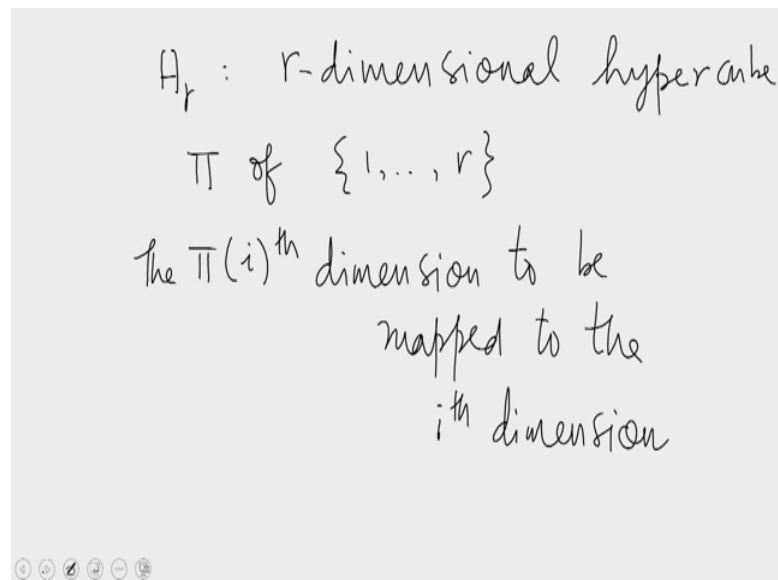


Parallel Algorithms
Prof. Sajith Gopalan
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati

Lecture – 30
Hypercube cont'd, butterfly network

Welcome to the 13th lecture of the MOOC on Parallel Algorithms. In the previous lecture, we were looking at some symmetry properties of a Hypercube. In particular we want to find an automorphism of a hypercube which will permute the dimensions the way we want and map a particular node to a chosen node.

(Refer Slide Time: 00:53)



In particular what we want is this, let us say we have a hypercube H_r which is an r dimensional hyper cube and let us say, we are given a permutation π of a the set 1 to r which are the dimension set. So, what we want is to realize the permutations? In other words, we want the $\pi(i)$ th dimension to be mapped to the i th dimension.

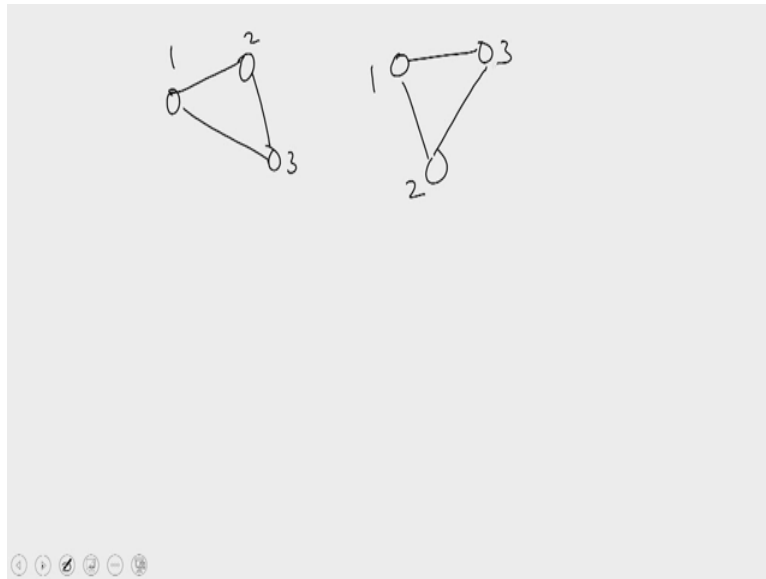
(Refer Slide Time: 01:53)

$$\begin{array}{l} u \longrightarrow u' \\ \hline \text{automorphism } \sigma \text{ of } G=(V, E) \\ \text{is } \sigma: V \rightarrow V \text{ so that} \\ \sigma \text{ is 1-1} \\ \{\sigma(u), \sigma(v)\} \in E \text{ for every} \\ \{u, v\} \in E \end{array}$$

And we want in particular that vertex u must be mapped to vertex u prime. So, we want an automorphism of the hypercube which will satisfy these conditions. An automorphism σ of a graph is a mapping from V to V the set of vertices to the set of vertices, it is a 1 to 1 mapping. And so that $\sigma(u) \sigma(v)$ belongs to the edge set of the graph for every u, v belonging to E . In other words an automorphism is a renaming of the vertices of the graph so that.

After the renaming after to the renaming the graph is exactly similar to the original graph it is exactly the same as the original graph. In spite of the renaming we get an exact replica of the original graph. If this is the case then we say that, σ is an automorphism clearly, every graph is not with an automorphism, only some graphs have an automorphism as an example that we saw in the last class.

(Refer Slide Time: 03:27)



A 3 clique in particular it is an automorphism; however, you renumber the vertices the graph remains the same; so, this is an automorphism some of the 3 clique. Similarly, what we want is an automorphism of the hypercube which will map a particular node u to u prime. So, let us say the vertex u has the binary representation, u is a node of an r dimensional hypercube therefore, you will have r bits to it.

(Refer Slide Time: 03:53)

$$\begin{array}{l} U = u_1 u_2 \dots u_r \\ U' = u'_1 u'_2 \dots u'_r \end{array} \left. \vphantom{\begin{array}{l} U \\ U' \end{array}} \right\} \begin{array}{l} \text{realise} \\ \Pi \end{array}$$

Two vertices should be adjacent
iff they differ in exactly
1 bit as per the new name

So, let u one be the most significant bit and u_r be the least significant bit. Similarly, let us say u prime is u prime 1 through u prime r . What we want is that? U must be mapped

to u prime and similarly along with that permutation π has to be realized and the resultant renaming should again be the naming of a hypercube. What it means is that? After the renaming two vertices should be adjacent precisely when they differ in exactly 1 bit.

As per the new name of course, this property is satisfied according to the old names because what we have is an r dimensional hyper cube with the conventional naming for the vertices. Now, what we want is to rename the vertices so, that even after the renaming this property will be satisfied there is two vertices are adjacent if and only if their new names differ by exactly 1 bit.

(Refer Slide Time: 05:38)

$$\begin{aligned} \sigma(x_1 \dots x_r) &= x_{\pi(1)} \oplus u_{\pi(1)} \oplus u'_1 \Big| \\ &\vdots \\ &x_{\pi(r)} \oplus u_{\pi(r)} \oplus u'_r \\ \hline u, u' \quad u &= u_1 \dots u_r \\ \sigma(u) &= \underbrace{u_{\pi(1)} \oplus u_{\pi(1)}}_0 \oplus \underbrace{u_{\pi(1)} \oplus u'_1}_{u'_1} \Big| \dots \dots \\ &\quad \underbrace{u_{\pi(r)} \oplus u_{\pi(r)}}_0 \oplus \underbrace{u_{\pi(r)} \oplus u'_r}_{u'_r} \Big| \dots \dots \end{aligned}$$

So, the naming scheme is this for a vertex named x_1 through x_r the new name is going to be this vertical bar represents a concatenation. Now, we claim that if the vertices are renamed according to the scheme, what we achieve is an automorphism which will permute the dimensions according to π and map u to u prime.

Now, let us see why the claims write. In particular consider two vertices; consider the two vertices u and u prime, u is u_1 through u_r . Let us see where u gets mapped to $\sigma(u)$ of u happens to be $u_{\pi(1)}$ exclusive $u_{\pi(1)}$ exclusive u_1 prime concatenated by bits obtained in a similar manner. Now, look at this expression $u_{\pi(1)}$ exclusive or $u_{\pi(1)}$ is 0 a bit exclusive odd with itself will give us 0; exclusive are being associative we can evaluate this expression anyhow, we can parenthesize them anyhow.

So, when we parenthesize the first two we get a 0, then the expression reduces to 0 exclusive or u 1 prime which is nothing but u 1 prime; so, this reduces to u 1 prime and the second bit reduces to u 2 prime and so on.

