, P = {36, 40, 50, 60} is assigned 0 and P ={70, 80} is assigned 1 at the root level. In the level 1 (level 1) all 0 bits are sent to the left child and all the 1 bits are sent to the right child. Therefore the root level has two children. For the left child of level 1, P = {36, 40} is assigned 0 and P =50, 60 is assigned 1. Here again all 0 bits are divided as left child (level 2) and all 1 bits are divided as right child for the left child of level 1. This partitioning is continued till leaf consisting of distinct data is achieved.

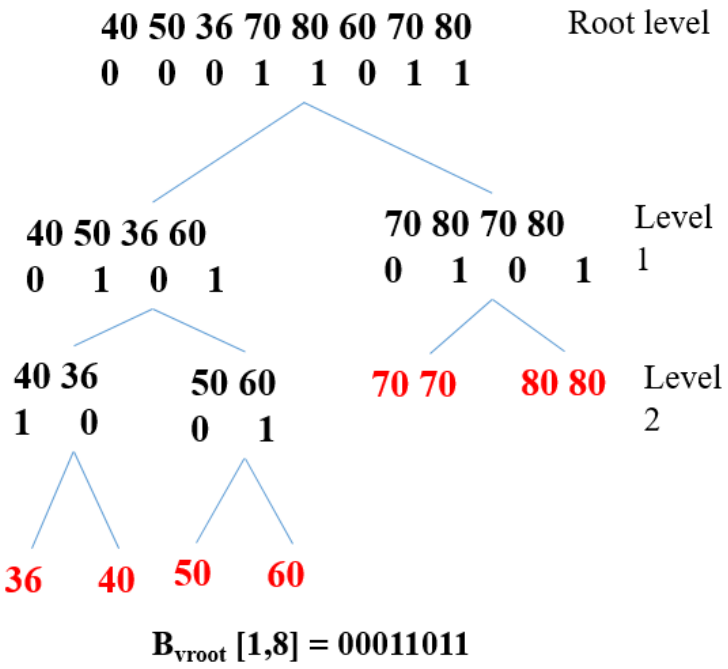

Figure 4: Wavelet tree for the data set P= "40 50 36 70 80 60 70 80". For understanding purposes, the data set is illustrated.

The data set P can be recovered from the above figure Fig 4, similar to that of string wavelet tree. To get P[i], we have to check for $B_{v_{root}}[i]$ value. If the $B_{v_{root}}[i] = 0$, the then the tree traversal is done to the left of the bitmap else to the right of the bitmap. In the bitmap the position P[6] contains **'0'** bit. Therefore the data is tracked downwards towards the left of the wavelet tree. Number of '0's in $B_{v_{root}}[i]$ till P[6] = 4. The 4^{th} bit is evaluated in level 1. It observed that the 4^{th} bit is occupied by '1' and hence the data is tracked downwards towards right of the level 1 node. Number of '1's in level 1 is 2. The 2^{nd} bit is evaluated in level 2. It is observed that the 2^{nd} bit consists of bit'1' and the data is traced downwards towards right of level 2 node. The distinct data is reached and it has data '60'. It implies that data 60 has occurred once till the position P[6], Fig5 demonstrates tracking down symbol from root position to the leaf. Rank operation can be computed in O(log σ) time.

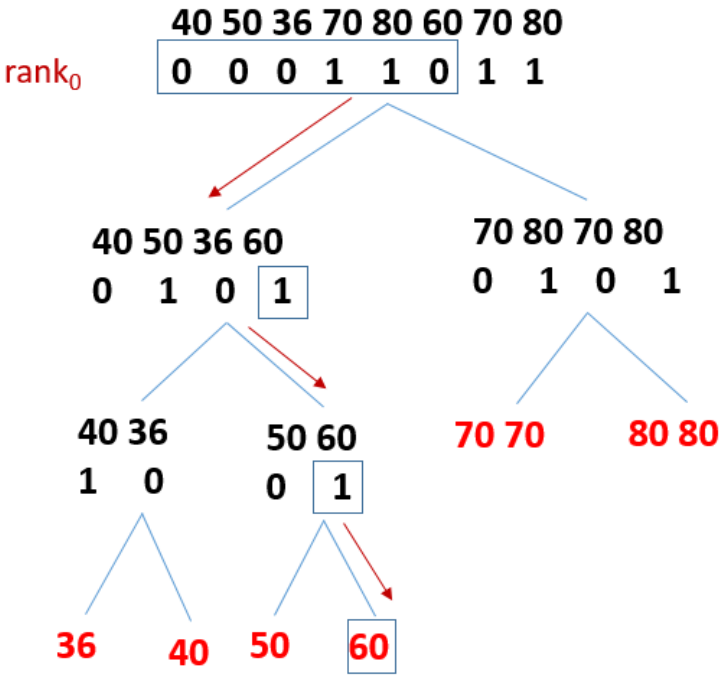

Figure 5: The data at position 6, P[6] is tracked down to leaf labeled **'60'** which has occurred once. Therefore data at position P[6] at the root is **'60'**.

