

Spatial Informatics
Prof. Soumya K. Ghosh
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

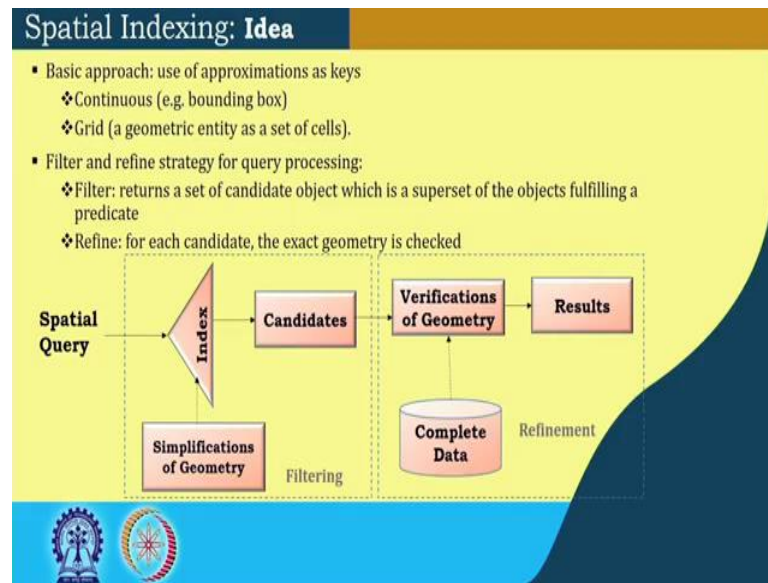
Lecture - 18
Spatial Indexing – II

Hello, so we are discussing about Spatial Indexing right, as we discussed that these are one of the very fundamental aspects which need to be looked into from the context of what we say back end management, right. So, like when a query comes we want the query to be efficiently resolved right, efficiency both in terms of time and space if you look at the space it in terms of time.

So, there is for spatial data we have issue for both it is computers and IO intensive. So, for the data which is stored in which is pretty large volume stored over a lot of over a different disk sectors. So, it becomes more time consuming when those are scattered, so or when there are lot of scanning of the data across that disk. So, this indexing helps us in for our traditional data set also in order to look at this data in a much efficient way, right.

So, spatial indexing what we are trying to see if you recollect your previous or previous discussion on this topic. That how it help in query resolution or how it help in spatial a queries involving spatial operators or spatial different other spatial aspects of the things like topological operator so and so forth.

(Refer Slide Time: 01:56)



So, will look at the what is the fundamental of the basic idea is that use of approximation at the accuse like continuous, like say bounding box right, grid a geometric entity for a set of cells, right. So, what we are doing that instead taking that object whether I can form a approximate objects of the things, right. As we are discussing I want to find out whether to say polygon overlaps or a route or a particular polyline and a polygon overlaps.

So, in; so, one way is that I go pixel, by pixel or I check every point that whether it is overlapping otherwise I create a bounding box right or popularly known as MBR or MOBR that is maximum that bounding box I create. And if those binding bounding box overlaps then there is a chance that embedded polygon on embedded geometry structure inside it may overlap. If those two bounding box does not overlap then there is no way that embedded thing will overlap.

So, instead of taking two polygons I take two MBRs which is much easier to calculate right, a rectangle maximum bounding rectangle is easy to calculate you require only two coordinate points, right. And then if the and checking also is much easier, so if the overlap is fine, if they do not overlap then I can discard that the embedded or the polygon which is whose MBR we have generated will not overlap.

So; that means, and second other is grid I put a grid over the structure right, like it is as good as we can think of as a sort of a square grid graph paper transparent graph paper

over the things. And see that which pixels and where the type of things right and then can create different buckets and do some sort of a grid analysis. Now, filter and refine strategy is for query processing like one is filter return a set of candidate objects which are super set of the objects filling the predicate, right.

So, filtering I want to find out that candidate objects which is superset of all the objects which are in the queries and refine for each candidate object exact geometry is checked, right. Like, what we say that spatial query I fall back to the index and then looking at that simplification of the geometry like creating a MBR instead of the actual geometry of the things, right.

So, simplification of the geometry and within the simplification of the geometry if it is satisfying my condition then I say those are the prospective candidates. Now, in the refinement I do the verification of the geometry and take the complete data set, and do the very using the complete data set and the result is generated right.

