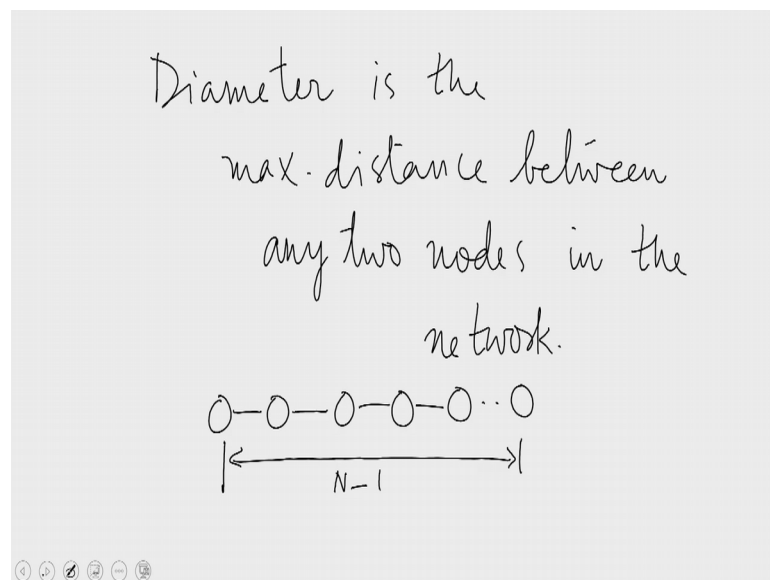


Parallel Algorithms
Prof. Sajith Gopalan
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati

Lecture – 27
Sorting on a 3D mesh

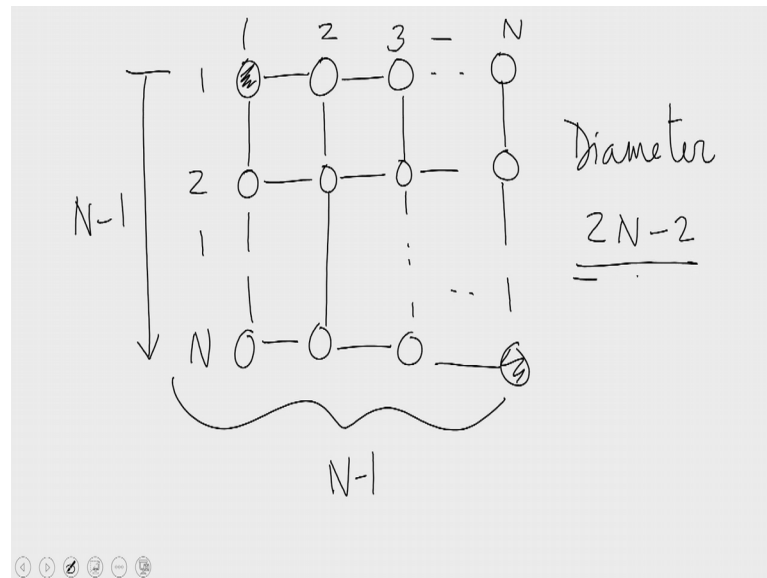
Welcome to the 27th lecture of the MOOC on Parallel Algorithms. In the previous few lectures, we have seen some algorithms that work on interconnection networks and we shall see some more today. When we discuss interconnection networks two of the parameters that are of relevance are the diameter of the network and the bisection width of the network.

(Refer Slide Time: 00:55)



What I mean by the diameter of a network is the maximum distance between any 2 nodes in the network. For example, when we consider a linear array of size N the maximum distance between any 2 nodes in the network is the distance between the first node and the last node which is N minus 1. So, the diameter of a linear array of size N is N minus 1.

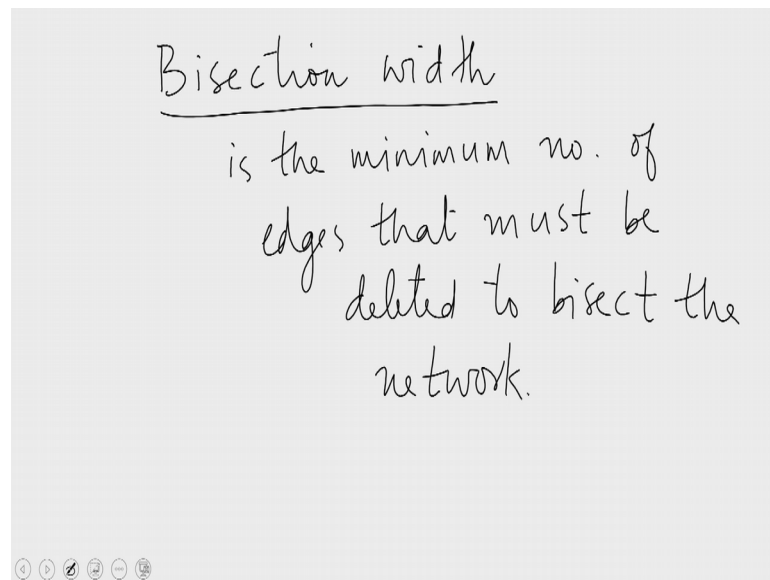
(Refer Slide Time: 01:59)



Instead when you consider 2 dimensional mesh that is N by N , when you consider a 2 dimensional mesh of size N by N , the maximum distance between any 2 nodes will be obtained by going from the top left to the bottom right for example, that is start at one corner and go to the diametrically opposite corner.

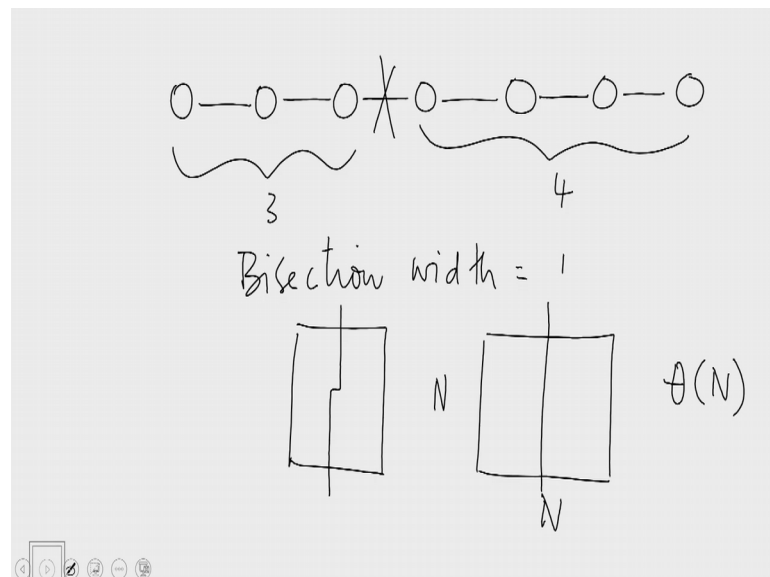
So, the distance traversed would be N minus 1 to come here and then N minus 1 to go here. So, the diameter is $2N$ minus 2. This is of course, one way of going from the top left corner to the bottom right corner. There are various other paths, as it happens in this case every single of these paths will have the same length $2N$ minus 2.

(Refer Slide Time: 03:19)



Another parameter that we shall often be talking about is the bisection width. The bisection width of a network is the minimum number of edges that must be removed to bisect the network. By bisection we mean we want to partition the network into 2 halves; so, that each half will be approximately of the same size.

