Electronic Design Automation

Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

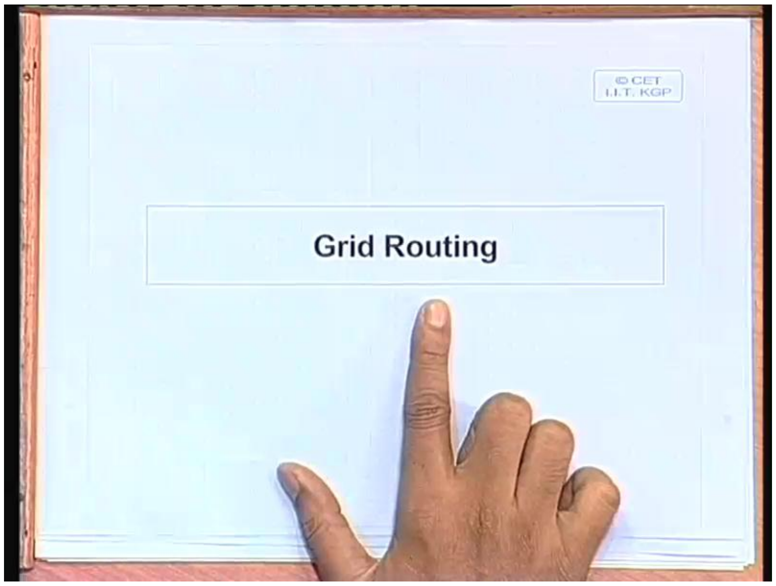
Lecture No #22
Backend Design Part-VIII

So after placement comes, the process of routing. So from today onwards we shall be looking at some of the different techniques and algorithms that are used for solving specifically the routing problem. But before going into the algorithm as such let us first try to understand what this problem of routing really involves well as you can understand by its name routing means interconnecting the different signal pins of the components on the surface of silicon. Well serves to interconnect that means with respect to net list we will have to connect them.

Now the problem is not as simple as it looks like. Because, you try to understand on the surface of the chip there may be thousands and thousands of such modules. There is no guarantee that one module will be connecting only to modules which I needs neighborhood. It may be possible that 2 modules which are far apart they also need to be connected. Say exactly how will the path or the routing path be decided or be found out. So instead of a treating the problem as one and trying to solve it in one go, it is normally tackled in a hierarchical fashion.

There is a process called global routing where we look at the global picture and try to find out a rough strategy using which we will be carrying out the routing. Then there is a process called detailed routing where based on the strategy we have just decided we will try to complete the detailed interconnecting means interconnection paths will be will be exactly finalizing the exact places where the routes or the wires have to be laid out. Now before going into those algorithms we look at a generic routing problem which is called grid routing. First look at some algorithms for grid routing. Then later we will find out that exactly where this grid routing fits in the overall picture of the routing problem.

(Refer Slide Time: 03:24)

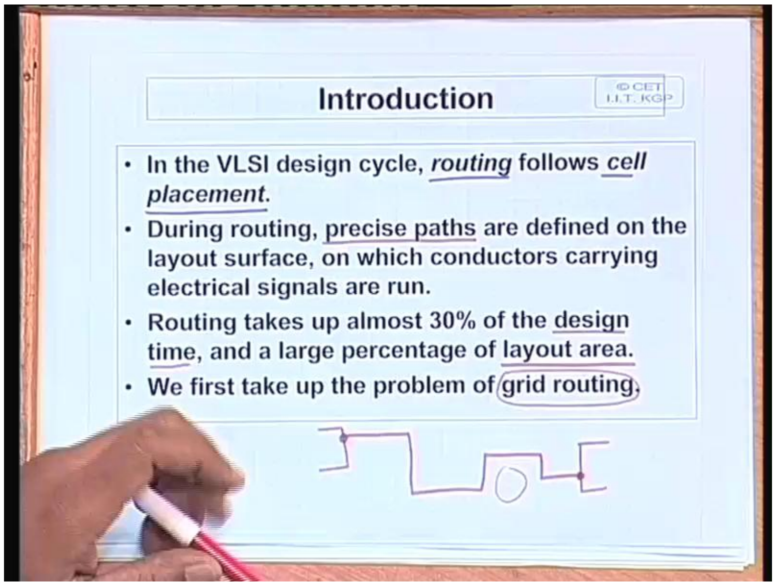


© CET
I.I.T. KGP

Grid Routing

So we will start today by having some discussion on the so called grid routing. Now as the name implies you can easily understand that we are talking about routing in some kind of a grid structure ok. So let us try to see what grid routing really is.

(Refer Slide Time: 03:44)



Introduction

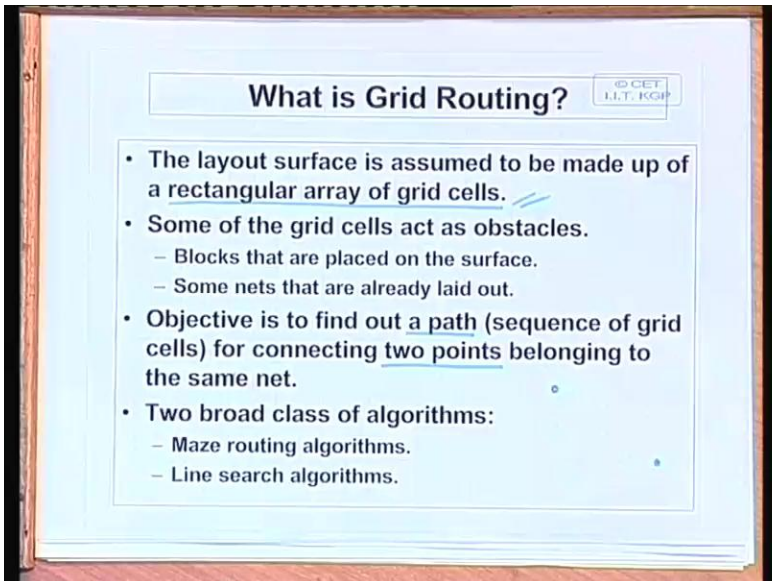
© CET
I.I.T. KGP

- In the VLSI design cycle, ***routing*** follows ***cell placement***.
- During routing, precise paths are defined on the layout surface, on which conductors carrying electrical signals are run.
- Routing takes up almost 30% of the design time, and a large percentage of layout area.
- We first take up the problem of grid routing.

But first let us look at the general problem once more. The process of routing will obviously follow the process of cell placement in the VLSI design cycle. Now when you say that we are trying to carry out routing we are actually trying to find out the exact or the precise paths through which the conductors will be carrying the signal from one point to the other. For example if this is one pin of a block this is another pin of a block well we may have to route the signal like this if there are obstacles in between. This may denote the routing path.

Well if there are some obstacles in between you cannot have a straight Manhattan path minimum distance path. Now this problem of routing is important because not only it takes up a substantial percentage of the total design time it also takes up a substantial percentage of the layout area. Well this percentage can be as high as fifty percent. If your placement is not good, so as I said we look at a sub problem called grid routing to start with then we would be looking at the different sub problems and going into the detail slowly.

(Refer Slide Time: 05:09)



What is Grid Routing?

© CET I.I.T. KGP

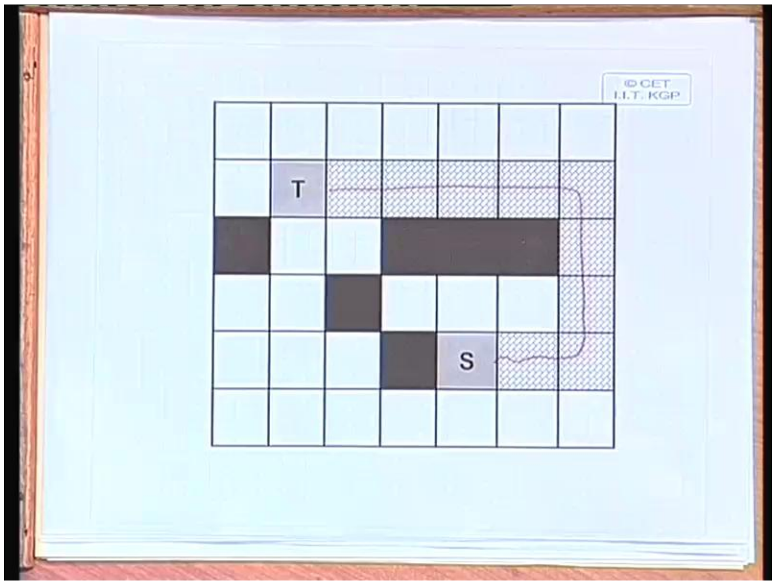
- The layout surface is assumed to be made up of a rectangular array of grid cells.
- Some of the grid cells act as obstacles.
  - Blocks that are placed on the surface.
  - Some nets that are already laid out.
- Objective is to find out a path (sequence of grid cells) for connecting two points belonging to the same net.
- Two broad class of algorithms:
  - Maze routing algorithms.
  - Line search algorithms.

Now in grid routing the problem is simple. Here the layout surface where we are trying to carry out the routing or the interconnection that is assume to be made up of a rectangular array of grid cells. This is the assumption in case of grid routing. So whatever be our surface we divide or split the surface into imaginary rows and columns those are our grids ok. Now in that imaginary array of grid cells some of the grid cells can act as obstacles because all of them are not free for routing.