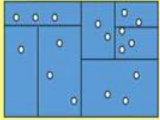
In this case, by doing this I can exclude lot of objects which are not within the thing. Like, I want to find out that say due to say flood in particular region how much of a particular say; how much of district of say a particular district say with Midnapore is in a network or agriculture field is getting affected. So, that if I take the whole districts of the database then is the whole India districts are there right or if I take the your flooded zone there may be different flooded zone etcetera.

So, this I can do, I can create this MBR which are much easier to calculate then with the Midnapore whichever the flood plains are overlapping I only consider those flood place rest I exclude. So, I reduce these candidates from a huge number to a manageable number, then actually I look at the geometry from the complete data and then calculate the results.



(Refer Slide Time: 06:19)

Spatial Indexing : Memory Organization

- A spatial index structure organizes points into buckets.
- Each bucket has an associated *bucket region*, a part of space containing all objects stored in that bucket.
- For point data structures, the regions are disjoint & partition space so that each point belongs into precisely one bucket.
- For rectangle data structures, bucket regions may overlap.



A *kd*-tree partitioning of 2d-space where each bucket can hold up to 3 points



So, that is; that where we are by indexing we can do a simplification of the geometry and this IO thing. So, on the context of memory organization a special index structure organization points into buckets, so points are contained in different buckets each bucket an associated bucket region. So, I say this bucket is particular this region is in the bucket whatever the things coming I put in that buckets, a part of the space containing all objects stored in that buckets, right.

That I want to find out that, so which are the medical store, which are the book stalls etcetera, in a particular region. Then if I have bucketize these things then I can hit that particular region and see that these are the buckets which are belonging to this region and then see that what are the content of the point feature etcetera.

So, for point structure the regions are disjoint right and partition space, so that each point belongs precisely in one bucket, right. So, I cannot; I do not I partition the space like that a particular point feature exactly fit into the one bucket. It cannot be either this bucket or that bucket right, those type of things are not there for rectangle structure bucket regions may overlap.


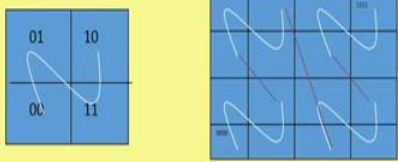
So, for rectangle structure or what we say other than points are like polygon structure or rectangle, where that MBR is rectangle for that particular object the bucket region may overlap, right. So, it is a if you can see the kd; a typical kd tree partitioning in 2d space where each bucket can holds up to 3 points in this case. So, where ever it is more than 3

point I again partition the thing region, into so that every bucket contents a max 3 points right, so in the 2d space, so that is kd tree partitioning.

(Refer Slide Time: 08:18)

Spatial Indexing: 1-D Grid approx.

- One dimensional embedding: Z-order or bit-interleaving
 - Find a linear order for the cells of the grid while maintaining “locality” (i.e., cells close to each other in space are also close to each other in the linear order)
 - Define this order recursively for a grid that is obtained by hierarchical subdivision of space



So, there is a one dimensional grid approach, so one dimensional embedding grid like Z order or bit interleaving one curve or bit interleaving approach. So, find a linear order for the cell in the grid while maintaining a locality that is the cell closer to each other in space are also closer to each other in the linear order, right. See our problem; our basically hunch is like whenever I m searching something which are if that is locality is one of the important aspects.

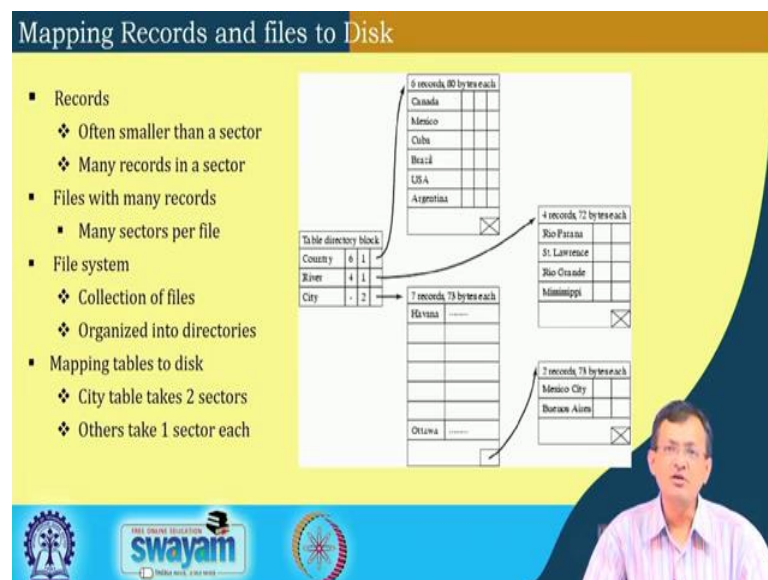
Like if you see the basic philosophy of this geo spatial data sets are nearby objects are more similar than distance object, right; temperature of IIT Kharagpur building one with respect to building two of IIT or a particular region r_1 of within IIT campus to r_2 within the IIT campus. They say temperature will be more near than IIT, Kharagpur region 1 with something say in Rajasthan or Simla or somewhere in north eastern side of things, right. So, more distance object a more closer objects are likely to be more similar type of property right, so that is one of the hunch.