(Refer Slide Time: 04:14)



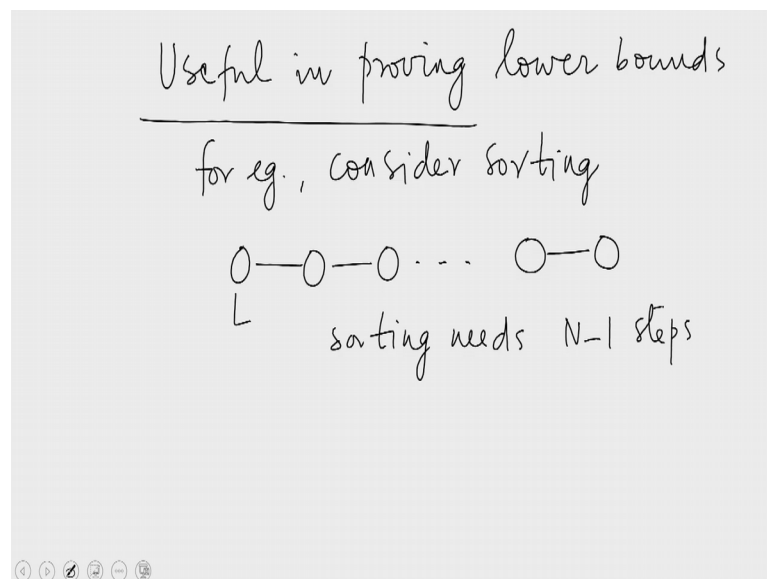
For example, when we are given a linear array of size 7 let us say. There are 7 nodes here and there are 6 edges. If you remove this edge the network is partitioned into

approximately 2 equal halves, in this side now we have 3 vertices and on this side we have 4 vertices.

So, this is as even as you can get in an in an network of odd size. So, in this case we say that the bisection width is 1. When you take an N by N mesh a division of the network like this when N is even we will ensure that the network is partitioned into 2 equal halves. If the network is of odd size then you can find a partition of the sort which will again divide the network into 2 equal halves approximately equal halves. So, you can work out the details, I am leaving the details for you to work out.

So, in either case we find that the bisection width is $\theta(N)$. So, in the case of a linear array the bisection width is 1 the, the bisection width is very low. In the case of an N by N mesh the bisection width is not very low it is not a constant it is $\theta(N)$. Now what are the relevances of diameter and bisection width?

(Refer Slide Time: 06:04)

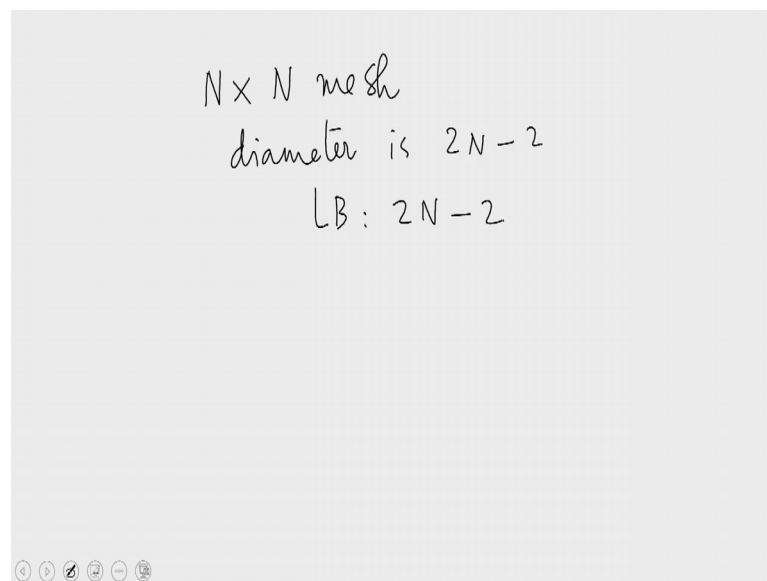


These are useful in proving lower bounds to many problems. For example, let us consider sorting. In the case of a linear array, where we want to sort the elements from the left to the right, let us say we keep the largest element in the first node. The largest element let us say L is kept in the first node; now when the sorting is over the largest element has to appear at the right end. For this to happen the element has to travel through a distance of N minus 1 which happens to be the diameter of the network.

So, here we are keeping the largest element a diameter away. So, that it will have to travel a distance equal to the diameter reach the destination. So, this ensures that the a lower bound for sorting is the diameter. So, on this in this case sorting requires at least N minus 1 steps. You can make the same argument about the general network. When we are given a general interconnection network, if we know the diameter of the network then for the problem of sorting we can take the element that is destined to a particular node and keep this node at it is diametrically opposite point. Then this element will have to travel through the diameter to reach it is correct position.

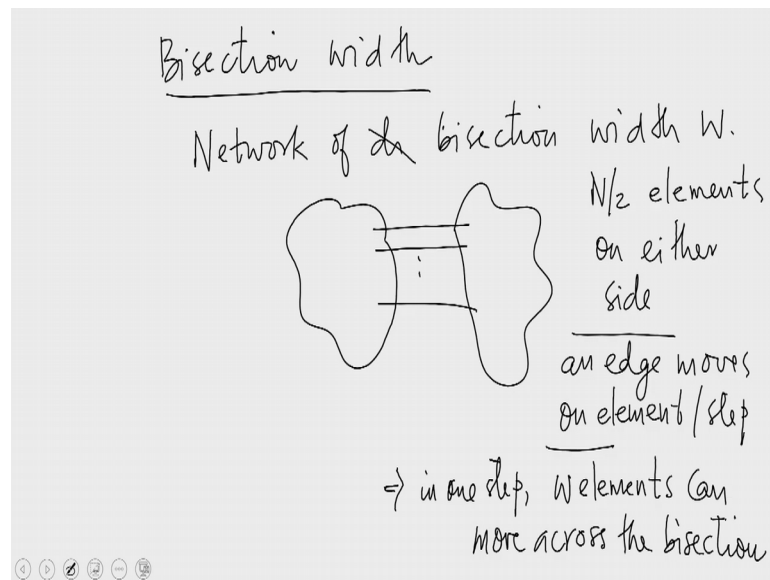
Therefore, the sorting algorithm is going to take at least that is that many steps. The time taken would be equal to the diameter. So, the same argument could be replicated for several other problems.

(Refer Slide Time: 08:06)



When we come to an N by N mesh, the diameter is as we have seen $2N$ minus 2. Therefore, a lower bound is $2N$ minus 2. This is of course, the argument we had seen earlier. And then we went on to improve this lower bound. So, in this fashion diameter can be used to prove several lower bound results.

(Refer Slide Time: 08:38)

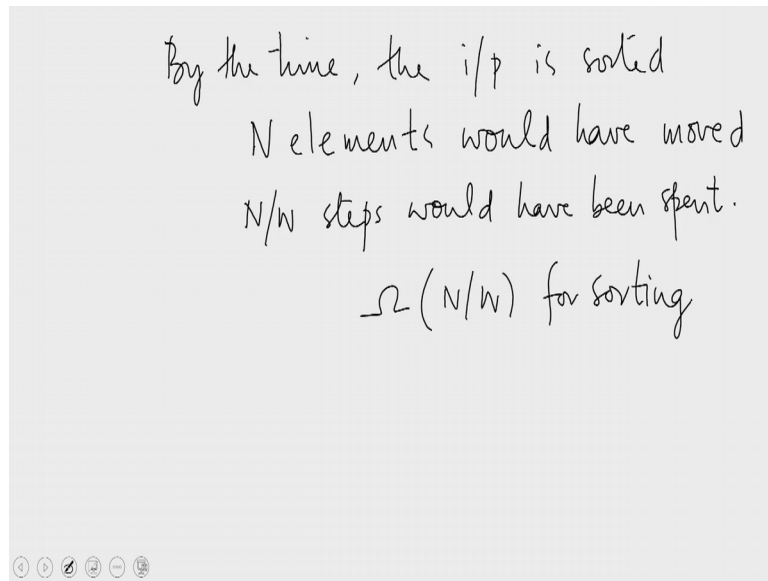


Then what is the use of bisection width? Let us consider a network of diameter d or bisection with W which means there are W edges; the removal of which will disconnect the network into 2 equal halves. Removal of these edges will disconnect the graph into 2 equal halves. Let us consider the problem of sorting looking at the network we can decide what should be the final sorted order that is where should the smallest element appear, where should the second smallest element appear and so on.