Now obstacles can be of 2 types there can be some blocks which are the circuit cells. So circuit cells can be treated as obstacles on the surface of the grid. And some of the interconnecting nets which you have just laid out that will also be acting as an obstacle for the future. Laying out of the nets so some nets which are already laid out they will also be acting as obstacles. Now in grid routing our objective is to find out a path connecting 2 points one is obviously belonging to the same net. So one characteristic of grid routing is that we consider 2 points at a time.

The basic grid routing algorithms they consider the sub problem of connecting 2 points one at a time. This kind of sub problem is taken one at a time we have only 2 points we have to connect them. Broadly grid routing algorithms can be classified as either maze routing or line search algorithms we will be looking into these in detail. Ok first whatever we are saying that the array of grid cells some blocks or the nets can act as obstacles I have a diagram to show you exactly what I mean.

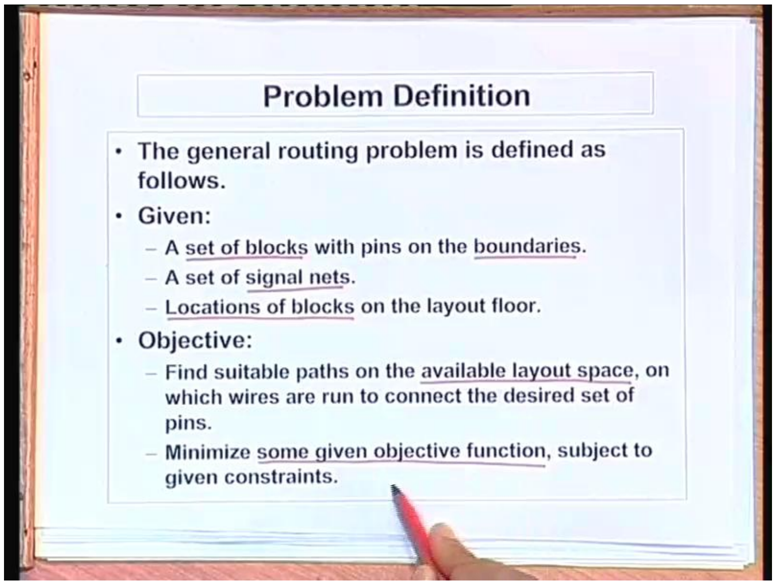
(Refer Slide Time: 07:15)



This is just a diagram for the sake of illustration you see here the total floor area we have divided up into some rows and columns. The black regions these are the obstacles this may be already pre paced pre placed cells or they may be interconnecting wires. This is the 2 these are the tow points you want to connect these sources and the target T and in this hatched area I am showing one possible path. This is one possible path through this grid structure. So when I pose this sub problem that given a set of obstacles placed on this area of grid given a source cell source grid cell and a destination or a target grid cell to find a path. Actually we will have to find out a sequence of empty grid cells which will lead us from S to T.

Now obviously this is not the only path there are other paths also this may be better or this may be shorter length also possibly. So there can be multiple paths through it. So the problem of grid routing attempts to find out one path. Well we will want a the path as good as possible the possible minimum cost. There can be other nets also which will be acting as obstacles. The order in which you lay out the nets that also becomes very important because once you layout net number i, that will act as obstacle for net number i plus one. Ok so the problem definition is well in case we are basically talking about the grid routing for the time being.

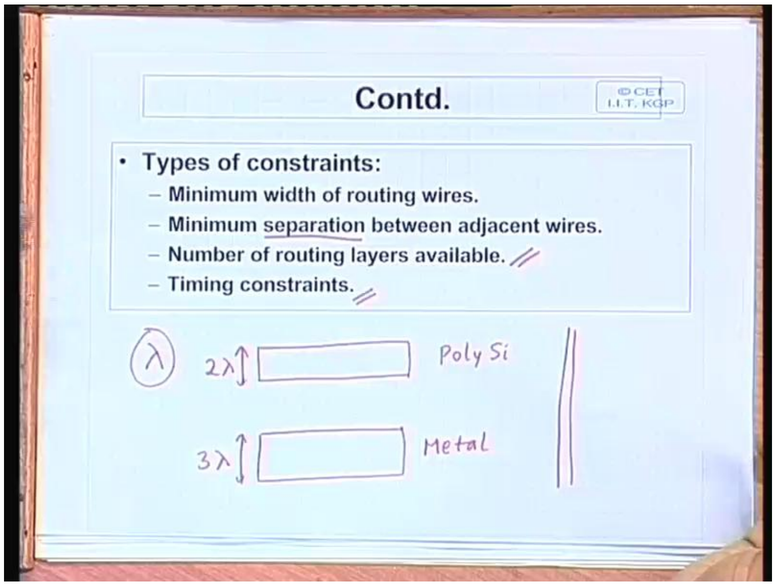
(Refer Slide Time: 09:13)


Problem Definition

- The general routing problem is defined as follows.
- Given:
  - A set of blocks with pins on the boundaries.
  - A set of signal nets.
  - Locations of blocks on the layout floor.
- Objective:
  - Find suitable paths on the available layout space, on which wires are run to connect the desired set of pins.
  - Minimize some given objective function, subject to given constraints.

We have a set of blocks with pins on the boundaries of the blocks, a set of signal nets and the precise location of the blocks which have been fixed during placement right. Our objective is to find suitable paths on the available layout area space. So the space which are already taken up by the blocks they are not available for the interconnection obviously and we will have to minimize some objective function while we are trying to carry out the interconnection. Now this objective function or the constraint can be of different types well we basically need to take care of several kinds of constraints when we are trying to complete the routing.

(Refer Slide Time: 10:04)



Contd.

- Types of constraints:
  - Minimum width of routing wires.
  - Minimum separation between adjacent wires.
  - Number of routing layers available.
  - Timing constraints.

©CET I.I.T. KGP
λ
2λ
Poly Si
3λ
Metal

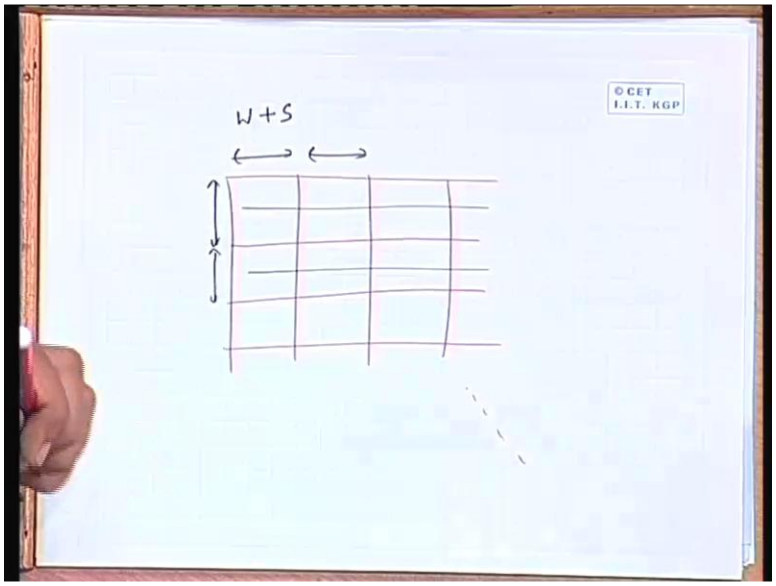
Some of the important constraints are like this minimum width of routing wires. Well if you look at the VLSI design layout level. There you may be knowing that signals may be carried on several different layers of course metal layer is an option. But for shorter distances you can even carry signals over poly silicon or diffusion layers. Now to the VLSI design some of well this process of laying out the basic cells on the surface of silicon and interconnecting them. This has been simplified by framing or formulating a set of geometrical design rooms.

The idea is that if your layout follows those geometrical design rules then your circuit will work correctly. These are some kind of rule of thumb like I am giving some examples well all measures in terms of the layout are computed in terms of a basic unit called lambda. Lambda is synonymous to the channel size. Typically the channel length is equal to twice lambda. Now some of the design rule says that if it is a for example poly silicon line which is carrying a signal then the minimum width of the poly silicon line has to be twice lambda. If it is a metal line if you are carrying signal over a metal line then the width has to be thrice lambda.

Now these figures have come up from the accuracy with which you can do fabrication at the different levels from experience not only at the with width rules given also some separation rules are given like if there are 2 poly silicon lines which are running side by side what will be the minimum separation between them. If a polysilicon and a metal line are running side by side of course on 2 different layers, what should be the minimum separation between them. So all this design rules have been framed up. Now you have a set of say geometrical rules and any layout will have to confirm to those rules so means when you are doing interconnection you must keep those in mind.

