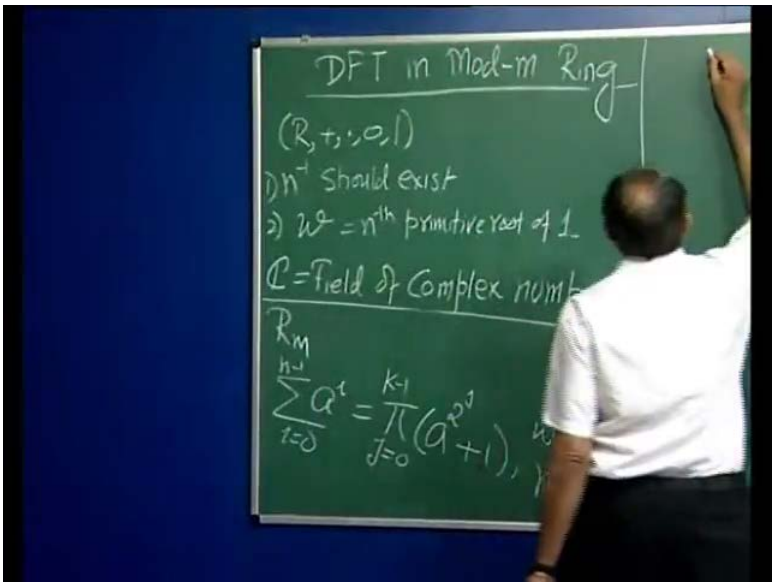
Computer Algorithms - 2
Prof. Dr. Shashank K. Mehta
Department of Computer Science and Engineering
Indian institute of Technology, Kanpur

Lecture - 25
Discrete Fourier Transform III

(Refer Slide Time: 00:42)



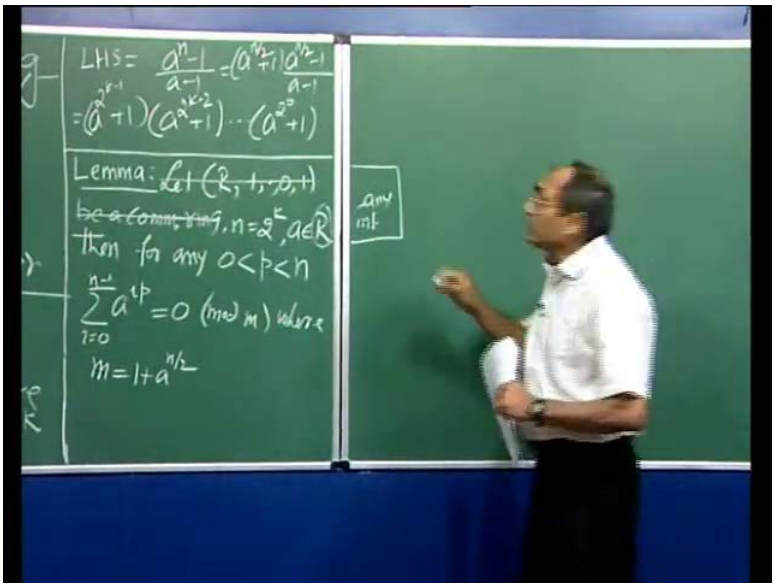
Hello, in last 2 lectures we have discussed this Discrete Fourier Transform. And first Fourier transform algorithm for it, and what we had said, is that discrete transform of an n vector, a vector of n elements can be computed, if the following conditions are true. First of all this domain in which we are working is a ring R which is committed to that is to say the multiplication is commutative and other properties of the ring hold. In addition to that n inverse should exits and 2 there should be a w which is the n'th primitive root of unity, n'th root of 1 this should also exits for as to be able to compute a discrete Fourier transform, of n component vectors. In addition to that we have assumed that n is a power of 2 in order to apply FFT algorithm. And in the last structure I have mentioned, that these things are there in the field of complex number. So, we can actually go a head an compute the discrete Fourier transform of vectors, where the components are complex numbers.

Now, today I am going to look at another domain, in which we can perform this computations and that is practically of interest because there we are going to be dealing

with integers only. So, today I am going to discuss such a ring, which is actually nothing but modulo m ring, where all the m is certain specifically chosen number integer. And the ring is, numbers ranging from 0 to m minus 1 and both these operation multiplication and addition will be done modulo m.

