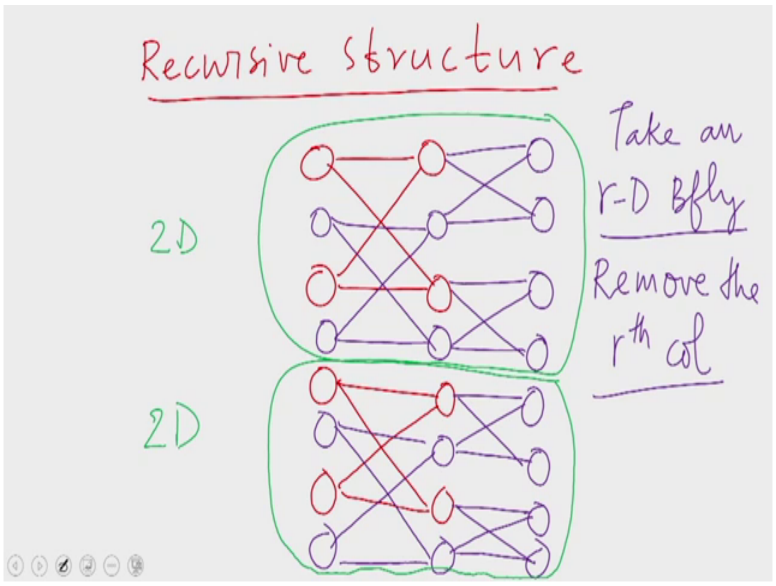
Parallel Algorithms

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

Lecture – 31
Butterfly, CCC and Benes Networks

Welcome to the 31st lecture of the MOOC on Parallel Algorithms. In the previous lecture we were seeing some interconnection networks that are related to hypercubes. In particular we had started with the butterfly network. Let us see some nice properties of the butterfly networks today.

(Refer Slide Time: 00:46)



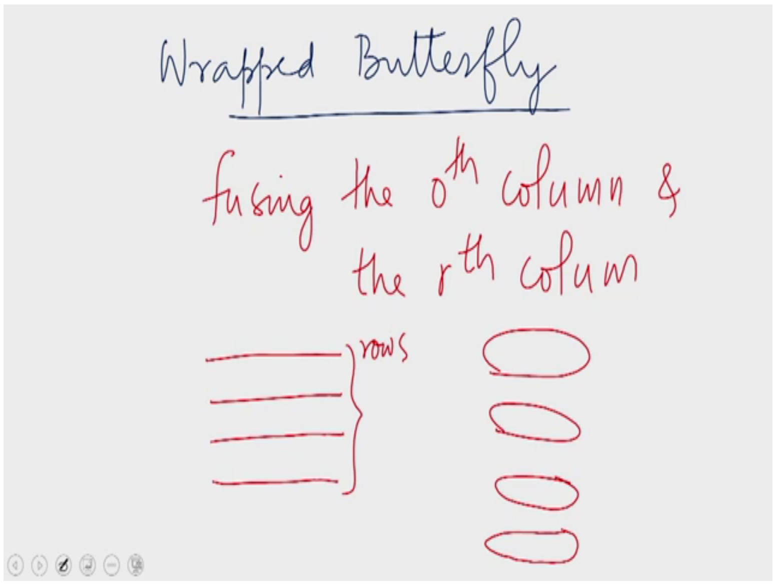
The butterfly network has some nice recursive properties. For example, let us consider 2 dimensional butterfly; a 2 dimensional butterfly has 3 columns, in each column we have 4 vertices, that is there are 4 rows. We have interconnections, cross connections of this sort; between the first 2 columns and between the last 2 columns we have edges of this sort, and then there are the straight edges. Now let me take an identical copy of this, and in disperse it with the original copy.

So, we have connections made like this and then let us say we add one more row here. And then let us add connections of the sort. So, what we have seen is that, a 3 dimensional butterfly can be realized out of 2 copies of 2 dimensional butterflies. Within the first 3 columns, the red vertices in the red edges define one copy of the 2 dimensional butterfly. The blue vertices and the blue called blue edges define another copy of a 2 dimensional butterfly. When we add another column of vertices to that and add edges as we see between the third and the fourth columns, we realize a 3 dimensional butterfly.

So, what this means is that, if we take an r dimensional butterfly and remove the last column, we get 2 copies of 2 dimensional butterfly, the red copy as well as the blue copy. This is one way of recovering an r minus 1 dimensional butterfly, out of an r dimensional butterfly. You remove the last column of vertices as well as the edges which are incident to these vertices. Another way of getting to r minus 1 dimensional butterflies is to remove the first column. If we remove the first column as well as the edges that are incident to it, first column and all the edges that are incident to it, we find that we have 2 copies of r minus 1 dimensional butterflies. So, an r dimensional butterfly can be decomposed into 2 r minus 1 dimensional butterflies in this fashion.

So, this is one 2 dimensional butterfly and this is another 2 dimensional butterfly. You either remove the 0th column or you remove the rth column, what you get is 1 copy of an r minus 1 dimensional butterfly. So, that is a nice recursive property of a butterfly.

(Refer Slide Time: 05:11)



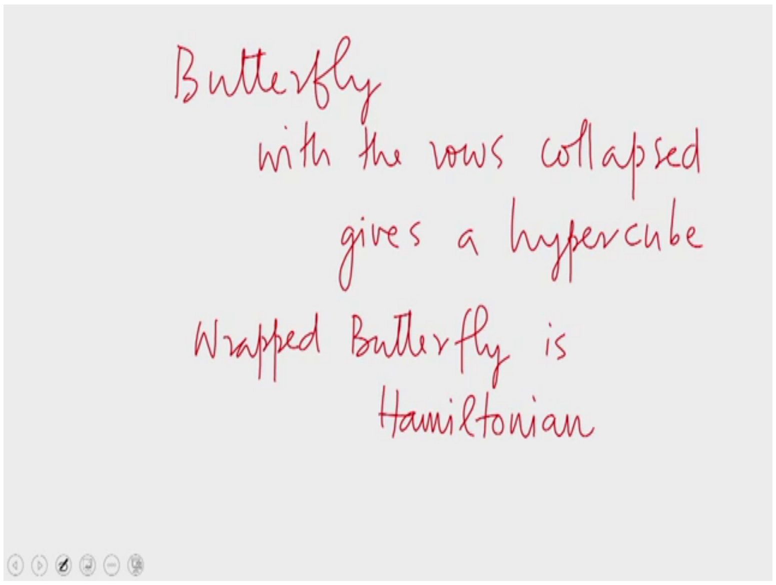
Wrapped Butterfly

fusing the 0th column & the rth colum

Now, let us see a related network of butterfly, what we call wrapped butterfly. A wrap butterfly is obtained by fusing the 0th row, 0th column and the rth column. In a butterfly we have a straight edges that define the rows, in a regular butterfly the straight edges defined the rows. In a wrap butterfly every row is converted into a cycle by fusing the first column and the last column. The remaining edges remain are the exactly. The same the 0th row and the rth row are fused into 1; therefore, all the straight edges will now defined a cycle.

So, there are 2 power r cycles, each cycle corresponds to a vertex in the hypercube. You would remember that, in our butterfly with the rows collapsed gives a hypercube.

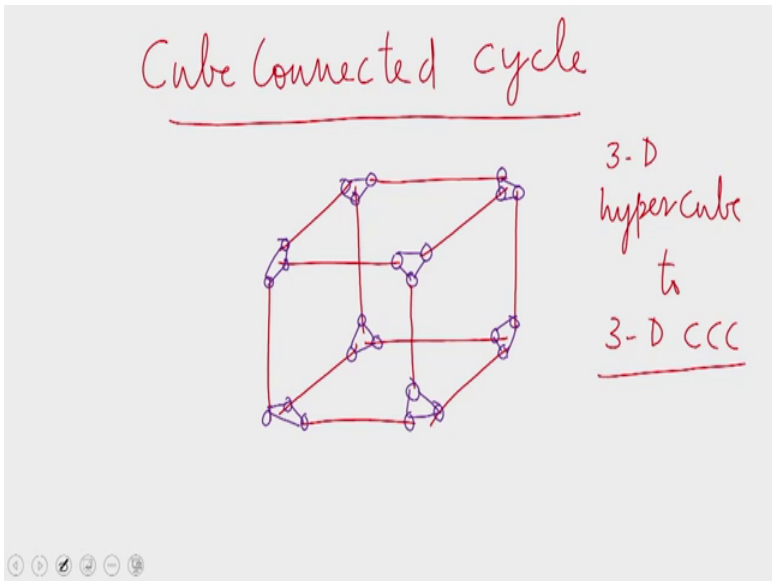
(Refer Slide Time: 06:32)



Butterfly
with the rows collapsed
gives a hypercube
Wrapped Butterfly is
Hamiltonian

In a wrap butterfly every hypercube vertex is replaced by a ring of size r and then the remaining cross edges are there. A wrap butterfly has some nice properties that even a butterfly does not have; a wrap butterfly for example, is Hamiltonian.