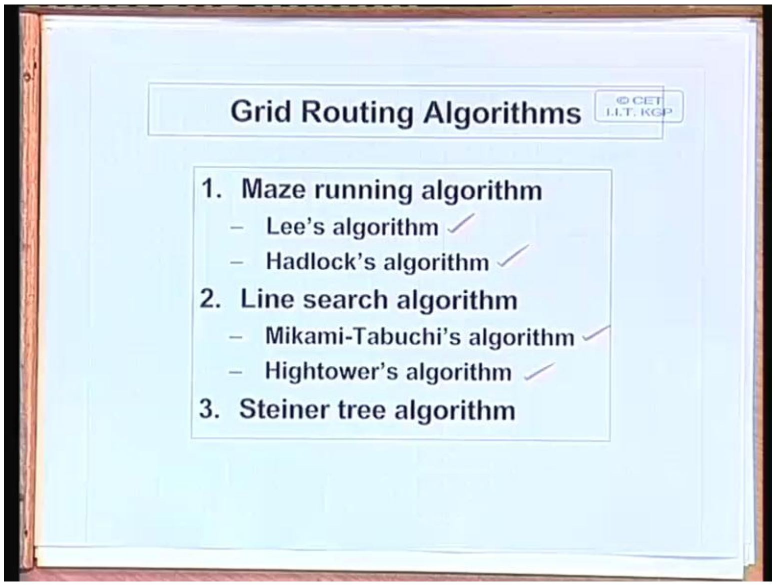
So, you cannot run a very thin interconnecting line which is not which is not confirming to the design rule. Metal the minimum width of the line should be thrice lambda for example. And of course other constraints are there how many routing layers are available to you this will be given as input and if there are any timing constraint the critical nets have to be given priority. Actually twice lambda is the channel length lambda is half the channel. Ok well the reason I am just mention this constraint is that well when we are talking about this grid routing.

(Refer Slide Time: 13:25)



So we are saying that the layout surface is divided up into rows and columns right. So it will be a big grid. Now the idea is that the grids are the sizes of the grids are chosen in such a way that if required I can layout 2 different wires on 2 adjacent rules. This means that my width, what the height whatever you call, of this grid cells must be at least equal to the minimum width plus the minimum separation of the lines. So that 2 adjacent rows or columns can carry 2 different signal lines belonging to 2 different nets. This is the way we select the size of the grid cells ok. So now let us come to the grid routing algorithms we had said those maze and line search algorithms are the 2 broad categories.

(Refer Slide Time: 14:28)



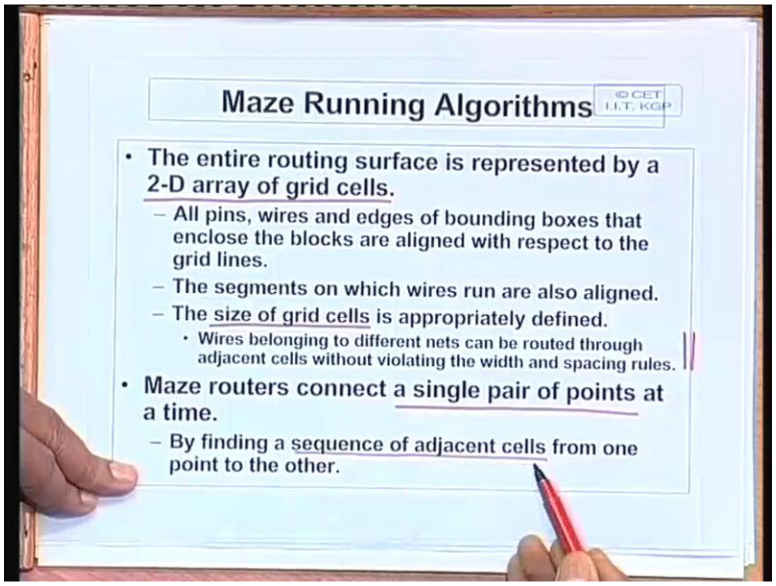
Grid Routing Algorithms

© CET I.I.T. KGP

1. Maze running algorithm
   - Lee's algorithm ✓
   - Hadlock's algorithm ✓
2. Line search algorithm
   - Mikami-Tabuchi's algorithm ✓
   - Hightower's algorithm ✓
3. Steiner tree algorithm

Well of course Steiner tree kinds of algorithms also exist but due to its complexity people very rarely use it. Maze running and line search algorithms are most popular. So we would be looking at some of the important representatives of these algorithms algorithm classes. Lee's, hadlocks, mikami tabuchi and high tower's algorithms. So we start with maze running algorithm. Well maze running algorithm you must have sort some maze kind of a puzzles where you enter the maze from one point you have to come out of the other point we will have to find a path through it. So the idea is something like this. Well we will have like here also you have a source you have a target you have some obstacles in between you will have to find out a path some kind of a search. So the idea behind maze routing algorithm is exactly that.

(Refer Slide Time: 15:27)



Maze Running Algorithms

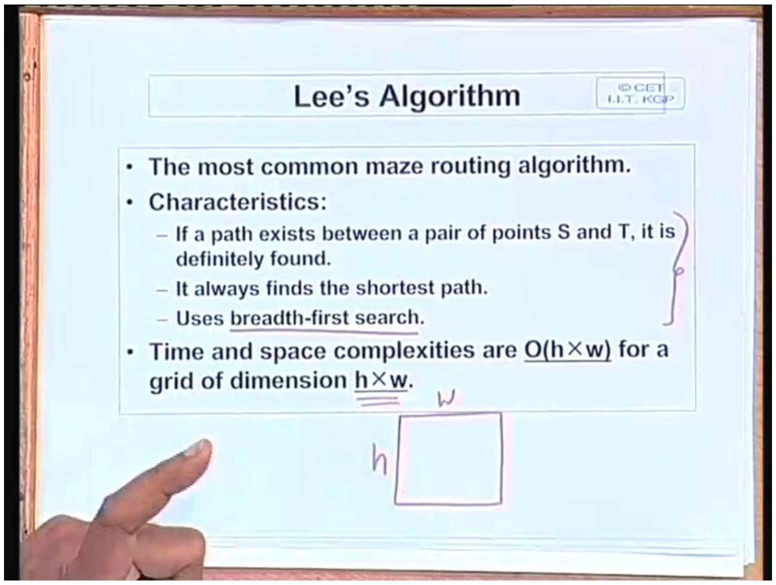
- The entire routing surface is represented by a 2-D array of grid cells.
  - All pins, wires and edges of bounding boxes that enclose the blocks are aligned with respect to the grid lines.
  - The segments on which wires run are also aligned.
  - The size of grid cells is appropriately defined.
    - Wires belonging to different nets can be routed through adjacent cells without violating the width and spacing rules.
- Maze routers connect a single pair of points at a time.
  - By finding a sequence of adjacent cells from one point to the other.

So will for the grid routing the entire routing surface we represent by a 2 dimensional array of grid cells right. Now when you say that we represent it by an 2 D array of grid cells this means that all the pins of the components they must be align to the boundaries of the grids all the wires. They must be laid out with respect to the grid boundaries and the edges of the bounding box that represent the blocks they also must follow on the boundaries, which means that that size of the grid will act as some kind of a basic unit. So everything will be aligned with respect to the grids.

Ok, so a block cannot go half way into a grid cell it will either occupy a grid cell only or it will not occupy this is the idea. The segments on which the wires are also aligned everything is aligned like to say. And as I just mentioned that the size of the grid cells are also appropriately defined so that wires belonging to different nets can be placed or routed through adjacent cells without evaluating the minimum width and separation constraints. So accordingly we fix up the size of the grid cells.

Now one characteristic of maze routers is that just has I have had again I had just mentioned before also that it connects a single pair of points at a time it does not look at the global routing problem and tries to solve it takes one small sub problem at a time. Given 2 points it tries to connect it. So I just said if it is represented as a 2 D array. So actually we have to find out a sequence of adjacent cells which will lead from the source to the target this is my objective. So first we look at the most popular algorithm in this category the lee's algorithm. So lee's algorithm was one of the first algorithms which were proposed it is a classical algorithm in that sense.

(Refer Slide Time: 17:41)



Lee's Algorithm

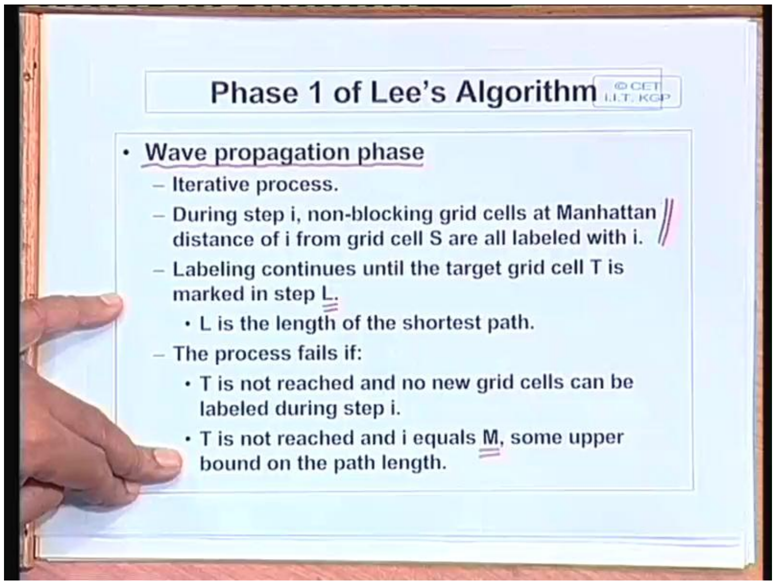
© CET I.I.T. KGP

- The most common maze routing algorithm.
- Characteristics:
  - If a path exists between a pair of points S and T, it is definitely found.
  - It always finds the shortest path.
  - Uses breadth-first search.
- Time and space complexities are $O(h \times w)$ for a grid of dimension $h \times w$.

