Spatial Informatics

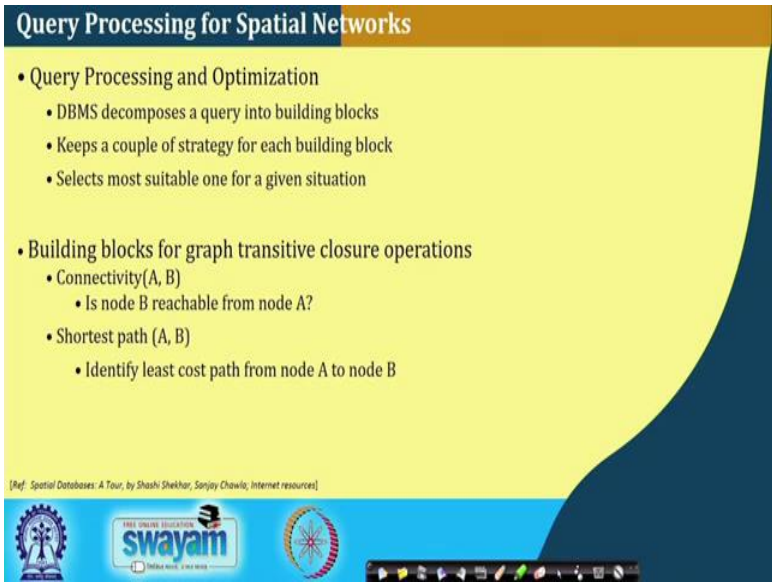
Prof. Soumya K. Ghosh
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 24
Spatial Networks – IV

Hello. So, we will continue our discussion on Spatial Networks one of the as we discuss is one of the major aspects of Spatial Informatics and one of the widely used in the spatial informatics application domains, right. So, like we are all used to finding a shortest path or finding a path between two source and destination and type of things right.

So, we are already discussing or different aspects of the things. Today we will see some more aspects of spatial networks, mostly which deals with spatial given a spatial network data; how to analyze the data and what are the specific building block for this type of applications and type of things right. So, those is our discuss focus today.

(Refer Slide Time: 01:10)



Query Processing for Spatial Networks

- Query Processing and Optimization
  - DBMS decomposes a query into building blocks
  - Keeps a couple of strategy for each building block
  - Selects most suitable one for a given situation
- Building blocks for graph transitive closure operations
  - Connectivity(A, B)
    - Is node B reachable from node A?
  - Shortest path (A, B)
    - Identify least cost path from node A to node B

[Ref: Spatial Databases: A Tour, by Shashi Shekhar, Sanjay Chawla; Internet resources]

So, one major aspects of spatial network is query processing and optimization right or anyway any spatial query, for that matter any database query; the one major aspect is a query processing and not only processing query optimization of the things. So, as if you recollect. So, what we mean by optimization, like if I have a query processing we have a query processing or query execution tree.

And I can have for the same query different sort of execution trees in the sense that I push the sometimes a select below the things, or non spatial operations before spatial operation etcetera. And based on that my overall cost may be dictated; that is because when we call whatever we are using at a much higher level like at the SQL, we are going select clause or some other like spatial joined or topological operation like overlap and so and so forth touch and type of things.

So, when we use that, those for those there are underlining different strategies which are being executed right. So, I use say spatial join operations. So, there can be two three algorithms for this spatial join operations. Now, out of this one, one will be picked up right. So, and every strategy, every algo has his own cost right. So, the algo which is picked up will be taken above the things, will be executed based on the cost.

Now, which algo need to be picked up also may depend on different type of parameters, like total like load on the database right, the cardinality of the database etcetera right. If the database size is pretty small I do not care that whether any of the algo could have been good all right. Or even spatial non spatial any of the functionality to execute maybe good; but if the database size is pretty large then I may want to execute the non spatial database, so the overall cardinally reduce and then etcetera.

So, these things are I need to have a query. So, here also it is true, like DBMS decomposition query into the building blocks; keeps a couple of strategies for each building blocks; selects more suitable for one for the given situations right that is the situation in the sense. So, that over based on the metadata.

Now, this is true for any database, for that matter this is more critical first any type of spatial database, because the data load is pretty high and we have seen that I O cost and compute costs are dictates together; it is sometimes difficult to put the whole data into the main memory, so there are lot of challenges into the things.

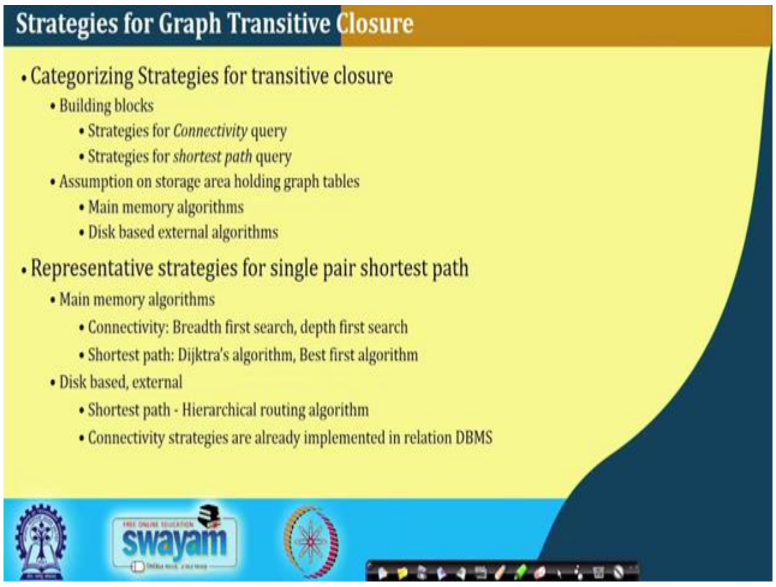
But again with the normals standard spatial database and with the spatial networks there are some difference, they are the here this basic building blocks may be different right. Here the there, here we have things like connectivity, finding a shortest path and type of these are the building blocks; which are not there in case of a say to polygon to polygon there may not be as such there is no meaning of shortest path unless you define something like that right or there is no meanings of like connectivity between these two polygon unless we define is as such there is no default type.

So, here the type of buildings blocks changes because of the behavior of the things right. Here the definition of proximity also changes right, as we have discussed because A and B if it is connected via a edge they are proximity rather than something which is not connected right. Because if you follow the spatial network or the physical proximity or the Euclidean distance maybe may not be the deciding point that which is near type of things right.