(Refer Slide Time: 07:32)

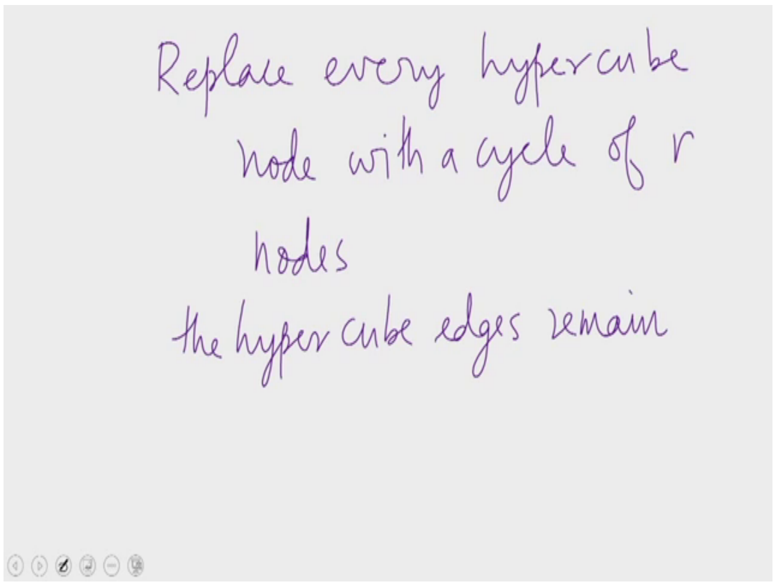


Cube Connected cycle

Another related network is a cube connected cycle. A cube connected cycle is obtained from a hypercube in this fashion. Let us take the case of a 3 dimensional hypercube, but the 3 dimensional hypercube has 8 nodes, as you know. To convert a 3 dimensional hyper tube into a 3 dimensional CCC, what we do is this.

We remove every single vertex of the hypercube, every node of the hypercube is removed and then these nodes are replaced by cycles. At each of these frayed ends we introduce a vertex. There is a missing edge here, and all these newly introduced vertices which are blue in color are connected using cycles. What we get is a 3 dimensional cube connected circle.

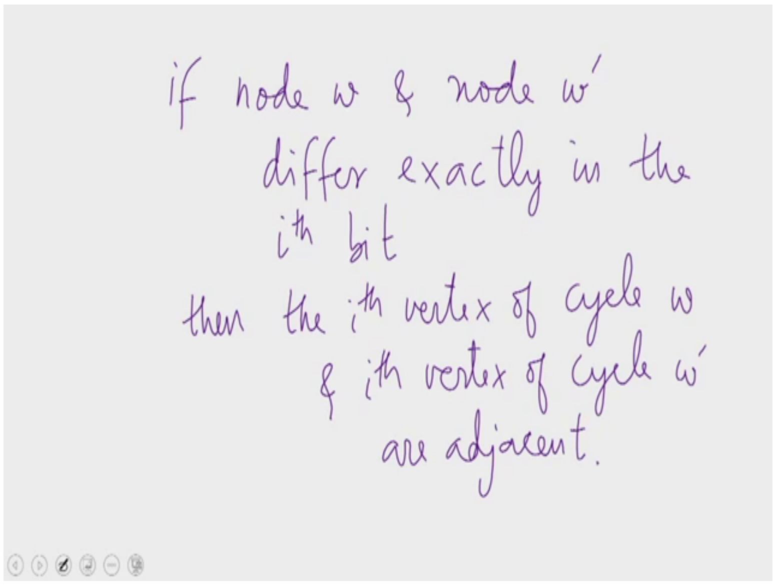
(Refer Slide Time: 10:01)


Replace every hypercube node with a cycle of r nodes

the hypercube edges remain

So, in general for an r dimensional case what we do is this, replace every hypercube node with a cycle of r nodes that is each hyper node hypercube node is replace with the cycle of size r, the hypercube edges remain, but now these edges are between vertices of the cycles.

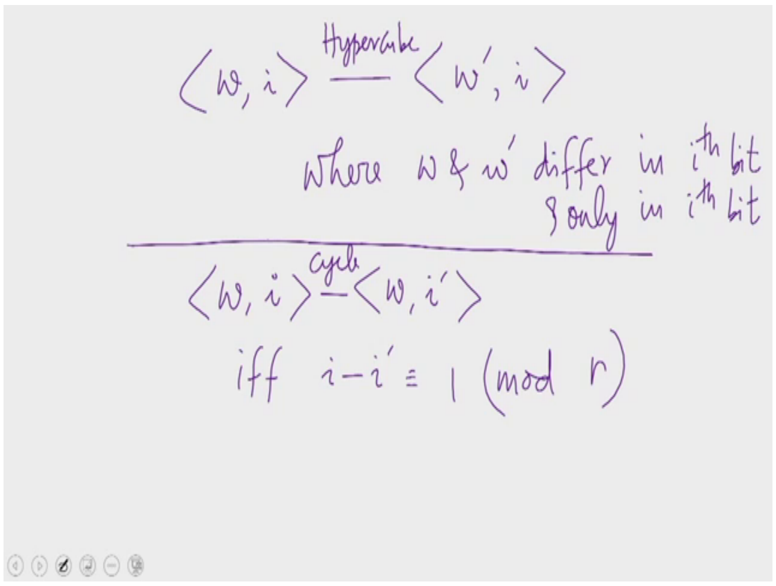
(Refer Slide Time: 10:51)


if node w & node w' differ exactly in the i^{th} bit

then the i^{th} vertex of cycle w & i^{th} vertex of cycle w' are adjacent.

In particular, if node w and node w prime differ exactly in the ith bit. Then the ith vertex of cycle w; w has been replaced with a cycle. The cycle also we would call w. So, the ith vertex of cycle w and the ith vertex of a cycle w prime are adjacent.

(Refer Slide Time: 11:50)



$$\langle w, i \rangle \overset{\text{Hypercube}}{—} \langle w', i \rangle$$

where w & w' differ in i^{th} bit & only in i^{th} bit

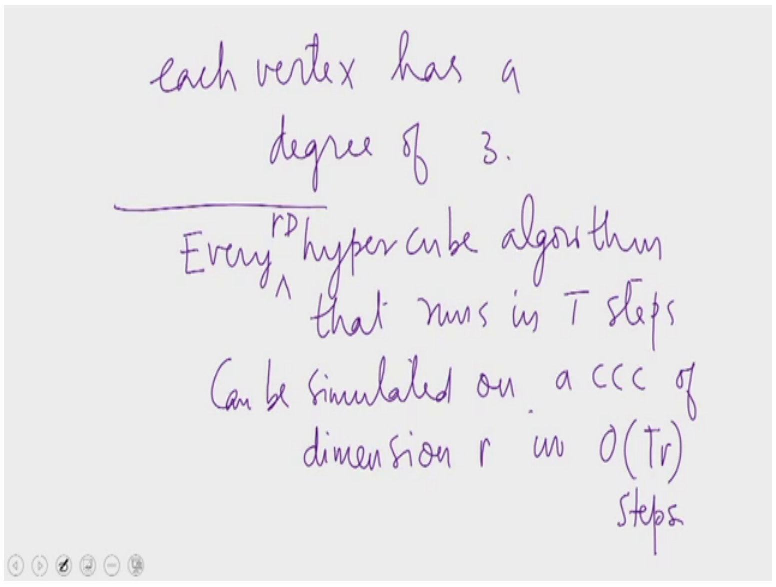
$$\langle w, i \rangle \overset{\text{cycle}}{—} \langle w, i' \rangle$$

iff $i - i' \equiv 1 \pmod{r}$

In other words, there is an edge from wi to w prime i where w and w prime, differ in the ith bit. And only in the ith bit, this edges the hypercube edge.