(Refer Slide Time: 07:47)

$$\sigma(u) = u' \quad \text{---} \quad \textcircled{1}$$

$$\sigma(x_1 \dots x_r) = \underbrace{x_{\pi(1)} \oplus u_{\pi(1)} \oplus u'_1}_{\dots} \dots$$

$$x_1 \dots x_{\pi(1)} \dots x_r$$

$$x_1 \dots x_{\pi(2)} \dots x_{\pi(1)} \dots x_r$$

Therefore, what we have obtained is that sigma of u is the same as u prime this is of course, 1 condition we wanted, we wanted u to be map to u prime. Now, the second claim was that the dimensions are permuted according to the way we wanted according to pi. So, let us establish that claim in particular for every node x 1 through x r, the new name is going to be the first bit of the new name is going to be this in particular that is x 1 is replaced with this or rather x pi 1 which happens to be the pi first bit of the present name.

So, in x 1 through x r we have x pi 1 somewhere. This bit is getting transposed to the first position in the new name with an appropriate modification. The bit will be transposed using u pi 1 exclusive or u 1 prime depending on whether along this dimension we want to turn the cube inside out or not. Similarly, the x pi 2 bit will appear at the second position which means the pi 2 dimension is going to the second dimension and so on.

So, what we find is that as far as the naming is concerned? The pi ith bit of the present name is going to be transposed into the ith bit of the new name. Of course, the bit will

undergo a transposition based on $u_{\pi(i)}$ or u_i . So, in the naming of course, the dimensions are being permuted exactly the way we want.

(Refer Slide Time: 09:55)

a $\pi(i)^{\text{th}}$ dimensional edge
of H_r

$$x = x_1 \dots x_{\pi(i)} \dots x_r$$

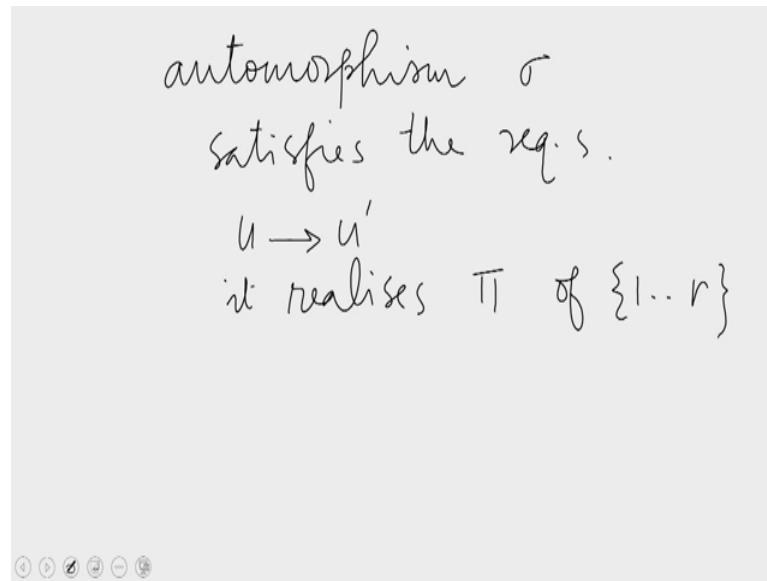
$$x' = x_1 \dots \overline{x_{\pi(i)}} \dots x_r$$

$$\sigma(x) = \underbrace{x_{\pi(i)} \oplus u_{\pi(i)} \oplus u_i}_{=} \dots \left| x_{\pi(i)} \oplus u_{\pi(i)} \oplus u_i \right|$$

$$\sigma(x') = \underbrace{\overline{x_{\pi(i)}} \oplus u_{\pi(i)} \oplus u_i}_{=} \dots \left| \overline{x_{\pi(i)}} \oplus u_{\pi(i)} \oplus u_i \right|$$

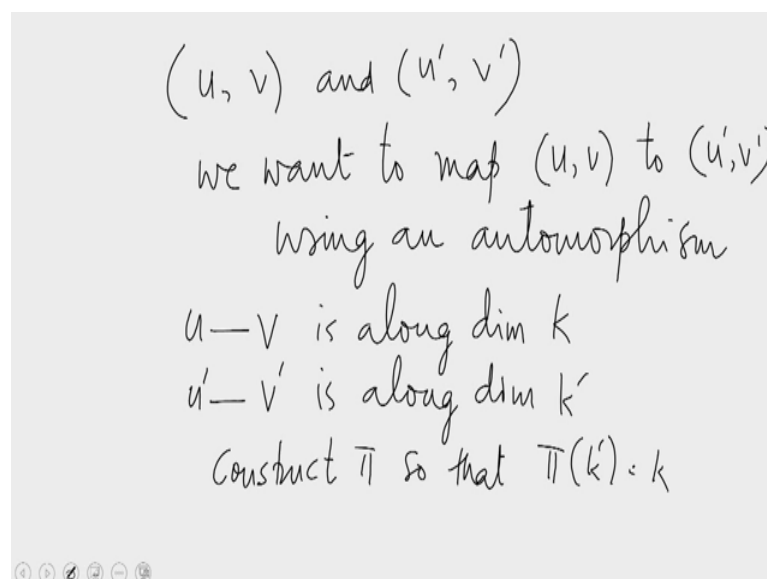
Now, let us consider an $\pi(i)$ th dimensional edge in particular let us consider the node x and the node x' let x' be identical to x except in the $\pi(i)$ th bit where it is the complement, at every other bit position x and x' agree, then what would $\sigma(x)$ and $\sigma(x')$ be. They will agree on all bit positions except the $\pi(i)$ th one that is because at the first position if $\pi(i)$ is not equal to i . At the first position we have this at the first bit of $\sigma(x)$; which is identical to the first bit of $\sigma(x')$ provided that $\pi(i)$ is not equal to i , but at the $\pi(i)$ th position here we have $x_{\pi(i)}$ whereas, here we have $\overline{x_{\pi(i)}}$ complemented exclusive ORed with the same expression.

(Refer Slide Time: 11:57)



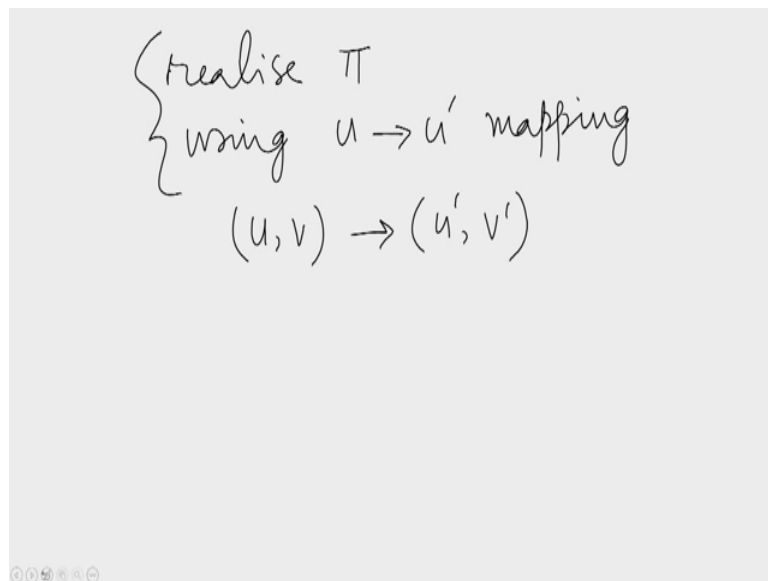
So, you can see that at the i th, but σ of x and σ of x prime differ at every other bit position they are identical. Therefore, the automorphism σ satisfies the requirements namely that it maps u to u prime it realizes the permutation of the dimensions. So, there indeed exists an automorphism which satisfies these requirements. Given any permutation of the dimensions and a pair of vertices u and u prime you can find an automorphism which realizes the permutation that is it permutes the dimensions according to π and also maps u to u prime, that is the new name under σ of u will be u prime.