So, the way of proximity things also differ or here the range queries etcetera also make differences, because it may not be the within this bounding box things are not always true. And for that we have different little different kind of dealings for this; like here the building blocks as we see that for graph transitive closure operation is the connectivity right, one is the connectivity is node B reachable from A that is the thing right

So, I require it has a deep closure right A say A and B, A to C, C to D, D to E, E to B then A to B is reachable right. Shortest path A B identify the least cost path from the node A to node B right. So, there is the shortest path between two node A B in a thing; rather than these rather I should say that it is more delete we are finding more delete of a graph theoretic approach, which are not generally required for our standard or spatial databases.

(Refer Slide Time: 06:20)



Strategies for Graph Transitive Closure

- Categorizing Strategies for transitive closure
  - Building blocks
    - Strategies for *Connectivity* query
    - Strategies for *shortest path* query
  - Assumption on storage area holding graph tables
    - Main memory algorithms
    - Disk based external algorithms
- Representative strategies for single pair shortest path
  - Main memory algorithms
    - Connectivity: Breadth first search, depth first search
    - Shortest path: Dijktra's algorithm, Best first algorithm
  - Disk based, external
    - Shortest path - Hierarchical routing algorithm
    - Connectivity strategies are already implemented in relation DBMS

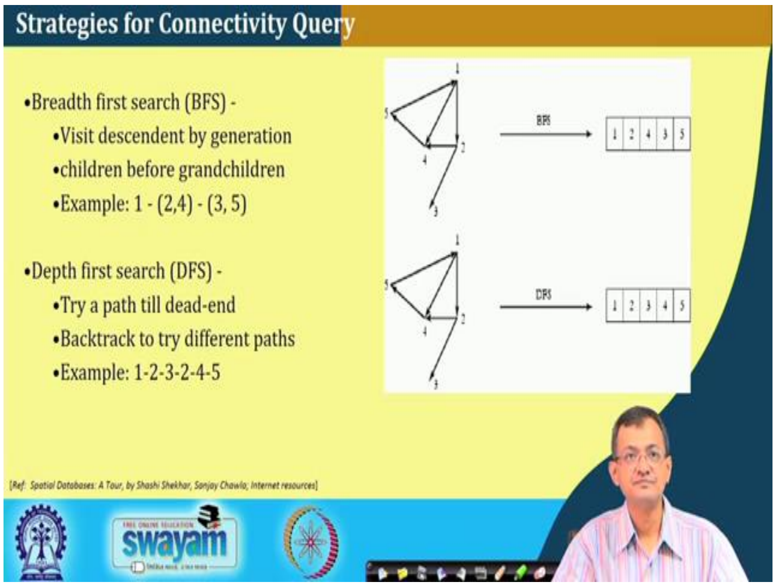
Now, we have different strategies for graph this transitive closure categorizing strategies for transitive closure, building blocks strategies for connectivity query, strategies for short shortest path query, assumption on the storage area holding the graph tables, main memory algorithms, disk based external algorithms right. So, these different type of algorithms can be there.

So, representative strategies for single pair of shortest path, if I want to have a one pair of shortest path, main memory algorithm. So, connectivity we know that BFS, DFS are very values and standard breadth first search and depth first search; whereas there are for the shortest path we have the this legendary Dijkstra's algorithm. And there is a variant where we use heuristic of finding a best first algorithm where we you some evaluation function finding the predicting the that cost of from V for a node V to the destination D and type of things right.

So, if you see these are the things which are standard for our graph theoretic operations also right. These are the things which we use in our graph theoretic operations, like and so if you have representative strategies for single pair shortest path, main memory algorithm connectivity and shortest path and this disk based and external shortest path is hierarchical routing algorithms right. So, you will see that instead of doing taking everything on the things I can do hierarchical, like I want to find out from say point in Kolkata to point in Bombay find out the shortest path between that is two right.

Now, keeping the whole network, loading the whole network into the main memory may not be feasible. So, we need to go for some sort of a hierarchical routing things; in some scenario it may not be the best solution or the optimal solution I may not be able to find out, but this some optimal solution is pretty good to work with right. So, connectivity there are other things, like connectivity strategies are already implemented in relational DBMS like there are several strategies which are already implemented in the like BFS DFS etcetera which are there in the standard DBMS.

(Refer Slide Time: 09:04)



Strategies for Connectivity Query

- Breadth first search (BFS) -
  - Visit descendent by generation
  - children before grandchildren
  - Example: 1 - (2,4) - (3, 5)
- Depth first search (DFS) -
  - Try a path till dead-end
  - Backtrack to try different paths
  - Example: 1-2-3-2-4-5

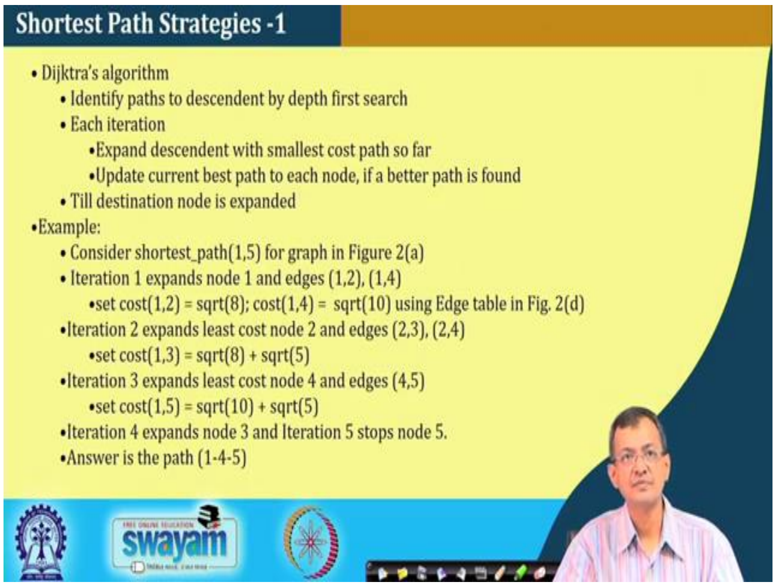
1
5
2
4
3
BFS
1 | 2 | 4 | 3 | 5
1
5
2
4
3
DFS
1 | 2 | 3 | 4 | 5

[Ref: Spatial Databases: A Tour, by Shashi Shekhar, Sanjay Chawla; Internet resources]

So, again coming back this breadth first search these are known just to quickly. So, visit descendent of the by generation and children before the grandchildren right. So, first children, then the grandchildren like in this case. So, 1 2 4 5 4, 1 2 4 3 5 right these are the things the way we do this BFS in case of a DFS try a path till the dead end and backtrack to find the different paths like 1 2 3 4 5 and this is the way we look at it right.