So, the cube connected cycle will have 2 kinds of edges, there are the hypercube edges and there are the cycle edges. The hypercube edges are of the sort, then the cycle edges are like this; 2 vertices of the same cycle wi and wi prime are adjacent if and only if i minus i prime equal to 1 mode r, that is either i equal to i plus 1 mod r or i prime equal to i plus ir, i prime equal to i plus 1 mod r. These edges are called the cycle edges. So, consequently we find that each vertex has a degree of 3.

(Refer Slide Time: 13:24)



each vertex has a degree of 3.

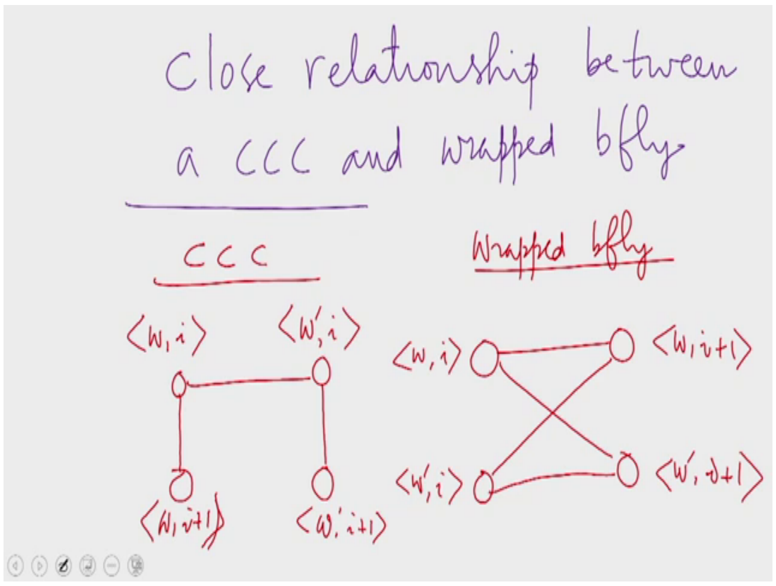
Every rD hypercube algorithm that runs in T steps can be simulated on a CCC of dimension r in $O(Tr)$ steps.

We increase the number of vertices extrinsically over the number of vertices in the hypercube, but then now the guarantee that we have is that the every vertex has a degree of 3.

So, you can readily see that every hypercube algorithm, the transcend T steps can be simulated, on a CCC of the same dimensions. So, let us say we are considering a hypercube of dimension r. So, every rD hypercube algorithm, the transcend T steps can be simulated on a CCC of dimension r, in order of T r steps. That is because every single node of the hypercube has been replaced by a cycle of size r.

So, now, the information which is within 1 node of the hypercube has to be broadcast over a cycle of length r. Therefore, the single step of the hypercube algorithm can be simulated in order of our steps on the CCC. Therefore, an algorithm which runs in T steps can be simulated in order of Tr steps.

(Refer Slide Time: 15:15)



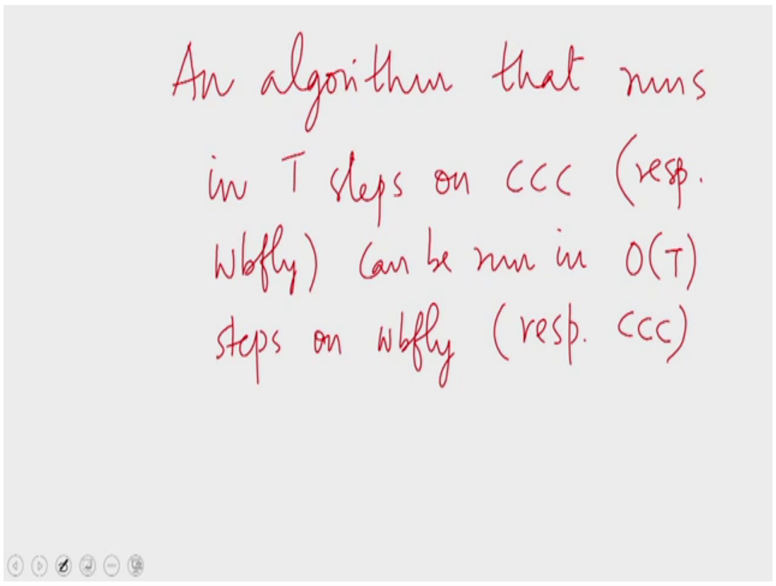
But then surprisingly, there is a close relationship between a CCC and a wrap butterfly. Looking at the diagram of the 2 networks it they would not appear similar, but then if you look closely you find that they are essentially the same. In a CCC as well as a butterfly a hypercube node is replaced by a cycle. Every row of a butterfly corresponds to node of the hypercube. From a butterfly we go to a wrap butterfly by pasting the row end to end. So, the row becomes 1 cycle here.

So, corresponding to every hypercube node, now we have a cycle the same as the case with the CCC. The only difference is that the hypercube edges in CCC do not correspond exactly to the cross edges of a butterfly, but still these 2 networks can simulate each other very easily. For example, in a CCC as we have seen, a node wi and a node w prime i, where w and w prime differ in exactly the ith bit are adjacent and w prime i is adjacent to w prime i plus 1 in particular and wi is adjacent to wi plus 1, through cycle edges.

So, when you consider these 4 nodes. Wi, w prime i, wi plus 1, w prime i plus 1. The interconnection amongst the 4 is in this fashion. Whereas, in a wrap butterfly, wi and wi plus 1 our vertices of the same row similarly w prime i and w prime i plus 1 are also in the same row, but the connection between them is of the sort.

So, what we find is that the locality that is defined by Wi, w prime i, wi plus 1 and w prime i plus 1 can be transposed to into a locality of the sort. If you do that for every possible 4 tuple of vertices of the sort, then a CCC converts into wrap butterfly or in other words.

(Refer Slide Time: 18:24)

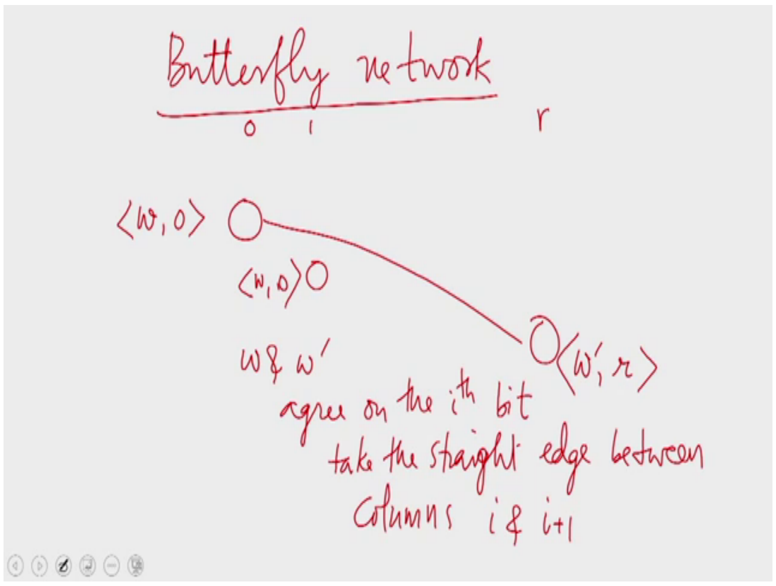


An algorithm that runs in T steps on CCC (resp. wbfly) can be run in O(T) steps on wbfly (resp. CCC)

An algorithm that runs in T steps on CCC or respectively wrap butterfly can be run in asymptotically the same amount of time. On the other model, that is an algorithm on CCC can be simulated on wrap butterfly and an algorithm on a wrap butterfly can be simulated on CCC. So, these 2 models are capable of simulating each other at an order 1 factor.

So, these 2 models are very close to each other indeed.