Now some of the characteristic of lee's algorithms are well first thing is that it is based on a breadth first search strategy. And this method is optimal in the sense that firstly if a path exists between sources and target it will always be found. Secondly it will always find the shortest possible path. So in this sense lee's algorithm is optimal. So the reason it is optimal it is not very difficult to understand once we explain the algorithm you need to see because it tries to explore all possible paths in parallel. The first path which touches or hits the target it is the, what which is chosen.

So since you are exploring all possible paths in parallel so if a path exist it will be found out and the first path to hit will be the minimum cost path ok fine. Now the characteristic of lee's algorithm is that if your grid size is h by w then the time complexity of this algorithm is order h into w and also the amount of memory requirement this space complexity is also h by w ok. So we will see how because the basic data structure you are using is a 2 dimension array of size h by w. That is why the space complexity will be h by w. Yes. Number of rows and columns. Now a lee's algorithm actually works in 3 different phases. First let us look at the phases and then we will take an example to work it out.

(Refer Slide Time: 19:35)



Phase 1 of Lee's Algorithm

© CET I.I.T. KGP

- Wave propagation phase
  - Iterative process.
  - During step i, non-blocking grid cells at Manhattan distance of i from grid cell S are all labeled with i.
  - Labeling continues until the target grid cell T is marked in step L.
    - L is the length of the shortest path.
  - The process fails if:
    - T is not reached and no new grid cells can be labeled during step i.
    - T is not reached and i equals M, some upper bound on the path length.

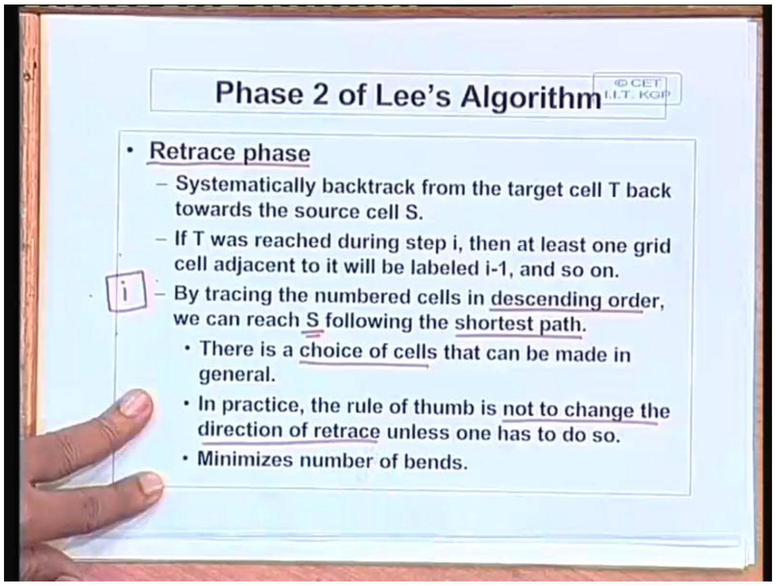
The first phase of the algorithm performs something called wave propagation. Well this is called wave propagation because starting from the source as if we are emitting or starting to emit some kind of an wave front in all possible directions. And with each step the wave front advances one cell at a time. So just through all possible available path the wave front will try to expand in all directions. The idea is that sometime this wave front will be hitting the target. Ok so let us see what this wave propagation phase you consist of. Well this is an iterative process because I told you it in each step it is advancing one step that is why it is iterative process.

Well during the wave propagation phase what we do we label the empty cells in the grid array by some numbers one 2 3 4. The number indicates during step number I we will be putting the number I during step one you will be labeling some of the cells with one during step two. We will be labeling some of the cells as 2 and so on. So during step I will be labeling them as I some of the cells. During step I all non-blocking grid cells at a distance of I from the source they will be labeled with I this is how the labeling is carried out. So all possible cells from the source which are at a distance of I subject to the obstacles of course they will be labeled with the number i.

This is how the labeling is carried out. This labeling will continue until the target cell T is reached. Say in step number L well if T is reached in cell number I it means that L is the length of the shortest path right. The process will fail if either of these 2 things happen. Well you see that beyond the point you cannot label any cell any further and you have not yet reached T so no path exists to T. T is not reached and no new grid cells can be labeled during some particular step i. Or secondly well if the user also specify some upper bound to the path length say M you see that even after iteration number M you do not find a path to T then also you stop it failure.

So under these 2 conditions you don not find a path. So the first step of lee's algorithm is weight propagation where by starting from the source you are systematically going on labeling the grid cells one 2 3 4 sequentially with numbers till you hit the target ok. Yes (( )) (22:50) Some kind of actually without obstacles it will look like a diamond expanding it will look like a diamond shaped thing. (( )) (23:02) I will take, I will take an example if you meet an obstacle you do not level the obstacle cells you try to find out the other cells which are free I will show an example. Ok the first phase is weight propagation well once you hit the target the next step is to systematically trace back from the target and find out one possible path.

(Refer Slide Time: 23:28)



Phase 2 of Lee's Algorithm

- Retrace phase
  - Systematically backtrack from the target cell T back towards the source cell S.
  - If T was reached during step i, then at least one grid cell adjacent to it will be labeled i-1, and so on.
  - By tracing the numbered cells in descending order, we can reach S following the shortest path.
    - There is a choice of cells that can be made in general.
    - In practice, the rule of thumb is not to change the direction of retrace unless one has to do so.
    - Minimizes number of bends.

There may be multiple paths. So you try to do a retrace and find out one of the feasible paths because all the paths that will be getting re tracing from the target well if the target is labeled as L so all the paths will have a length of L. The retrace phase for le lee's algorithm is very simple. Well you systematically back track from T back towards S. Well the idea is that if the target cell T was reached during step I what that means that, the target cell T was labeled with some number I. Some of the neighboring cells must have been labeled with i minus 1.

So you select one of the neighboring cells which are labeled I minus one. Similarly from i minus one you select one of its neighboring cells which is labeled I minus 2 and so on. Finally you will be reaching some cell which will be labeled as one that one is adjacent to S you have reached S ok. Because the way you are carrying out the labeling given a cell with a label I we will always find a neighbor with label I minus 1. Because it was from that neighbor only you came to this cell. (( )) (24:48) Yes, yes, yes. DFS. Yeah. No during retrace we are doing DFS. He is right.

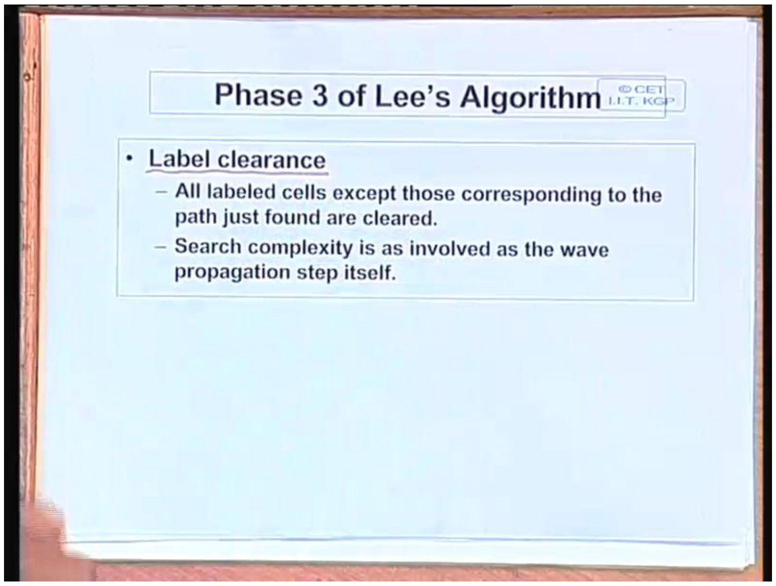
Because during back track we are interested in selecting only one path we are we are not interested to look at all paths. Yes. So by tracing this numbered cells in descending order we can reach S and any path you follow that will be the shortest path. Because starting with the number

L you are going to L minus one L minus 2 back to one all paths will have a length of L. But some of the paths will be more desirable than the other so there is a rule of thumb while you are back tracking that you do not change direction until it is necessary unless it is necessary because zigzag wiring is not very desirable while interconnecting.

Well if you can find a straight line path where numbers are descending try to follow that straight line. So in general there will be a choice of cells which you can follow during back trace. So you try not to change the direction of retrace this is a heuristic. This will minimize the number of bends in the wires. Minimize number of bends means your reliability will increase because more the number of bends. There is a chance of the junction getting thinner. (( )) (26:14) Yes (( )) (26:16) Right now we are not talking about lee we are assume there are single layer routing. No, no.

Here in lee's algorithm we are treating we are assuming there is a single layer there is no concept of multiple layers right now. There is a single grid array through which we have to carry out all the routing multi-layer thing will be coming to later yes. (( )) (26:46) Forward trace I will work out an example that means you will get an exam you will get your answer well. Now after retrace you will be getting one path but during the forward trace you have made the array dirty you have labeled so many cells with the numbers. So before you start connecting another pair of points you will have to clear those numbers then start it again.

