Big Data Analytics

Homework 2 (Z-order and R-tree Family)

Instructor : Ruoming Jin

Name: Shiva Kumar Peddapuram

KSU ID: 811235874 speddapu@kent.edu

3.) Please list at least 2 differences between B^+-tree and R-tree. [10 points]

S.NO	B+- Tree	R -tree
1.)	B+ Tree can be used for one dimension structure of elements	R tree can be used for multi-dimensional structure of elements like 2d 3d
2.)	Used for storing large amount of data that cant be stored in the man memory.	Example: we can use this tree for representing related to geographical like latitudes and longitudes
3.)	Only one operation needed to be done at a single time.	We can perform operations like inserting and deleting at a time
4.)	Time complexity is : O(log(n))	Time complexity for this also almost same but it differs for N-dimensional range to access the mentioned keys.
5.)	B+ trees allows us to search orderable items in secondary memory.	R trees allows us to search for elements that are near or at a particular point.
6.)	B+ Tree identifies all the keys which are in within the certain range	This is a Spatial index, that is it can identify all the close values of keys in mentioned dimensions.
7.)	Both of them stores the data in the leaf	These tree also stores the data in the leaf.
8.)	Linked list is bound the nodes in this kind of trees.	MBRs that is Minimum bounding rectangle is used to bound the nodes in R trees.

2. For Z-ordering curve, with 16 × 16 pixels (or cells), please use the bit-shuffling method to find the mappings between cell locations and Z-values below. [20 points]

2a. Please illustrate how to calculate the Z-value for the cell with the 2D location (6, 15) *[note: cell locations start from 0 on each dimension]*. [10 points]

Solution:

From the above scenario the given 2D location are (6,15)

Let us calculate the Z-value for the mentioned 2D cell by using Bit Shuffling method.

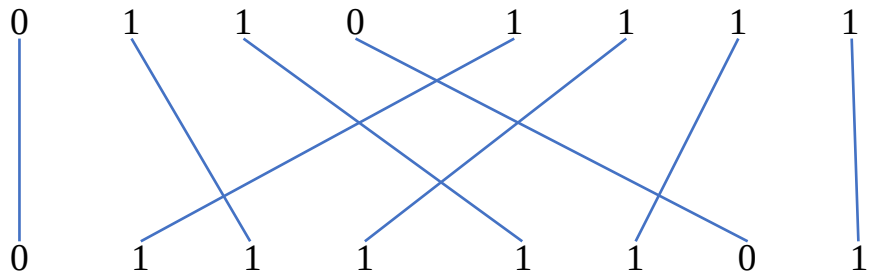
We can represent decimal 6 in 4-bit binary numbers as = 0 1 1 0

Same for the decimal 15 we can represent them in 4-bit binary number as = 1 1 1 1

Let us consider the (0110) as (x1, x2, x3, x4) and (1111) as (y1, y2, y3, y4)

After Bit shuffling it is written as (x1, y1), (x2,y2), (x3,y3), (x4,y4)

- **Bit Shuffling: representation**



Bits after shuffling:

$$(0\ 1\ 1\ 1\ \ 1\ 1\ 0\ 1\ )_2$$

- **Decimal Value = 2^0*1 + 2^1*0 + 2^2*1 + 2^3*1 + 2^4*1 + 2^5*1 + 2^6*1 + 2^7*0**