So, this is the standard DFS, BFS and the algorithms are there in any standard book or internet media you can also find these algorithms in this referred book what we are referring to, this if you are not knowing to that; but DFS, BFS all we have used widely.

(Refer Slide Time: 10:10)

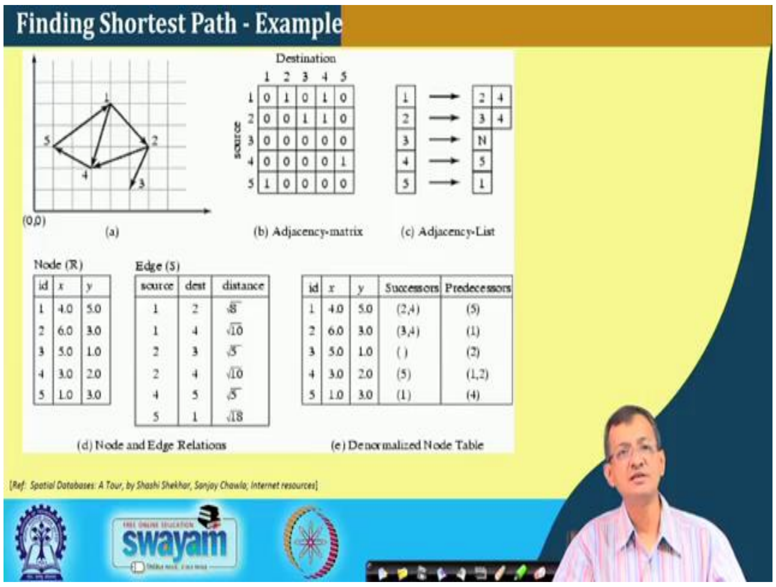


Shortest Path Strategies -1

- Dijktra's algorithm
  - Identify paths to descendent by depth first search
  - Each iteration
    - Expand descendent with smallest cost path so far
    - Update current best path to each node, if a better path is found
  - Till destination node is expanded
- Example:
  - Consider shortest_path(1,5) for graph in Figure 2(a)
  - Iteration 1 expands node 1 and edges (1,2), (1,4)
    - set cost(1,2) = sqrt(8); cost(1,4) = sqrt(10) using Edge table in Fig. 2(d)
  - Iteration 2 expands least cost node 2 and edges (2,3), (2,4)
    - set cost(1,3) = sqrt(8) + sqrt(5)
  - Iteration 3 expands least cost node 4 and edges (4,5)
    - set cost(1,5) = sqrt(10) + sqrt(5)
  - Iteration 4 expands node 3 and Iteration 5 stops node 5.
  - Answer is the path (1-4-5)

And Dijkstra’s algorithm that is a identify path to descendent by depth first search, each iteration expand the descendent with the smallest cost path so far; like so each level, we find out the smallest cost and go to the next level. Update the current best path to each node, if the better path is found right. So, we go on updating the things this is a standard, like if you considered shortest path 1, 5 for the graph that what we have seen here, then what we see expand it in 1, 2; 1, 4; 1, 2; 1, 4 then cost of 1, 2 is square root of 8; 1, 4 is square root of 10 and the edge table of 2 d will see here right.

(Refer Slide Time: 11:04)



Finding Shortest Path - Example

(a)

Destination

source	1	2	3	4	5
1	0	1	0	1	0
2	0	0	1	1	0
3	0	0	0	0	0
4	0	0	0	0	1
5	1	0	0	0	0

(b) Adjacency-matrix

```
1 → 2 4
2 → 3 4
3 → N
4 → 5
5 → 1
```

(c) Adjacency-List

Node (R)

id	x	y
1	4.0	5.0
2	6.0	3.0
3	5.0	1.0
4	3.0	2.0
5	1.0	3.0

Edge (S)

source	dest	distance
1	2	$\sqrt{8}$
1	4	$\sqrt{10}$
2	3	$\sqrt{5}$
2	4	$\sqrt{10}$
4	5	$\sqrt{5}$
5	1	$\sqrt{18}$

(d) Node and Edge Relations

id	x	y	Successors	Predecessors
1	4.0	5.0	(2,4)	(5)
2	6.0	3.0	(3,4)	(1)
3	5.0	1.0	( )	(2)
4	3.0	2.0	(5)	(1,2)
5	1.0	3.0	(1)	(4)

(e) Denormalized Node Table

[Ref: Spatial Databases: A Tour, by Shashi Shekhar, Sanjay Chawla; Internet resources]

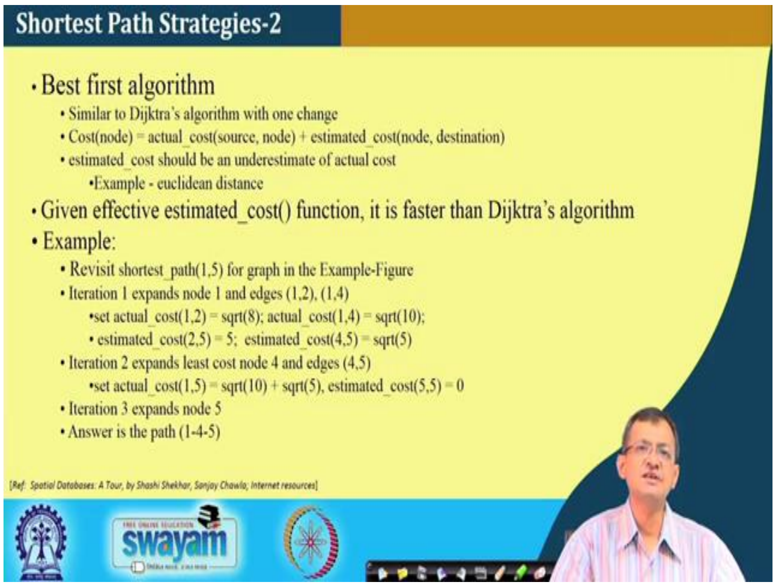
So, this is 1, 2 1, 4 sorry. So, we found select 1, 2 with 8 and 4 and we found at particular edge table and then set iteration 2 expands with the node 2 and edges 2, 3, 2, 4 similarly, 2, 3, 2, 4 and then set cost 2 4 to the edge 4, 5 and iteration 4 iterate to the 3 to the node 4, 5 and this.