Fig 6 explains the select operation on data set wavelet tree. 2^{nd} data **'70'** is tracked upwards to the root node. The data '70' is a left child of level 2 as demonstrated in the figure, hence the occurrence of bit '0' for the 2^{nd} time in the parent node is evaluated. the 2^{nd} bit '0' is in the 3^{rd} position in the parent node of level-1. In level-1, '70' is right child and hence the occurrence of bit '1' for the 3^{rd} is evaluated. It occurs at 7th position in the bitmap.Select operation also takes O(log σ) time. The tracking is done until the operation reaches the bitmap.

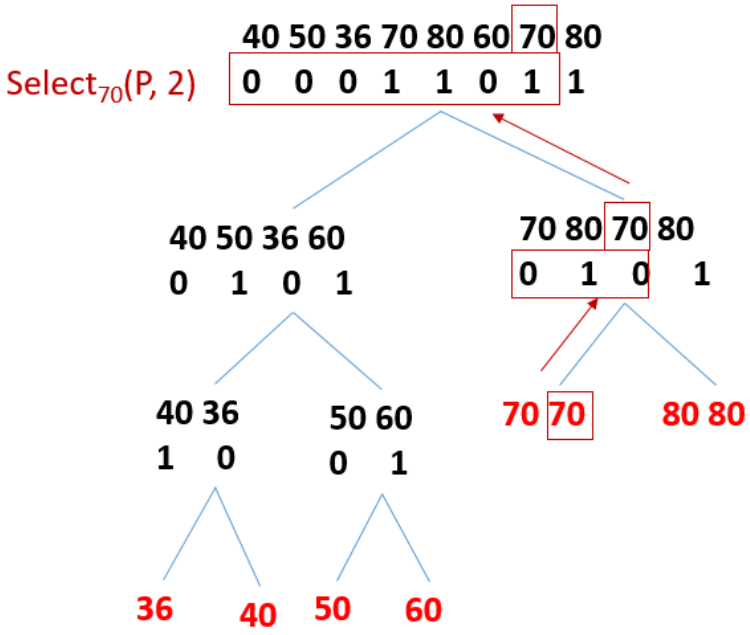

Figure 6: The position of the data '70' is tracked upwards to the root level and it is at the position P[7] in the bitmap. .

2.2 Range counting using wavelet trees

Range counting is a query operation performed on 2D grids. Wavelet tree of data consisting of $'n'$ points can be converted into a grid consisting of 'c' columns and 'r' rows. In this grid, each cell might have or might not a point. Now this irregular grid is converted into $'n'$ x $'n'$ grid with $'n'$ points. This grid has exactly one point per row per column. The data is first sorted according to the x-coordinate where $x_i < x_{i+1}$ and is defined later according to y-coordinate i.e P[1...n] = $\{y_1, y_2, ...y_n\}$.

The unsorted data P[1,n] is-

$P_{unsorted}$ =

Age	20	32	50	22	45	59	52	45
Salary	40K	36K	60K	50K	80K	80K	70K	70K

The wavelet tree for the sorted and defined data is constructed later.

P_{sorted} =

Age	20	22	32	45	45	50	52	59
Salary	40K	50K	36K	70K	80K	60K	70K	80K

Considering same wavelet tree in Figure 4, the 2D grid can be created as shown below.

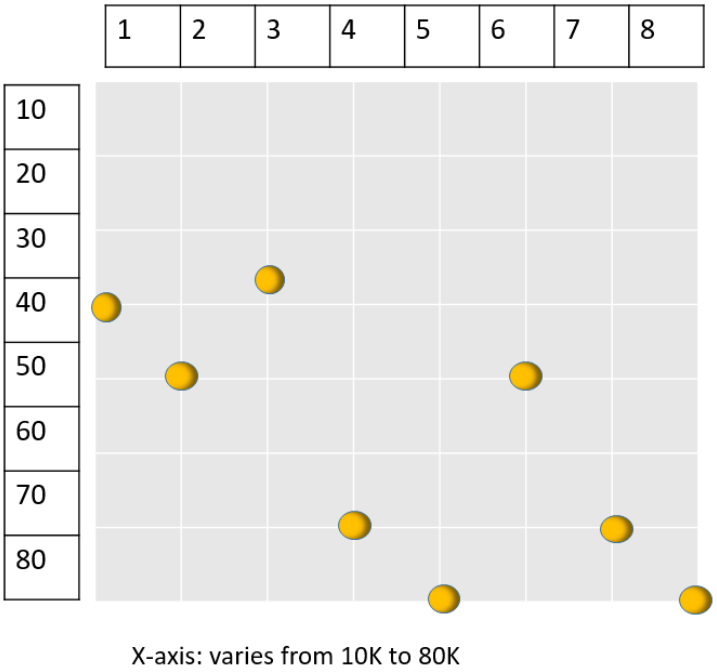

Figure 7: The 2D representation of data P sorted according to x-coordinate at the root node.