(Refer Slide Time: 27:11)

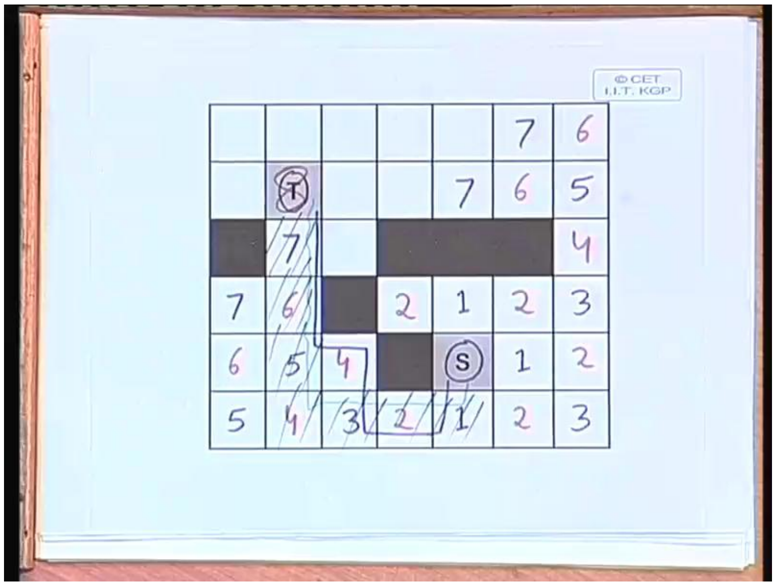


Phase 3 of Lee's Algorithm

- Label clearance
  - All labeled cells except those corresponding to the path just found are cleared.
  - Search complexity is as involved as the wave propagation step itself.

So the third step is the so called label clearance step just other than the path you have just found out the other labels which you are temporarily put in they will have to be cleared. So all label cells except those corresponding to the path just found they will have to be cleared or initialized to some value. This also is an invert process. The search complexity is almost as high as the forward propagation wave propagation phase. So these are the 3 steps of the lee's algorithm. So let us have an example and work out so if you have any if you have doubts it well get cleared. Ok let us take this same example.

(Refer Slide Time: 27:57)



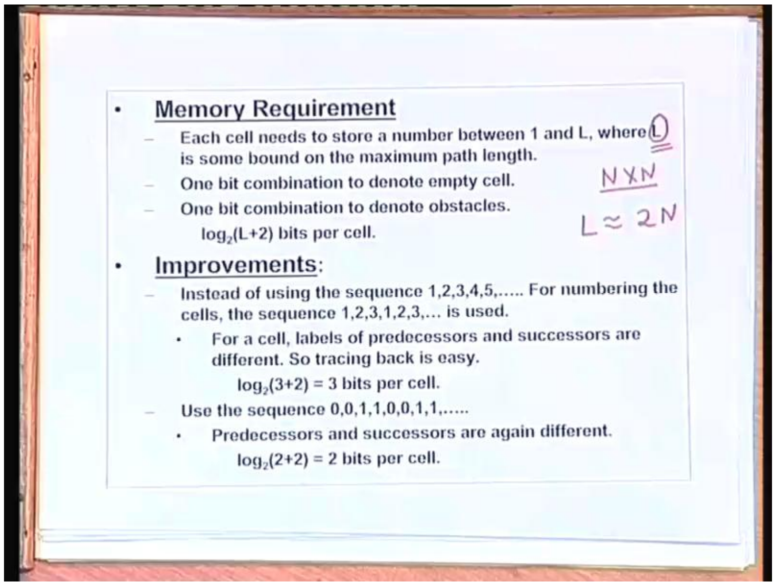
These are the obstacle cells this is our source this is the target. So we start with the source the neighboring cells of S other than the obstacles will be labeled with one so this will be one this will be one this will be one these are the 3 neighbors. Next step all the neighbors of ones will be labeled as two's so the, you will have to maintain some kind of priority queue through which you will be doing this breadth first search. So this will be 2 this will be 2 2 2 and 2. Then the neighbors of twos will be labeled as three's this will be three's this 2 has obstacles on al all side it cannot expand any further 3 3 then 4.

So you come 4 4 4 then 5 5 5 5 then 6 6 6 6 and 6 then 7 7 7 7 7 then while trying to label eight we will heat up on the target. So target is at a distance of eight from the source. So from this you can follow since there are a number of choices you have 8 7 6 5 4 3 2 1. That means you can follow a path like this if you want. If you want you can follow a path like this also because this is also a feasible path. 7 6 5 4 3 2 1 S. But as I said it is better to minimize the bend so this will be a more desirable path. So this is the basic idea behind lee's algorithm. This is how you carry out the labeling and after this labeling is done and this path has been fixed out you mark this entire path as an obstacle for the next step this entire path will be marked as an obstacle now. (( ))(30:25) Yes (( )) (30:27)

But after 4 you will have to go to 3 you are always moving to a lower number 5 is not a choice. (( )) (30:39) Available Yeah. Yeah from I you will be moving to I minus one always. If a neighbor is labeled by some other number that is not a candidate for visiting you do not go that that connection ok. So this is the basic concept behind lee's algorithm it is very simple to implement but of course you can see that this is very complex in terms of the memory requirements and the number of cells you expand or label.

Now let us just look into the complexity of this method. First with respect to memory, then with respect to the computation steps you require. Well first you look at the memory requirements. So of course well in terms of the rows and columns. Well if it is an n by n, array of course you require an n by n because you call n by n array to store it. But each element of the array how many bits you require to represent. That is one issue. So you just look at the memory requirement critically.

(Refer Slide Time: 31:54)



- **Memory Requirement**
  - Each cell needs to store a number between 1 and L, where $\underline{L}$ is some bound on the maximum path length.
  - One bit combination to denote empty cell.
  - One bit combination to denote obstacles.

    $\log_2(L+2)$ bits per cell.
- **Improvements**:
  - Instead of using the sequence 1,2,3,4,5,..... For numbering the cells, the sequence 1,2,3,1,2,3,... is used.
    - For a cell, labels of predecessors and successors are different. So tracing back is easy.

      $\log_2(3+2) = 3$ bits per cell.
  - Use the sequence 0,0,1,1,0,0,1,1,.....
    - Predecessors and successors are again different.

      $\log_2(2+2) = 2$ bits per cell.

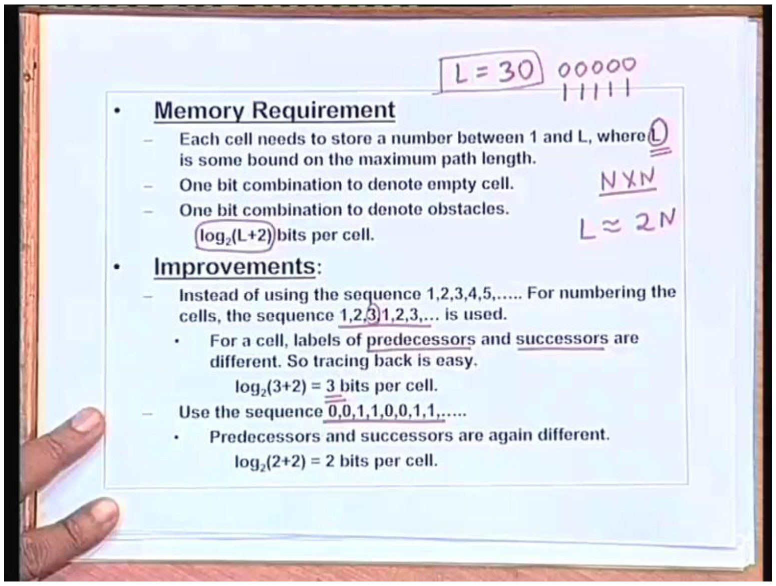
$\underline{N \times N}$

$L \approx 2N$

Now suppose the user has specified L as a bound to the maximum path length. Well if the user has not specified anything then if it is an n by n array then in the worst case L will be of the order of twice N. Well if the 2 cells having the 2 diagonally opposite corners the path will be equal to twice N minus one that length. But user can specify some other value less than that also because I want paths only with a maximum length of this. So if this L is given by the user then the maximum label you can put on a cell is L.

So the valid labels are between one and L. Now in addition you also have to indicate the empty cells also have to indicate obstacles so you require 2 other bit combinations. So actually L plus 2 different things you have to store. Suppose I have certain number of bits there will be a one bit combination which will be indicating empty cell may be 0 there will be another bit combination which will be indicating blocked or obstacles and other bit combinations can be used for labeling. So actually we need log L plus 2 so many bits per cell actually ceiling of that. Yes (( )) (33:28) to an (( )) (33:30)

(Refer Slide Time: 33:34)



$\boxed{L = 30}$ 00000 11111

- **Memory Requirement**
  - Each cell needs to store a number between 1 and L, where $\underline{\underline{L}}$ is some bound on the maximum path length.
  - One bit combination to denote empty cell.
  - One bit combination to denote obstacles.

    $\boxed{\log_2(L+2)}$ bits per cell.

$\underline{N \times N}$

$L \approx 2N$

- **Improvements**:
  - Instead of using the sequence 1,2,3,4,5,..... For numbering the cells, the sequence $\underline{1,2,3,1,2,3,...}$ is used.
    - For a cell, labels of $\underline{\text{predecessors}}$ and $\underline{\text{successors}}$ are different. So tracing back is easy.