So, I am going to denote that by R m. So, we will discover this ring in which we can perform this DFT and for that, we will have to establish certain results, to the first one that I am going to show is that a power i, i going from 0 to n minus 1 this geometric series is equal to the product, from j going from 0 to k minus 1 a power 2 power j plus 1 where n is 2 power k.

(Refer Slide Time: 03:54)



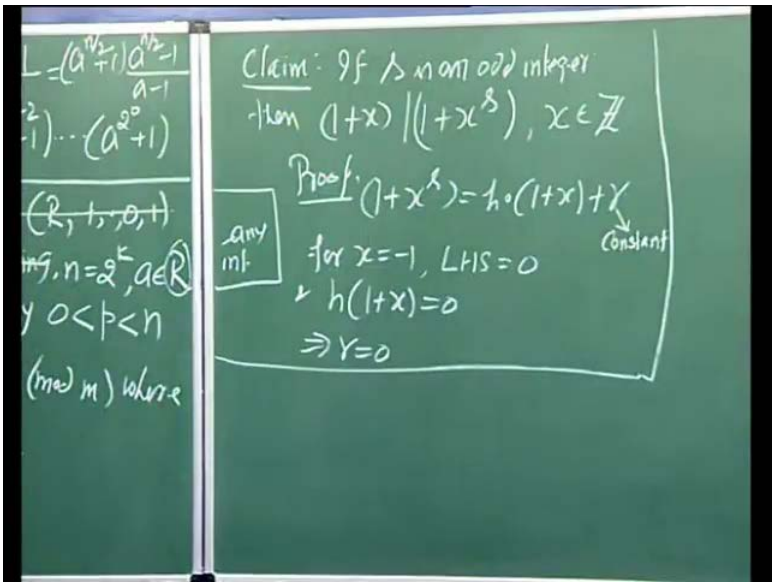
So, the left hand side the geometric sum is a power n minus 1 over a minus 1. So, I can factor this and write down this is a power n by 2 plus 1 into a power n by 2 minus 1 a minus 1. So, this can be further broken down, assuming that now of course, we have assume that n is a power of 2. So, this is a power 2 power k minus 1 plus 1 the first half then we will have, this power to 2 power k minus 2 plus 1 all the way up to a power 2 power 0 plus 1 and you will be left with a minus 1 at the end over here, which will cancel the denominator; so this preciously what the right hand side.

The next claim, we want to establish is says that let R plus dot 0 1 be a commutative ring remember by commutative a ring a multiplication is commutative a times b is equal to b times a, n is 2 power k a is sum element of the ring, then for any p strictly between 0 and

n. So, this is any integer come 1 to n minus 1 remember, since 1 is present in the ring, then 2 is also present, 3 is present, so all integers are present in the ring.

Then we claim, that sum a to the power i p for i going from 0 to n minus 1is 0 mod m, where m is 1 plus a to the power n by 2. In fact, I can take this more generally and what I really need is, in the ring of integer itself. So, even if I ignore this and we generally take n to be 2 power k and a is any integer, this is any integer then again this should hold, where m is 1 plus a power m by 2.

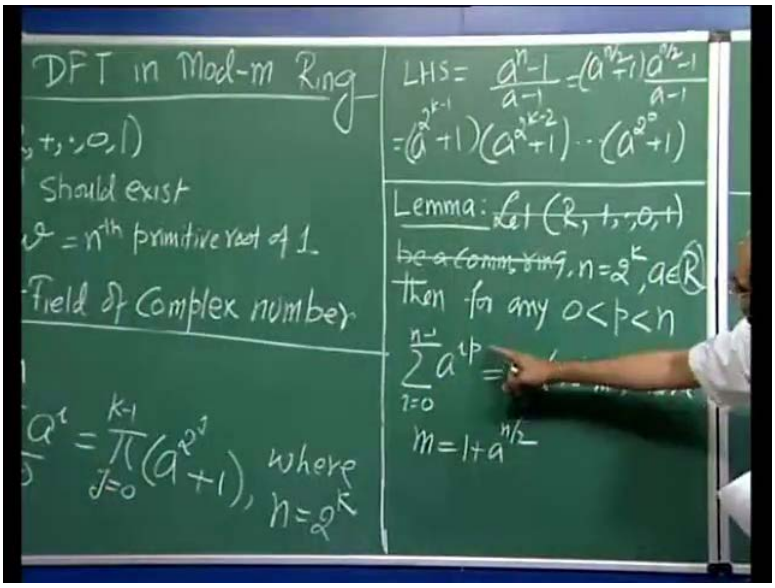
(Refer Slide Time: 07:40)



Now, lets first establish a small sub claim if s is an odd integer, then 1 plus x divides 1 plus x power s, where for any x belong into set of integers, this denotes set of integers. So, to show this let as, divide 1 plus x power s by 1 plus x and we can write down 1 plus x power s as sum h times 1 plus x plus r, this I am treating x as a variable. So, this is a polynomial of degree s, this is polynomial of degree 1. So, we can always make a division, such that this us degree less than the degree of 1 plus x.