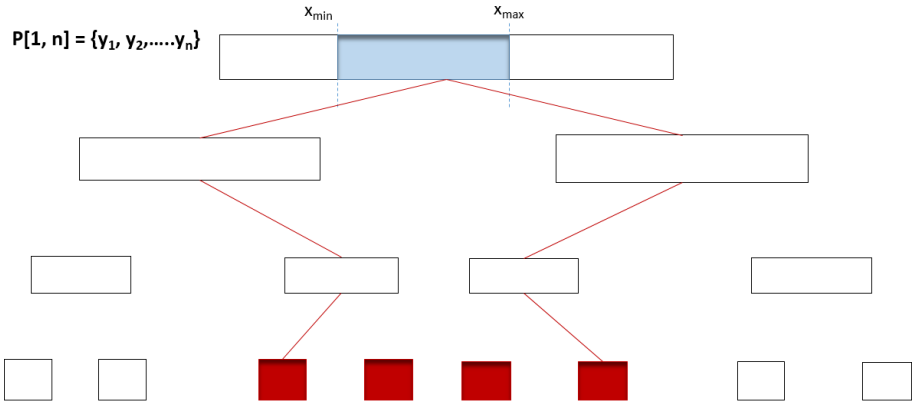

Figure 8: In the figure, the range of the query rectangle $[x_{min}, x_{max}]$ x $[y_{min}, y_{max}]$ is shown. the root level is sorted by x-coordinate and distinct data by y-coordinate.

Pre-processing-

(i)All the 'n' points are sorted in x-coordinate order(i.e $x_i < x_{i+1}$).

(ii)The wavelet tree for the sequence S[1, n] = $\{y_1, y_2, \ldots . y_n\}$ is constructed.

Algorithm-

(*iii*) Given query rectangle range $[x_{min}, x_{max}]$ x $[y_{min}, y_{max}]$, the searching mechanism starts at the root bitmap $B_{v_{root}}$[i][x_l, x_r] = [x_{min}, x_{max}].

(*iv*) Now the interval to the left and right are mapped.

(*v*) The number of points present in the query rectangle range is counted in which the interval of the leaves is in the range $[y_{min}, y_{max}]$.

(*vi*) The mapping of points is repeated until the- (**a**) value $x_l > x_r$, i.e the interval $[x_l, x_r$] is empty.(**b**) interval of leaves say $[a, b]$ will have no intersection with $[y_{min}, y_{max}]$. (**c**) interval of leaves say $[a, b]$ is present in $[y_{min}, y_{max}$].

In case (**a**) and (**b**), the points returned back are zero and in case (**c**) number of the points returned back are $(x_{min} - x_{max} + 1)$. Time taken by the query is O(log n). It should be noted that only O(log n) wavelet tree nodes are visited before stopping all the recursive calls[16](the proof can be read in[8])

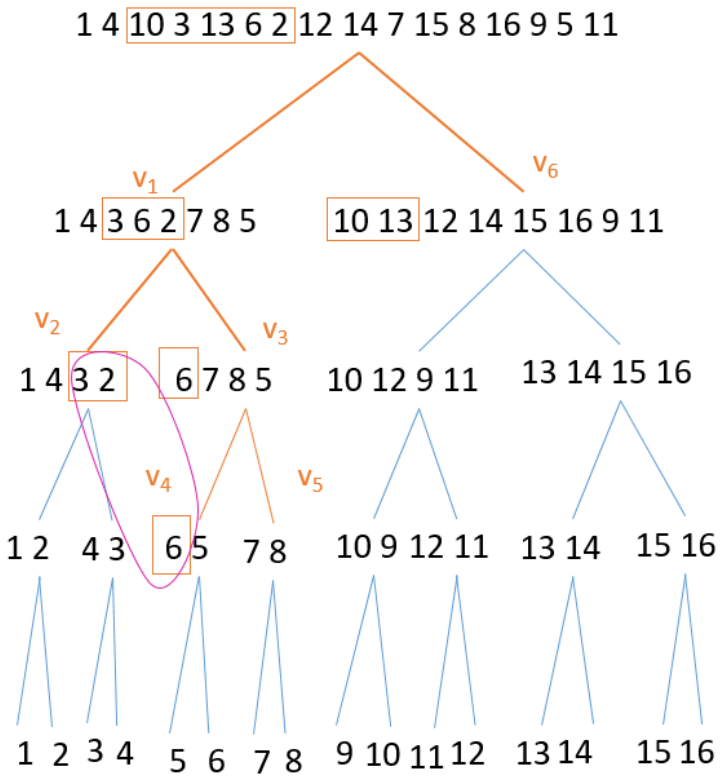

Figure 9: The boxed region in the figure represents how the x-range is sub-divided in the nodes. The number of points present is explicitly shown in the diagram. The nodes v_2 and v_4 cover the y-range of the query rectangle.

To explain range counting on 2D grid let us consider the bitmap $B_{v_{root}}[i] = 0100101110110111$ from the figure 1 and is converted into 16 x 16 grid. The wavelet tree for the points in grid is as shown in the above figure 9.