      $\log_2(3+2) = \underline{\underline{3}}$ bits per cell.
  - Use the sequence $\underline{0,0,1,1,0,0,1,1,}$.....
    - Predecessors and successors are again different.

      $\log_2(2+2) = 2$ bits per cell.

Zigzag whatever you do suppose you have your source on this site and target on this site there your Manhattan distance can never be less than 2 N minus one whatever we follow. Diagonally you cannot go you cannot go diagonally (( )) (34:00) Yeah. More than that. Yeah. If you if in a path you are moving away from that. It can be more than that. But this is (( )) (34:10) They the way you put it. Ok there is not necessary to. Suppose if someone says that you need only a this L is 30, say then you can say that well the well I use only 5 bits the 0 0 0 0 0 combination I will be using to indicate empty cell 1 1 1 1 1 combination.

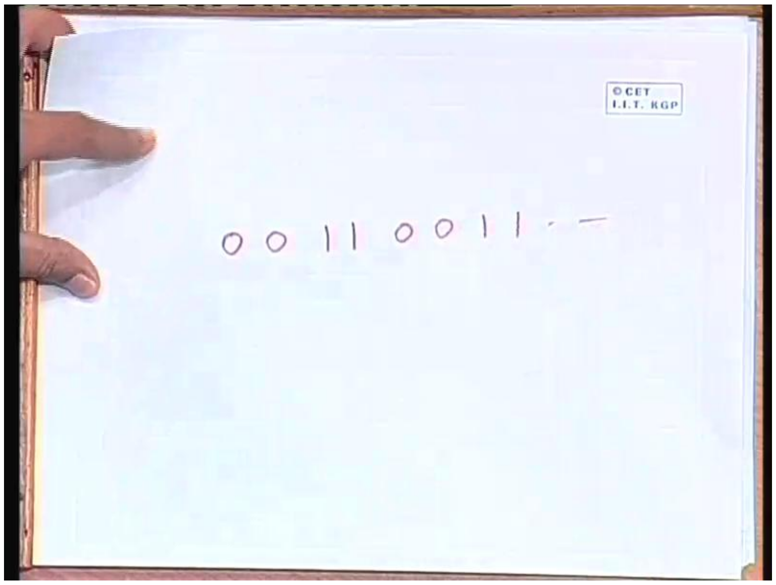
I will be using to indicate obstacles and the other numbers between means 1 and 29 that will be using for that. So it actually L plus 2 log 2 of that how many bits for that. Now with respect to memory requirement some interesting improvements have been suggested. See just if you look at the way lee's algorithm works well we have label the number 1 2 3 4 why because during retrace there is no ambiguity where to return back. For each particular cell which has been labeled the successor and the predecessor they have 2 different label numbers because if I have I the successor will be I plus one the predecessor will be I minus one. So during back trace I exactly know where to go right.

But suppose instead of this 1 2 3 4 5 this kind of numbering well I use an alternate numbering how? 1 2 3 1 2 3 1 2 3 in that order I repeat. 1 2 3 again I use 1 2 3 again 1 2 3. See if I use a numbering scheme like this, suppose I am currently labeling the cells with this number 3 then I know with respect to 3 my successor will be labeled as one my predecessor will be labeled as two. So at each level my successors and predecessors are distinct. So as long as I can distinguish between the predecessor and successor it is fine. So I can reduce the number of bits required per cell if I use a labeling scheme like this 1 2 3 1 2 3 1 2 3. So I can very uniquely distinguish between the predecessor and successor and here I will be requiring 3 bits per cell because log 3 plus 2 ceiling of that will be 3 bits. (( )) (36:44) That is a number you are putting.

Suppose you have been you are labeling a cell with a number 3 then you search its neighbors. Some of the cell some of the neighbors will be labeled as one some of them will be labeled as two. You know 2 is the predecessor and one is the successor. You will retrace you will always move from 3 to 2, 2 to 1, 1 to 3, 3 to again 2, 2 in that way. (( )) (37:15) Yes (( )) (37:17) Ok (( )) (37:18) In breadth first search for a particular node you can land up from 2 different, yes you can. (( )) (37:34) It might be different from 2 different paths but see I am telling you what will be the reason.

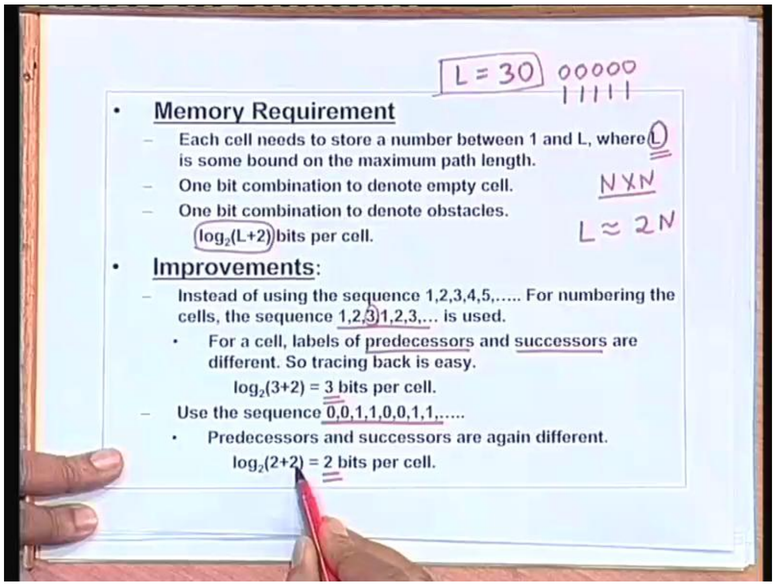
So mean your question is that the way you are leveling the label of a particular cell might be different for 2 different paths but the shorter path I have already labeled first. Once I have labeled that cell is not a candidate for re labeling. (( )) (38:07) Which that sort of thing (( )) (38:10) See, no, no, no. It can land on a target but not via that cell it can have some other path and land into the target from there. Ok so this, another interesting improvement of lee's algorithm that you do not use one 2 3 just use 2 numbers 0s and ones and use a labeling scheme like this 0 0 1 1 0 0 1 1 0 0 1 1 in this order. See in this sequence you again observe.

(Refer Slide Time: 38:55)



Say 0 0 1 1 0 0 1 1 in this order it goes on you take any number say you take the second 0 the predecessor is 0 successor is one they are different you take this one. Predecessor is 0 successor is one they are different you take the second one predecessor is one successor is 0. So for each particular step the predecessor and successors are different. But you will have to remember that which 0 or which one you are in you are in is it the first one or the second one that you will automatically remember during back trace or forward trace you will be remembering that anyway. But the way we are labeling the predecessor and the successors they are always different one of them is 0 other is one. So during back trace there is no ambiguity.

(Refer Slide Time: 39:47)



L = 30 00000 11111

- **Memory Requirement**
  - Each cell needs to store a number between 1 and L, where L is some bound on the maximum path length.
  - One bit combination to denote empty cell.
  - One bit combination to denote obstacles.
    - $\log_2(L+2)$ bits per cell.

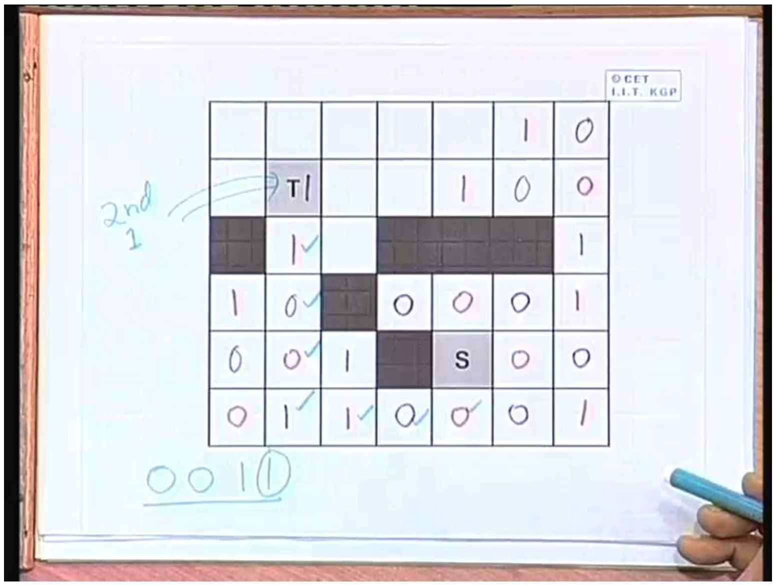
N X N

$L \approx 2N$

- **Improvements**:
  - Instead of using the sequence 1,2,3,4,5,..... For numbering the cells, the sequence 1,2,3,1,2,3,... is used.
    - For a cell, labels of predecessors and successors are different. So tracing back is easy.
      - $\log_2(3+2) = 3$ bits per cell.
  - Use the sequence 0,0,1,1,0,0,1,1,.....
    - Predecessors and successors are again different.
      - $\log_2(2+2) = 2$ bits per cell.

So if you do this the number of bits becomes only 2 per cell log 2 plus two. Ok just I am taken an example to illustrate this process with the same example right.