Now, in order to do that instead of going say raster scan, then raster scan like this I can basically do it in a space filling way right, so like say linear order of the cell of a grid while maintaining locality. So, I make a ordering like 00, 01 somewhere this is I ordering instead of maintaining the locality, so it is within the locality. Instead otherwise I have to

go like this and they again scan back like this etcetra this is 2 by 2, but it can be more say 10 by 10 type of things. And that is what is the cells close to each other in space or also close to each other in the linear order, right.

So, that I that closeness in the space I want to capture by my this ordering technique or indexing technique, right. So, this is a typically a Z curve is one of the example there is a Hilbert or H curve also you see. Define this order recursively on a grid that is obtained by hierarchical sub division of a space right, so I can do a recursively use that. And then I can generate like one is this one, then go there and this one, then come back and this one etcetra right, so I can do a hierarchical subdivision of space right, so that is possible.

(Refer Slide Time: 11:26)



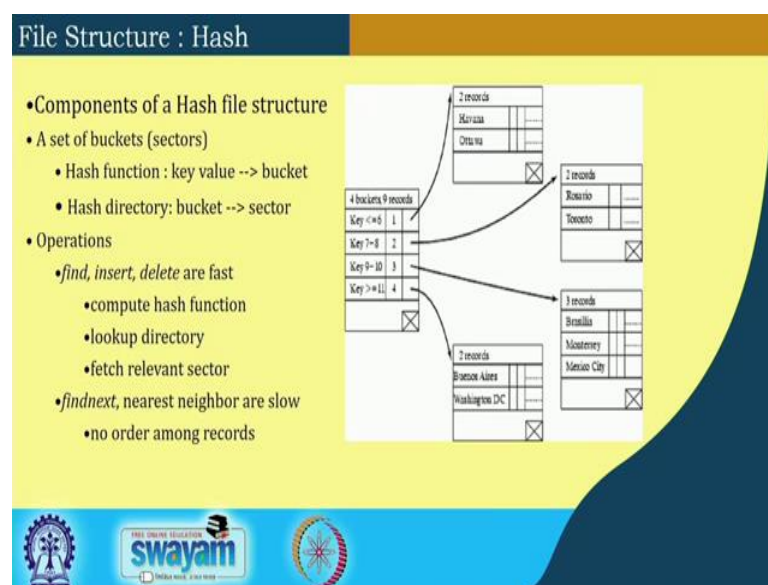
So, now again looking at some of the standard typical approaches what is followed in our standard database management system or any traditional databases were some of the nomenclature which is definitely valid here. So, records often smaller than a sector, so we have sectors and the records are often many sectors, many records in particular sector, so a particular sector in the say disk contain many records.

Files with many records with many records so many sectors per file, so a particular sector with many records; many records, a many sectors per file and a file system is collection of files. So, I have a collection of files, then many sectors in a file and then

many records in a sector right. So this is the way it organize, organized into directories etcetera mapping tables to disk.

So, city tables take two sectors others takes one sector each and type of things right. In this particular things like in this county country table takes one sector each city table takes two sector and type of things and it basically points to where the things. So, if you see it is a hierarchical arrangement of the things, this is which is used in our traditional data sets.

(Refer Slide Time: 12:59)



There is another popular things what we what is being used is the hash or hash function right, hash file structure; a set of buckets or sectors has function key value mapped to the particular bucket. So, I have the key value on this has hash function as a that if I apply this function it will be map to that bucket has directory bucket to a sector, right. So, particular key value to a bucket and for hash directory this bucket to a particular sector ok.

So, operations, so what it is fine, so hash function and hash directory one is the key value to the bucket and the bucket to the sectors, right. So, this is a hashing standard hashing operation and operations there are for looking at the operations like find, insert, delete are pretty fast when you use this type of things right are operator referred target.