So, what we do every step if you look at. So, initially 1, 2 and 1, 4 right and then finding out the cost 2 and 4 in the things, as from the 2, 3 and 4, 3 does not lead to anywhere 4 to 5 and 5 to 1. So, this is my adjacency list matrix and while calculating what we are having 1, 2, 4 root 8 10 for this and then using this table we calculate this consecutively shortage path right. So, every layer we are expanding and the shortest Dijkstra's says that the which is the node with the least cost will be again expanded and go on updating the table as and when we go.

So, that is a denorm form of the things, node table, like we have. So, this is the representation of the graph one is the adjacency matrix, another one is a list, finding the node coordinates and the edge relationship and from these I can calculate the different things right; like here how we calculated right 2, 2, 3, 4 and iteration 4 to edge 4, 5 and iteration 4 expands the node 3 to the 4, 5 and stops at 5, if we as we find.

So, my shortage path here it becomes 1, 4, 5 right, because of this it is the least cost right. So, there is a concept of I in order to finding the predecessor successors I can have a denormalized tables right. So, it may not be a normalized table and denormalized tables. So, that it if I have a denormal table the searching maybe higher, maybe faster in some of the cases.

(Refer Slide Time: 13:41)



Shortest Path Strategies-2

- Best first algorithm
  - Similar to Dijktra's algorithm with one change
  - Cost(node) = actual_cost(source, node) + estimated_cost(node, destination)
  - estimated_cost should be an underestimate of actual cost
    - Example - euclidean distance
- Given effective estimated_cost() function, it is faster than Dijktra's algorithm
- Example:
  - Revisit shortest_path(1,5) for graph in the Example-Figure
  - Iteration 1 expands node 1 and edges (1,2), (1,4)
    - set actual_cost(1,2) = sqrt(8); actual_cost(1,4) = sqrt(10);
    - estimated_cost(2,5) = 5; estimated_cost(4,5) = sqrt(5)
  - Iteration 2 expands least cost node 4 and edges (4,5)
    - set actual_cost(1,5) = sqrt(10) + sqrt(5), estimated_cost(5,5) = 0
  - Iteration 3 expands node 5
  - Answer is the path (1-4-5)

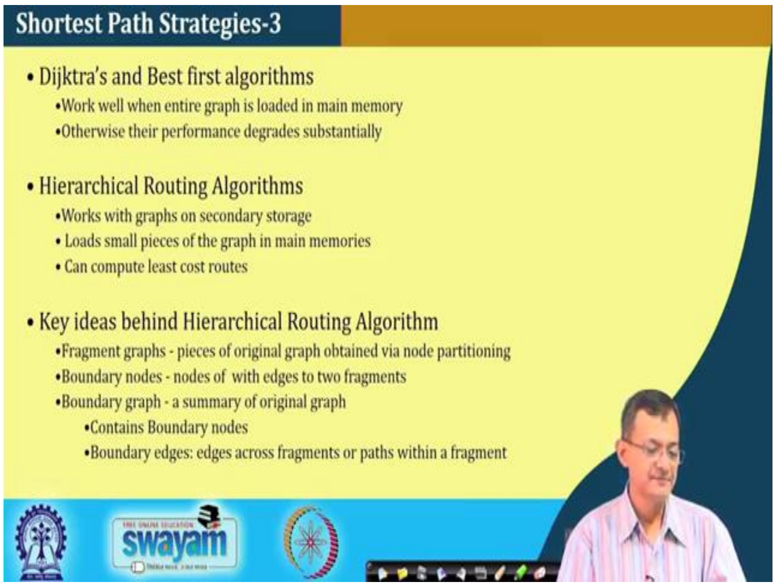
[Ref: Spatial Databases: A Tour, by Shashi Shekhar, Sanjay Chawla; Internet resources]

So, there is a variant of the Dijkstra's which is as best first algorithms, which uses heuristic, like given an effective estimated cost function it is faster than Dijkstra's algorithm to finding that what is the estimated cost function. If you revisit, the shortest path of the things iteration expand 1, 2, 4 square root and estimated cost 2 and 2 to 5 is 5 estimated cost to 4 to 5 is square root of 5, 2 expand the least cost 4 to the thing. So, there is a estimated cost at the looking at the front.

So, if we if that cost function is a strong function or it is a means the cost function is a good one, then you may have a faster convergence than Dijkstra's algorithm right. So, otherwise similar to Dijkstra's with change that cost node is actual cost between source and node plus estimated cost between node and destination. So, I do a look ahead type of things, what is the estimated cost in things.

Estimated cost should be and under estimate of the actual cost, so example Euclidean distance and type of things right. So, that I do a estimated cost of the things right. So, this is the variation of the things, again this is a standard strategy for any graph theoretic application and you can always consult any standard book or even this referred book.

(Refer Slide Time: 15:10)



Shortest Path Strategies-3

- Dijktra's and Best first algorithms
  - Work well when entire graph is loaded in main memory
  - Otherwise their performance degrades substantially
- Hierarchical Routing Algorithms
  - Works with graphs on secondary storage
  - Loads small pieces of the graph in main memories
  - Can compute least cost routes
- Key ideas behind Hierarchical Routing Algorithm
  - Fragment graphs - pieces of original graph obtained via node partitioning
  - Boundary nodes - nodes of with edges to two fragments
  - Boundary graph - a summary of original graph
    - Contains Boundary nodes
    - Boundary edges: edges across fragments or paths within a fragment

So, if we look at the shortest path strategies, so Dijkstra's and best first algorithms, like work well when the entire graph is loaded in the main memory right. So, if we these are pretty good where it the enter graph in the main memory, otherwise the performance degrades substantially. So, if the graph is not in the main memory. So, there is a degradation of the performance of these things. So, that is a challenge.