(Refer Slide Time: 19:20)



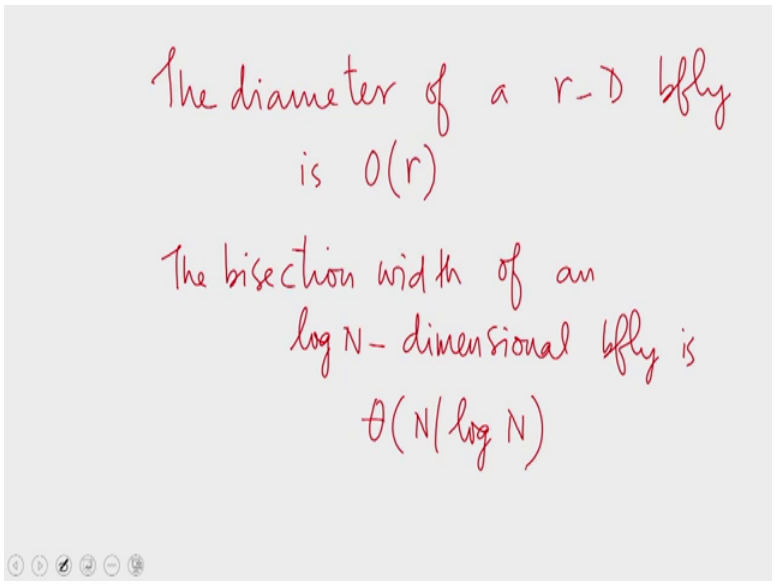
A butterfly network has some other nice properties too. Coming back to the butterfly networks, consider a node belonging to the 0th column. So, we denote this as w 0. This node belongs to the wth row and the 0th column let us say and let us consider a node in the rth column, let us denote it as w prime r.

So, we are picking the wth node of the 0th column and the w prime th node of the rth column. The butterfly network provides a unique path from this node to that node that is the network allows you to go from w 0 to w prime r, for any choice of w and w prime using a unique path. How do you find this path? If w and w prime agree on the ith bit, then take the straight edge between columns i and i plus 1, if they do not agree on the ith bit, then take the cross edge.

So, that is all the algorithm to find the unique path that connects w 0 to w prime r for any choice of w and w prime you start with w 0, if w and w prime agree on the 0th bit. Then take the straight edge if they do not agree then take the cross edge. Now you are in the first column and the node at which at which you are suppose that is w 1 0, w 1 0 has the property that w 1 agrees with w prime in the 0th bit and it agrees with w on every other bit. Now what we have done is to fix the first bit of W, by taking either the straight edge if w and w prime agree on the 0th bit and taking the cross edge, if w and w prime differ on the 0th bit what we have done is to the reach and no w 1, so that w 1 agrees with w on the 0th bit and agrees with w prime on every other bit which means, now we are morphing w into w prime and have gone through the first step of the morphing.

So, if you continue this process of taking straight edges or cross edges accordingly we will end up reaching w prime, that is w will morph into w prime by the time we come to the rth column, we would have reached w prime r. So, the algorithm is this take a straight edge, if the ith bit of w and w prime agree when you are between columns i and i plus 1 or take the cross edge. So, this is one nice property of the butterfly network.

(Refer Slide Time: 22:48)

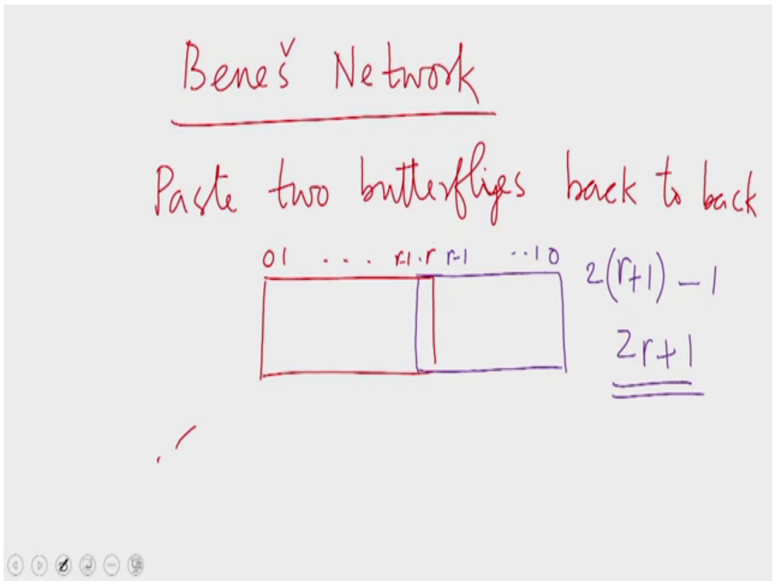

The diameter of a r-D bfly is $O(r)$

The bisection width of an $\log N$- dimensional bfly is $\theta(N/\log N)$

What this establishes is that the diameter of a butterfly an r dimensional butterfly is order of r. Since, you can go from any node in the first 0th column to any node in the rth column in r steps. The maximum distance between any 2 nodes in the network is order of r. For example, if you take 2 nodes belonging to the 0th column to go from 1 node to the other even though there is no direct edge between them you can start from the 0th column grow to an arbitrary node in the rth column and then come back to the destination node in the 0th column.

So, the total distance taken would be 2 r. So, the diameter of an r dimensional butterfly is order r. Then the bisection width of a log in dimensional butterfly is theta of n by log n, we will not prove this here, and stating it without proof, but what it establishes is that a butterfly network has a small diameter and a large bisection with. Both of which are desirable properties, these are quite similar to a hypercube. So, a butterfly network as well as a cube connected cycle possess, many of the nice properties of a hypercube architecture, but they have the additional advantage that the vertex degree of every vertex is a constant.

(Refer Slide Time: 24:42)



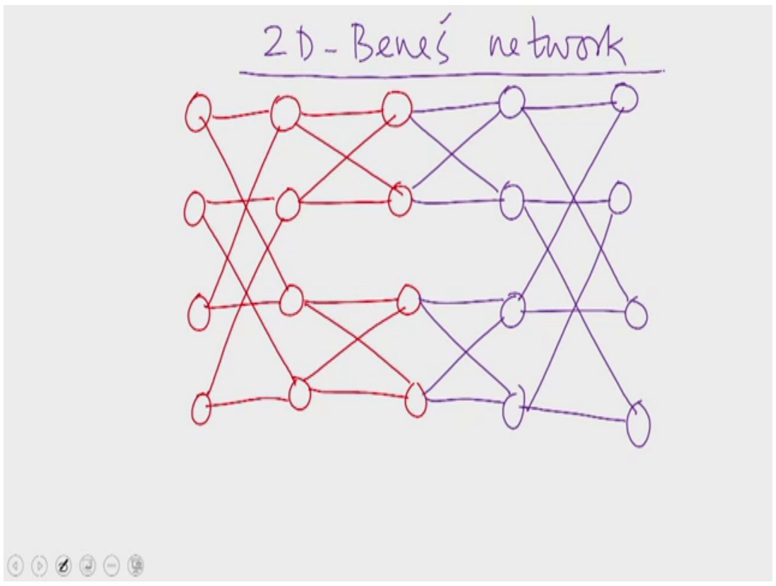
Beneš Network

Paste two butterflies back to back

In the case of a cube connected cycle that is 3, in the case of a wrap butterfly or a butterfly it is 4. Now let us see a network which is obtained from a butterfly and has some nice reconfigurable properties this is called a Benes network. A banes network is obtained by pasting two butterflies back to back. What we do is this, we take 1 copy of a butterfly with r columns and we take another copy of a butterfly of the same dimensionality with again r plus 1 columns and fuse the rth columns of the 2 butterflies together. So, on the whole the total number of columns here would be 2 r plus 1.