The query rectangle $[x_{min}, x_{max}]$ x $[y_{min}, y_{max}]$ = [3,7] x[1,6]. The data interval of the data set present in the query rectangle is always checked for its existence in query range $[y_{min}, y_{max}]$. The data interval of root node is [1, 16] and the character interval is [3,7]. Now the data interval [1,16] is checked with the query interval [1,6]. It can be seen that data interval [1,16]is neither disjoint nor contained in the query interval [1,6]. Therefore both nodes of root node is followed downwards recursively in the next level. Level-1 has two nodes, node v_1 and v_4. Node v_1 is checked for the existence of the query interval in the data interval. The data interval of node v_1 is [1, 8]and the character interval is [3,5]. The data interval [1,8] is neither disjoint nor contained in the query interval [1,6]. Both nodes of node v_1 is followed down recursively. Noe v_2 has data interval [1,4] and character interval [3,4]. The data interval [1, 4] is contained in the query interval[1,6]. Hence the points are calculated = 4-3+1 = 2. Therefore two points from node v_2 are present. Now the right child of node v_1 is checked for the points.Node v_3 has data interval [5,8] and character interval [5,5]. The data interval [5,8] is neither contained or is a disjoint of query interval [1,6]. Therefore both nodes of node v_3 followed recursively. Now the data interval of left child of node v_3 is [5,6] and character interval is [5,5]. The data interval [5,6] is present in the query interval [1,6] and hence the points are calculated here. Number of points = 5-5+1 = 1.

The nodes v_5 and v_6 has data interval [7,8] and [9,16] respectively. Both the data intervals are not present in the query interval [1,6] and hence nodes v_5 and v_6 are not checked for the points existence and they are ignored.

Therefore the count of total number of points in the query rectangle [3,7] x [1,6] = 2+1 = 3 points

The time taken to report the points back is O(log n).

3 RELATED WORK

Now a days, Range searching is the most often difficulty faced in many applications. This difficulty is extensively monitored in computational geometry and spatial databases. For example, consider set objects. A normal range query task is to report all the objects which is intersecting with query object. There are numerous applications in which the inputs are also divided into categories and a query is sent to report categories of those objects which intersect the query object[1]. Applications like network routing and databases inputs are divide into categorized objects and the reporting query is operated on the objects. Usually the object is categorized by colors. When the category range-searching problem is applied on a grid, the data structures use near linear space and answer a query in O(log log U + k) time, where k is the output size[1] and space of O(n $\log^2$ U).

There is another approach for the categorical range reporting. In this approach, all the categories belongs to σ and a query is sent to report back every point with its color C along with its position in the array of data. This approach analyses two parameters- Threshold T and top-k version[19]. This approach has produced O(n log n) space and O(log n+k) time for the queries where 'k' the size of the output.

The main application of categorical range reporting is as a database and categorical range queries find important applications in spatial data[15]. Consider this application in the field of DNA sequences. There is a high possibility that the documents containing DNA sequences data are needed to be accessed frequently to extract a particular string of sequence. That particular sequences might have occurred more than one time but need to reported only once. Also, with some more requirements the document containing the sequence can be accessed by sending queries. These queries can be approached with the help of categorical range reporting queries and with their other forms. Colored range reporting queries and their variants were extensively studied in the main memory model[18]. The examples can be referred in the papers[12, 4, 10, 9, 3, 14, 15, 13]

Previously, the data structures described in Gupta et al. [1995][10] and Janardan and Lopez [1993][12] support two-dimensional colored range-reporting queries in O(log N + K) time1 and use O(N log2 N) words of space.[17]

4 THE PROPOSED APPROACH

4.1 Structure

The proposed data structure consists of pre-processing the given categories and then applying the algorithm. Consider a set of points which have 'n' distinct categories of 'n' different frequencies i.e $c[i] = \{c_1, c_2, ...c_n\} \in \sigma$ and each category has frequencies of $\{f_1, f_2, ...f_n\}$ respectively. A wavelet tree on these categories is constructed as follows-

(i) In the beginning of the set, the root node consists of all the colors.

(ii) Now the interval of the 't' categories is divided equally into two different subset of the categories.

(iii) This hierarchical partitioning is repeated until distinct categories are obtained.

(iv) Each set consists of wavelet tree for the points $c[i]$ which belongs to that particular interval.