Now, we are given an input and the goal is to rearrange the elements permute the elements. So, that they reach the required positions. Now if we ensure that all the elements that should go to the right side are placed in the left side. And all the elements that should go to the left side or placed from the right side then every single element will have to cross the bisection. So, if the network has a size of N we have $N/2$ elements on either side give or take one. These elements have to cross over to the other side.

Similarly, there are $N/2$ elements on the other side that have to cross over to this side. So, we have a total of total traffic of N ; that should be going through these W edges, but then each of these edges can move only one element per step. We assume that an edge moves one element per step. This implies that in one step W elements can move across the bisection, but then we have a total traffic of N to be transported.

(Refer Slide Time: 11:19)



So, by the time the sorting is over, N elements would have moved across the bisection. In each step no more than W elements would have moved. Therefore, we would have spent at least N by W steps. So, this establishes a lower bound of N by W for sorting. So, likewise we can decide we can find the lower bound for many other problems.

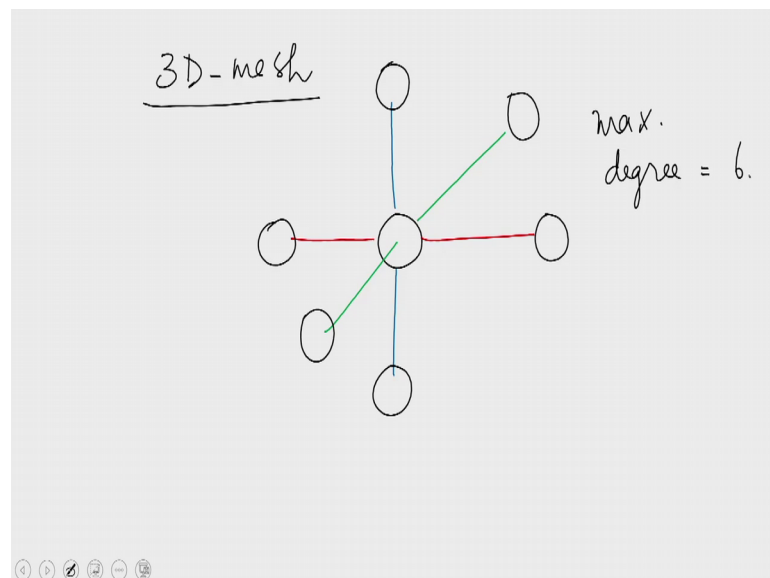
We estimate the total traffic that the problem will have to solve and we will arrange the inputs in such a way that the traffic will be maximized. Then the maximum traffic that is required to solve the problem divided by the bisection width will be the worst case lower bound for the running time of an algorithm for the problem on the given network.

(Refer Slide Time: 12:45)

Sort on a $\sqrt{N} \times \sqrt{N}$ mesh
Bisection width $W = \Theta(\sqrt{N})$
traffic = N
 $LB = \Omega(N/\sqrt{N}) = \Omega(\sqrt{N})$
Higher dimensional mesh ?

So, in particular when you want to sort on a root N by root N mesh as we have seen just now, the bisection width is theta of root N the net traffic required is N . Therefore, the lower bound is omega of a N by root N which is omega of root N . That is you cannot expect to sort faster than root m . Of course, we have already seen a precise lower bound for this purpose, but in particular if we are looking at a higher dimensional mesh we can use this method to evolve a lower bound, but what is a higher dimensional mesh.

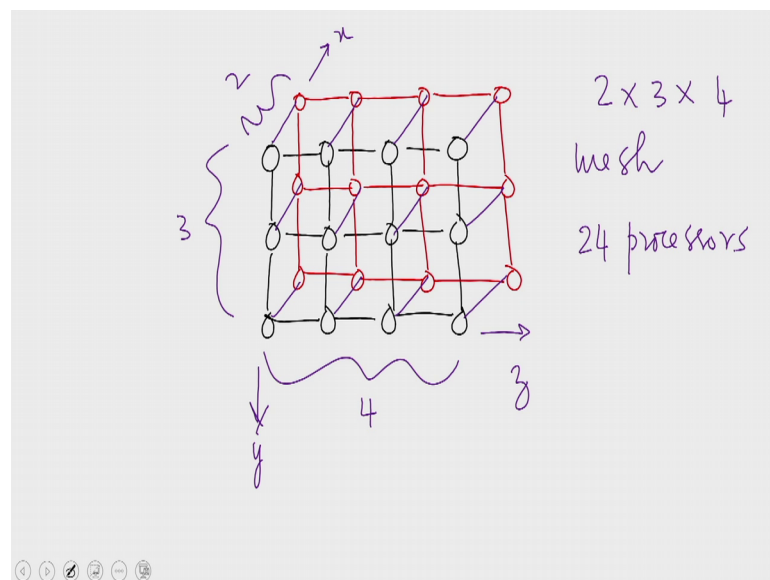
(Refer Slide Time: 13:58)



First let us consider a 3 dimensional mesh. A 3 dimensional mesh is a straightforward generalization of a 2 dimensional mesh. In a 3 dimensional mesh we have 3 dimensions a node has 2 neighbors along one dimension, and it has 2 neighbors along a second dimension, and then it has 2 more neighbors along a third dimension. So, a node could have 2 dimension a 2 neighbors along the blue dimension 2 neighbors along the red dimension and 2 neighbors along the green dimension.

So, the maximum vertex degree this 6, but of course, as we know from the 2 dimensional case, the corner nodes will have smaller vertex degrees. A corner node in this case can have a vertex degree of 3. So, the vertex degrees will range from 3 to 6.

(Refer Slide Time: 15:03)

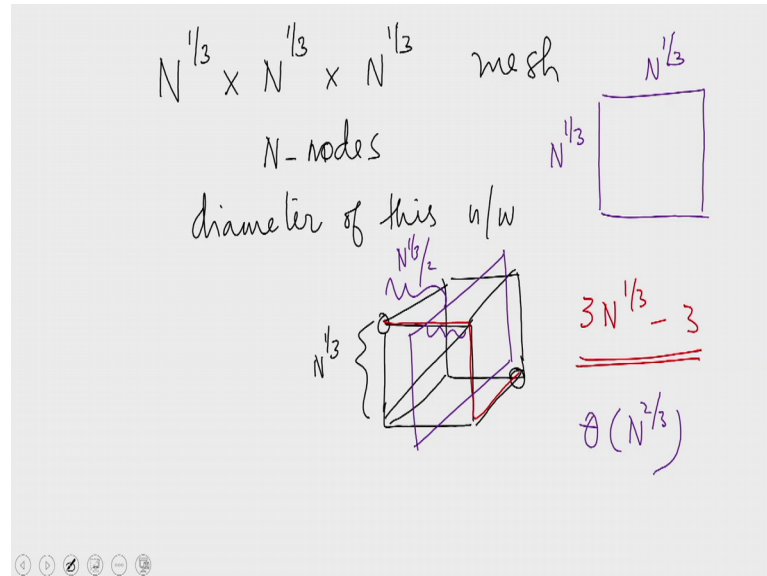


So a 3 dimensional mesh will look like this. You have multiple 2 dimensional meshes, and we have connections between the corresponding nodes that forms a 3 dimensional mesh. So, in this case there are 3 nodes along this dimension, 4 nodes along this dimension and there are 2 nodes along this dimension. So, if you call this is the x dimension this is the y dimension and this is the z dimension then we can say this is a 2 by 3 by 4 mesh.