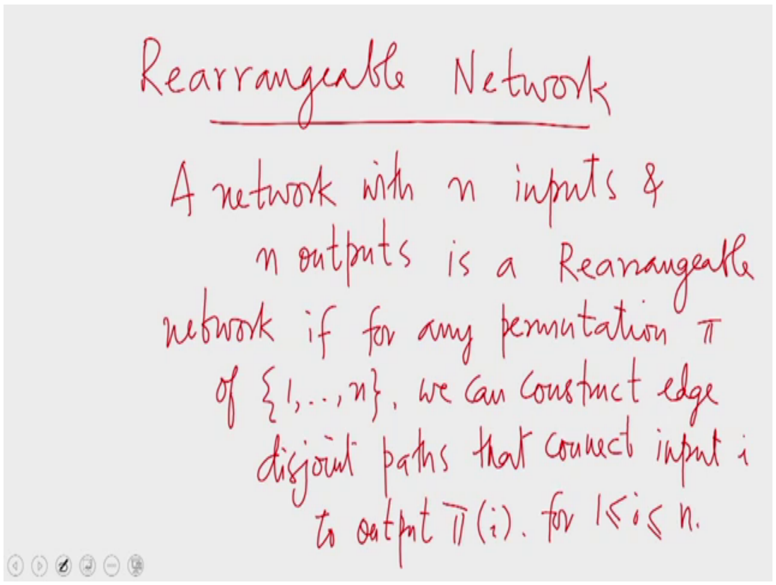
So, let us see how a Benes network would look like.

(Refer Slide Time: 26:10)



In the case of a 2 dimensional Benes network, we take 2 copies of 2 dimensional butterflies. This is 1, 2 dimensional butterfly. Then you take another 2 dimensional butterfly, but make sure that the last columns of the 2 are fused together. What we get is a 2 dimensional Benes network. So, in general an r dimensional Benes network is obtained by pasting to r dimensional butterfly networks back to back,

(Refer Slide Time: 27:26)



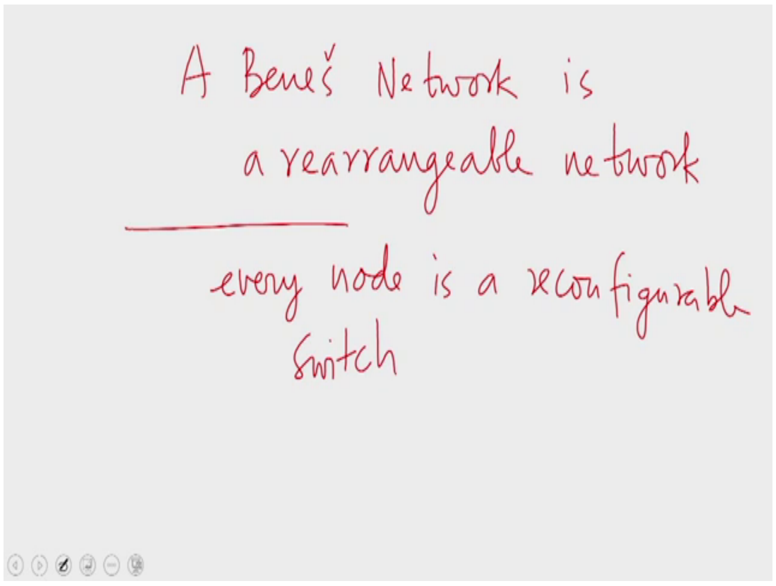
Rearrangeable Network

A network with n inputs & n outputs is a Rearrangeable network if for any permutation π of $\{1,..,n\}$, we can construct edge disjoint paths that connect input i to output $\pi(i)$. for $1 \leq i \leq n$.

By fusing the rth column of the 2 together we get an r dimensional Benes network a banished network has certain nice properties, 1 being that it is a rearrangeable network. A network with the n inputs and n outputs is a rearrangeable network. If for any permutation pi of the set 1 to n, we can construct edge disjoint paths, that connect input i to output pi i for every i. 1 less than or equal to i less than or equal to n.

So, if this can be done for every single pi, then we say that a network is a rearrangeable network. In other words it is capable of any 1 to 1 mapping of the inputs to the outputs. So, what we claim is that a Benes network is a rearrangeable network.

(Refer Slide Time: 29:16)

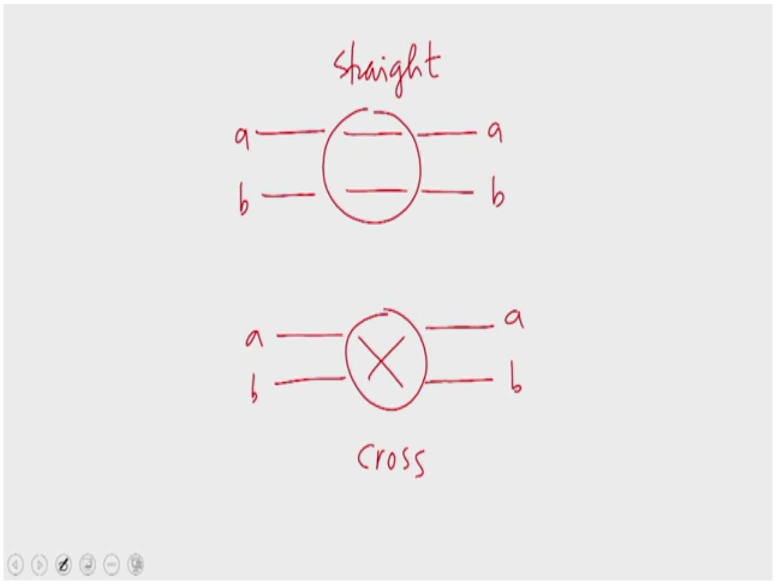


A Beneš Network is a rearrangeable network

every node is a reconfigurable switch

So, let us now try to prove this. Let us take a Benes network and ensure that every node is a reconfigurable switch, what I mean is this.

(Refer Slide Time: 30:03)

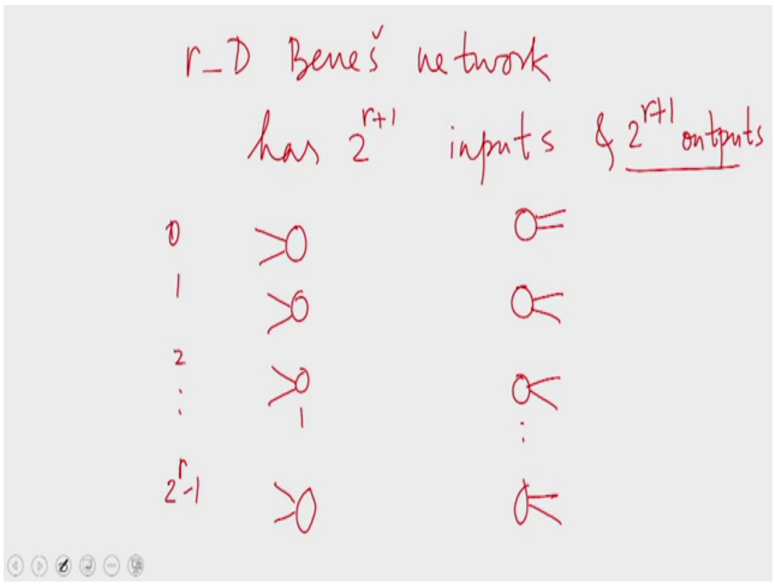


Consider a node of the Benes network. Assume that this node has 2 inputs and 2 outputs the node has 2 inputs and 2 outputs. Let us say the node has 2 possible configurations, it could have a straight configuration. In a straight configuration input a is connected to output a and input b is connected to output b.

So, what is in input a will appear on output a and what is on input b will appear on output b, as opposed to this there is the cross connection. In a cross connection input a will appear on output b and input b will appear on output a.

So, let us say we replace every node of a Benes network, using reconfigurable switches of the sort. That can take on either a straight configuration or a cross configuration. Then an r dimensional Benes network, has 2 r inputs that is because when you consider the 0th row of this network.