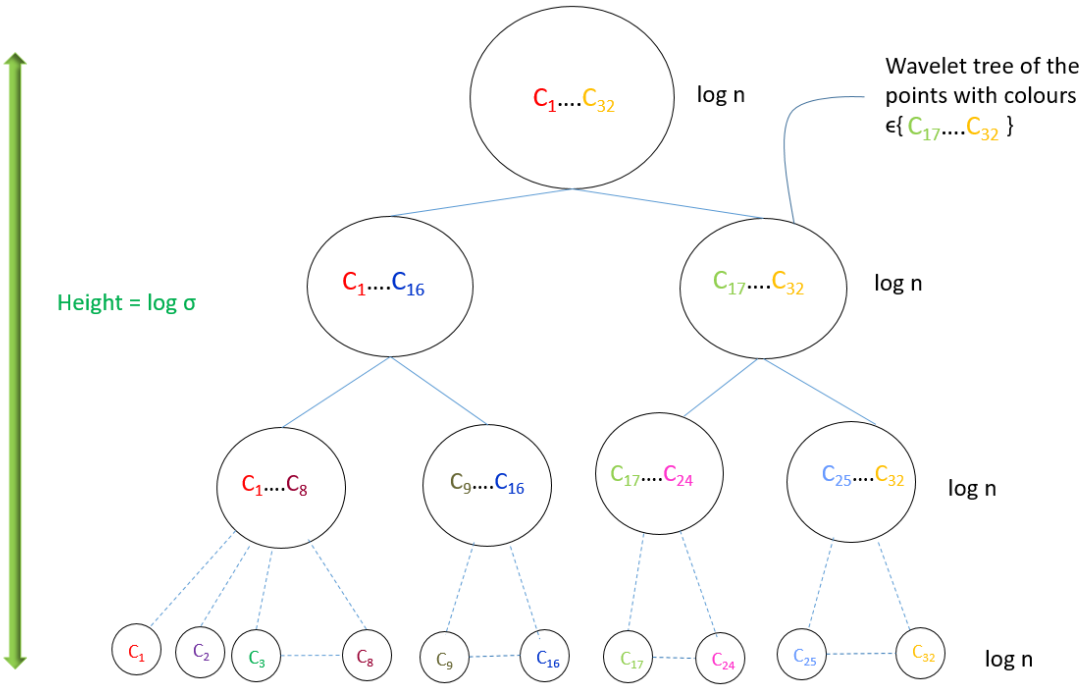

Figure 10: The figure illustrates the construction of the external tree on the given distinct 32 categories.

In the above figure, 'n'= 32. In the main set node, wavelet tree for all the points is constructed. The interval 't' = 32, is divided equally and the points associated with the interval are distributed to next level. Now the level-1 consists of two sets. The left set consists of wavelet tree of points with categories $C_1....c_{16}$ and frequencies $f_1....f_{16}$. The right set consists of wavelet tree of points with categories $C_{17}....c_{32}$ and frequencies $f_{17}....f_{32}$. This partitioning of

categories is repeated on both sides of the tree until points with different categories are sorted out as demonstrated in the Figure 7.

The approached algorithm is as follows-

(i) The query range within the interval of root set is sent.

(ii) Now each set is checked for the existence of a single point within the given range using a query.

(iii) If Query $= \begin{cases} 1 & \text{Follow on both sides of the parent node downwards} \\ 0 & \text{Ignore the set node} \end{cases}$

(iv) This querying is repeated until distinct colors in the given range are reported once to output.

4.2 Query analysis

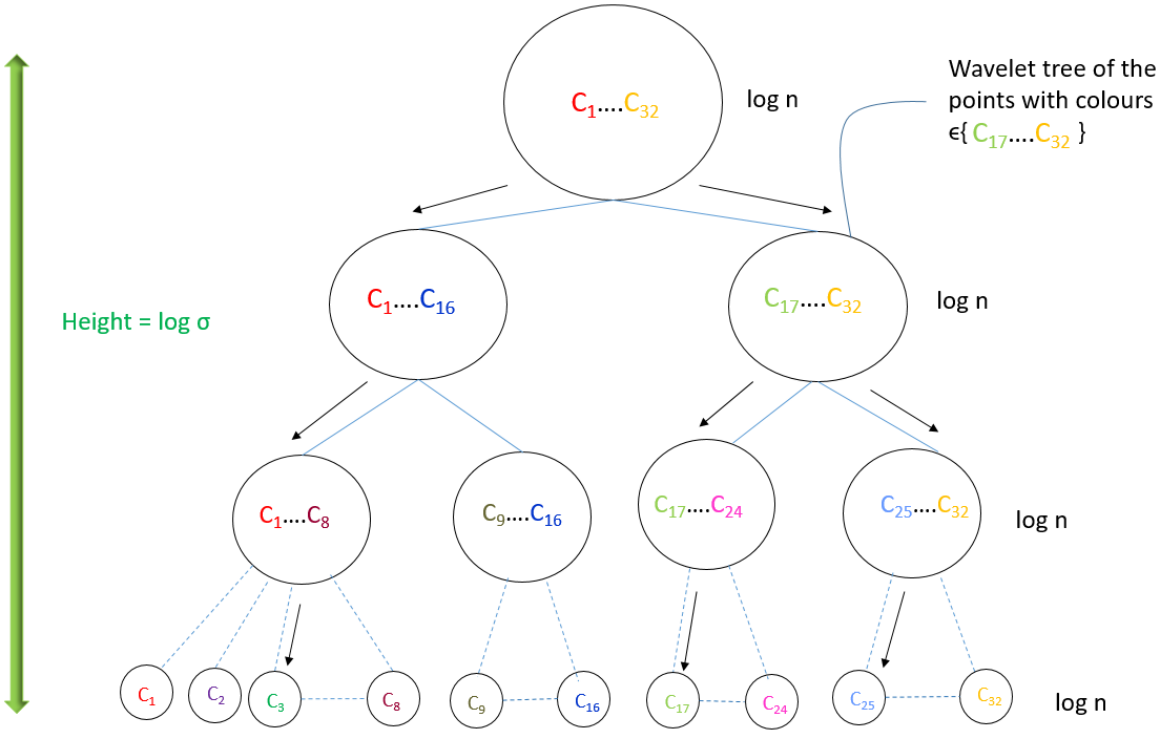

Figure 11: The colors C_3, C_{17}, C_{25} are the three colors associated with the points present in the given range query.