  = 1 + 0 + 4 + 8 + 16 + 32 + 64 + 0

  = **125**

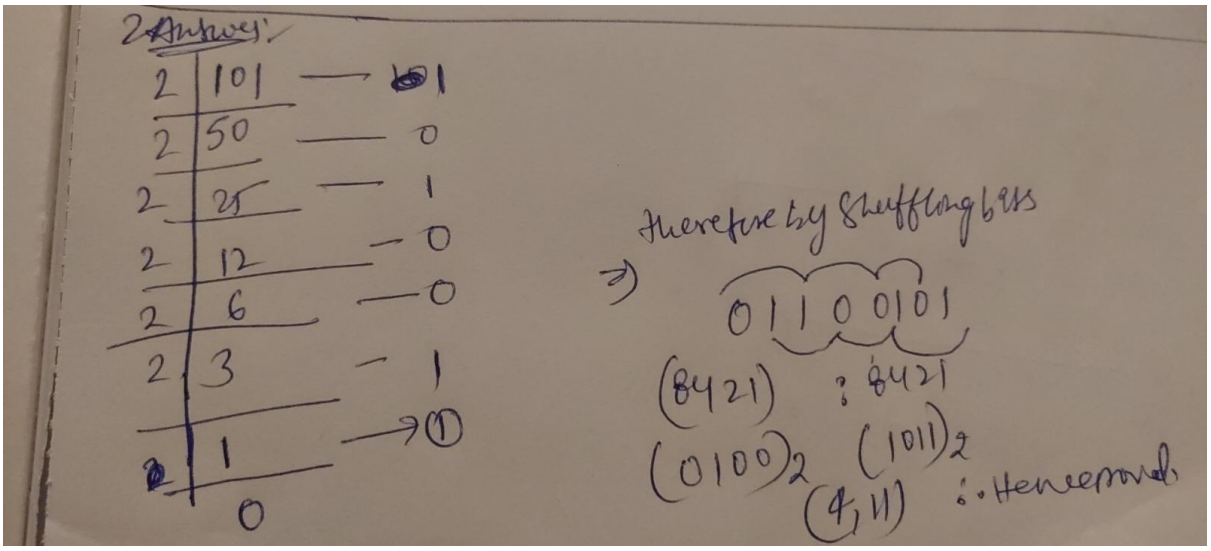
- **Hence, Decimal Value corresponding to cell (6,15) = 125**

2b. Given a Z-value 101 (decimal number), please identify the 2D location of the cell corresponding to this Z-value. [10 points]

Answer:

Solution 2b:-

- From the above question the Given Z-Value is = 101
- First let us convert $(101)_{10}$ to 2D location by using Bit Shuffling method.
- Binary representation of 101 will be: 0 1 1 0 0 1 0 1
- The above binary representation value is in the form of (x1,y1), (x2,y2), (x3,y3), (x4,y4) = ((0,1),(1,0),(0,1),(0,1))
- We can represent the 2D Location value as (x1x2x3x4,y1y2y3y4) = (0100,1011)
- So now when we convert the above value from binary to decimal form then it becomes as below (4,11).
- At final The Co-ordinates are as (4,11).
-



3.) Given two 4-dimensional *minimum bounding rectangle* (MBRs) in an R-tree, *A* = (2, 10; 40, 60; 32, 40; -2, 11) and *B* = (3, 4; 30, 50; 30, 40; 20, 23), please use an MBR, *E* to minimally bound both MBRs *A* and *B*. [20 points]