Compute hash function look up directory, fetch relevant records and type of thing, so it computes the hash function go to that particular look up directory, fetch the relevant record and goes on like this, right. So, find next is the nearest neighbor are slow here like, so when we have a nearest neighbor type of things then we have to go for find next type of things and this is slow no order among the records.

There is no inherent ordering among records like, I do not know that where the next will come into play and type of things, right. Here it is everything based on this hash function I do not have this near nest of the locality mapped into the things right. So, this is there, so that find next may be costly or slow process.

(Refer Slide Time: 14:46)

Spatial File Structures: Clustering

- Motivation:
 - Ordered files are not natural for spatial data
 - Clustering records in sector by space filling curve is an alternative
 - In general, clustering groups records
 - accessed by common queries
 - into common disk sectors
 - to reduce I/O costs for selected queries
- Clustering using Space filling curves
 - Z-curve
 - Hilbert-curve

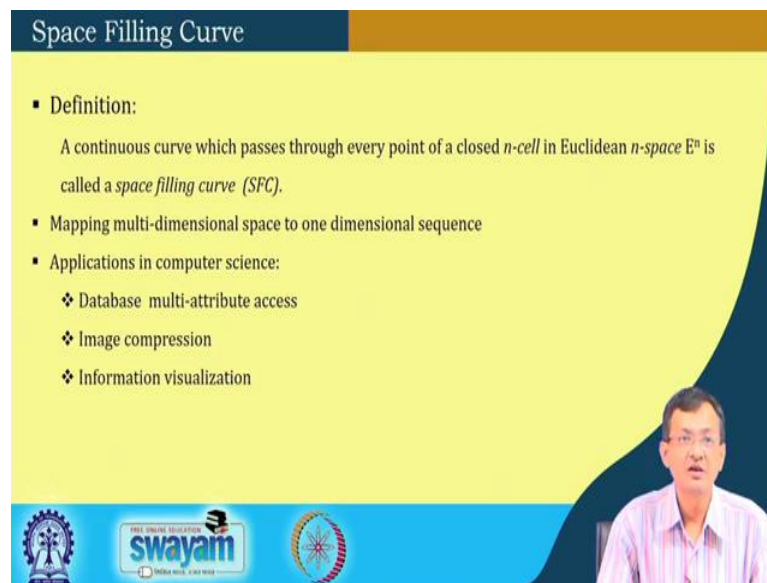
The slide features a yellow background with a blue header and footer. A video inset in the bottom right corner shows a man in a striped shirt speaking. The footer contains logos for 'swayam' and other educational institutions.

So, another aspects of spatial file structure is the clustering right, motivation; ordered file are not natural for spatial data, right. So, though it is natural for our traditional data set ordered file are not so natural for this, so there is more of the nearness of the locality etcetera are there.

Clustering orders in sector by space filling curve is an alternatives right, in general clustering group records, accessed by common queries, into common disk sectors to reduce IO costs for selected queries, right. So, in general clustering groups records, so when you do a cluster a groups into records of similar type what we say cluster of this cluster of this access by common queries.

So, that why we group them it is accessed by the common queries into common; if I can put them into common disk sectors, so the access rate will be high and reduce IO cost for selected queries. So, if I have a set of queries which are looking at the thing then it may help in reducing IO cost. So, the though two popular technique for clustering using space filling curve one is Z curve, and Hilbert curve, right. So, these are H curve or Hilbert curve these are the two popular clustering using space filling curve or SFC what we say.

(Refer Slide Time: 16:17)



Space Filling Curve

- **Definition:**
A continuous curve which passes through every point of a closed n -cell in Euclidean n -space E^n is called a *space filling curve (SFC)*.
- Mapping multi-dimensional space to one dimensional sequence
- Applications in computer science:
 - ❖ Database multi-attribute access
 - ❖ Image compression
 - ❖ Information visualization

At the bottom of the slide, there are logos for 'swayam' and 'INDIA RISES WITH EDUCATION'.

So, when you talk about space filling curve, so what this it a continuous curve which passes through every point of a closed n cell, so n number of cell in a Euclidean in space of e to the power n or E^n is called a space filling curve, right. So, I have a close thing and it should a curve which will go to the every points in a particular fashion, right.

So, if that traversal index helps me in better clustering of the points based on the characteristics of the features of that particular region of interest, then it helps in clustering, access rate, better quick access rate and type of things. So, mapping multi dimensional space into one dimensional sequence is one of the thing.