As explained in the section 2.2, range counting is used to count the number of points present in the given range. Assume the given range query consists of at-least one point of categories C_3, C_{17} and C_{25}. There total of 32 categories when divided equally results in 16 categories which is used to further classify categories into two subset nodes. Now root set is checked for

the presence of at least point for the given range query. The given range consists of at least one point associated with categories C_3, C_{17} and C_{25}. Category C_3 falls in the interval less than t/2 and categories C_{17} and C_{25} belongs to interval greater than t/2, where 't' represents total number of categories. Thus category C_3 is sent to the left subset of the root set node and categories C_{17} and C_{25} are sent to the right subset of the root set node. Here the subsets are checked for the existence of at least one point associated with the categories in the given range. Now the interval of these subsets are divided equally again. The points exists in the given range. The category C_3 falls in the interval less than t/4 and hence is sent to the left subset of left node in level-1 and is evaluated further. But the right node of root set does not contain at least one point in the given range and hence it is ignore and further the node is not evaluated. The categories C_{17} and C_{25} in the interval greater than t/4 but when evaluated from the interval 17-32, category C_{17} is sent to left subset of the right node in level-1 and category C_{25} is sent to right subset of the right node in level-1. This is repeated until categories C_3, C_{17} and C_{25} are distinctly found as leaf nodes. When they are found distinctly, the categories C_3, C_{17} and C_{25} are reported once giving the count of the categories present.

4.3 Experimental Analysis

The proposed approach is analyzed by sending various queries on different size of data. Consider a data set S consisting of 1000 points which are defined by age varying from 18-60, salary varying from 30K-500K and 100 departments. Departments are the categories which have to be reported back when a query rectangle is sent. Query rectangle [a,b] x [c,d] is given by the user. Now to analyze the approach a program is written to create a bit vector for the given data, to find the median of the bit vector, to create a wavelet tree for the given data, and to perform the select operation on the data to report the departments once; if included in the query rectangle [a,b] x [c,d].

Our approach has a tree in which each node consists of internal wavelet tree until all leaf nodes are sorted by unique categories and each node is divided based on the category which is illustrated in Figure 10. At the first node, the bit vector for all 1000 points is created based on the salaries in the first node. A wavelet tree for this bitvector is created. Now, in the root bitvector the median is checked for its occurrence in the salary range. If the median is in the salary range then the internal tree is traversed on both the sides. If the median is less than the salary range, then the internal tree is traversed only to the right side in the next level and if the median is greater than the salary range, then the internal tree is traversed only to the left

in the next level.

This method helps us in reaching a particular salary value at the end of the internal wavelet tree. Now this particular salary value is checked for its existence in the given range[c,d] using select operation. If at least one point with the given salary value exists then, we traverse on both the nodes of the parent node. Now each node is checked for the existence of at least one point in it using select method. If the point exists, then it is traversed down till an unique department is reached.

DATA SIZE	Output per time for 20 queries
10K	430.9655
20K	436.43986
40K	664.80285
80K	772.6583418
160K	1123.04055

Figure 12: The table consists of data size and the average of output per time of 20 queries for 5 different data set.

This approach was tested on 5 different sizes of data. All sets of data consisted of age varying from 18-60, salary varying from 30K-100K and 100 departments. The first data set consisted of 10000 points, the second data set consisted of 20000 points, the third data set consisted of 40000 points, fourth data set consisted of 80000 points and the fifth data set consisted of 160000 points. For each data set, 20 queries consisting of age range [a,b] and salary range [c,d] were sent to report back the number of categories in the given range [a,b] x [c,d]. The output per time was calculated for each query and the average of output per time was determined for all 5 data sets. The data size and output per time is tabulated above and the graph for data size vs output per time is as shown in the figure below.

It can be seen in the graph that as the input size increases the output per time of the queries also increases linearly.