It has a total of 24 processors and the maximum vertex degree is 6 and the minimum vertex degree is 3. Such a mesh is called a 3 dimensional mesh. Then of course, we can go on from a 3 dimensional mesh to a 4 dimensional mesh. You take points in 4

dimensions and connect them up in general you can talk about a k dimensional mesh for any number k .

(Refer Slide Time: 16:52)



Now let us consider a 3 dimensional mesh which is $N^{1/3}$ by $N^{1/3}$ by $N^{1/3}$. This has a total of N nodes. And these nodes are arranged in the form of an $N^{1/3}$ by $N^{1/3}$ by $N^{1/3}$ mesh. Each node has at most 6 neighbors. What would be the diameter of this network? The maximum distance between any 2 nodes think of these networks as a cube, where each side is $N^{1/3}$.

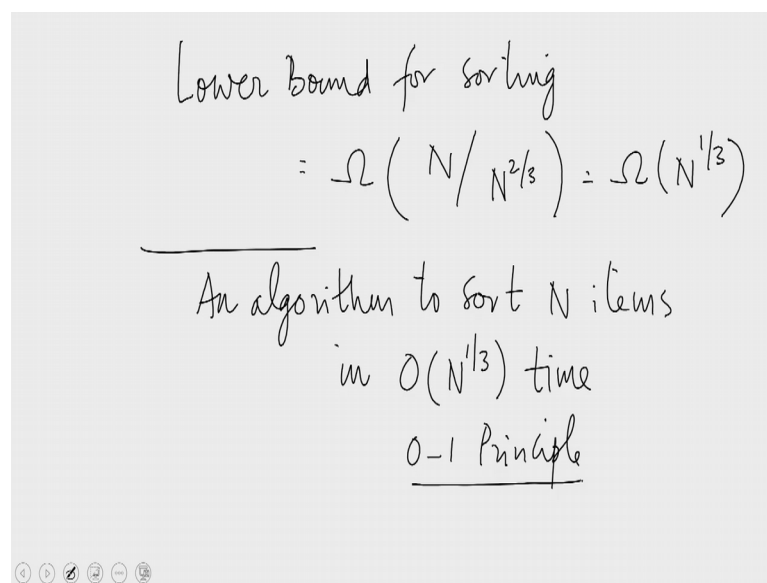
So, if you start at a node and go to the diametrically opposite node then you can traverse any one of the possible paths. This is one way of going from a node 2 it is diametrically opposite node. So, the distance traversed here is $3N^{1/3} - 3$. So, this is the diameter of this mesh. Now what would be a bisection width? I leave a rigorous proof to you, but to find the rigorous find the bisection width, what you can do is to cut the mesh along this direction and using a plane of the sort with $N^{1/3} - 1$ elements on the side and $N^{1/3} - 1$ elements on this side. For each x line and x line is a line of processes that are parallel to the x axis.

So, let us say we are cutting every x line into 2 equal halves. Then the number of processes on the left side and the number of processes will on the right side will be approximately equal. If $N^{1/3}$ happens to be an odd number then you will have

to introduce a kink a planar cut will not be enough, but still you can device a cut which ensures that the number of vertices on either side is approximately the same. And the number of edges disconnected is of the same order as the number of edges on a plane.

So, we are now cutting perpendicular to the x axis using a y z plane. So, the number of edges disconnected will be the number of edges that are on 1 plane. So, when you consider a plane of the sort, the number of edges here would be theta of N power 2 by 3. So, the bisection width of the network is going to be theta of N by N power 2 by 3.

(Refer Slide Time: 20:11)



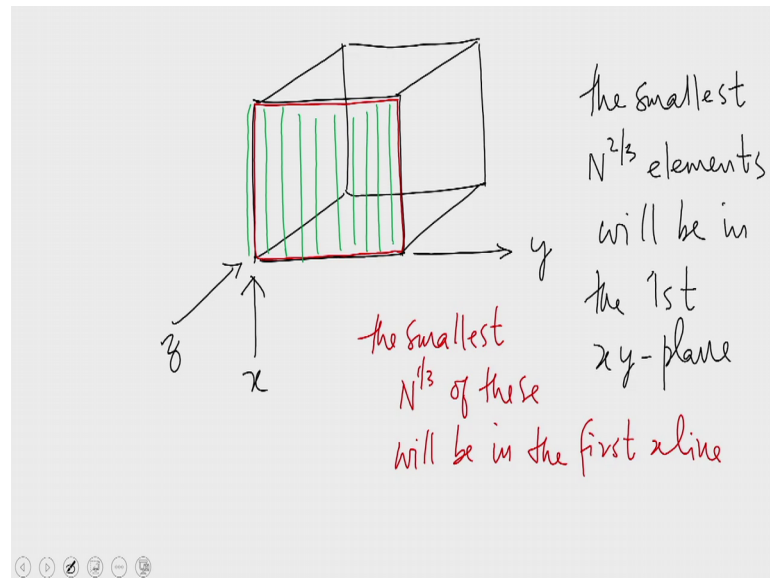
Lower bound for sorting
 $= \Omega\left(\frac{N}{N^{2/3}}\right) = \Omega(N^{1/3})$

An algorithm to sort N items
in $O(N^{1/3})$ time
0-1 Principle

So, using this once again we get a lower bound for sorting. It is the total traffic divided by the bisection width which is omega of N power 1 by 3. So, you cannot expect to sort N elements faster than N power 1 by 3 asymptotically on this model. Now let us an algorithm that actually does this that is it runs in order of N power 1 by 3 time. So, that is what we are going to see next. In order of N power 1 by 3 time on an N power 1 by 3 by N power 1 by 3 3 dimensional mesh.

As I described the algorithm I will also prove it is correctness for proving the correctness I will resort to the 0 one principle which asserts that if an algorithm works correctly on all binary sequences it should work correctly on all sequences drawn from a linearly ordered set. So, we assume that we are sorting a binary sequence if the algorithm works correctly on the binary sequence it should work correctly on any sequence drawn from a linearly ordered set.