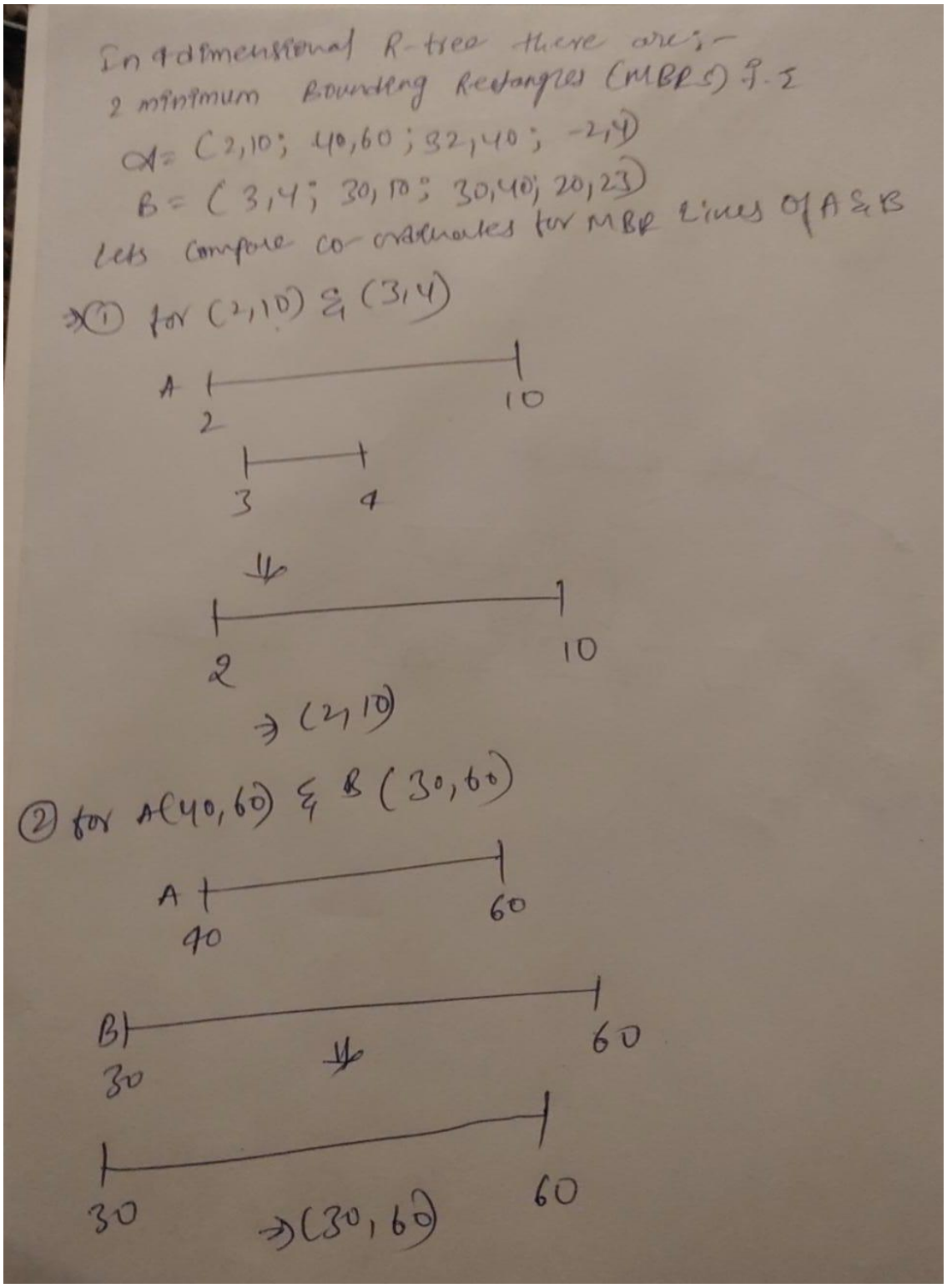
Solution:

From the above question the given 4-dimensional minimum bounding rectangle (MBRs) in an R-Tree are :

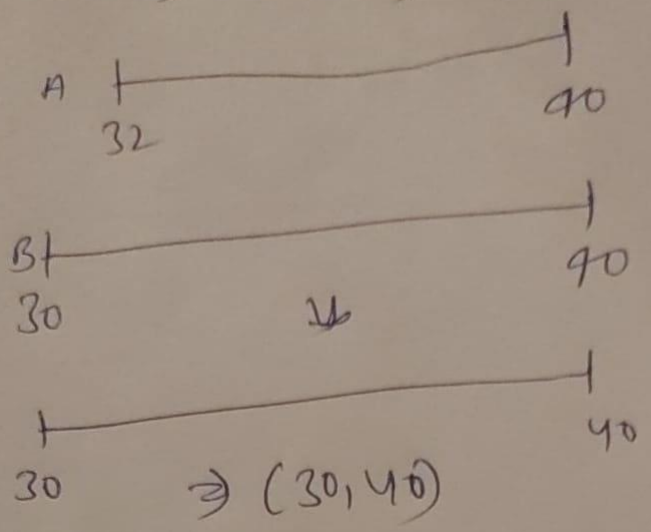
A = (2,10; 40,60; 32,40; -2,11)