(Refer Slide Time: 12:52)



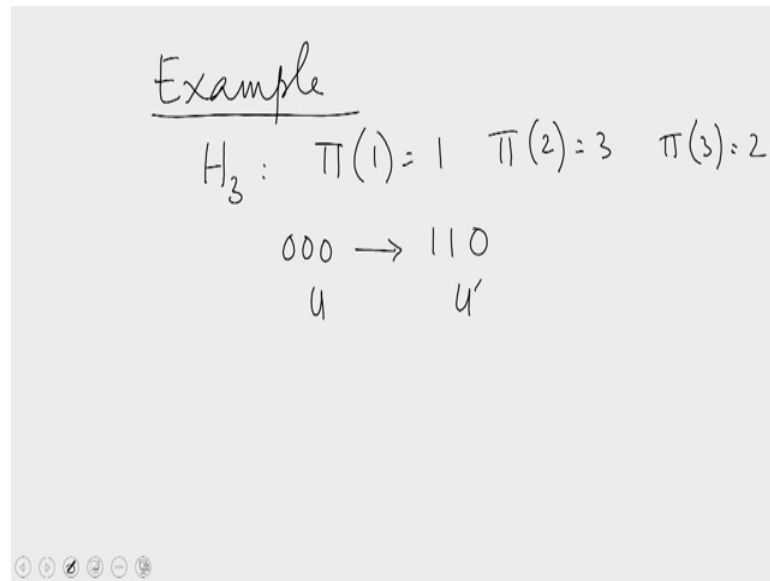
As an extension to this let us say we are given two edges $u v$ and $u' v'$. And let us say we want to map $u v$ to $u' v'$ using an automorphism. Now, this is easily achieved suppose $u v$ is along dimension k that is the binary representations of the numerals u and v differ exactly in the k th bit and only at the k th bit. And let us say $u' v'$ it is along dimension k' , then we can construct a π so that π of k prime equal to k' .

(Refer Slide Time: 14:09)



In that case, if we realize π using the mapping u to u' , we would have achieved what we wanted that is the edge uv will be will have been mapped to $u' v'$, then we will have to edge uv mapping to $u' v'$. So, let me clarify what I have been saying using an example.

(Refer Slide Time: 14:42)



Example

$$H_3: \pi(1) = 1 \quad \pi(2) = 3 \quad \pi(3) = 2$$
$$\begin{array}{ccc} 000 & \rightarrow & 110 \\ u & & u' \end{array}$$

The image shows a whiteboard with handwritten text. At the top, the word "Example" is written and underlined. Below it, the permutation H_3 is defined as $\pi(1) = 1$, $\pi(2) = 3$, and $\pi(3) = 2$. Below the permutation, a mapping is shown: $000 \rightarrow 110$, with u under 000 and u' under 110 . At the bottom left of the whiteboard, there are several small circular icons.

Let us say we have a 3 dimensional hyper cube. In this hypercube let us say we have a permutation π defined like this π_1 equal to 1, π_2 equal to 3 and π_3 equal to 2. That is we want to interchange the dimensions to n_3 , that is dimension 2 should occupy the position of 3 and dimension 3 should occupy the position of 2.

So, we want to turn the hypercube around in this manner and in particular let us say, we want node 0 0 0 to map to node 1 1 0. So, 0 0 0 is u and 1 1 0 is u' . So, given a permutation π of the sort and a pair of vertices u and u' in this manner let us say we want to find an automorphism σ .

(Refer Slide Time: 15:42)

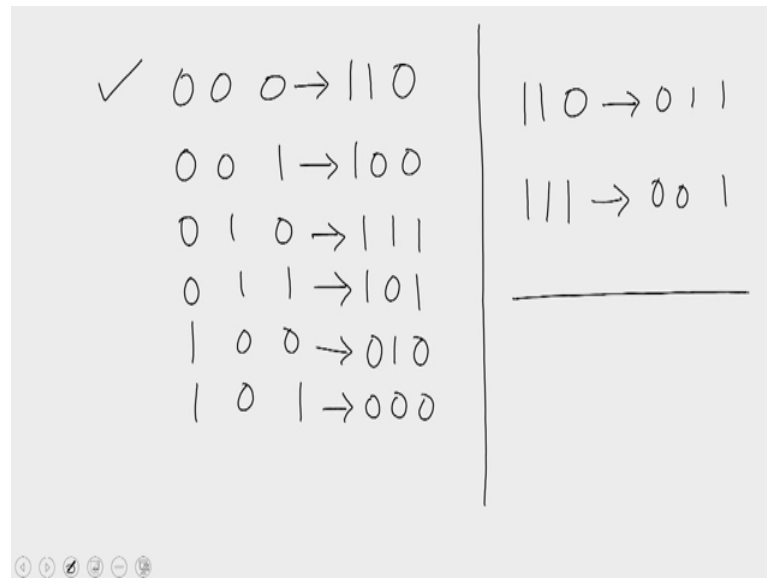
$$\begin{aligned}
 \sigma(x) &= \begin{array}{c} x_1 \quad \oplus \quad 0 \quad \oplus \quad u_1' \\ x_3 \quad \oplus \quad 0 \quad \oplus \quad u_2' \\ x_2 \quad \oplus \quad 0 \quad \oplus \quad u_3' \end{array} \Bigg| \begin{array}{l} u = 000 \\ u' = 110 \end{array} \\
 &= \begin{array}{c} x_1 \oplus \\ x_3 \oplus \\ x_2 \oplus \end{array} \Bigg| \begin{array}{c} | \\ | \\ 0 \end{array} \\
 &= \underline{\underline{\bar{x}_1 \quad \bar{x}_3 \quad x_2}}
 \end{aligned}$$

The automorphism sigma can be obtained like this sigma of x for any vertex labeled x which is x 1, x 2, x 3 the automorphism will have to be x of pi 1 exclusive or u of pi 1; u of pi 1 happens to be u 1 indeed exclusive or u 1 prime. Concatenation x of pi 2 exclusive or u 1, u of pi 1 which is u 3 exclusive or u prime and then x of pi 3 exclusive or u 2 exclusive or u 2, u 3 prime. But of course, we know that pi 1 is 1 and pi 2 is 3 and pi 3 is therefore, we can simplify the expression to be like this.

Now, let us substitute the values of u 1 nu, u 1, u 2 and u 3 and u 1 prime, u 2 prime and u 3 prime. Now, u is 0 0 0, u prime is 1 1 0 therefore, all these u 1, u 2, u 3 will go to 0 therefore, the expression simplifies to 0 here; which means, these can be simplified to u 1 prime, u 2 prime and u 3 prime respectively. So, this becomes x 1 exclusive or u 1 prime; u 1 prime is 1, x 2, x 3 exclusive or u 2 prime which is also 1 and then x 2 exclusive or u 3 prime which is 0.

So, this simplifies to x 1 bar, x 3 bar and x 2. So, this will be the new name of the vertex which is named x now, this will be an automorphism as we argued earlier.

(Refer Slide Time: 17:46)

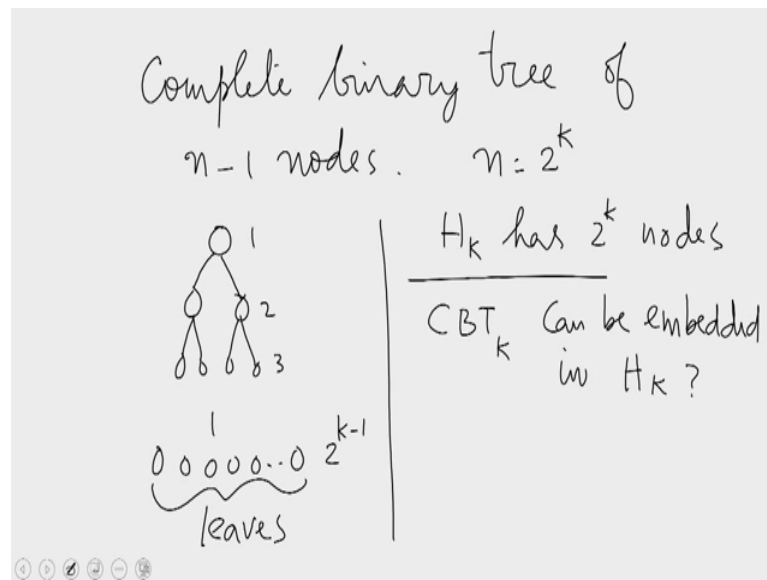


So, let us see how the mapping goes the eight hypercube vertices, we have to flip the first bit and the third bit and then contact me in the second bit over that. So, this is indeed as we wanted we wanted 0 0 0 to be mapped to 1 1 0. Here, we have the first bit and the third bit flipped second bit replicated, 1 0 flips to 0 1 and then 0, 1 1 flips to 0 0 and then 0, 1 0 flips to 0 1 and then 1 1 1 flips to 0 0 that is 1. So, this is how the mapping will go.