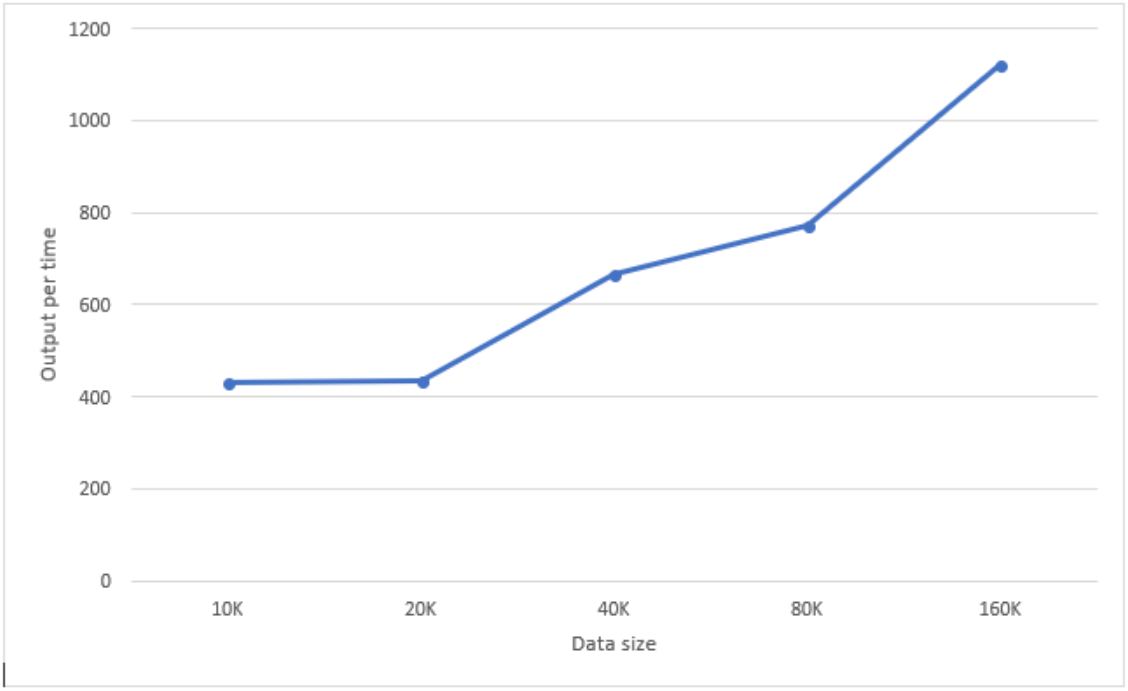

Figure 13: The graph is plotted for data sizes and averages given in the above table (Figure 12).

4.4 Space Complexity

The space complexity of the proposed data structure is O(n logσ) which is better than the previous available solutions.Here each level stores 'n' points till the distinct leaf nodes. The height of the tree is logσ. Hence the structure takes O(n logσ). Each point belongs to log σ different sets. When σ is smaller, the space occupied is O(n logσ) which is more efficient than O(n log n).

4.5 Time complexity

The height of the tree is bounded by logσ. Each level store *'n'* points and the last level stores at most *'n'* points. The time taken by each wavelet tree for reporting query i.e to check for the presence of at least one point in the given range is O(log n). The traversal to find the presence of distinct category takes log σ time. The number of categories to be found in the given query region is the output size. Therefore the output is bound by the the factor 'k'. The space used to store the pints is O(n logσ) and the time taken to find the number of categories proportional to the output size is O(k logn logσ).

5 CONCLUSION

The wavelet tree along with range counting on the data set of the wavelet tree produced an efficient space complexity for the categorical range reporting of points in 2D. The previous available best solutions for categorical range reporting of the points has space complexity of O(n logn) space and O(log n+k) time. Our approach in the proposed structure is space efficient when σ is smaller. The space complexity of the proposed structure is O(n logσ) which much more efficient when σ is smaller than the O(n logn) space. It can be observed that the time complexity of the proposed structure is not efficient as the time complexity of the previous available solution. Our aim was to provide better space complexity for categorical range reporting which is obtained when wavelet tree and range reporting operations are used together. Future work can be done on improvising the time complexity of the proposed structure.

REFERENCES

[1] Pankaj K. Agarwal, Sathish Govindarajan, and S. Muthukrishnan. "Range Searching in Categorical Data: Colored Range Searching on Grid". In: *Proceedings of the 10th Annual European Symposium on Algorithms*. ESA '02. London, UK, UK: Springer-Verlag, 2002, pp. 17–28. ISBN: 3-540-44180-8. URL: http://dl.acm.org/citation.cfm?id=647912.740825 (cit. on p. 11).

[2] Ali N. Akansu and Richard A. Haddad. *Multiresolution Signal Decomposition: Transforms, Subbands, and Wavelets*. Orlando, FL, USA: Academic Press, Inc., 1992. ISBN: 012047140X (cit. on p. 1).

[3] Panayiotis Bozanis et al. "New Results on Intersection Query Problems". In: *Comput. J.* 40 (1997), pp. 22–29 (cit. on p. 11).

[4] Panayiotis Bozanis et al. "New upper bounds for generalized intersection searching problems". In: *Automata, Languages and Programming*. Ed. by Zoltán Fülöp and Ferenc Gécseg. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 464–474 (cit. on p. 11).

[5] C. Sidney Burrus, Ramesh A. Gopinath, and Haitao Guo. *Introduction to Wavelets and Wavelet Transforms: A Primer*. Prentice Hall, 1997, p. 268. URL: http://www.amazon.com/Introduction-Wavelets-Wavelet-Transforms-Primer/dp/0134896009 (cit. on p. 1).

[6] David Richard Clark. "Compact Pat Trees". UMI Order No. GAXNQ-21335. PhD thesis. Waterloo, Ont., Canada, Canada, 1998 (cit. on p. 4).