(Refer Slide Time: 31:08)

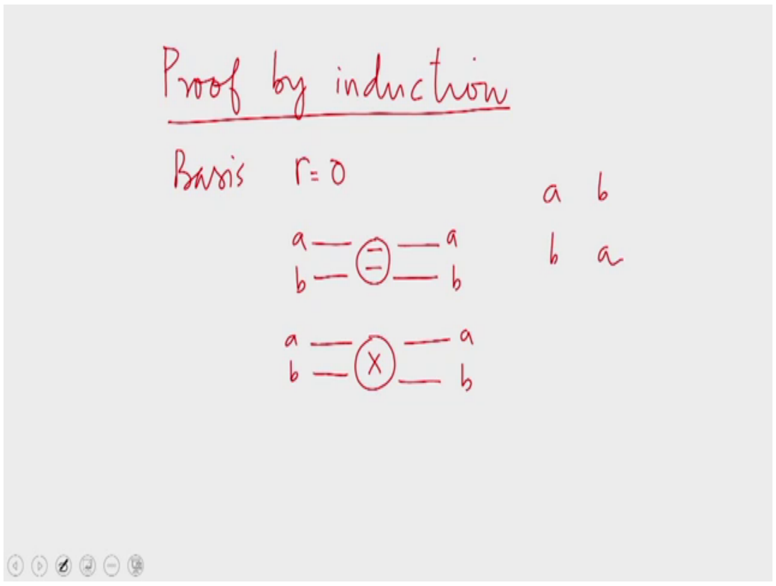


r_D Beneš network

has 2^{r+1} inputs & 2^{r+1} outputs

. There are 2 power r nodes here each 1 will have 2 inputs, sorry this is 2 power and r dimensional Benes network has 2 power r plus 1 inputs. Similarly the rth column, column number r will have to outputs each. So, this is a Benes network of 2 power r plus 1 inputs and 2 power r plus 1 outputs. What we want to claim is that, this is a rearrangeable network. The proof is by induction.

(Refer Slide Time: 32:18)



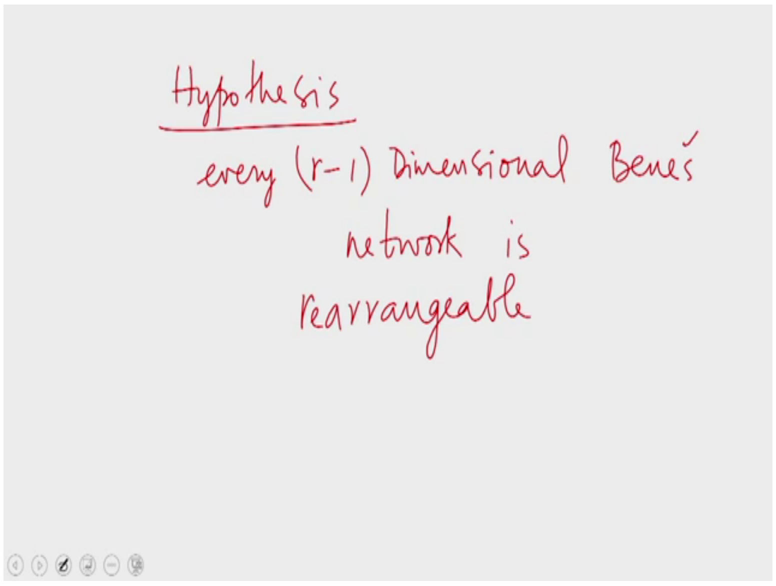
Proof by induction

Basis r=0

The basis is where r equal to 1 or where r equal to 0, when r equal to 0 we have exactly 1 node in the Benes network and there are 2 inputs and there are 2 outputs. This is indeed a rearrangeable network, if you use a straight connection, you realize the permutation a b that is input a will be connected to output a and input b will be connected to output b. If you use the cross configuration, then we will be realizing the permutation b a, when you have only 2 inputs there are only 2 permutations possible.

So, when r equal to 0, every possible permutation is realizable by reconfiguring the nodes of the network. So, there is an edge disjoint path that between every pair of inputs and outputs for any permutation that we choose therefore, the network is rearrangeable, when r equal to 0. So, this forms the basis.

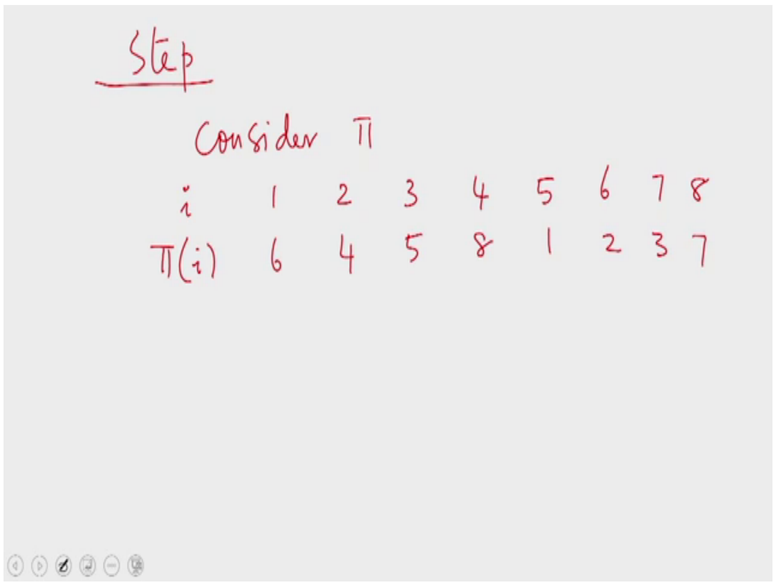
(Refer Slide Time: 33:41)


Hypothesis

every $(r-1)$ Dimensional Beneš network is rearrangeable

Now, the hypothesis assumes that every r minus 1 dimensional Benes network is rearrangeable. So, this is r hypothesis.

(Refer Slide Time: 34:16)



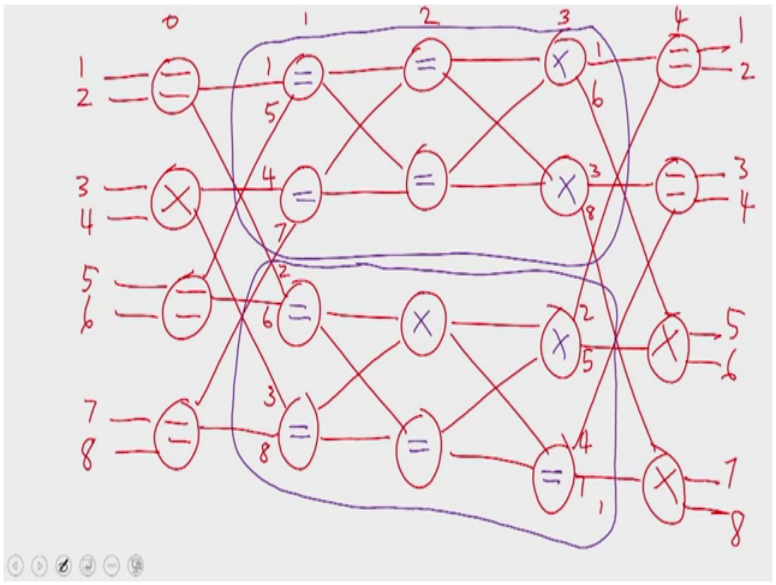
Step

Consider π

i	1	2	3	4	5	6	7	8
$\pi(i)$	6	4	5	8	1	2	3	7

Now, coming to the induction step, let me demonstrate the induction step using an example first and then we will generalize the example. Let us say we want to realize one particular pi, pi is defined like this. Let us say this is the permutation that we want to realize in this permutation input 1 is to be connected to output 6, input 2 is to be connected to output 4 and so on.

(Refer Slide Time: 35:02)



To realize this permutation, what we do is this we consider the Benes network of dimension 2. Let us consider the connection between the first column and the second column. Similarly we have and exactly analog as connections between the last 2 columns. And then every node in the last column has 2 outputs and every node in the first column has 2 inputs. Let us number the inputs in this fashion 1, 2, 3, 4, 5, 6, 7 and 8 and the outputs are also numbered like this. Now let us see because let me draw the remaining edges as well.

So, you can see here that when we consider columns, 1, 2 and 3, the columns is numbered starting from 0. When you consider columns 1, 2 and 3, the edges are all either in the upper half or in the lower half. This has certain significance the certain significance. Now look at the permutation once again, input 1 has to be connected to output 6.

So, let us arbitrarily choose a straight configuration for this one switch. The switch to which input 1 is connected. So, this ensures that input 1 is transmitted to on this line and comes to the first node of the first column, input 1 is to be connected to the 6th output. Now input 1 is going to the upper half of the network, that is columns 1, 2 and 3 have an upper half and a lower half, every edges in the in the network is either in the upper half or in the lower half.