So, if the nodes are we numbered in this fashion then we find that the hypercube properties are satisfied that is the same adjacencies are maintained and what we get us therefore, an automorphism. So, this is a nice symmetric property of a hypercube, you can permute the dimensions while mapping a particular node to a particular node. If our goal was merely permuting the dimensions, then we would have multiple options. For example, instead of mapping u to u prime for the same permutation of the dimensions we could realize eight possibilities, that is for each bit whether we want to flip the bit or not will decide what the automorphism would be.

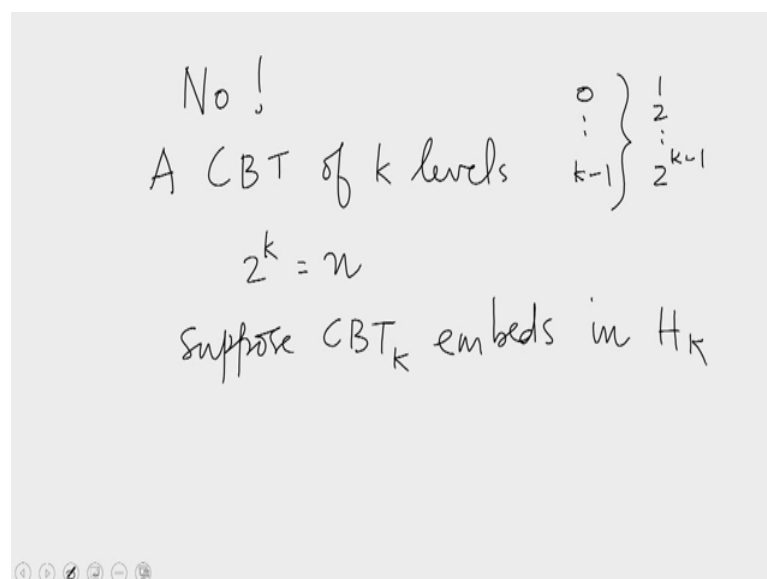
So, there would be eight automorphism realizing the same permutation of dimensions, but when we want to map a particular node to a particular node that is in this case u_2 u prime triple 0 to 1 1 0 then there is only 1 automorphism which realizes that. Now let us see an interesting consequence of this establishes some nice embedding properties, but first a negative result.

(Refer Slide Time: 19:47)



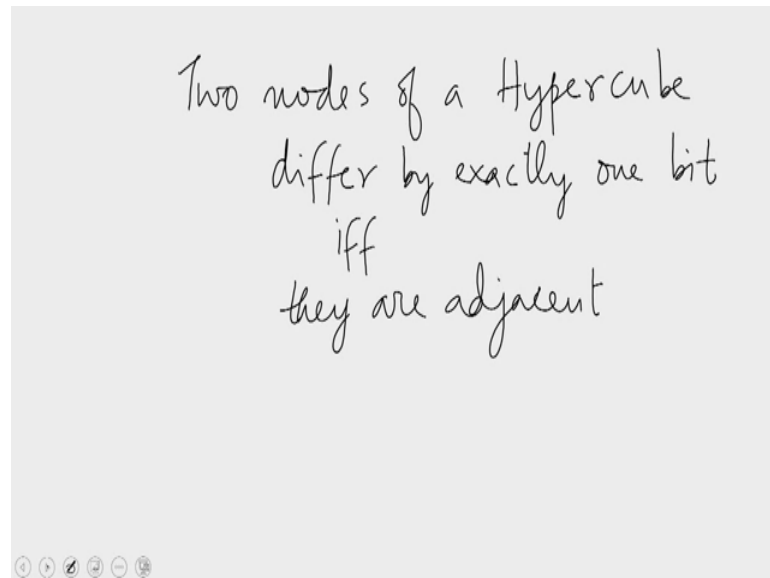
Consider a complete binary tree of n minus 1 nodes, then naturally n is a power of 2 so, in the complete binary tree we have 1 root, 2 children to the root, 4 grandchildren to the root and so on. At the lowest level we have 2^k minus 1 leaves. The smallest hypercube with more nodes than this would be H_k ; H_k has 2^k nodes. So, it would be interesting to check if a complete binary tree with k levels can be embedded in H_k the answer happens to be know.

(Refer Slide Time: 21:10)



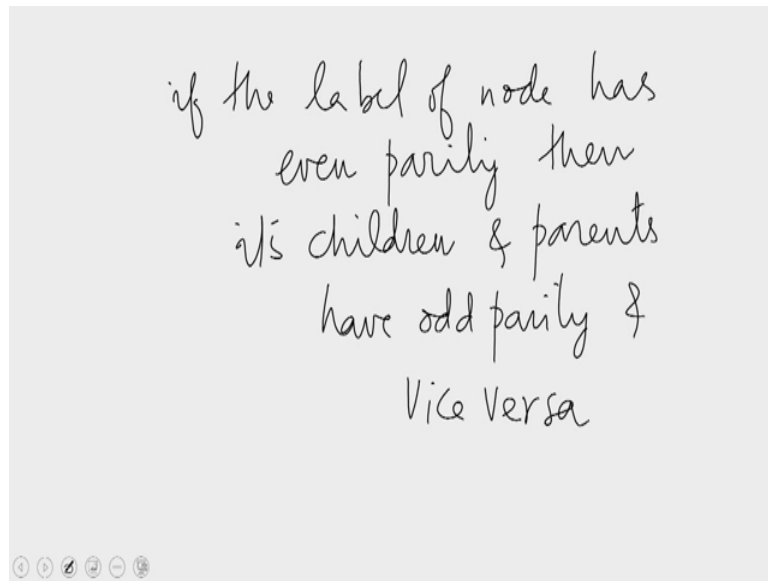
A complete binary tree of k levels with the level numbers going from 0 to k minus 1 and the root at the 0th level we have 1 node at the next level we have 2 nodes and so on, at the bottommost level we have 2^k nodes. Let us say, 2^k is n so, we consider the complete binary tree of k levels. Let us hypothesize that, this can be embedded in a hypercube of k dimensions.

(Refer Slide Time: 22:04)



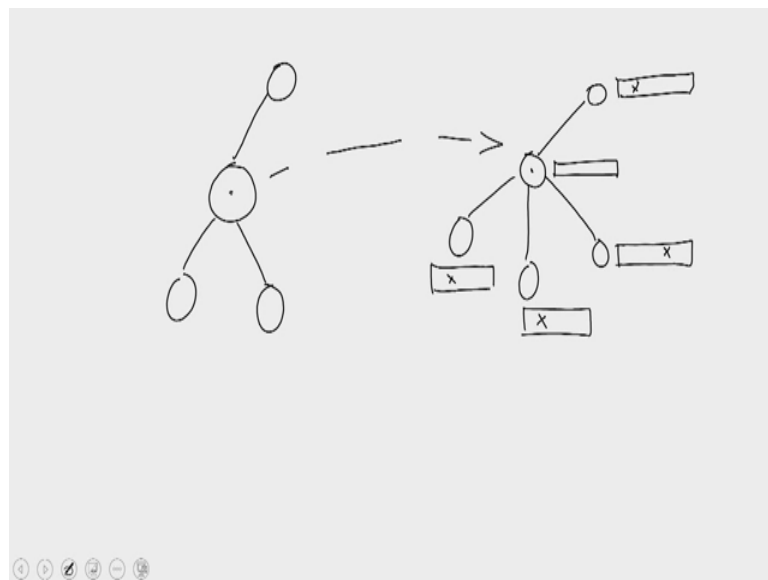
Suppose a complete binary tree of k levels embeds in H^k . Now, what we know about the hypercube is this? Two nodes of hypercube differ by exactly one bit if and only if they are adjacent. Now, if the tree embeds in a hypercube then the adjacent vertices of the tree will be embedding two adjacent nodes of the hypercube therefore, their labels would be differing by exactly one bit position.

(Refer Slide Time: 22:55)



What this means is this? If the label of a node has even parity then its children and parents have odd parity and vice versa.