[7] Francisco Claude and Gonzalo Navarro. "Algorithms and Applications". In: ed. by Tapio Elomaa, Heikki Mannila, and Pekka Orponen. Berlin, Heidelberg: Springer-Verlag, 2010. Chap. Extended Compact Web Graph Representations, pp. 77–91. ISBN: 3-642-12475-5, 978-3-642-12475-4. URL: http://dl.acm.org/citation.cfm?id=2167962.2167967 (cit. on p. 4).

[8] Travis Gagie, Gonzalo Navarro, and Simon J. Puglisi. "New algorithms on wavelet trees and applications to information retrieval". In: *Theoretical Computer Science* 426-427 (2012), pp. 25–41. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2011.12.002. URL: http://www.sciencedirect.com/science/article/pii/S0304397511009625 (cit. on p. 9).

[9] Prosenjit Gupta, Ravi Janardan, and Michiel Smid. "Efficient Algorithms for Generalized Intersection Searching on Non-iso-oriented Objects". In: *Proceedings of the Tenth Annual Symposium on Computational Geometry*. SCG '94. Stony Brook, New York, USA: ACM, 1994, pp. 369–378. ISBN: 0-89791-648-4. DOI: 10.1145/177424.178096. URL: http://doi.acm.org/10.1145/177424.178096 (cit. on p. 11).

[10] Prosenjit Gupta, Ravi Janardan, and Michiel Smid. "Further Results on Generalized Intersection Searching Problems: Counting, Reporting, and Dynamization". In: *J. Algorithms* 19.2 (Sept. 1995), pp. 282–317. ISSN: 0196-6774. DOI: 10.1006/jagm.1995.1038. URL: http://dx.doi.org/10.1006/jagm.1995.1038 (cit. on p. 11).

[11] Sreenivasa Jammalamadaka. "A Primer on Wavelets and Their Scientific Applications by James S. Walker". In: 95 (Sept. 2000) (cit. on p. 1).

[12] RAVI JANARDAN and MARIO LOPEZ. "GENERALIZED INTERSECTION SEARCHING PROBLEMS". In: *International Journal of Computational Geometry & Applications* 03.01 (1993), pp. 39–69. DOI: 10.1142/S021819599300004X. eprint: https://doi.org/10.1142/S021819599300004X. URL: https://doi.org/10.1142/S021819599300004X (cit. on p. 11).

[13] Marek Karpinski and Yakov Nekrich. "Top-K Color Queries for Document Retrieval". In: *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '11. San Francisco, California: Society for Industrial and Applied Mathematics, 2011, pp. 401–411. URL: http://dl.acm.org/citation.cfm?id=2133036.2133068 (cit. on p. 11).

[14] S. Muthukrishnan. "Efficient Algorithms for Document Retrieval Problems". In: *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '02. San Francisco, California: Society for Industrial and Applied Mathematics, 2002, pp. 657–666. ISBN: 0-89871-513-X. URL: http://dl.acm.org/citation.cfm?id=545381.545469 (cit. on p. 11).

[15] Alexandros Nanopoulos and Panayiotis Bozanis. "Categorical Range Queries in Large Databases". In: *Advances in Spatial and Temporal Databases*. Ed. by Thanasis Hadzilacos et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 122–139. ISBN: 978-3-540-45072-6 (cit. on p. 11).

[16] Gonzalo Navarro. "Wavelet Trees for All". In: *J. of Discrete Algorithms* 25 (Mar. 2014), pp. 2–20. ISSN: 1570-8667. DOI: 10.1016/j.jda.2013.07.004. URL: http://dx.doi.org/10.1016/j.jda.2013.07.004 (cit. on pp. 1–4, 9).

[17] Yakov Nekrich. "Efficient Range Searching for Categorical and Plain Data". In: *ACM Trans. Database Syst.* 39.1 (Jan. 2014), 9:1–9:21. ISSN: 0362-5915. DOI: 10.1145/2543924. URL: http://doi.acm.org/10.1145/2543924 (cit. on pp. 1, 11).

[18] Yakov Nekrich. "Space-efficient Range Reporting for Categorical Data". In: *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. PODS '12. Scottsdale, Arizona, USA: ACM, 2012, pp. 113–120. ISBN: 978-1-4503-1248-6. DOI: 10.1145/2213556.2213575. URL: http://doi.acm.org/10.1145/2213556.2213575 (cit. on p. 11).

[19] Manish Patil et al. "Categorical Range Maxima Queries". In: *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. PODS '14. Snowbird, Utah, USA: ACM, 2014, pp. 266–277. ISBN: 978-1-4503-2375-8. DOI: 10.1145/2594538.2594557. URL: http://doi.acm.org/10.1145/2594538.2594557 (cit. on p. 11).

[20] German Tischler. "On Wavelet Tree Construction". In: *Proceedings of the 22Nd Annual Conference on Combinatorial Pattern Matching*. CPM'11. Palermo, Italy: Springer-Verlag, 2011, pp. 208–218. ISBN: 978-3-642-21457-8. URL: http://dl.acm.org/citation.cfm?id=2018243.2018264 (cit. on p. 1).