Application in computer science in CS is there are things database multi attribute access, image compression if they are together then I can have a possibility of compression like I can store incremental data etcetera like I know that these values are in the same cluster. They are near by values then I store one and those are other just incremental values

across the thing. And better information visualization, so there are several application if we look at the CS point of view.

(Refer Slide Time: 17:51)

Z-Curve

- What is a Z-curve?
 - A space filling curve
 - Generated from interleaving bits
 - x, y coordinate
 - Connecting points by z-order
 - looks like Ns or Zs
- Implementing file operations
 - similar to ordered files

So, now we look at the Z curve alright, so it is a space filling curve generated from interleaving bits right. So, x y coordinate are interleaved, connecting points by z order alright, that is looks like n or z. Like if you see here it is a z-order right, z then this again this z order and type of things z order or looks like n also. Implementing file operations similar to ordered files, right.

(Refer Slide Time: 18:33)

Hilbert curve: Geometric Generation

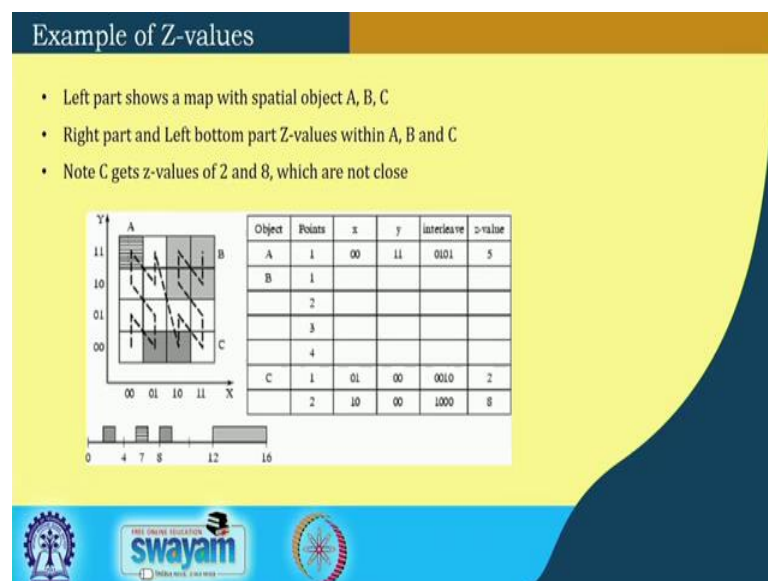
- If I can be mapped continuously on Ω , then after partitioning I into four congruent subintervals and Ω into four congruent subsquares, each subinterval can be mapped continuously onto one of the subsquares. This partitioning can be carried out ad infinitum.
- The subsquares must be arranged such that adjacent subintervals are mapped onto adjacent subsquares.
- Inclusion relationship: if an interval corresponds to a square, then its subintervals must correspond to the subsquares of that square.
- This process defines a mapping $f_h(I)$, called the Hilbert space-filling curve.

There is another category of curve which is Hilbert curve it is geometric generation. So, if I can be mapped in a continuously on a sigma, then after partitioning I into four congruent intervals and the sigma into four congruent subsquares, each sub interval can be mapped continuously onto one of the subsquare right. We will see that example; this partitioning can be carried out at infinitum, so I go on partitioning like that then I generate those type of curve.

The sub square must be arranged such that adjacent sub intervals are mapped into a adjacent sub squares alright, so that I mapped them into adjacent sub squares. So, there is a inclusive relationship that if a interval correspond to a square then it is sub interval must correspond to the subsquare of that square, right.

So, if it is; there is a inclusion relationship if the interval corresponds to a square then sub interval must corresponds to the subsquare of that square. This process is defined by the mapping f h of n or what we say Hilbert's space filling curve may be, right.

(Refer Slide Time: 19:48)



Like let us look at it, so the example of Z-curve like in this case C A, let us this A is having value of 00 11. So, that is interleaved 01 00 and then we end up to a value of 5; so Z value is 5, though it is 00 11 Z value is 5. Similarly, you can calculate for B, like in this case B is having these are the values, right.

So, if this value is 10 10 then I if I interleave 10, so it becomes 1100 right, so it is 8 to the power 3 plus 4 8 plus 4 then you get 12, right. So, I get a value of these values mapping to the two corresponding values, like if we see it is 0 1 0 0 interleaved 0011 2 and this one is 00 1000. So, after interleaving also it is 1000 8 and here we get this is 12 13 14 15 for the your B value.