B = (3,4; 30,50; 30,40; 20,23)

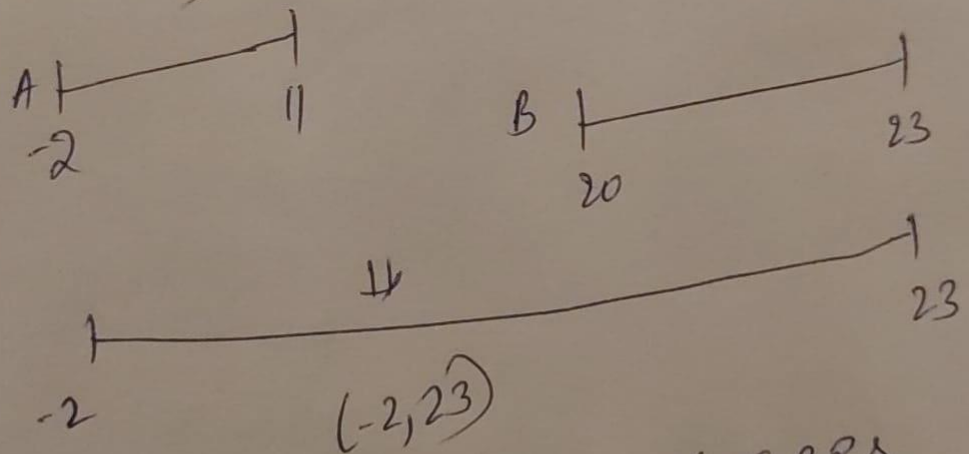
Let's me consider the 4 Dimensions



(3) for A(32,40) & B(30,40)



(4) for A(-2,11) & (20,23)



Hence 4 dimensional MBR's of A & B's

E = (2,10; 30,60; 30,40; -2,23)

4.) Given two *d*-dimensional MBRs, E_1 = (x1min, x1max; x2min, x2max; …; xdmin, xdmax) and E_2 = (y1min, y1max; y2min, y2max; …; ydmin, ydmax) (note: both E1 and E2 are represented by an array in the code), please give the pseudo code to check whether or not two MBRs are intersecting with each other. If yes, please return true; otherwise, return false. [20 points]

Answer:

Here Intersect is used as Boolean Variable

Var Intersect = false;

//MBR are stored in the form of array was a two items per dimensions.

Array E_1();

Array E_2();

// To determine whether two MBR intersect or not we use for loop.

for(int k = 0;k<d; k++) {

If(E_1[i]>=E_2[i] && E_1[i] <= E_2[i+1]) Or (E_1[i+1] >= E_2[i] && E_1[i+1] <= E_2[i+1])

Or

(E_2[i] >= E_1[i] && E_2[i] <= E_1[i+1]) Or (E_2[i+1] >= E_1[i] && E_2[i+1] <= E_1[i+1])

){

// If condition is true, intersect variable value will be changed to true; or the loop will examine additional dimension.

Intersect = true;

Return Intersect.

}

Return Intersect.

}

// If two MBR are not intersecting it will return false.

5.) Write an algorithm to perform a range query (with rectangular shape, Q) in a quad-tree (please write comments to explain the meaning of your pseudo code). [30 Points]

Ans :-

- A partitioning index called Quad Tree will divide the 2d data space into four equally sized quadrants.

- Each quadrant is divided recursively into four equal quadrants, and then each quadrant is divided into four regions named North West, North East, South West, and South East.

- This Quad tree can be utilized to respond to range-related queries in Range Query on Quad tree.
- If we have a rectangle shape Q aligned with a coordinate axis, range searching is a straightforward technique.
- We initially begin our search at the root, and if we are unsuccessful there, we move on to the sub-quadrants, where we check each point to see if it is a part of the range or not.
- If it falls in our range, then the outcome will include that point; otherwise, it will pass through other points.
- The result will be displayed after the navigation is done.