(Refer Slide Time: 23:39)

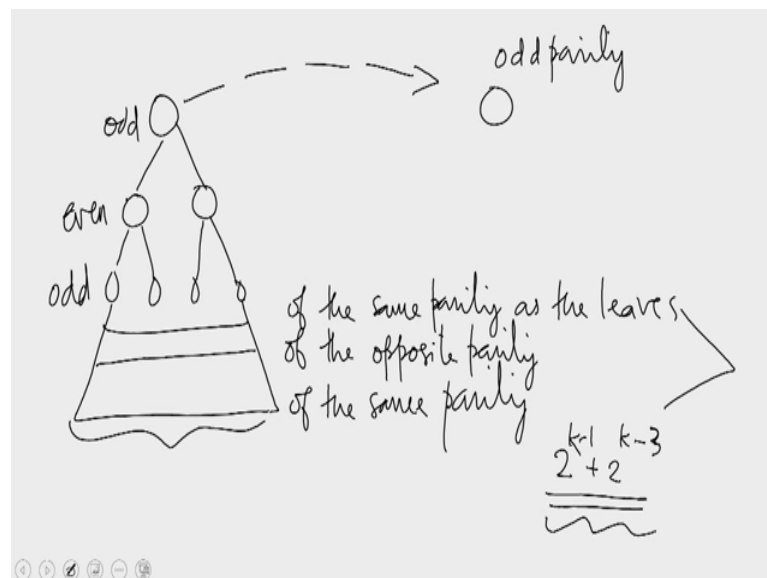


So, what I mean by this is this consider a tree node; a node of the complete binary tree of k levels consider its parent and its children. Let us say, this node maps to a particular node of the hypercube of the k dimensional hypercube. Now, this hypercube node has some name given to it, this node also has k adjacent nodes in the hypercube, one along

each dimension there are k dimensions. So, this node has k neighbors along each dimension.

Now, all these neighbors differ from this node in exactly one bit position therefore, all of them will have a different parity; if this node has an odd parity then all its neighbors would have an even parity because all of them differ in exactly one bit. Whatever be the bit that you flip the parity would flip, if this node has an odd parity then all the neighbors would have even parity and if this node has an even parity then all the neighbors would have an odd parity. Therefore, what we know this? If an embedding is possible then for every single node of the complete binary tree the parent as well as the children are of different parity after the embedding in the hypercube.

(Refer Slide Time: 25:12)



Now, let us say the root maps to a node of odd parity then all its children are of even parity, then all the grandchildren are of odd parity and so on. If the root were of an even parity then all the children will be of odd parity, all the grandchildren would be of an even parity and so on. In any case, as you hit the bottom depending on whether k is odd or even or on the parity that you chose for the root what we find that all the leaves are of the same parity. And all the penultimate nodes all the parents of the leaves are of the opposite parity and then all the grandparents of the leaves are of the same parity as the leaves.

What that means is that, the leaves as well as their grandparents are of the same parity, but in a tree of 2^{k-1} nodes, there are 2^k leaves and there are 2^{k-2} grandparents which means there are $2^k + 2^{k-2}$ nodes of the same parity. So, the number of leaves is 2^{k-1} and the number of their grandparents is 2^{k-3} . So, these many nodes are of the same parity.

(Refer Slide Time: 27:13)

$$\frac{n}{2} + \frac{n}{8} = \frac{5n}{8} > \frac{n}{2}$$

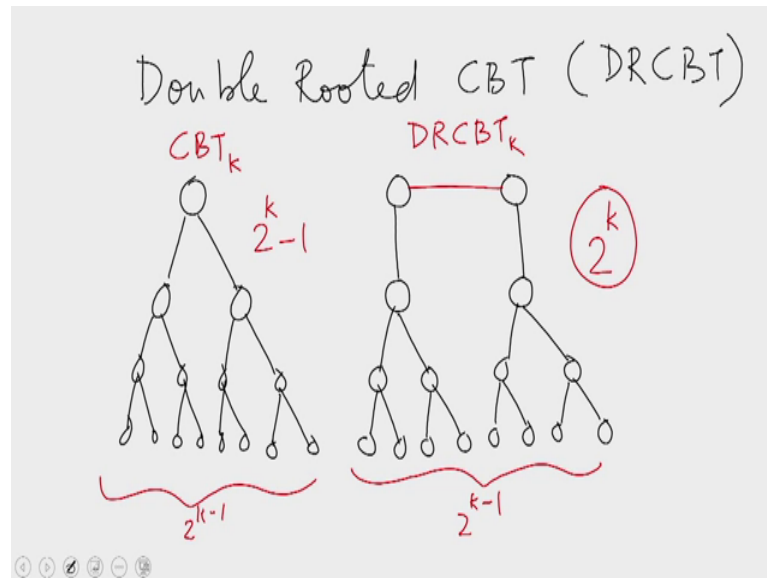
$> \frac{n}{2}$ nodes of H_k are of the same parity

Contradiction

But 2^{k-1} is $n/2$ and 2^{k-2} is $n/8$ this is $5n/8$ greater than $n/2$; what it means is that? More than $n/2$ nodes of H_k are of the same parity which cannot be this is the contradiction that is because in a hypercube of k dimensions there are 2^k nodes. Every possible k bit binary string will be used as a name by some vertex or the other of the hypercube therefore, the number of vertices of even parity is exactly equal to the number of vertices of the odd parity.

Therefore, if you consider a hypercube of n -nodes the number of odd parity vertices is $n/2$ and the number of even parity vertices is $n/2$. Here, what we find is that from the embedding of the complete binary tree in H_k . We conclude that more than $n/2$ vertices of H_k are of the same parity which is a contradiction which means, the complete binary tree just cannot be embedded in a hypercube of 1 extra node. But then the situation is not quite hopeless as far as complete binary trees are concerned, a minor variant of a complete binary tree called a double rooted.

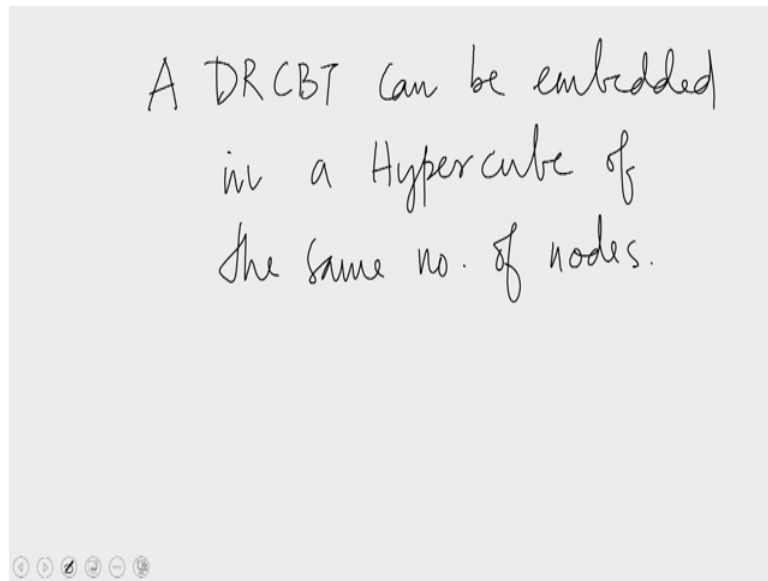
(Refer Slide Time: 28:41)



Complete binary tree is embeddable in a hypercube. A double rooted complete binary tree can be obtained in this fashion consider a binary tree this can be converted into a double root complete binary tree by splitting the root into 2 nodes. The left root will be the parent of the earlier left child and the right root will be the parent of the earlier right child, the rest of the tree remains exactly the same.

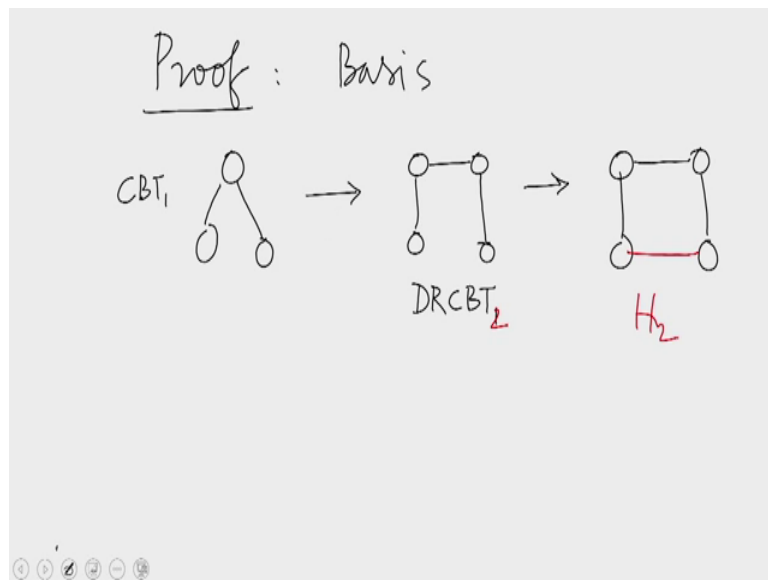
So, all that has happened to the tree is that the root splits into two separate nodes. So, this modified tree has 2 roots let us say we interconnect the roots using a new edge. So, the 1 single root that the complete binary tree had is replaced with an edge. So, in a complete binary tree of k levels we will have $2^k - 1$ leaves here also we have $2^k - 1$ leaves, the tree has changed only at the root, the total number of nodes here is $2^k - 1$ whereas, the total number of nodes here is 2^k .