Now, if I look in if I try to look it if it is indexed in that fashion means if it is indexed in the Z values, then what I have a if I put it like this scale. So, A I get 5 then here we get another value here actually the displays a this thing is not there. So, it is basically on 5 right and then 8 and 2 are for C and 12 to 16 are for B.

Now, see this is though C is having 2 and 8 values, so left part shows the map of the spatial object A B C and right part this or left bottom is the Z value of the A B C. Now, what we say C gets A Z value 2 and 8 which are not close other things are close; that means, if I want to extract B and if it is ordered in this sequence and put on say on the disk then you wants to extract B. And I need to now for this vanilla example I need to access 12 13 14 15 or so in the scale of 12 13 14 15, right.

Last one is 1111 interleaved with 1111 is 15 right, so, this is 12, then we have 13, then 14, then 15, ok. So, I get in a one go, it is not separated across things, but there is a there is a challenge here also it is not like that all will able to map on a. It depends on that how your data is distributed in the space and then for this typically here for the C we get to and 8.

(Refer Slide Time: 23:23)

Hilbert Curve

- A space filling curve
- More complex to generate
 - ❖ due to rotations
- Implementing file operations
 - ❖ similar to ordered files

Logos: IIT Bombay, SWAYAM, and a circular logo with a star.

Hilbert curve is space filling curve more complex to generate due to rotations right, so it is Hilbert curve is a space filling curve is more complex to generate. Implementing file operations similar to ordered files right, like here this 0 0 0 1 1 0 and this is the way it is implemented and it goes on doing the things, right.

So, it is a best filling curve, but it is difficult to generate it is not the very standard logic as we have seen in the Z-curve, like here if you look at it will be rotated in the this fashion when if you look at Hilbert implementation algorithm.

(Refer Slide Time: 24:11)

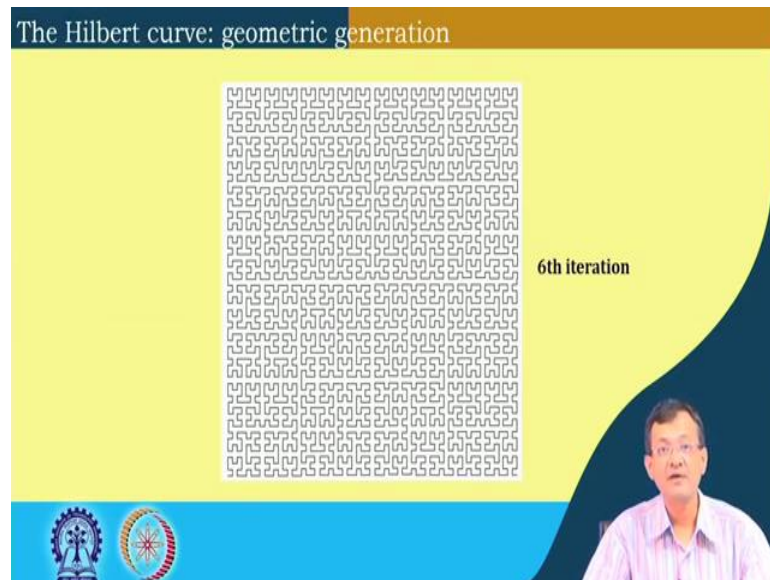
The Hilbert curve: Geometric Generation

1st iteration 2nd iteration 3rd iteration

Logos: IIT Bombay, SWAYAM, and a circular logo with a star.

But it is space filling right, so we have this sort of things that 1 iteration we get these, next iteration more things and after finite number of iterations the curve is a space filling curves, right.

(Refer Slide Time: 24:24)



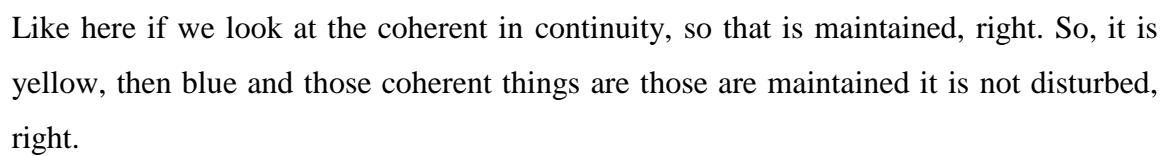
After 6th iteration we get this type of a recursively if it go on iterations, then this sort of geometry is generated.

(Refer Slide Time: 24:34)




So, what we found this properties of this Z curve and Hilbert curve if we try to look at, it is coherent in continuity both are coherent in continuity. If we look at the clustering

(Refer Slide Time: 25:07)






Clustering property



The left diagram shows a grid with points and lines connecting them, with a red shaded region. The right diagram shows a grid with points and lines connecting them, with a red shaded region. Below the diagrams are two horizontal bars labeled Z and H, representing different clustering metrics.