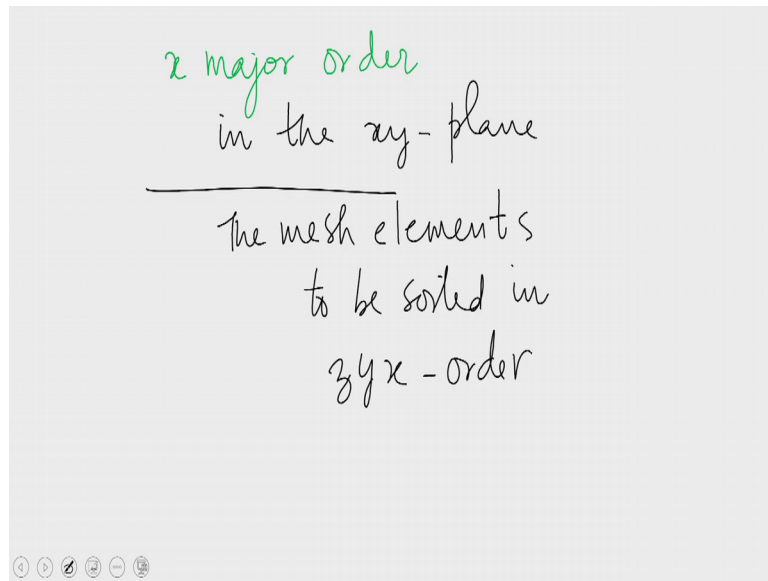
(Refer Slide Time: 21:42)



So, we are given a mesh. Let me represent the mesh using a cube. To indicate the 3 dimensions. Let us say the vertical dimension is the x dimension, the horizontal dimension is the y dimension and the dimension into the board is the z dimension. So, let us say we want to sort the elements so, that the smallest $N^{2/3}$ elements will be in the first x y plane. So, if the planes are numbered from top to bottom, then the first x y plane happens to be or if the planes are numbered from front to back then the first x x y plane happens to be this.

So, we assume that the smallest elements are going to be the smallest $N^{2/3}$ elements will be on this plane. And then in addition we will assume that the smallest of the smallest $N^{1/3}$ of these will be in the first x line which is this one. The first x line let us say is this one. So, the first x y plane is made up of several x lines. An x line is a line of processors that are parallel to the x axis. So, the one that is marked now is the first x line then we have the second x line third x line and so on. These x lines will form the first x y plane.

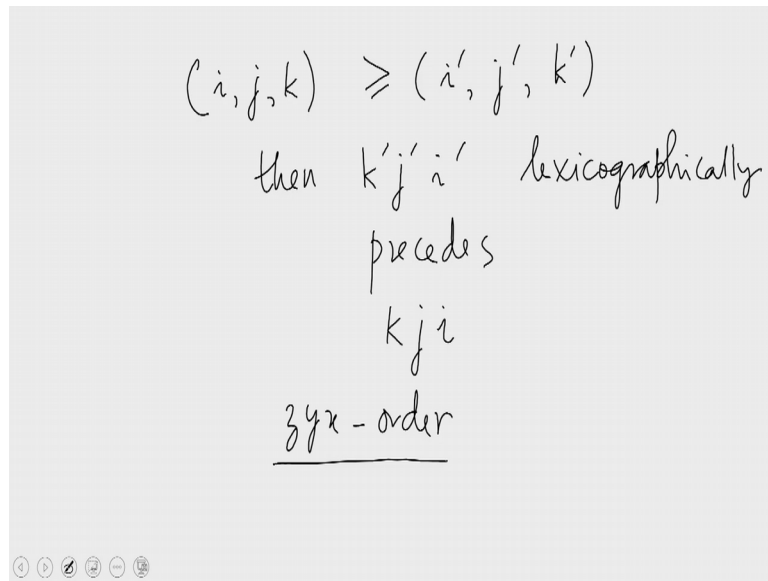
(Refer Slide Time: 24:03)



So, in the first x y plane, we want the elements to be sorted in x major order. In general, we want the elements in the mesh must be sorted in the z y x order. What it means is that x is the first major y is the second major and z is the third major. In other words, when we look at the elements in sorted order, the x dimension will move first that is to find the next element you should go down the x dimension to the next element.

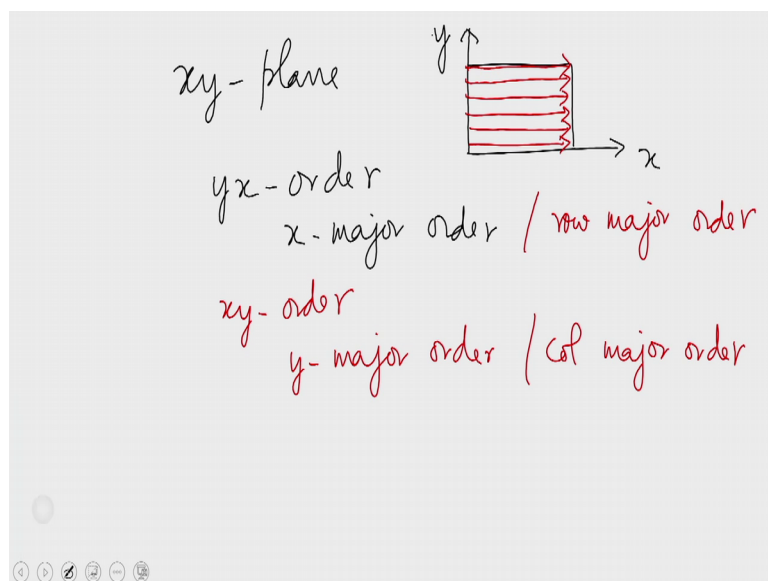
Once the x dimension is exhausted you should move along the y dimension to find the next. And then once the y dimension is also exhausted that is once the x y plane is over then you should move into z dimension and go to the next x y plane. So, this is the order in which we want the elements to be sorted.

(Refer Slide Time: 25:17)



Or, in other words what we want to say is that if the i j k th element is greater than or equal to the i prime j prime k primeth element, then k prime j prime i prime lexicographically precedes $k j i$ that is the formal statement. We want the elements to be finally, arranged in such a way that if the element that appears at position $i j k$ is greater than or equal to the element that appears at i prime j prime k prime, then the pair the order triplet k prime j prime i prime lexicographically precedes $k j i$. So, this is the same as the $z y x$ order.

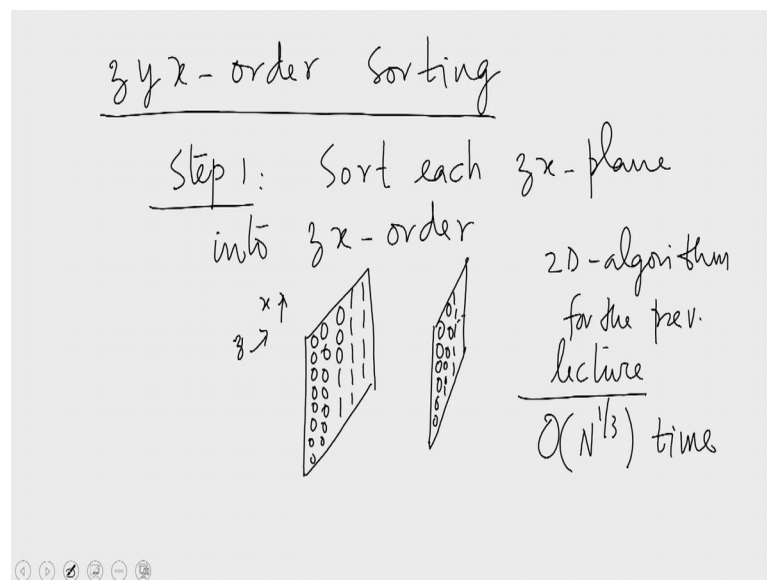
(Refer Slide Time: 26:26)



So, when we talk about a 2 dimensional plane using the standard notation if I write x along the horizontal axis and y along the vertical axis, and if this is how I look at a plane the x y plane. If I sort in y x order which is the same as the x major order x would be moving first; means to find the next element for an element you should go along the x x dimension. Once the x dimension is over you should move in the y dimension and then continue so on.

So, this is the x major order which according to this figure if you consider the rows horizontal and columns vertical. This is the same as the row major order. The x y order happens to be the y major order. So, this is the same as the column major order.