Hierarchical routing algorithms works with a graphs on the secondary storage, right. So, there is a another strategy which is the hierarchical routing algorithms. So, at the my number of size of the graph is pretty large. So, it in the secondary storage loads small pieces of the graph in the main memory, can compute least cost routes within that particular things; that means, I look at the things in a hierarchical fashion.

So, if you look at that basic idea behind these hierarchical routing algorithm. So, fragment the graphs, pieces of original graph obtained via node partitioning right. So, I want to do the fragmentation of the graph; boundary nodes, node of with edges of two fragments right. So, I have some of the things at the boundary nodes and boundary graph, the summary of the original graph. So, contains boundary nodes, boundary edges, edges across fragments or the paths within the fragments right.

So, just to reiterate; so, what we have seen that Dijkstra's and best first algorithms are good and widely used, but the one of the major constant is that they perform well when the whole thing in the main memory; but if you look at our typical spatial networks. So, this is a the sometimes the networks are pretty large.

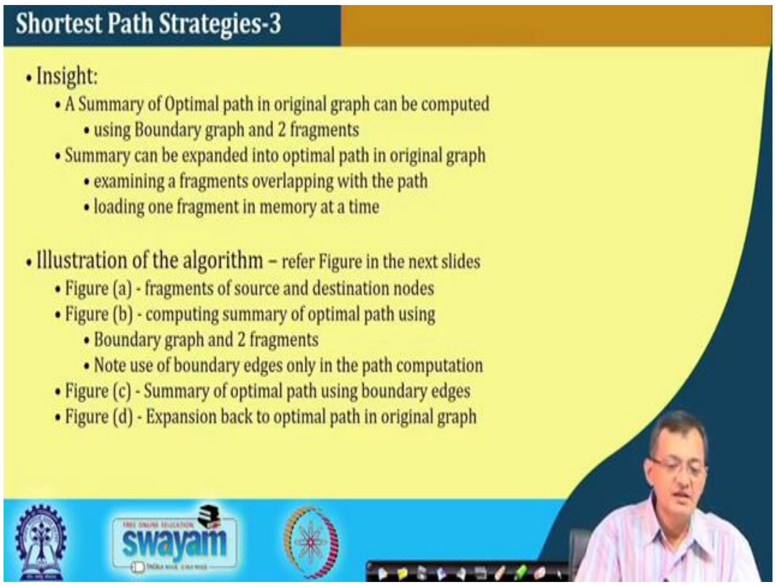
So, it is difficult to feed the whole thing into the main memory so; that means, you have to do lot of disc accesses right. So, it is in the secondary storage. So, the they in that cases the Dijkstra's and the best path best first algorithms performance pretty, performance degrades substantially when we have lot of disc accesses like that.

Now, that is why one of the another values algorithm is a hierarchical routing algorithm. So, works with graphs at the secondary storage, load small pieces of the graph in the main memory and can compute the least cost route on that graph, on that particular loaded graph and I can have some functionality of finding the whole route of the whole from source to destination.

So, basic idea, that the fragment the graph is the original graph via node partitioning. So, some sort of a node partitioning, we partition the graph. Boundary nodes; nodes with edges to two fragment. So, if I have some nodes which are on the edge of the two fragments both here and there, then I considered those are the boundary nodes. And boundary graph, the summary of the original graph is the boundary graph. So, instead of considering those say I want to find out the a point in Kolkata to a point in Mumbai right, shortest path.

Now, I consider this Kolkata as a particular boundary. So, whatever the road network or transport network is there within the Kolkata is as considered as a one sort of a boundary graph and Bombay may be. And then I find at the least cost between these two and then I expand within this Kolkata thing and Bombay thing to find out the shortest path right. So, contain boundary nodes boundary edges, edges across fragments or the paths within the fragments right. So, boundary edges are the, edges across the fragments and paths within the fragments which are there in the boundary edges.

(Refer Slide Time: 19:26)



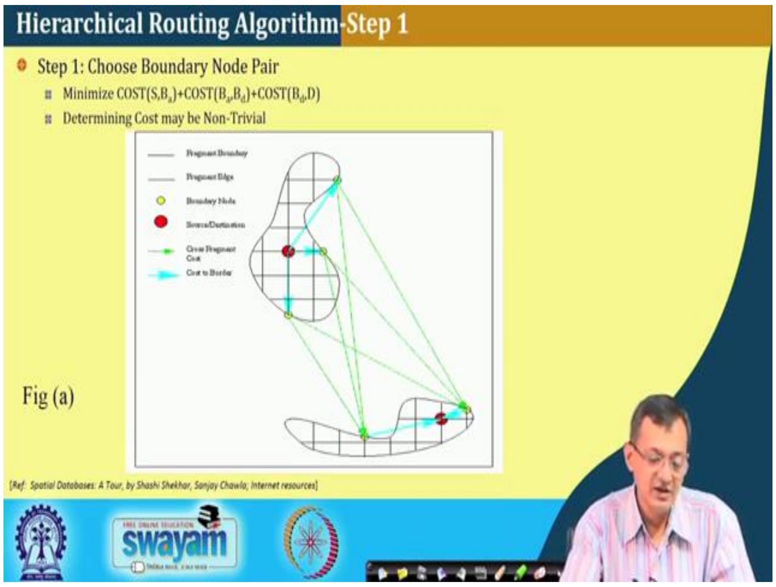
Shortest Path Strategies-3

- Insight:
  - A Summary of Optimal path in original graph can be computed
    - using Boundary graph and 2 fragments
  - Summary can be expanded into optimal path in original graph
    - examining a fragments overlapping with the path
    - loading one fragment in memory at a time
- Illustration of the algorithm – refer Figure in the next slides
  - Figure (a) - fragments of source and destination nodes
  - Figure (b) - computing summary of optimal path using
    - Boundary graph and 2 fragments
    - Note use of boundary edges only in the path computation
  - Figure (c) - Summary of optimal path using boundary edges
  - Figure (d) - Expansion back to optimal path in original graph

So, looking at that thus a summary of optimal path in original graph can be computed using boundary graph and the thing for the hierarchical. Summary can be expanded to the optimal path to the original graph, examining a fragment overlapping with the path right, loading one fragment in the memory at a time. So, I fragmented. So, loading at the at a time finding the this minimal or shortest path and then considering those things to calculate the actual shortest path.