Pseudo Code

// The given range is stored in the variable,

var given_range;

// Here, we define a function which performs all the range query which is related to the specific quadrant.

function Range_Query(given_range) {

we use an array to store all the results of the range query that is running in the current quadrant.

Quadrant border is stored in the variable var bound.

Arr result=null;

// If the result is true, then the present quadrant will be checked to see if it is in the range. If it is not in the range then it will just pass over the area.

If(bound.intersects(given_range)= =false)

Return result;

//we use for loop to iterate thoroughly every point in the current quad.

We determine If children are available or not..

for (int k=0; k<curr_quad.size; k++) {

If(range.contains(curr_quad.points[]))

Result.append(curr_quad.points[])

}

// if the child quadrant is not present we must return the current result

If (curr_quad.NorthWest= =NULL) {

Return result; }

Else {

//Range query function is called recursively on all other regions of the quadrant if the child quad is inside the range.

```
Range_query.append(NorthWest.Range_query(given_range));
Range_query.append(NorthEast.Range_query(given_range));
Range_query.append(SouthWest.Range_query(given_range));
Range_query.append(SouthEast.Range_query(given_range));
}

//This returns the final result of a range query.
Return result.
}
```

Bonus Question [20 extra points]

6. Read Sections 1 and 2 of the following paper and write a short survey about the formal definition of *reverse nearest neighbor* (RNN) query and solutions of RNN processing (including KM, YL, MVZ, SAA, and SFT). Use one paragraph for each solution (*note: please use your own words to describe the problem definition and solutions; DO NOT copy any sentences from the paper*).

Yufei Tao, Dimitris Papadias, and Xiang Lian. Reverse *k*NN Search in Arbitrary Dimensionality. In *Proceedings of the Very Large Data Bases Conference* (VLDB'04), pages 744-755, Toronto, Canada, Aug. 30-Sept. 3, 2004. *Located in the Library Course Reserves on the left-hand course menu*

Answer:

RNN Query (Reverse Nearest Neighbor Query):

- This query returns all data points that are closest to the specified point p.
- There are a few issues with how the RNN query is currently processed, including:
- They are unable to support arbitrary k values.
- Only 2D data can be used in this query.
- An algorithm was created to execute the RkNN query in order to get around these issues.
- An RkNN query uses a point q and a multidimensional dataset p to discover all of the k points that are closest to that point.

KM Algorithm:

- It is often known as KM00, was created by Korn and Muthu Krishna.
- It is an original of RNN approach that computes the data in advance for each data point, determines its nearest neighbor, and locates a circle nearby.

- Next, an R-tree is used to efficiently retrieve the nearest neighbor by indexing the minimum bounding rectangle of all the circles.

YN Algorithm:

- Yang and Lin combine two indices as a result.
- For that, the RdNN-tree was used.
- This tree has the same structure as an RNN-tree and can process RNN and NN tree queries more quickly.

MVZ Algorithm:

- The MVZ Approach, which is restricted to 2D spaces, is another pre-computation algorithm that aids in identifying the worst-case constraints.
- The issue with this algorithm is that it will not function correctly with any type of insertions or deletions.
- As long as the procedure is ongoing, insertions and deletions are permitted.

SAA Method:

- Similar to the SAA algorithm, a 3-dimensional data set makes the SAA algorithm's performance inefficient due to the increased number of regions that must be examined.
- SFT03 is a multi-step method that operates effectively on multidimensional datasets. This approach starts with the closest k neighbors pointing p in the multidimensional dataset.
- Second, all the points that are nearer to other points than q are eliminated.
- In the end, SFT calculates query q RNN over all points that are similar to q in the multidimensional dataset using a Boolean range query.

SFT METHOD:

- The SAA Algorithm loses its efficiency when applied to the #D data sets.
- The SFT03 method has different multiple steps and effectively handles multidimensional datasets.
- In this approach, the closest k neighbors to a point p are found first in the multidimensional dataset.
- Then Secondly, it eliminates every point that is nearer to another point than q. In the end, SFT calculates query q RNN over all points in the multidimensional dataset that are similar to q using a Boolean range query.