(Refer Slide Time: 40:04)



From the source you start with 0s. The red color indicates the first 0 ok then again another 0 I am indicating with the different color. Then one the first one then the second one. (( )) (40:31) That this you can keep by just by using a single flag whether it is order even during forward trace you just remember that nothing else whether it is the first one or the second one you are labeling. Similarly 0 0 0 0 this is the first 0 then the second 0 then the first one then the second one. So finally you will see that you land up the target with the second one.

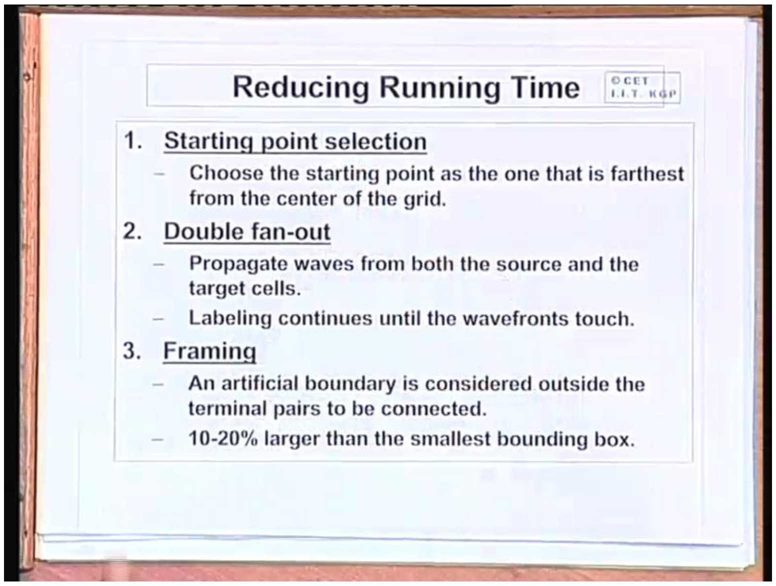
So now during back tracking you will be remembering this that you have reach T with the second one. Which means your predecessor will be one with respect to 0 0 1 1. This subsequence you have landed up with this 1. So the predecessor will be a one. So from here search for a one in the neighbor you come here from here you search for a 0 come here from here you search for another 0 come here then come to a one then one then 0 then finally you reach S. So just if you remember whether is the first one or the second one back tracing is easy right.

So these are improvement of lee's algorithm you can have with very little modified. (()) (41:57) Previous one (( )) (41:59) Yeah. This example one ok yes. (( )) (42:08) This one (( )) (42:15) Yes (( )) (42:21) Yes but yeah, yeah. (( )) (42:29). No there is not be the first one you are labeling see there may be another path see means finally this S can also label it following some other path round this round this round this round this. But this is the shorter path which has reach the cell first ok after reaching it has already labeled with some number and any other wave front coming from some other direction will not disturb it any further.

So here what I am saying is that the shorter wave front whenever it hits you just store that with you. Now if anyone else hits you later we ignore that that way I will always keep track of the shortest path of every cell from the source ok fine. There are a few other heuristics which are used to improve the running time of lee's algorithm this is with respect to memory. But the number of cells you have to label that can go very large you can easily understand.

So there are some heuristics. What I am saying is that whatever we have talked so for that is with respect to the memory requirement. But with respect to running time the execution time will depend on how many cells you have to labeled during the breadth first search scheme. So if you can reduce that by some means then then the so the execution time will be faster. So there are several heuristics which have been proposed to reduce the running time.

(Refer Slide Time: 44:10)



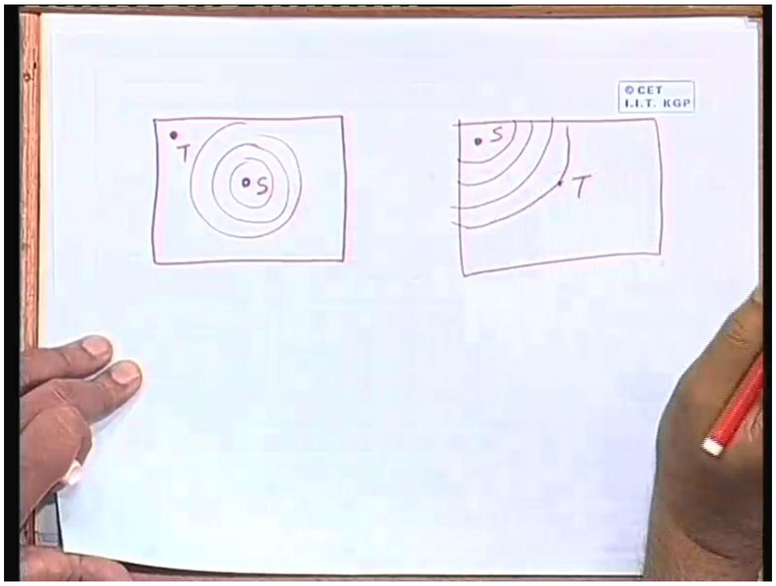
Reducing Running Time

©CET I.I.T. KGP

1. Starting point selection
   - Choose the starting point as the one that is farthest from the center of the grid.
2. Double fan-out
   - Propagate waves from both the source and the target cells.
   - Labeling continues until the wavefronts touch.
3. Framing
   - An artificial boundary is considered outside the terminal pairs to be connected.
   - 10-20% larger than the smallest bounding box.

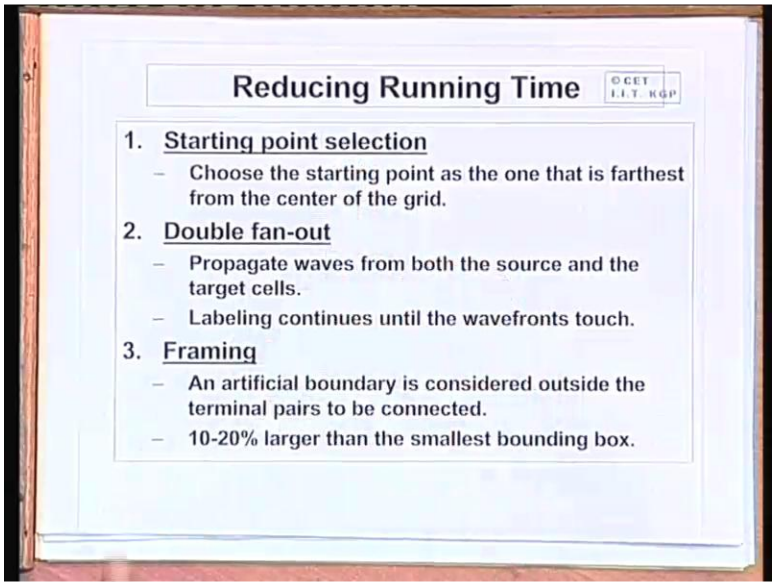
The first one says that you should choose the starting point suitably that is one is source and one is target. Fine. But which one you should select as source and which one as target that will influence the running time how.

(Refer Slide Time: 44:27)



Suppose you have the grid array like this. Ok. One of the points you want to connect is towards the corner and other is towards the center. Now if you select this as the source and this as the target then your wave front will move in all possible direction starting from S. But if you choose the other way around if you choose the one near the corner as the source and this as the target and the boundary will clearly not allow the wave front to expand in all directions. So the number of cells will be labeling will be much less one fourth of this approximately. So the first heuristic says that you choose a point near the corner as the source.

(Refer Slide Time: 45:17)



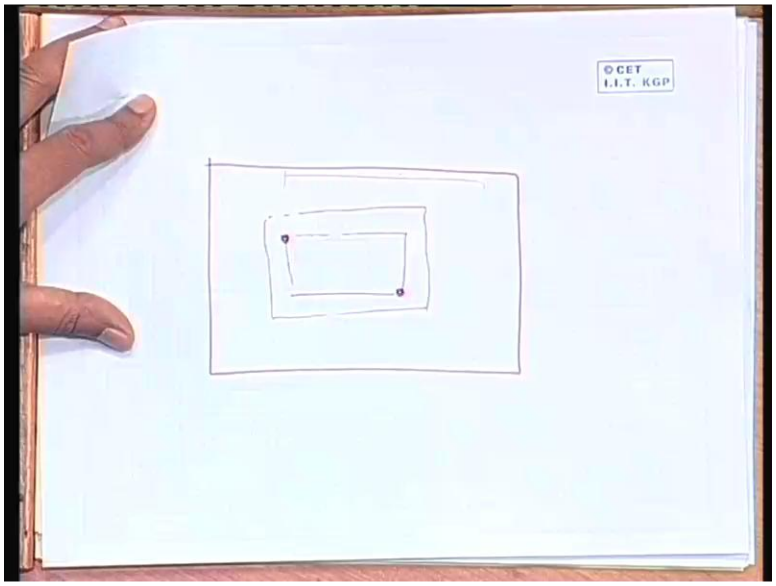
Reducing Running Time

© CET I.I.T. KGP

1. **Starting point selection**
   - Choose the starting point as the one that is farthest from the center of the grid.
2. **Double fan-out**
   - Propagate waves from both the source and the target cells.
   - Labeling continues until the wavefronts touch.
3. **Framing**
   - An artificial boundary is considered outside the terminal pairs to be connected.
   - 10-20% larger than the smallest bounding box.