(Refer Slide Time: 27:59)



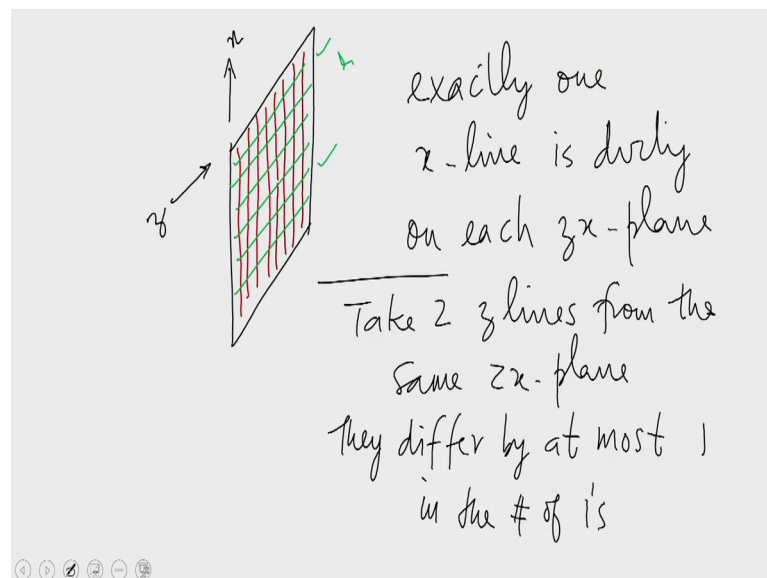
So, what we want is that the elements must be sorted in z y x order. So, the algorithm proceeds in this fashion. As I described the algorithm I will also explain the algorithm using an instance of 0s and 1s. If the algorithm works correctly on a binary sequence it should work correctly on any sequence. So, in step one of the algorithm, what we do this we sort each z x plane into the z x order. What it means is that we consider the z x planes the upper dimension is the x dimension and the into the board dimension as the z dimension.

So, we consider the z x planes and we sort each z x plane of the mesh in z x order which means x moves first. So, in the so, if you have sorting from top to bottom we will be sorting the elements in those fashion. This is the z x order sorting of a number of bits that

appear in this $z \times x$ plane. There is a elements are moving first in the x order the sorted order is moving first in the x order. Once the x dimension is exhausted we move in the z order.

So, in another $z \times x$ plane, in which the sorting is proceeding in parallel we will have the results in this fashion (Refer Time: 30:08). So, let us say the $z \times x$ planes are sorted in this fashion. This sorting can be done using the 2 d algorithm we saw earlier. Therefore, this step will take order of N power 1 by 3 time. So, in this case we are sorting N power 2 by 3 elements on each $z \times x$ plane. Each $z \times x$ plane is essentially an N power 1 by 3 by N power 1 by 3 2 dimensional mesh. So, the algorithm of the last class is exactly applicable. So, the sorting will be over in order of N power 1 by 3 time.

(Refer Slide Time: 31:04)



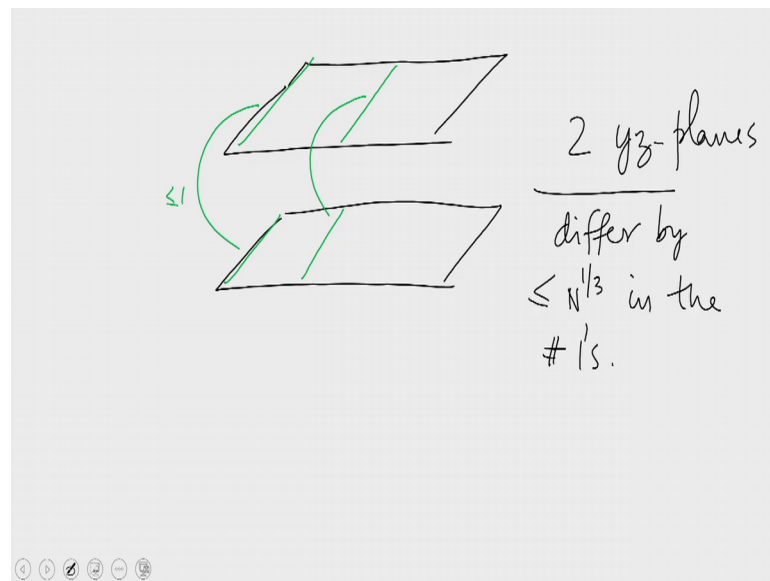
Now, look at a $z \times x$ plane. This $z \times x$ plane has been sorted. What it means is that in this every x line the x line is a line which is parallel to the x axis. Every x line is now sorted. So, I have used 2 colors the light red and the dark red to indicate 0 and one what we find is that there is exactly one x line which is dirty on each $z \times x$ plane.

I have one sample $z \times x$ plane here. Remember x is the vertical dimension and z is the into the both dimension. So, when you consider the x lines we find that all of them except one are clean. The earlier 1s are 0 lines and the later 1s are one lines. Now let me consider the x lines here the z lines here. The z lines are the lines that are parallel to the z axis. A line of processors that are parallel to the z axis will form what we call a z line. So, let us

consider the z lines. When we compare 2 z lines here we find that they have approximately the same number of ones. There can be a difference of at most one in the number of 1s between 2 z lines from the same z x plane.

So, what we find is that take 2 z lines from the same z x plane, they differ by at most one in the number of 1s. 2 z x planes differ by 2 z lines from the same z x plane differ by at most one in the number of ones.

(Refer Slide Time: 33:53)



Now let me consider two y z planes. Consider any 2 y z planes of the mesh. When you consider 2 y z planes, we discovered that these y z planes are made up of 4 corresponding z lines. Take these 2 z lines they belong to the same z x plane. Similarly, you take 2 z lines here they belong to the same z x plane.

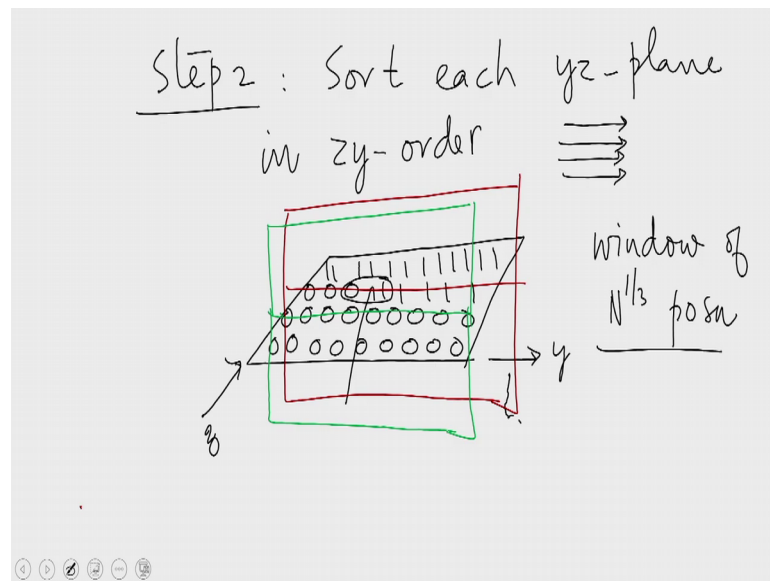
So, these 2 y z planes are made up of pairs of corresponding z lines, where each pair belong to the same z x plane. Now from our conclusion in the previous slide what we know is that between these 2 pairs between the elements of each of these pairs, there is a difference of at most one in the number of 1s. That is these 2 z lines differ by at most one in the number of 1s. So, do these 2 and so on.