So, what we see that, will do a small illustration also available in this particular referred book. Figure in some of the figures in the next slide, so figure a; fragment the source and destination nodes.

(Refer Slide Time: 20:12)



Hierarchical Routing Algorithm-Step 1

- Step 1: Choose Boundary Node Pair
  - Minimize $COST(S,B_a)+COST(B_a,B_d)+COST(B_d,D)$
  - Determining Cost may be Non-Trivial

Fig (a)

[Ref: Spatial Databases: A Tour, by Shashi Shekhar, Sanjay Chawla; Internet resources]

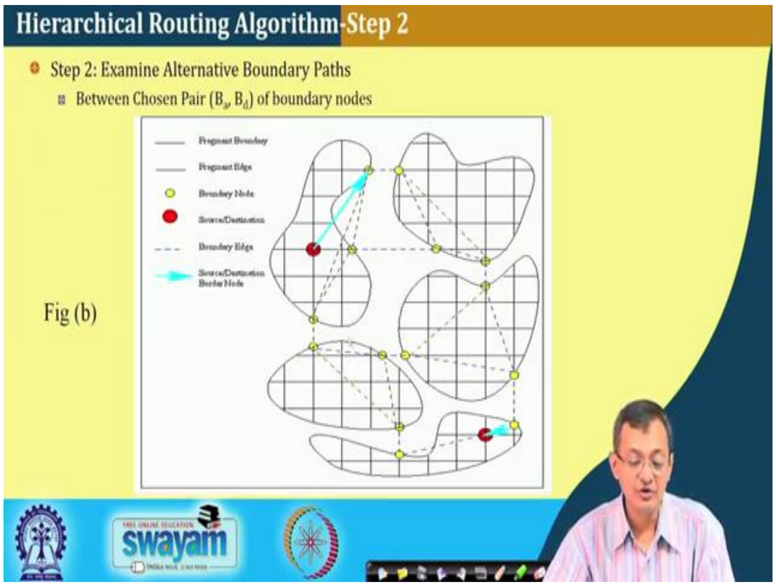
So, like here as we are discussing that. So, we have a one fragment a say and then another there right. So, figure b. So, in this case what we consider, you see there is a choose boundary node pair minimum cost S, B a to cost B a B d equal to cost plus cost of B D. So, they are two some of these boundary nodes, boundary nodes to reach this boundary nodes of this particular a. So, if this is a and say b and then from here to this destination. So, I may have this is my optimal minimum path, from there this is a minimum path and there we go to the destination.

So, this can be a one of the path. So, minimize the cost of S B a, the boundary of this particular set a, cost of this a boundary of a to boundary of b, cost of boundary of d to boundary of d determining the cost maybe a nontrivial right. So, this is the way we try to handle the thing. So, I fragment the graph into the things and it makes also obvious, like if you think of Kolkata and Mumbai right. So, there are some defined exit point from the Kolkata right, it is not like that anywhere you come out of the things right. So, also there are exit-entry point in Mumbai and Kolkata they are different.

Now, this network is again connected by national highways and type of things right. So, if we take the whole thing whereas, the complexity of the network within the city zone or this metro cities will be pretty complex right. So, and this at most of the times, this distances of this type of large distances dictates more than this is a shorter distance. So, if you look at logically also then I we can we take this Kolkata thing and find out this from destination what are the different, which is the minima, which has the different cost to go to this boundary nodes right. Here also what are the different cost go to the boundary nodes and then I connect the things right. So, this make lot of sense into the things right.

So, this is the figure a, fragment the source and destination nodes and then figure b, compute summary of the optimal path using boundary graph and two fragments; note use of boundary nodes only in the path computation right.

(Refer Slide Time: 22:47)



Hierarchical Routing Algorithm-Step 2

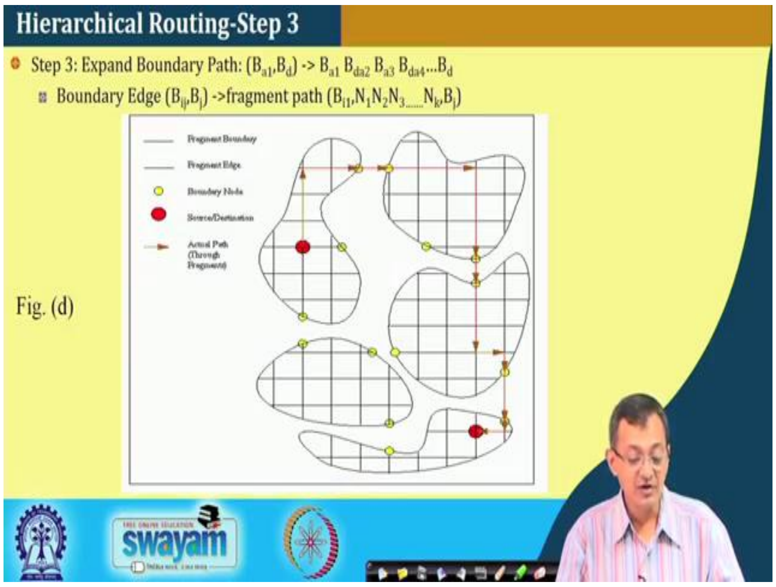
- Step 2: Examine Alternative Boundary Paths
  - Between Chosen Pair (B_s, B_d) of boundary nodes

Fig (b)

So, now, here we are not considering the inside nodes all the things right. So, first of all we have fragment, we have considering only these boundary nodes which is pretty finite and can be have less memory loading and find out this to find out this minimum path computation, right. So, I fragment this things and this boundary nodes used for the fragmentation.

Then what we do summary of the optimal path using boundary edges right. So, in the c, so we summarize these optimal path using this boundary. So, this to this so, but there is no path par se like this, I need to go somewhere like this right here also there is no Euclidean path from we have to jump into the things, we have to follow something right. Nevertheless I can find from the source to destination using this boundary nodes and this boundary paths right. So, summary of the optimal path using boundary edges and finally, expand that back to the optimal path in the original graph.