(Refer Slide Time: 30:34)



Now, what our result is? This a double root at complete binary tree can be embedded in a hypercube of the same number of nodes.

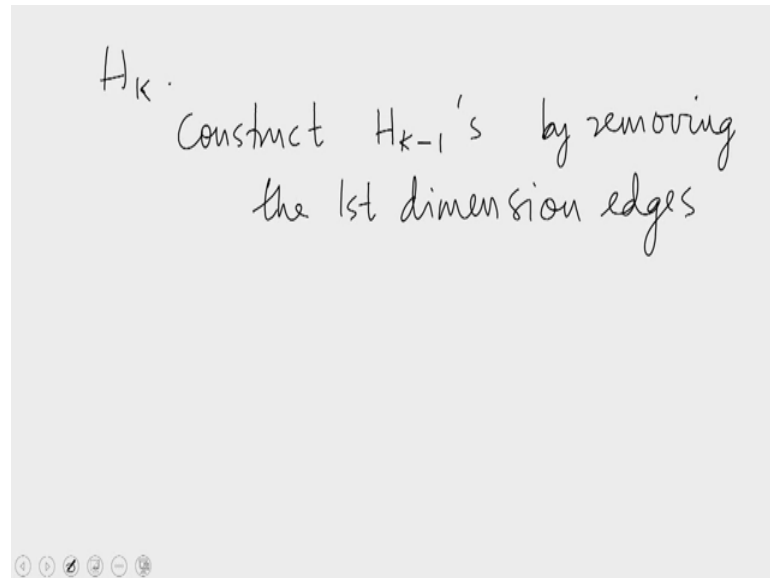
(Refer Slide Time: 31:05)



The proof is by induction in the basis case we consider a complete binary tree of 2 leaves which is CBT 1, when this is double rooted we get this tree instead this is DRCBT 1. Quite clearly, this is embeddable in a hypercube of 2 dimensions sorry this is DRCBT 2.

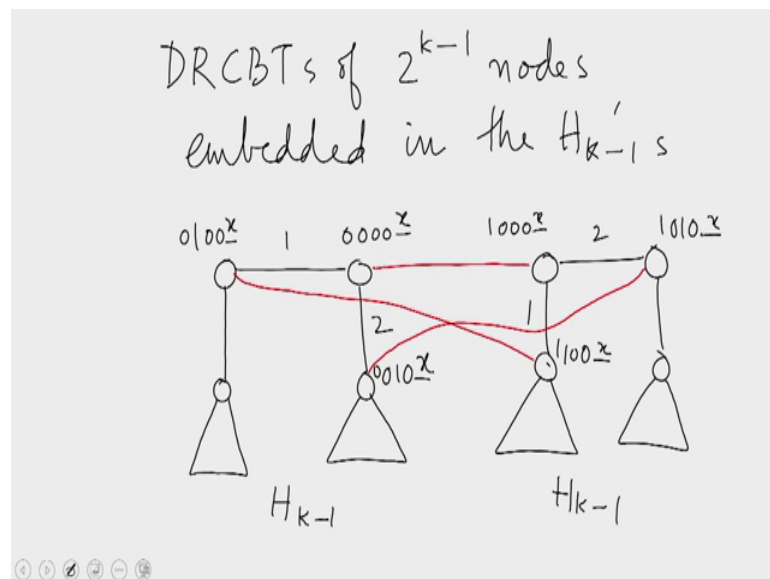
So, DRCBT 2 is embeddable in a hypercube of k dimensions; now, let us consider a hypercube of k dimensions and show that we can have a double root complete binary tree of the same number of nodes embedded in it.

(Refer Slide Time: 32:03)



So, let us consider H_k . From H_k we can construct two smaller hyper cubes by removing the first dimension edges.

(Refer Slide Time: 32:47)



In particular let us say we have double root complete binary trees of $2^k - 1$ nodes embedded in the $H_k - 1$'s. That is by removing all k dimension, for all

first dimensional edges we get 2^{H-k-1} 's. In each of these 2^{H-k-1} 's, we embed a double rooted complete binary tree.

So, let us say the embeddings are like this. So, these double rooted complete binary trees have been embedded in 2^{H-k-1} s. So, this is 2^{H-k-1} and this is another 2^{H-k-1} . Let us say, this is dimension 1 and this is dimension 2, the embedding that we use have these properties. Now let us take a mirror image of this embedding for the other hypercube. So, therefore, this is dimension 2 and this is dimension 1, that is after embedding a double rooted complete binary tree of 2^{k-1} nodes in 2^{H-k-1} , we take two such copies one being the mirror image of the other. And then in the second copy we realize a permutation of the embeddings the dimensions.

So, that 1 and 2 get flipped; dimensions 1 and 2 are interchanged in the second embedding. So, the now the embeddings look like this, let us see how the namings of the nodes would be, let us say those node is named starting with triple 0 and this is some name starting with 1 double 0, the blank could be anything and this let us say is starting with 0 1 0. So, the blank has to be some same x everywhere that is in the embeddings, I have chosen 1 double 0 x , 2; 2 0 double 0 x to be the root of the double rooted complete binary tree.

Similarly, on this side since this is a mirror image I will have a triple 0 x as the name of this node, one of the roots and since here dimensions 1 and 2 have been interchanged here I will have 0 1 0 x as the name of this node and the name of the left child would be 1 0 0 x .

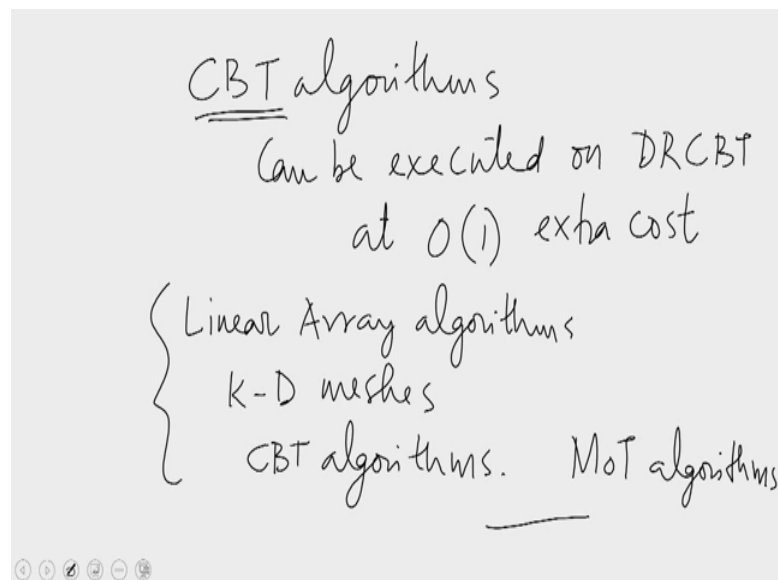
Now, these are the respective names in the 2 copies of 2^{H-k-1} s what we have assumed is that double root complete binary trees of 2^{k-1} nodes are embeddable in the 2^{H-k-1} 's, I assume that the 2^{H-k-1} 's are mirror images of each other. And then in the second 2^{H-k-1} , I permute a dimensions 1 and 2, therefore, the names of the vertices would be as they are now. Then, for the first dimension I will add a 0 here to all the nodes and I will add a 1 here to all these nodes, then we find that 0 1 double 0 x is adjacent along the first dimension 2; 1 double 0 x .