So, if you aggregate over the entire y z plane, the 2 y z planes we find that they differ by at most N power 1 by 3. These 2 y z planes differ by at most N power 1 by 3 in the number of 1s. That is because every y z plane is made up of z lines. Therefore, if you

take 2 y z planes for every z line here take the corresponding z line on the other y z plane these 2 z lines will fall on the same z x plane and therefore, they differ by at most one in their number of ones.

Therefore, if you aggregate over these 2 y z planes, we find that they differ in at most $N^{1/3}$ in the total number of ones. Because the total number of z lines is $N^{2/3}$.

(Refer Slide Time: 36:05)

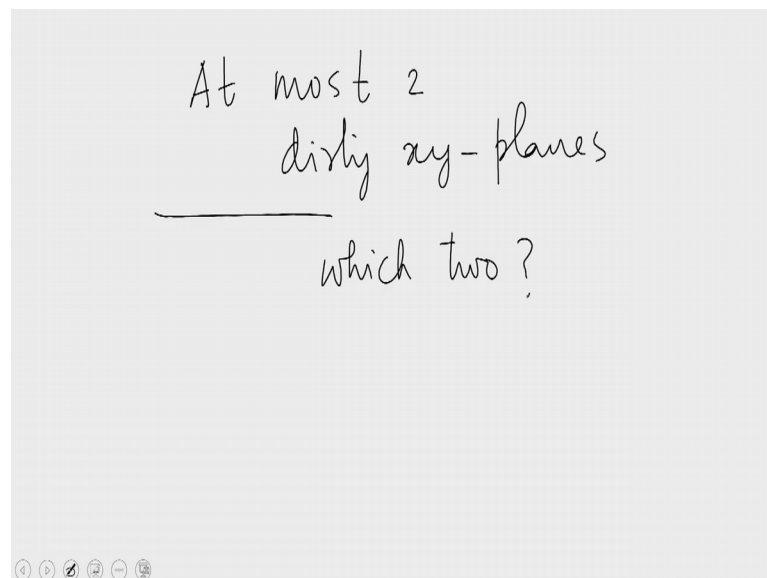


Now what we do in step 2 is to sort each y z plane in z y order. We sort every y z plane in z y order; mind you in this sorting as well as the sorting of the previous step. We assume that the sorting is not done in a snake like order, but in this order. When we sort a plane here we assume that every row is sorted in the same direction, but this is not difficult. If you manage to sort them in snakelike fashion, then we can switch the order of every alternate row in order of $N^{1/3}$ time.

Therefore, the time taken for this first step as well as this step is order of $N^{1/3}$. So, we have now sorted every y z plane in z y order. So, when you look at a y z plane, what we find is that it is sorted in this fashion. The y z plane is now sorted in this fashion y is the horizontal dimension and z is the into the both dimension. So, every y z plane is sorted in this fashion. When I take another y z plane we find that the kink in that y z plane, the kink is the point at which 0s transition into 1. So, the kink here happens at this point the switch from 0s to 1.

So, the position at which the kink happens in 2 y z planes differ by at most $N^{1/3}$. So, if I take another y z plane I find that the kink in that y z plane is happening roughly at the same place. It could be in the adjacent row. So, that is the earliest when you when I take the earliest y z plane and the latest y z plane in terms of the position of the kink, we find that they differ by at most $N^{1/3}$. Or in other words all the kinks are happening within a window of $N^{1/3}$ positions. We have a window of $N^{1/3}$ positions in which all the kinks happen. What this means is that, when you look at the entire network the dirty y lines in every z y plane there are only 2 such aggregating over all the z y planes of the mesh.

(Refer Slide Time: 39:09)

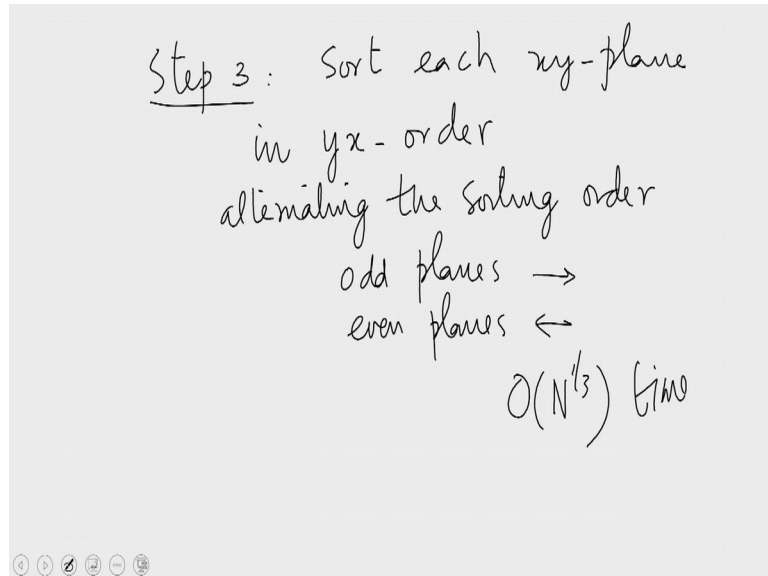


In other words, there are at most 2 dirty x y planes, y lines from all the z y planes aggregated together will form an x y plane. That is if I take the first y line from here if I take the first y line from the next z y plate and so on if I aggregate all of them I will get the first x y plane. Now here I find that the this plane has exactly one dirty row dirty y line. When I go to another y z plane the dirty y row there could be either the same one or the adjacent one, it cannot be any other.

Therefore, there could be at most 2 dirty x y planes in the entire mesh this could be one dirty x y plane or this could be another dirty x y plane. So, there could be at most 2 dirty x y planes in the entire mesh. So now, all that is left is for us to fix these 2 x y planes, but the problem is that we do not know which are these x y planes. We know that there are

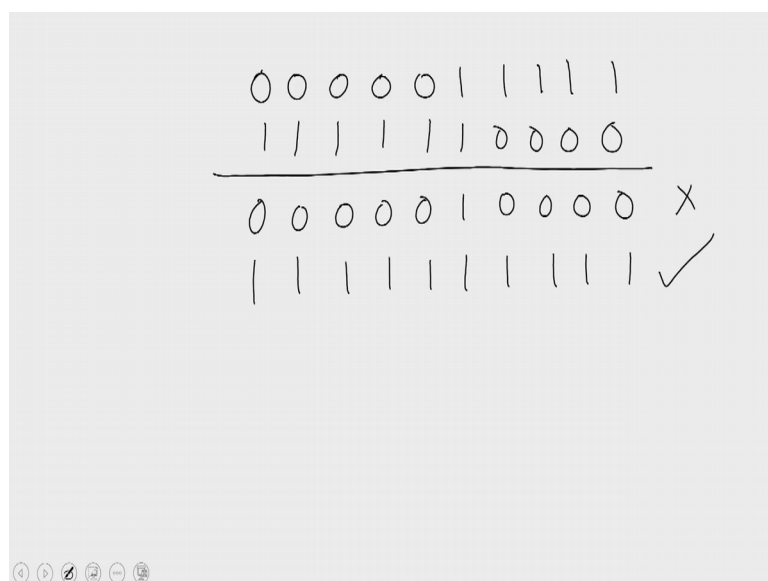
only 2 x y planes in the worst case, but we do not know which 2. Since we do not know which these 2 dirty x y planes are what we do is this.