(Refer Slide Time: 23:52)



Hierarchical Routing-Step 3

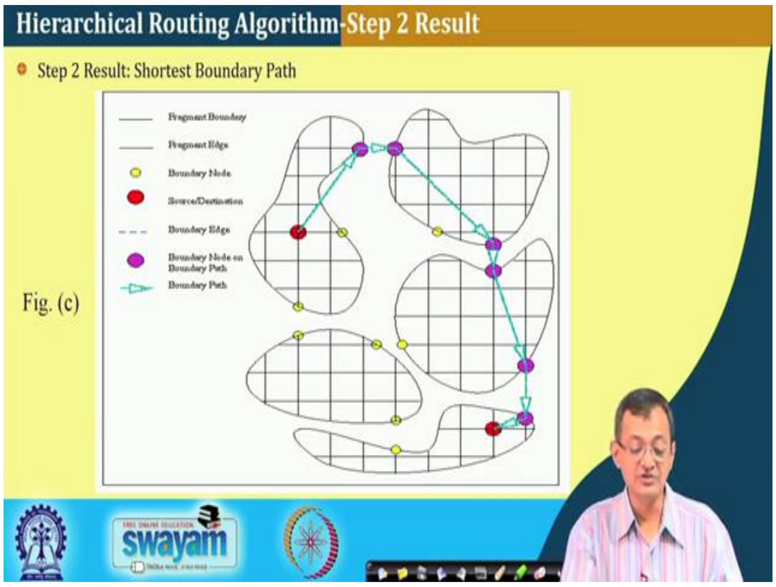
- Step 3: Expand Boundary Path: (B_{a1},B_d) -> $B_{a1}\ B_{da2}\ B_{a3}\ B_{da4}\dots B_d$
  - Boundary Edge (B_{ij},B_j) ->fragment path $(B_{i1},N_1N_2N_3\dots N_k,B_j)$

Fig. (d)

Now, the finally, what we do, then now we find out from here to here going to the what is the optimal path, shortest path into the things right; then finding the shortest path and shortest path and going to the things. So, this becomes my actual path between the source s and destination d right. This is this way, we can now handle a large graph or a which are maybe true for when we work with spatial graph of a large area, finding from here to Mumbai or here to Bangalore or Trivandrum etcetera it is a large graph. So, we have different fragments and type of things.

Just to again recap. So, initially we divide into different batches or the, what we say different smaller graphs. So, fragments, the source and destination smaller graphs; then this boundary nodes, based on these boundary nodes we try to calculate this shortest path right from the source and destination.

(Refer Slide Time: 25:01)



Hierarchical Routing Algorithm-Step 2 Result

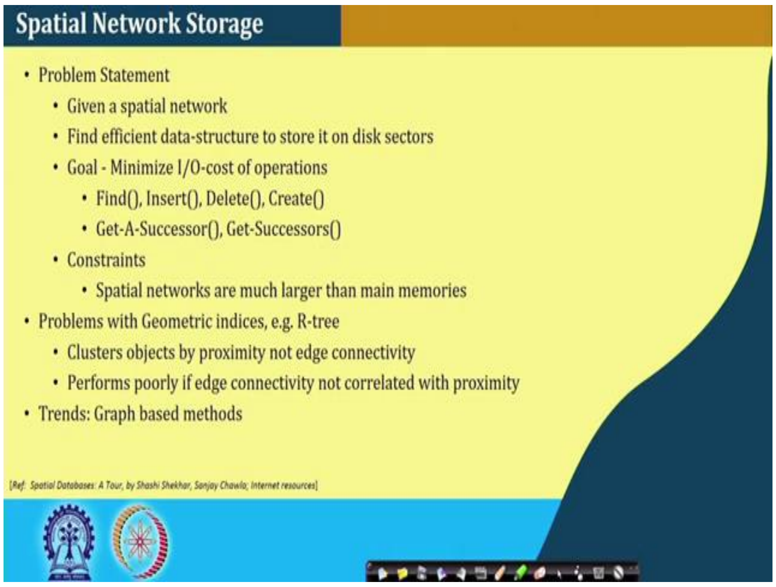
- Step 2 Result: Shortest Boundary Path

Fig. (c)

And then find out the shortest path based on the boundary nodes and the source and destination and then expand those into the graph all right. So, in doing so if you look at we oppose the problem in a hierarchical fashion right. Initially at the bottom in the hierarchy we have those number of graphs and this, otherwise the number of nodes will be pretty large; but here the number of nodes are much lesser and type of things right.

So, by this we can have a hierarchical strategy to find out the shortest path between a node in some network fragment is a to another fragment b and type of things b or d a something right, like as we giving that from Kolkata to Mumbai and doing.

(Refer Slide Time: 25:56)



Spatial Network Storage

- Problem Statement
  - Given a spatial network
  - Find efficient data-structure to store it on disk sectors
  - Goal - Minimize I/O-cost of operations
    - Find(), Insert(), Delete(), Create()
    - Get-A-Successor(), Get-Successors()
  - Constraints
    - Spatial networks are much larger than main memories
- Problems with Geometric indices, e.g. R-tree
  - Clusters objects by proximity not edge connectivity
  - Performs poorly if edge connectivity not correlated with proximity
- Trends: Graph based methods

[Ref: Spatial Databases: A Tour, by Shashi Shekhar, Sanjay Chawla; Internet resources]

So, now given set all these things, we need to look at that how do I store data right. Like how do I manage and store data. If you recollect we used to do in a something space filling curve when there are queries or the range queries, point queries and type of things; but here those things type of queries are less. Rather here that queries are find, insert, delete definitely get a successor of a node, get a predecessor of a node, get a set of successor, set of predecessor these are the things which are predominant.