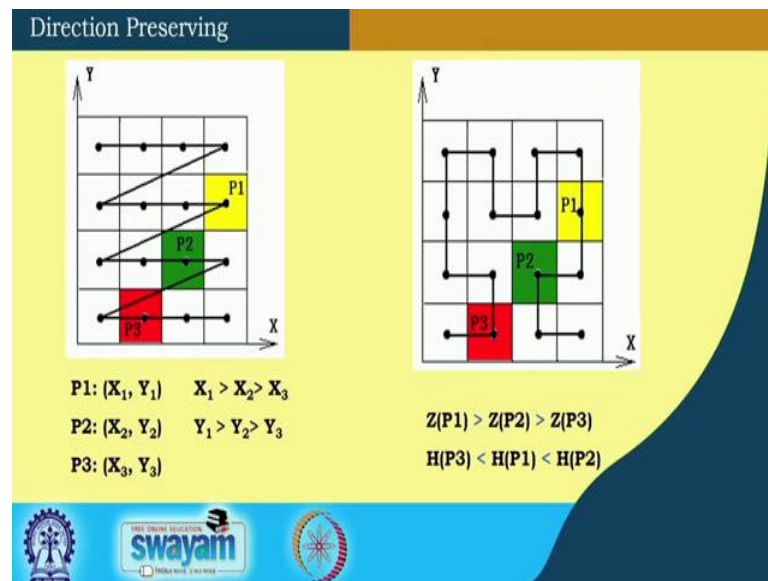
Z

H



Clustering property if you see Z curve, so it is say if the rate thing is it in this though the blue is clustered in a single cluster, but these are on different patches. Whereas, this patch is better, so we have only two one cluster for this red patch whereas, two cluster for this blue patch, right, so this is much is better than the things right. Is again very synthetic example, but we can try to show that this type of properties are.

(Refer Slide Time: 26:00)





Now, if you look at the directional property if there are three points, P1, P2, P3; where P1 is X_1, Y_1 ; P2 is X_2, Y_2 ; P3 is X_3, Y_3 and also it contains X_1 greater than X_2 , greater than X_3 ; Y_1 greater than Y_2 greater than Y_3 then we have a P X_1, Y_1 ; X_2, Y_2 ; X_3, Y_3 is in some fashion, right. And now, if you see Z of P1 if you calculate Z value of P1 it is more than the Z value of P2 greater than the Z value of P3.

So, that this direction property is maintained, what we direction preserving what we try to, so the direction preserving is z curve is more direction preserving right. Whereas, in case of Hilbert that H of Hilbert value of P3 is less than Hilbert value of P1 is less than Hilbert value of P2, right, so it is this what we see that Z is more direction preserving. So, it all depends that what is the major focus of that particular problem we want to address and type of things.

(Refer Slide Time: 27:17)

Spatial Indexing: 1-D Grid approx.

- Any shape (approximated as set of cells) over the grid can now be decomposed into a *minimal* number of cells at different levels (using always the highest possible level)
- Hence, for each spatial object, we can obtain a set of "spatial keys"
- Index: can be a B-tree of lexicographically ordered list of the union of these spatial keys



There is a one grid approach that any shape approximate as a set of cells over grid can now be decomposed into a minimal number of cells in different levels using always the highest possible levels, right. So, for each spatial object we an obtain a set of spatial keys.

Index can be done by B-tree or lexicographically can be a B-tree or lexicographically ordered list of the union of these spatial keys, right. So, I can have these that is we can say that these are my keys and then I can do a B-tree construct a B tree out of this by for indexing into the things using those things.

(Refer Slide Time: 27:57)



Spatial indexing: 2-D points & Rectangles

- **Data structures representing points :**
 - kd-tree and its extensions (KDBtree and LSDtree)
 - Grid file (organizing buckets into an irregular grid of pointers)
- **Spatial index structures for rectangles:** unlike points, rectangles don't fall into a unique cell of a partition and might intersect partition boundaries
 - Transformation approach: instead of k-dimensional rectangles, 2k-dimensional points are stored using a point data structure
 - Overlapping regions: partitioning space is abandoned & bucket regions may overlap (e.g. R-tree & R*-tree)
 - Clipping: keep partitioning, a rectangle that intersects partition boundaries is clipped and represented within each intersecting cell (e.g. R+-tree)

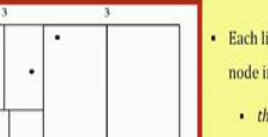
Now, if we have a 2-D points and rectangle points like kd tree and its extensions like KDB tree and LSD tree type of things or grid file organizing bucket into irregular grids of pointers right, so I can we can have different buckets we store the this pointers. So, spatial index structures for rectangle unlike points rectangle do not fall on a unique cell right I cannot put them in a grid or in a grid cell and type of things of a partition and might intersect partition boundaries, right.