So, we have a hypercube edge of the sort and 0 double 0 1 0 x is adjacent to 1 0 1 0 x along dimension 1 so, we have an edge of this sort as well and double 0 double 0 x is adjacent to 1 doubles 1 triple 0 x therefore, we have an edge of this sort also. All the red

edges are dimension 1 edges; so, we have dimension one adjacencies of the sort. Now we see that, we are coming pretty close to a double rooted complete binary tree of 2^k nodes, if only we delete a couple of edges delete this edge as well as this edge.

What we now find us that, a double rooted complete binary tree with double 0, double 0 x and 1 0 double 0 x as the double roots this now formed within the k dimensional hypercube. So, this is establishes that a 2^k nodes double rooted complete binary tree can be embedded in a hypercube of k dimensions.

(Refer Slide Time: 37:42)

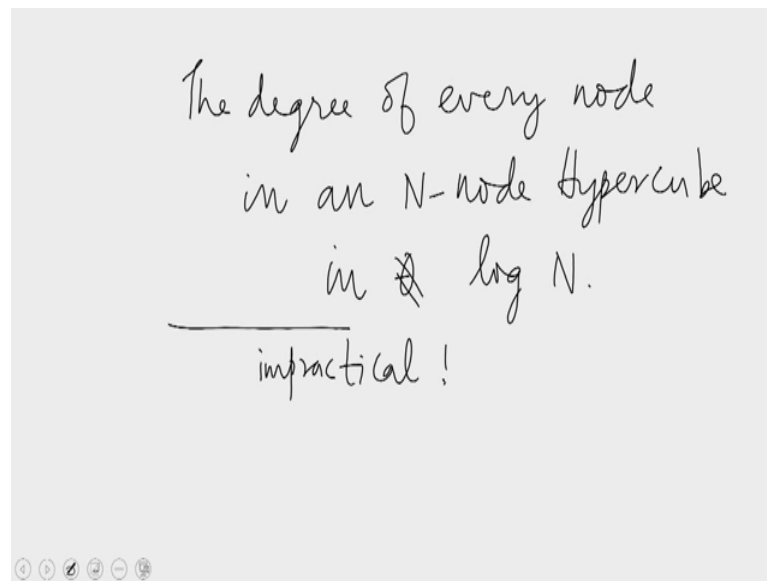


In other words all complete binary tree algorithms can be executed on double rooted complete binary trees that order 1 extra cost; order 1 factor extra cost. That is because all the operations of the root of the complete binary tree can be executed in twice the time by the double roots together, that is the 2 roots together will function as a single node and they can send messages to each other and with 1 exchange of messages in 2 steps the double root can simulate every operation of the single root of a CBT.

Therefore, every CBT algorithm can be simulated on a double root of complete binary tree in twice a time. And a double root to complete binary tree is embeddable on a hypercube therefore, whatever you do on CBT can be executed on a hypercube at most twice the time.

So, this establishes a versatility of hypercubes as far as simulations are concerned. So, we now know that linear array algorithms, multi-dimensional meshes, complete binary tree algorithms and as an extension to that mesh of trees algorithms and how going to prove this here, but mesh of trees algorithms all can be simulated on a hypercube, at no additional cost that is only order 1 extra factor would be necessary for all these simulations. However, the hypercube architecture has a disadvantage.

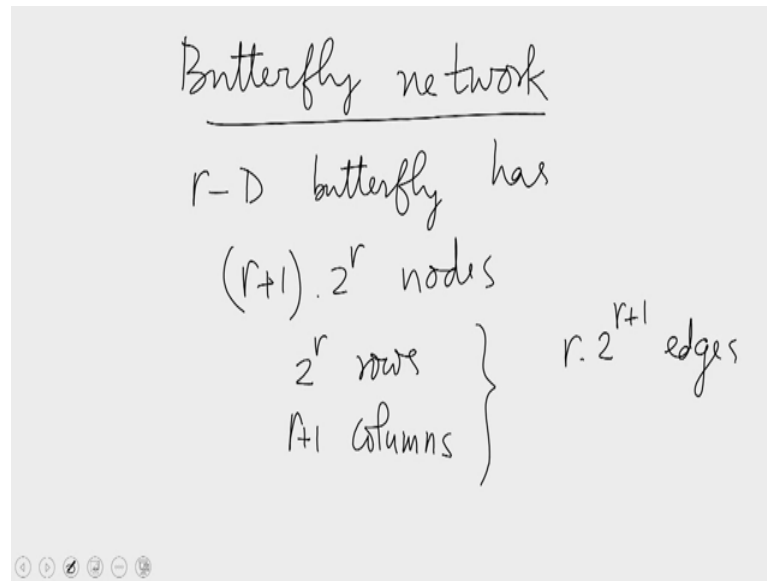
(Refer Slide Time: 39:41)



The degree of every node in an N-node hypercube this order of $\log n$ this theta of is exactly $\log N$. Whereas, all the other interconnection networks that we were talking about the linear arrays, k dimensional meshes for a constant k and the complete binary trees or mesh of trees for a constant dimension in all these cases if the degree of every single vertex is a constant.

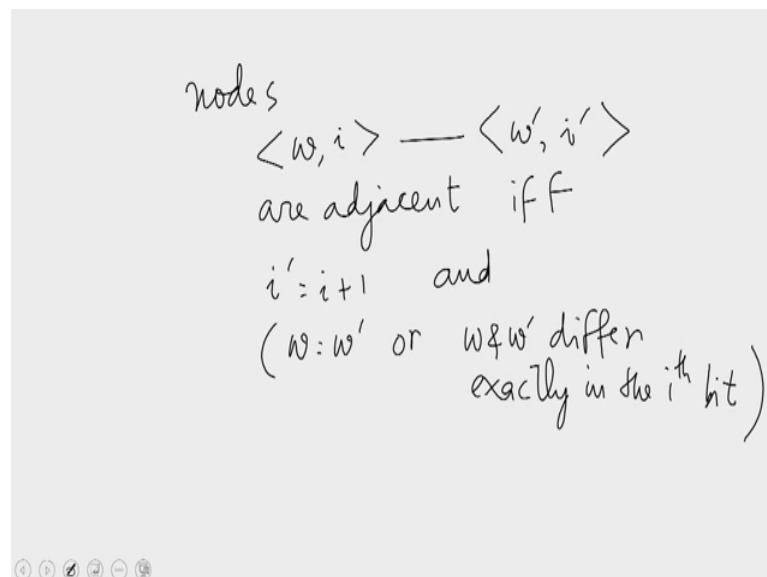
So, these are all constant degree graphs whereas, a hypercube is a $\log N$ degree graph when the number of nodes is N. Therefore, a hypercube is something of an impractical network in spite of it is versatility. But then there are networks which can simulate hypercube algorithms, but those networks are still bounded degree networks. So, they have the advantage of both sides they retain the versatility of hypercube networks, but at the same time they are more practical in the sense that the maximum vertex degree of every single node is a constant.

(Refer Slide Time: 41:11)



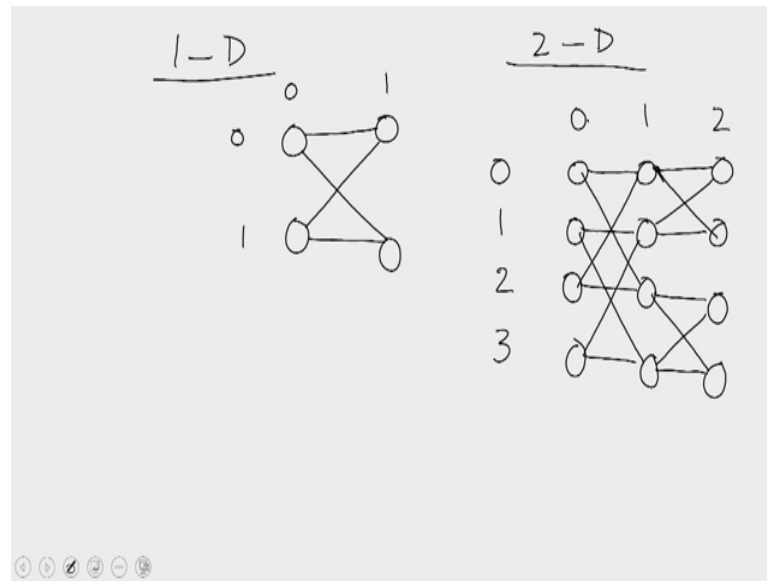
One such network is the butterfly network. In an r dimensional butterfly network has r plus 1 into 2 power nodes. The nodes are arranged in 2 power r rows and r plus 1 column and we place r into 2 power r plus 1 edges among them.