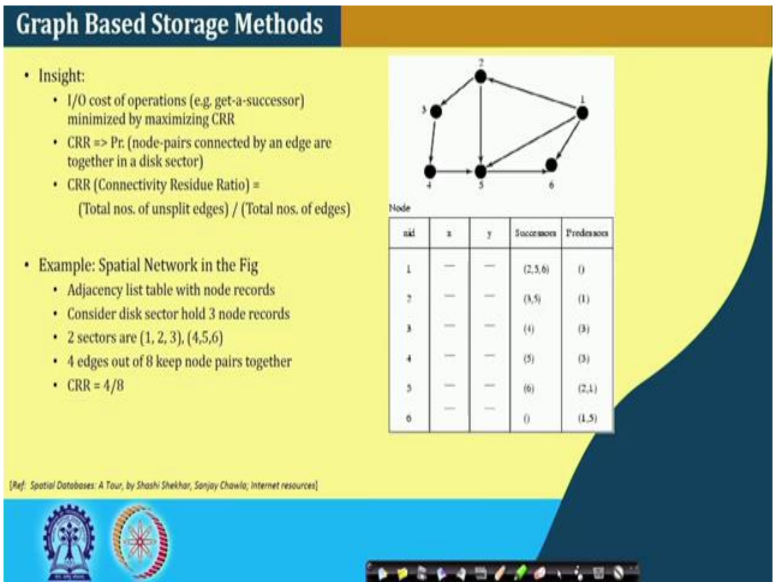
So, my, so rather I need the node and that connectivity of the node; if it is a directed graph, then I know need to know that from where to where it goes and type of things right. So, given a spatial network find the efficient data structure to store it on the disk sectors right. So, minimize I O cost of the operations. So, find, insert, delete, create these are the operations for finding, inserting data, deleting a removing data, or create something table and type of things.

And there are functions like get a successor, get set a get successors, or get predecessor, get predecessors and type of things right. So, there is a constraint that the spatial network are much larger in many cases than the main memory. In other sense I cannot store the data affront on the main memory and work on the things. So, that is one of the major challenge out here.

So, problem with geography geometric indices like R tree n type of things; the cluster objects by proximity not by edge connectivity right. So, it is by proximity if not the edge connectivity. So, it is more of a how code, uncode Euclidean distance by it is proximity, but the edge if I do the road network or the edge connectivity maybe much larger. So, our typically R tree type of structure looks at that proximity rather than connectivity.

Performs poorly if the edge connectivity is not correlated with the proximity. So, if the edge connectivity something and the proximity something, then it is perform poorly right. And so, we the trend is to or what we try to use is to look at graph based methods or graph theory application of graph theory or graph theoretic approaches and doing so.

(Refer Slide Time: 28:38)



Graph Based Storage Methods

- Insight:
  - I/O cost of operations (e.g. get-a-successor) minimized by maximizing CRR
  - CRR => Pr. (node-pairs connected by an edge are together in a disk sector)
  - CRR (Connectivity Residue Ratio) = (Total nos. of unsplit edges) / (Total nos. of edges)
- Example: Spatial Network in the Fig
  - Adjacency list table with node records
  - Consider disk sector hold 3 node records
  - 2 sectors are (1, 2, 3), (4,5,6)
  - 4 edges out of 8 keep node pairs together
  - CRR = 4/8

Node

nid	x	y	Successors	Predessors
1	—	—	(2,5,6)	()
2	—	—	(3,5)	(1)
3	—	—	(4)	(3)
4	—	—	(5)	(3)
5	—	—	(6)	(2,1)
6	—	—	()	(1,5)

[Ref: Spatial Databases: A Tour, by Shashi Shekhar, Sanjay Chawla; Internet resources]

So, inside I O cost of operation that is get a successor, minimized by maximizing CRR right. I O cost of operations get a successor that is minimized by CRR, what is CRR? So, CRR which is connectivity residue ratio right, which is basically a probability of the node pairs connected by an edge are together in a disk sector right.

So, our everything is based on edges. So, what is the chance that if I take a something on the form a disk sector what is the chance that the node pair are can be retrieved on a one call right. So, that is a thing or in other sense connectivity ratio we define the total number of unsplit edges, total number of edges right, if we this is the ratio of the things.

Like in this example, suppose I have a graph like this right 1, 2, 3, 4, 5, 6 and type of things right; now this adjacency list table with node records. So, this is the adjacency list tables with node records. So, 1 is connected to 2, 5, 6; 2 is connected to 3 and 5; 3 is connected to 4; 4 is connected to 5; 5 to 6 and 6 to no one right. So, this is the successor and predecessor is accordingly on the other side like 1, 3; for 1 there is no predecessor; for 2 the predecessor is 1; for 3 is the predecessor for 3 the predecessor is 2 it is 2 may be this type over this is not very clear.

So, for 3 the predecessor is 2; for 4 the predecessor is 3; for 5 the predecessor is 4 sorry 5 is one is 4 one is 2 another is 4 ah, for particular 5 actually for 5 the predecessor is 1, 2, 4 right these are the three predecessor. Again there is a missing thing here and for 6 the predecessor is 1 and 5 right. So, there is few type of. So, we this should be 2 this should be 1, 2, 4 will correct it.

Now, a consider a disk holding three nodes records; say let us considering a particular disk is holding three node records. So, it is let us consider the selection as 1, 2, 3 and 4, 5 6. So, if it is 1, 2, 3 see that what how many unsplit edges you can take. So, total number of edges 1, 2, 3, 4, 5, 6, 7, 8, so total number of edges is 8, if it is 1, 2, 3 in the one disk. So, what are the unsplit edges. So, 1 to 2 2 is 1; 2 to 3 is 2. So, 2 unsplit edges if 1, 2, 3 for 4, 5, 6 on the other disk unsplit edges are 4 to 5 and 5 to 6 another two.

So, 2 plus 2, 4, right. So, 4 divided by 8 is my connectivity residual ratio right or CRR right. So, ideally CRR if it is had it been all are in a single go, then it would have been much easier to do the things right. So, if instead of that if I have the CRR, if I have kept 1, 2 say 1, 2, 5 into one disk and 3, 4, 6 in other disk right if there is arrangement. So, 1 to 2 will be 1; 2 to 5 will be 1; 2 to 5 will be in the 1 edge and 1 to 5 will be another edge. So, 3, count is 3 and as it is 2, 3, 5 6, so 3 to 4 one, then there is another is 2, 3, 4 6, so 3 to 1, so that we do not gain anything out here.

But I can have, what we mean to say choosing this node appropriately I can have this different values of the CRR. In other sense this dictates that how I that gives me a possibility that how I store the data into the storage appropriately. So, that I get a better means better performance in terms of the I O retriever.

So, in today’s discuss what we tried to look at is that what are the different building blocks, what are the different strategies for the things and also at the aspects of what we has thing that connectivity CRR, connectivity residue ratio by what will be the back falling back to the things. We also looked into the hierarchical approach where the spatial networks are pretty large and then how to handle are all those things right.

So, with this let us conclude our discussion today. We will little bit look into this again this CRR calculation and associated thing, so related to disk in our subsequent lecture ok.

Thank you.