(Refer Slide Time: 40:52)



In step 3 we sort each x y plane in y x order. That is the x major order. Every x y plane is now sorted in the x y order. We know that there are only 2 dirty x y planes both of these x y planes are now sorted, but then here we do not sort every x y plane the same way. We alternate the sorting order we sort the odd planes one way, the even planes the other way.

(Refer Slide Time: 41:58)



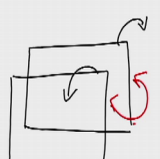
Our purpose is to do exactly that what we did in the case of shear sort. Let us say in the in a 2 dimensional case consider the 2 dimensional case let us say we have 2 dirty rows that are sorted in the opposite direction. We saw that if we compare and exchange them position by position, with the smaller element going to the top and the larger element coming to the bottom, we saw that at least one of the rows will become a clean row. In this case the bottom row has become a clean row. Whereas, the top row continues to be dirty, but if the number of 0s and the number of 1s exactly matched then we will have to clean rows.

This we had seen earlier in the previous lecture. So, we are going to use essentially the same concept here. The situation here is that we have 2 dirty $x \times y$ rows, but we do not know which of them are dirty which those 2 are. Therefore, we sort every $x \times y$ plane and $y \times x$ order alternating the sorting order. The odd planes are all sorted in one direction the even planes are all sorted in the opposite direction. Then the 2 dirty $x \times y$ planes will be in a situation that is similar to those of these 2 rows. They are pitted against each other. And then if we compare and exchange in this fashion we can ensure that at least one of them will turn a clean $x \times y$ plane.

So, we will be left with only one dirty $x \times y$ plane if we manage to do this, but then the trip is that, but then the problem is that we do not really know whether they form an odd even pair or an even odd pair.

(Refer Slide Time: 43:51)

Step 4 Do ~~2~~ 2 steps of
OETS on each z -line



$O(1)$ time

There is at most one dirty xy -plane

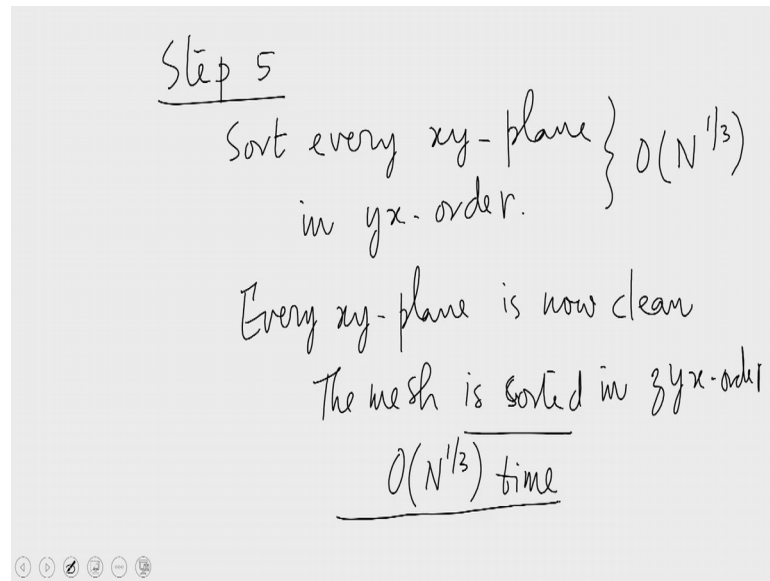
Navigation icons: back, forward, search, etc.

Therefore, to rectify this problem in step 4 what we do is this. We do 2 steps of odd even transposition sort on each z line. That is, we are going across the x y planes. So, if we look at the 2 dirty x y planes, it could be that in the first step they are paired of with others. It could be that the dirty x y plane at the back is paired with the plane which is further to the back and the one in the front is paired with the one which is further to the front. Then the compare exchange will not create any difference. That is the first step of the OETS that we do will not make any difference.

Even if this is the case in the next step these 2 will be paired together. When these 2 are paired together at least in the second step of the odd even transposition sort that we do here there will be an exchange. If they happen to be paired in the first step, no worry one of them will become clean in the first step itself there could be at most one dirty plane in the next step. So, in the next step it will be paired with either to the front or to the back that will not make any difference because, to the front we have all 0's and to the back we have all 1's.

So therefore, a compare exchange in the second step will then be a waste will be unnecessary it will not make any difference to the mesh, but either way once we do this that is we do 2 steps of odd even transposition sort on every z line. Once we do this what we ensure is that there is at most one dirty x y plane, but of course, we do not know the identity of this x y plane. We do not know which of the x y planes is dirty therefore, what we do in the next step.

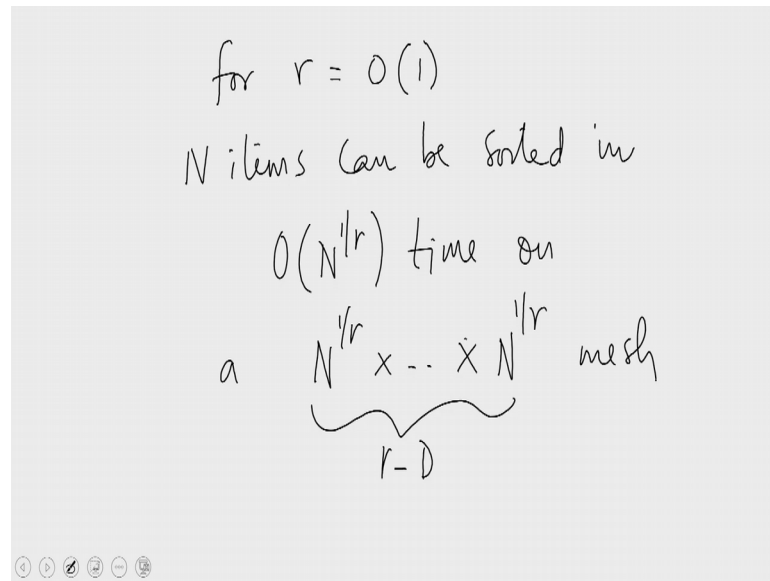
(Refer Slide Time: 46:17)



In step 5 we sort every x y plane in y x order. This ensures that every x y plane is now clean. And the entire mesh is sorted. So, this algorithm works correctly on a binary sequence. So, doing a quick recap the steps of the algorithm are just these we sort every z x plane in the z x order this will take order of N power 1 by 3 time, by the algorithm that we saw in the previous class. So, after sorting every z x plane in the second step we sort each y z plane in the z y order. Which again we will take order of N power 1 by 3 time using the same algorithm. And then in step 3 we sort every x y plane, but alternating the order of the odd planes and the even planes. This will again take order of N power 1 by 3 time we are using exactly the same algorithm once again.

Then in step, 4 we do 2 steps of odd even transposition sort along each z line. This will take order of one time. There are only 2 steps to perform and then finally, in step 5 we sort every x y plane in y x order. This will ensure that the entire mesh is sorted in z y x order. So, this also we will take order of N power 1 by 3 time. Therefore, on the hold this algorithm runs in order of N power 1 by 3 time. Which means N items can be sorted in order of N power and by 3 time on a 3 dimensional mesh.

(Refer Slide Time: 48:49)



Now, going further it can be shown that even though we will not show it here for a constant r N items can be sorted in order of N power $1/r$ time on an r dimensional mesh which is regular. An N power $1/r$ side at r dimensional mesh N items can be sorted in order of N power $1/r$ time which is the generalization of the result we have seen just now for constant r . Of course, the result will not work when r is not a constant. So, there is a result for sorting on multi dimensional meshes; that is it from this lecture. Hope to see you in the next lecture.

Thank you.