(Refer Slide Time: 42:05)



The adjacent c is are established in this manner nodes w, i and w prime, i prime are adjacent. If and only if i prime is equal to i plus 1 and either w is equal to w prime or w and w prime are differ exactly in the i th bit.

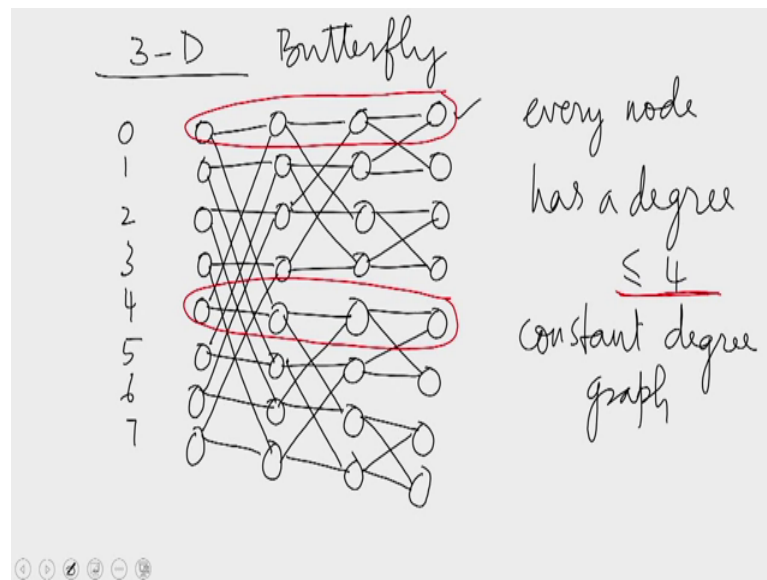
(Refer Slide Time: 43:02)



So, let us see how some butterflies look like. Let us consider the first dimensional butterfly; in a first dimensional butterfly we have to power 1 nodes to power 1 rows numbered 0 and 1 respectively and we have 2 columns named 0 and 1. And then we have interconnections like this. So, you can verify that these interconnections are according to the definition we have seen just now.

In a 2 dimensional case we have a three columns numbered 0 1 2 and we have four rows; numbered 0 1 2 3 and the interconnections are made in this fashion and the interconnections are made in this fashion. Once again verify that they are according to the definition that is nodes w_i and $w_{i'}$, i' are adjacent precisely according to the definition here.

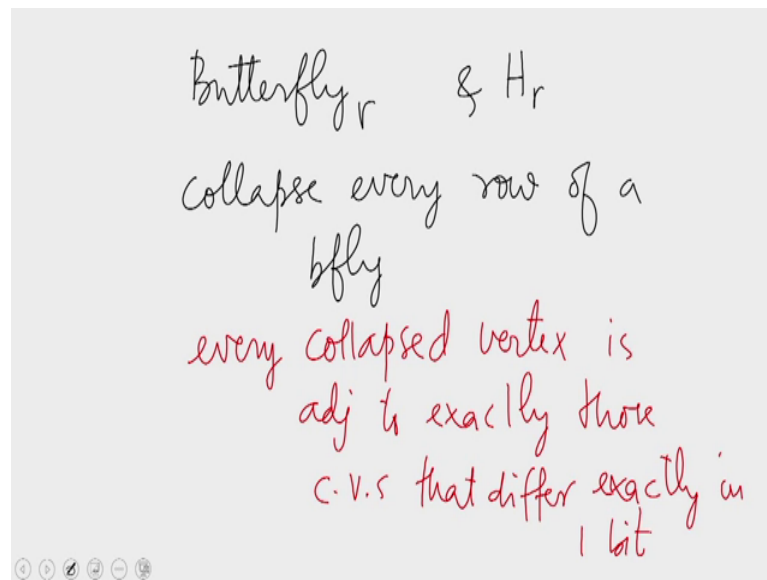
(Refer Slide Time: 44:34)



So, now, you can see why the network is called a butterfly network. When we go to the 3rd dimension, we will require eight rows and four columns $r \times s \times 3$ here therefore, $r + 1$ is 4; so, we require 4 columns between the first column in the second you have to draw edges in this fashion so, this is a 3 dimensional butterfly network.

So, what is the advantage of a butterfly network over a hypercube? You can see that every node has a degree less than or equal to 4. The vertices of first column as well as the last column have a degree of 2 each, all the other vertices have a degree of 4 each. So, we have a constant degree graph indeed which is more practical than a hypercube which has and non constant vertex degree.

(Refer Slide Time: 47:08)

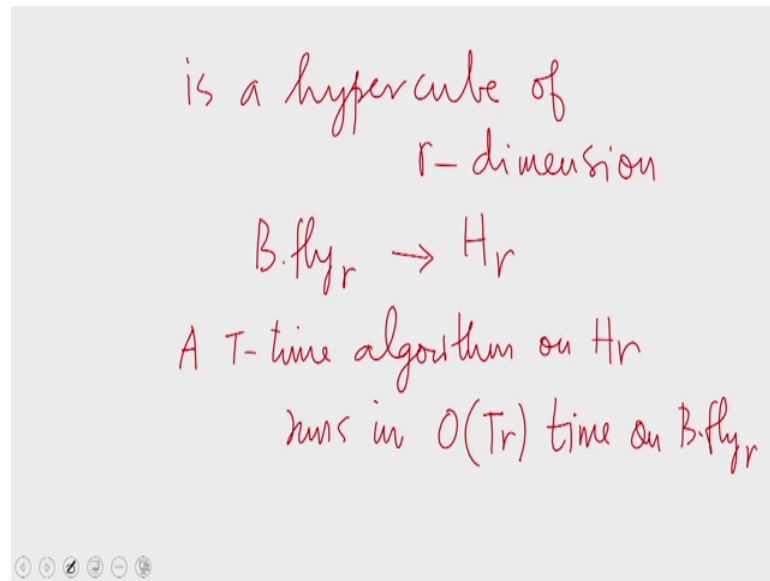


Now, what is the relationship to a butterfly and hypercube. Let us consider an r dimensional butterfly and an r dimensional hypercube. When you look at the picture we find that, if you collapse every single row of a butterfly by fusing the vertices together, then we find that the cross edges of the between the first column and the second will form the first dimensional edges for the fused vertices, that is when all the vertices of the first row are fusing together to form one single vertex.

So, this we label by the name of the row 0 so, this is the 0'th row of the modified graph and let us consider the 4'th row, the vertices of the 4'th row will also fuse to form one single vertex. Now we find that this fused vertex of name 0 and the fused vertex of name 4 are adjacent using the cross edges of the butterfly network that are there between the first column and the second column. Similarly, when I consider nodes 0 and 2 the fuse nodes 0 and 2 we find that they are adjacent using the cross edges between columns 1 in 2 and nodes 0 and 1 are adjacent using the cross edges between columns 2 and 3.

So, what this establishes the is that? Every fused edge; every collapsed edge, collapse vertices vertex is adjacent to exactly those collapse vertices that differ exactly in 1 bit. In other words, the graph that you obtain when you fuse all the rows into 1 single vertex is a hypercube of our dimensions, that is you can convert a butterfly of r dimensions into a hypercube of r dimensions by collapsing every single row into a single vertex.

(Refer Slide Time: 49:33)



Therefore, any algorithm that you run on a hypercube of dimensions r can be run on a butterfly of r dimensions in order of r prime. One single node of a H_r is converted into 1 entire row of butterfly r . Therefore, a T -time algorithm on H_r runs in order of Tr time on butterfly r or in other words if you consider a hypercube of n -nodes r is $\log n$ therefore, we will have an order $\log n$ extra factor in the running time, whatever you do in T -time on hypercube of n -nodes can be done in order of $T \log n$ time on a butterfly of dimension $\log n$.

So, butterfly is quite versatile whatever you do on a hypercube can be similar to run a butterfly at a small extra cost and then it also has this nice property that every vertex is of a constant degree. More about butterflies and similar networks in the next class hope to see you in the next.

Thank you.