Transforming approach instead of k dimensional rectangle, 2k-dimensional points are stored using a point data structures and type of things. And we will see later on that we have partitioning space of is abandoned and bucket regions may overlap like, so I can if it is a box then the bucket region may overlap. So, we need to fall back to structure like R tree or R plus tree right, so there where it allows overlapping.


Now, clipping keeps partition rectangle that interacts partition boundaries are clipped and the represented within each intersecting things. So, there is a variant of R trees and R plus tree where you clip this where wherever the overlaps are there and put them into the with intersecting cells.

(Refer Slide Time: 29:24)


Division of Space by a k -d Tree



- Each line in the figure (other than the outside box) corresponds to a node in the k -d tree
 - the maximum number of points in a leaf node has been set to 1.*
- The numbering of the lines in the figure indicates the level of the tree at which the corresponding node appears.



anna university
chennai



swayam

free online education



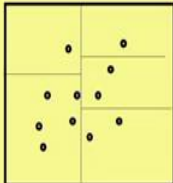
anna university, chennai

So, each cell in the figure other than outside box corresponding to a node in the kd tree right this is a point thing, in this case the maximum number of points in a leaf node has been set to 1. The numbering of the lines in the figure indicates the level of the tree at which the corresponding node appears, right.

So, the numbering in the lines like if there are 1 2 3 this numbering in the lines in the figure indicates the level of the tree at which the corresponding nodes appear and, so this partition and then goes to this partition and go on types of things.

(Refer Slide Time: 30:08)

- But: insertions/deletions are tricky (splits may propagate downwards **and** upwards)
- No guarantee on space utilization

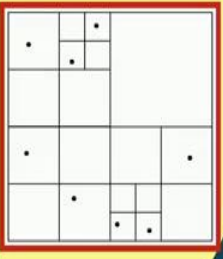



So, there are some disadvantages in kd tree what will what we can see insert, deletion are tricky; split may propagate downwards and upwards, right. So, there is very difficult no guarantee of space utilization, so there is the poor space utilization, so there is no guarantee of proper space utilization.




(Refer Slide Time: 30:29)

Division of Space by Quadrees

- Each node of a *quadtree* is associated with a rectangular region of space; the top node is associated with the entire target space.
- Each non-leaf node divides its region into four equal sized quadrants
 - ❖ correspondingly each such node has four child nodes corresponding to the four quadrants and so on
- Leaf nodes have between zero and some fixed maximum number of points (set to 1 in example).





So, we partitioned in another thing called quad tree. So, what we see for from today is discussion is basically. First of all indexing is one of the major component for better query retrieval or better query or efficient query processing. We see that two types of there is a one aspects of spatial filling curve, primarily because this your spatial data sets are that basic spatial philosophy or those spatial data are what we say that nearby objects are likely to be more closer in value than distant object.

So, it is the search is always in a more nearer to the nearby objects, so that space filling curve may help us in keeping them in a particular cluster right. We are just checked or we have just seen that Z curve and a Hilbert or H curve which help in this case. Also if it is a we have seen this grid and this bucket digestion, but the; what we see that when we if we for this point feature is fine.

But if it is for polyline, or polygon where we have rectangular feature then the problem becomes more of there may be a issue of overlapping into the things like I cannot put in that specifically in one bucket type of things right. So, our traditional B tree or B plus

tree may not hold true in handling this we look for another type of another category of tree which is R tree or R star family.

And then also see that there the weight which allows overlapping where at the different branches or down the down the line tree, right. So, there is a variation of the R tree or R plus tree or will see some examples in our subsequent classes.

So with this, what we see that for efficient management of this data sets or query processing, we will be using that the standard way that how this our traditional data sets are handled may not be suitable for that what we require is more efficient structures. Like R tree, or R plus tree or Z space filling curve like Z curve or Hilbert curve and type of things right. So, will let us conclude our discussion today and we will continue our discussion on spatial indexing in one or two more classes.

Thank you.