Now you find that input 1 is going into the upper half therefore, it will have to come out of the upper half. That is when it comes out of column 3, it is coming out of the upper half and it has to go to node 6. Therefore, node 6 will have to receive a value from the upper half, in other words node 6 will be able to receive a value from the upper half only if the switch which is connected to output 6 is not cross configuration.

So, 6 is in now in cross configuration, which means output 6 will have to be received from here. What it means is that output 5 in turn will have to be received from here. Output 5 is coming from the lower half. Now let us look at the permutation once again. Output 5 is input 3; input 3 has to appear on output 5.

So, output five is to be received from the lower half, which means input 3 has to be sent to the lower half. Which means this switch will have to be in cross configuration. Then that forces, so we have 5 coming here that says that 3 will have to be coming from it should be going into the lower half that ensures that 4 will be going into the upper half we will have 4 coming on this line at this point.

So, 4 is going input 4 is going into the upper half from our permutation we say that see that input 4 is to be connected to output 8. So, output 8 has to be coming from in has to be connected to input 4, which is going into the upper half. Therefore, this which will also have to be in cross configuration, it will be appearing here. Which means that 7 will be going to the lower half and 7 will be appearing here, since 7 is appearing here output 7 is connected to input 3. So, input 3 output 7 is connected to input 8 sorry, output 7 is connected to input 3, output 7 is coming to the lower half which means input 8 will have to go into the lower half. This which will have to be in a straight configuration, so 8 appears here and 7 appears here.

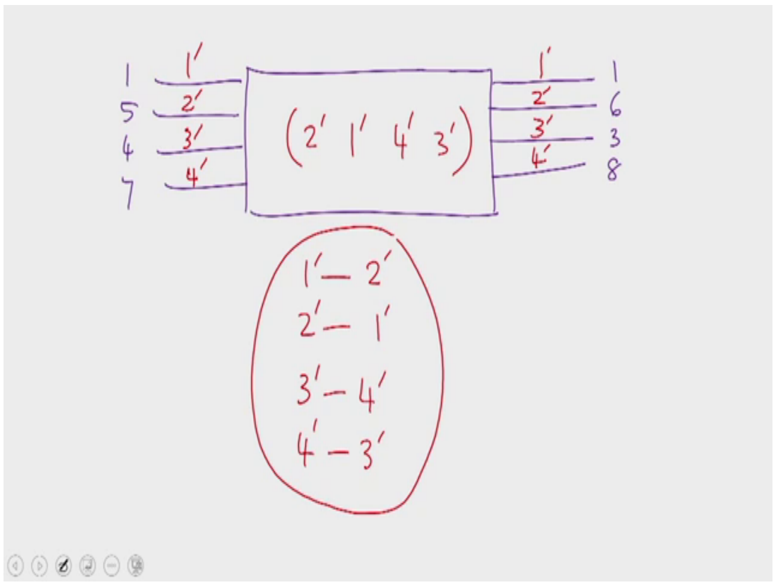
So, 7 is going into the upper half and input 7 is to be connected to output 3. So, output 3 has to receive its value from the upper half which means this which will have to be in the straight configuration. So, 3 will be coming in from here. Therefore, 4 will be going into the lower half we will have 4 appearing here. Now where does output 4 comes from, output 4 is coming from input 8. Sorry output 4 is coming from input 2. So, now we are looping back, input 2 has to be therefore, connected to the lower half. Which is indeed the case, input 2 is connected to the lower half by choosing to send 1 to the upper half, we had decided to send 2 to the lower half and that is consistent with what we have computed now we have looped back.

So, every switch in the first column in the 0th column and the 4th are not configured yet, but we have configured some of them and we have looped. Then what we need to do is to start with one of the remaining nodes. Let us consider then node 2 which inputs 5 and 6 are connected, if you choose to configure them in street then, we will have 5 appearing here and 6 appearing in the lower half, 6 will go here.

So, 5 is going to the upper half, input 5 is connected to output 1. So, output 1 will have to come from the upper half. Therefore, this switch will have to be straight. Which means 2 is going into the lower half and whereas, output to come from output 2 is to come from input 6.

So, 2 is going to the lower half therefore, input 6 also will have to go into the lower half, which is consistent with what we have already marked. Therefore, now we have entered up configuring all the nodes of the 0th column as well as the 4th column. Now let us focus on the upper and lower halves of the Benes network. This is the upper half and this is the lower half. So, in the upper half and the lower half, what we need to do is to permute 1, 5, 4, 7 inputs 1, 5, 4, 7 into 1, 6, 3, 8.

(Refer Slide Time: 42:38)



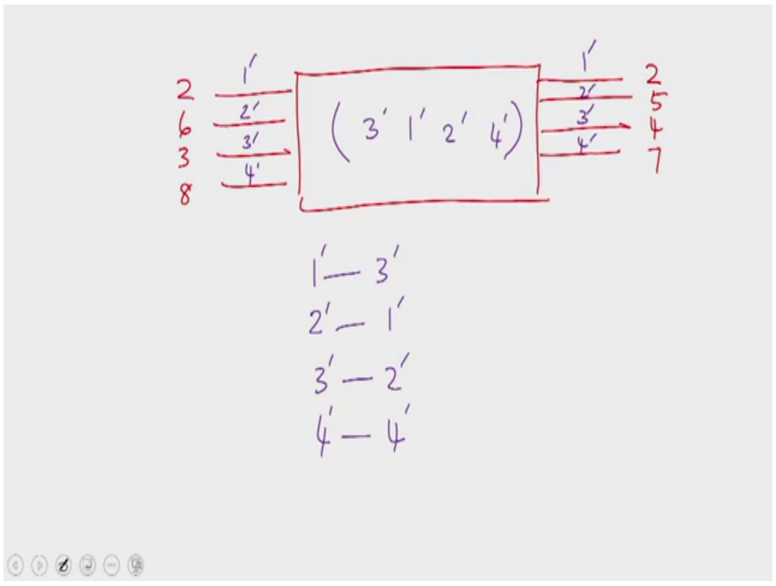
In particular; if I consider the upper half, here I have inputs 1, 5, 4, 7 coming in and I have outputs 1, 6, 3, 8 coming in this order. Now let me rename these inputs and the outputs. I rename 1 as 1 prime, 5 as 2 prime, 4 as 3 prime and 7 as 4 prime. And here output 1 is renamed as output 1 prime, output 6 as renamed as output 2 prime, output 3 is renamed as output 3 prime and output 8 is renamed as output 4 prime. So, now let us consider the inputs 1 prime, 2 prime, 3 prime, and 4 prime and 1 prime, 2 prime, 3 prime and 4 prime on the output side, how do you need to map the inputs to the outputs.

So, when you look at the permutation we find that, 1 prime has to be mapped to 2 prime. That is because 1 has to be input, 1 has to be mapped to output 6. Similarly 2 prime has to be mapped the 1 prime, that is because input 5 has to be mapped to output 1, input 5 has to be mapped to output 1. Similarly 3 prime has to be mapped to 4 prime and 4 prime has to be mapped to 3 prime. That is because input 4 has to be mapped to 8 and input 7 has to be mapped to 3.

So, this is the permutation that we have to realize, which is 2 prime, 1 prime, 4 prime and 3 prime. So, on the upper half of this network we have to realize the permutation; we have shown here 2 prime, 1 prime, 4 prime and 3 prime, but by are in depth induction hypothesis. It is indeed possible we know how to configure these switches.

So, that the required permutation is realized.