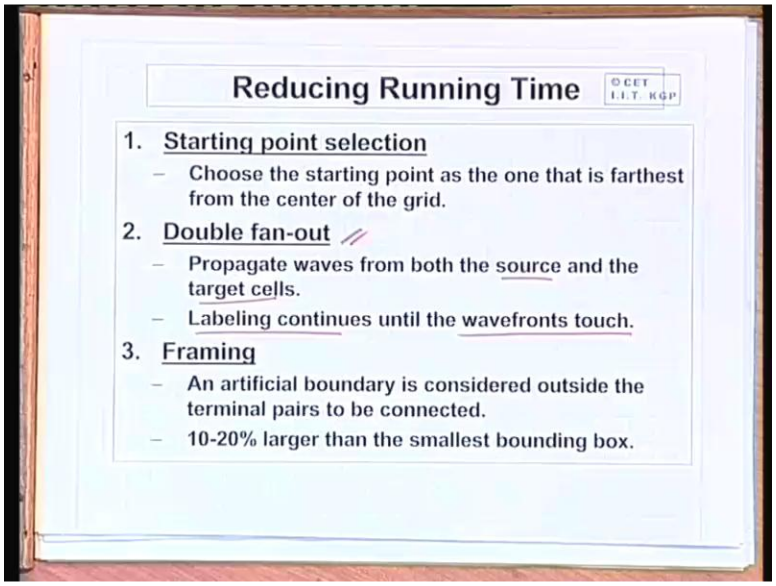
So this starting point as the one that is farthest from the center of the grid. The second one says well from the source as you go on expanding the number of cells you need to handle becomes larger and larger and larger. So as the second heuristic why do not you propagate waves from both the directions start from source and target both and as soon as the 2 wave fronts touch we stop and from that touching point you back trace the 2 directions this the other way around. This is double fan out propagate waves from both the source and the target cell labeling will continue until the wave fronts touch and the third one arises out of a realistic constraint.

(Refer Slide Time: 46:08)



Say the realistic constraint says suppose in this grid array you want to interconnect a point out here and a point out here. Now normally I would not have a path which will go like this. Then this is normally you if you imagine a rectangular bounding box certain percentage beyond that say ten percent or twenty percent beyond that. So you can say that I will not allow my net to go beyond that this is a realistic constraint this called framing.

(Refer Slide Time: 46:46)



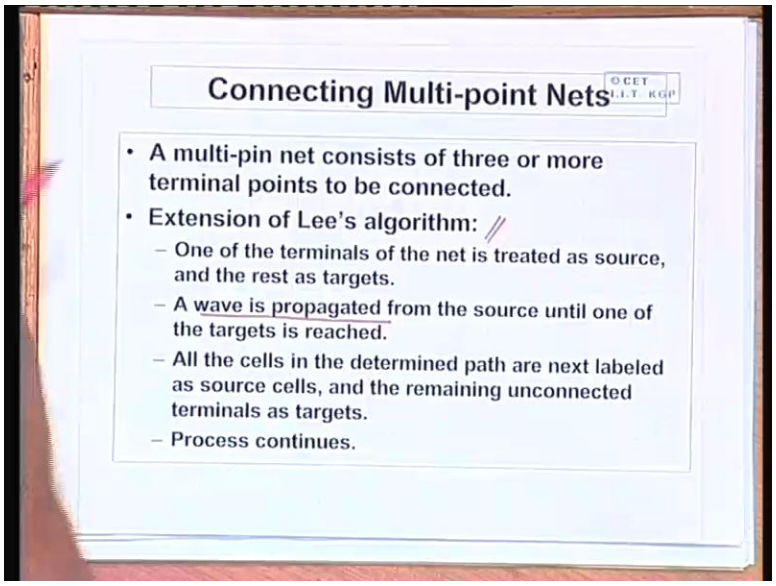
Reducing Running Time

©CET I.I.T. KGP

1. Starting point selection
   - Choose the starting point as the one that is farthest from the center of the grid.
2. Double fan-out
   - Propagate waves from both the source and the target cells.
   - Labeling continues until the wavefronts touch.
3. Framing
   - An artificial boundary is considered outside the terminal pairs to be connected.
   - 10-20% larger than the smallest bounding box.

So an artificial boundary considered outside the terminal pair to be considered to be connected and you can have some figure ten to twenty percent larger than that you will not label cells which are beyond that. So by using this heuristics you can improve the running point of the lee’s algorithm running time. (( )) (47:07) In this case you may not find a solution obviously. Yes but if you do not find a solution you can expand that imaginary box you can even redone it. Just one thing we will see later that these kind of greed routing algorithms are used only occasionally these are not used for routing all the nets. These are typically use to route the difficult nets ok. (( )) (47:36) Yes (( )) (47:38) Between the 2 routing means (( )) (47:46) No. That way not storing we are not keeping any history (( )) (48:00) That might of kept that means they modify it. (( )) (48:04) Yeah. We are forgetting again it yes true. So one thing you can you can slightly modify lee’s algorithm are to handle multi point nets also.

(Refer Slide Time: 48:23)

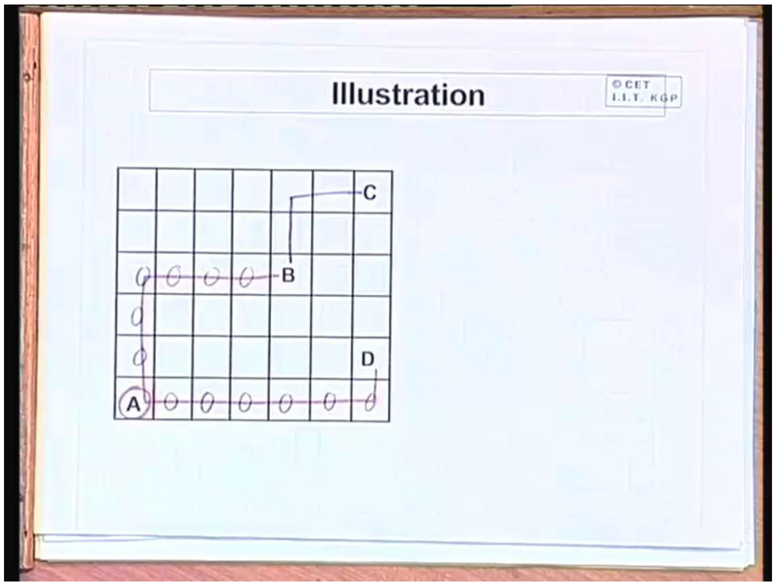


Connecting Multi-point Nets

- A multi-pin net consists of three or more terminal points to be connected.
- Extension of Lee's algorithm:
  - One of the terminals of the net is treated as source, and the rest as targets.
  - A wave is propagated from the source until one of the targets is reached.
  - All the cells in the determined path are next labeled as source cells, and the remaining unconnected terminals as targets.
  - Process continues.

Say also means multi point net is a net which will be connecting 3 or more terminal points. Now a simple extension of lee's algorithm can handle this how. See the extension of these algorithm will work like this one of the terminals of the net will be treated as source like the previous one and from there you start propagating wave till it hits one of the other points. So you have a partial net already laid out. Now all the grid cells you have already found out as part of this these all this cells will be acting as the source for the next step. So actually you are not starting from scratch in the next step you are starting from this partially completed layout and from all this points you are starting to propagate wave front to touch the new points and this process continues till all of them touch I have a small example to show you.

(Refer Slide Time: 49:35)



Illustration

Suppose I have a scenario like this A B C and D they all belong to the same net they all will have to be connected and incidentally A and D are at a distance 1 2 3 4 5 6 7 B is also at the same distance 1 2 3 4 5 6 7. So in the first step itself when you apply lee's you will possibly find a path like this and also find a path like this right. In the next step what I am saying is that say initially A was the target. Now source sorry source now in the next step when you are wanting to reach C not only A all the other cells which have been marked they will all act as sources. So wave front will go out from all those things and the nearest one to touch. So nearest will be from either D or C both will be there so you take one of them to reach C right.

So just by using an extension like this you can just have (( )) (50:50) Yeah. But the way you are doing it you are not taking that into account in that case you will have to reap up this net because your basic data structure is not efficient to do all those things. (( )) (51:10) Ok fine, fine. So instead of instead of having it from A to D and B both you take any one of them at a time from there ok. (( )) (51:29) There is other alternative. You take AD and from AD path you try to reach B and C do not take both these at the same time. Only one interval. Fine? Fine. There is another maze routing algorithm we will be discussing but that will do in our next class.

See there the concept is slightly different we will not be doing a blind search like the lee's algorithm. See this breadth first search or depth first search these are blind search algorithms we really do not know which side our solution lies we are exploding in all direction we are exploding all possibilities. Now we would be looking at an algorithm which is more like a goal directed search. Well we have the xy coordinates of the source and the target. So as we go on labeling the cells like we are doing in lee so we also know whether we are moving in the correct direction or moving out towards the wrong direction. So we define something like a cost function if you are moving away or not.

Because if you are moving away one step then to reach the target we again have to come back one step some while later. So more you move away you will have to move back that many steps to reach T. So this will be a priority kind of a thing that means you try to always move along direction where you are not moving away from the target. Well in some cases you will have to move away from the target you will try to minimize that how many minimum numbers of cells you can move away from the target and still you can manage that. Now this is the basic idea behind the hadlocks algorithm and this will be discussed in our next lecture.