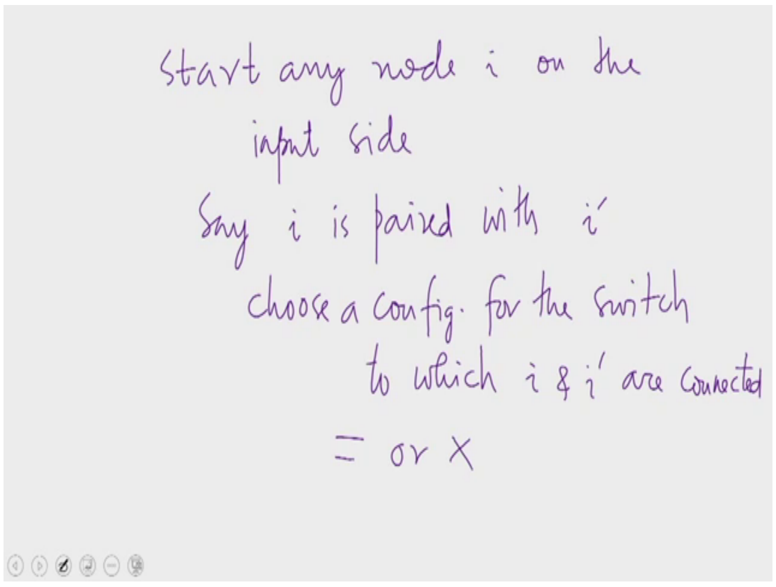
(Refer Slide Time: 44:37)



Similarly when we look at the lower half, we find that, the 4 inputs are 2, 6, 3 and 8 and the 4 outputs are 2, 5, 4 and 7. If you rename them as 1 prime, 2 prime, 3 prime and 4 prime on the input side and similarly on the output side, we find that we have input 1 prime to be mapped to 3 prime, input 2 prime to be mapped to 1 prime, 3 prime to be mapped to 2 prime and 4 prime to be mapped to 4 prime. In other words we have to realize the permutation 3 prime, 1 Prime, 2 prime and 4 prime, which can be realized again used by the induction hypothesis. So, now we have 2 recursive invocations, one on the upper half and the lower half. Once these 2 are solved we would have reconfigured every single switch in the network.

So, I will give you the solution which you can verify, this is one solution that will work, that ensures that every input is connected to every output.

(Refer Slide Time: 46:13)



Start any node i on the input side

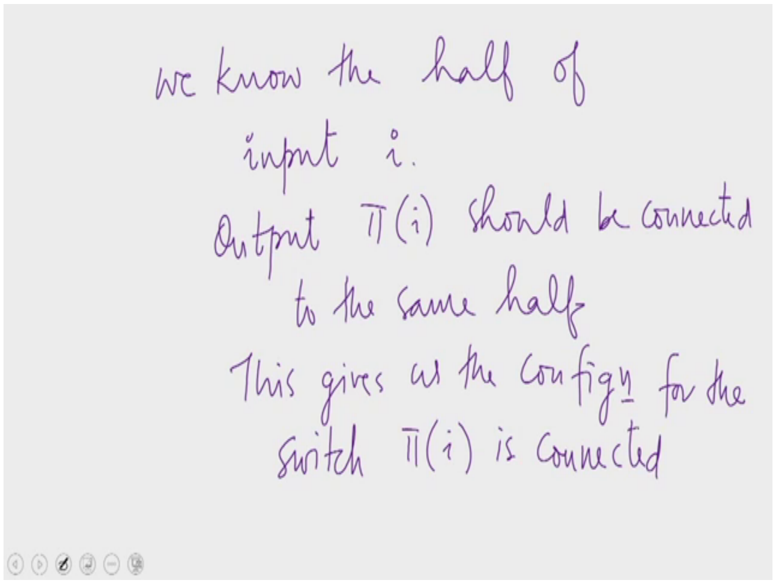
Say i is paired with i′

choose a config. for the switch to which i & i′ are connected

= or X

So, now in general The induction step to feet's in this manner, start at any node on the input side, say i is paired with i prime, i prime would be either i minus 1 or i plus 1. Choose a configuration for the switch to which i and i prime are connected, in our example the first i that we chose was 1 and i prime was two. So, we chose a configuration for the switch to each 1 and inputs 1 and 2 were connected. So, the configuration chosen could be either straight or cross.

(Refer Slide Time: 47:21)



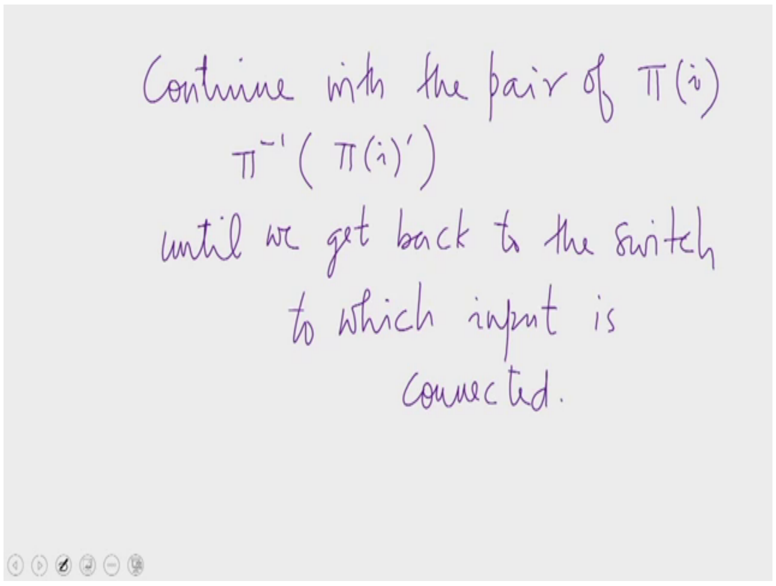
We know the half of input i.

Output $\pi(i)$ should be connected to the same half

This gives us the confign for the switch $\pi(i)$ is connected

So, depending on the choice, we know the half of input i, output pi, i should be connected to the same half. This gives us the configuration for the switch to which pi i is connected.

(Refer Slide Time: 48:27)



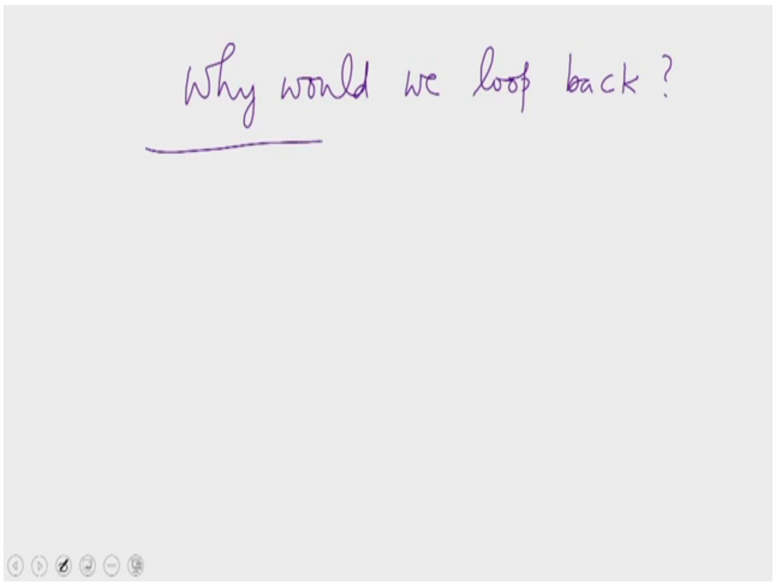
Continue with the pair of $\pi(i)$

$\pi^{-1}(\pi(i)')$

until we get back to the switch to which input is connected.

Now we continue with the pair of pi I, in our example pi i was 6. So, the pair of pi i was 5 and then we have pi inverse of pi i prime, now pi i prime is the pair of pi i that is the input which should be connected to pi i prime that will ensure the configuration for the switch to which that is connected and so on. We continue until we loop, until we get back to the switch to which i is connected input i is connected. Why would i loop back to i, I would leave that for you to answer think of bipartite graphs and even cycles.

(Refer Slide Time: 49:32)



Why would we loop back?

So, we would indeed loop back consistently, but then when we loop back it may not be guaranteed that every switch on in the 0th column as well as the rth column are configured. If every switch is not configured we will begin with any switch that is not configured yet and pick an input which is connected to that switch and continue. We continue until every switch in the 0th column and the rth column are configured, at this point we have 2 sub problems of dimension or minus 1, by induction hypothesis the dimension r minus 1 problem can be solved. Because an r minus 1 dimensional Benes network is rearrangeable, this establishes that even an r dimensional Benes network is rearrangeable. That completes the induction.

So, let us we will see some more nice properties of Benes networks in the next class hope to see you in the next class.

Thank you.