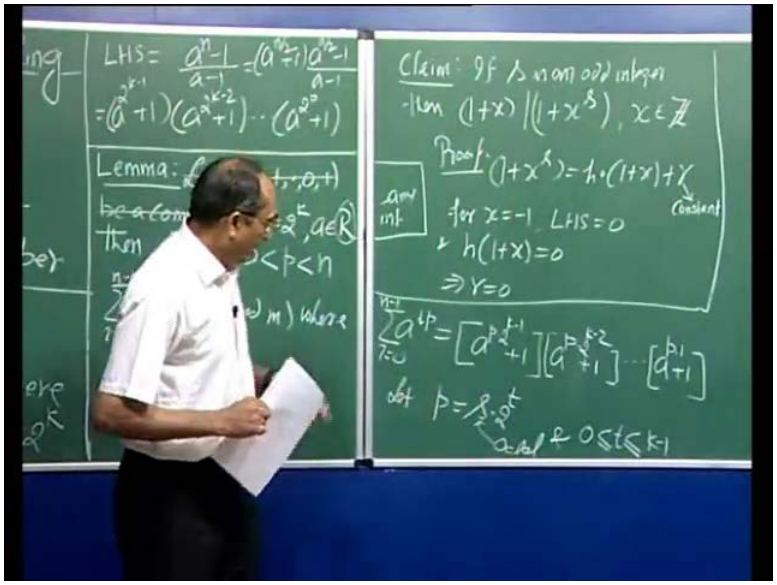
So, this has degree 0, so this is actually a constant, so in case if I set x equal to minus 1 then this term is 1 plus minus 1 power in odd power which is a minus 1, this is for x equal to minus 1 the left hand side is 0 this term is also 0 and h time 1 plus x is 0 hence, r which is a constant must be 0. So, what we notice is, that this is 0, so 1 plus x better divide, 1 plus x power s. Now, this is a side result that I will need to prove this claim.

(Refer Slide Time: 10:16)



So, now, let us take the expansion of this geometric says as wave done here.

(Refer Slide Time: 10:26)

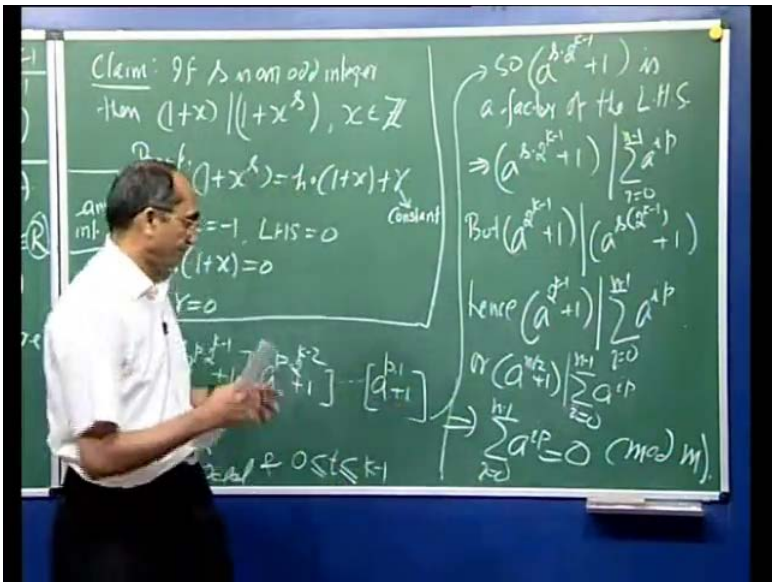


So, some a power i p, i going from 0 to n minus 1 we know is equal to the product some I am going to explicitly write down, this product which is a power p times 2 power k minus 1 plus 1 a power p 2 power sorry this is in the exponent. So, this will be p times 2 power k minus 2 plus 1 all the way up to, a power p times 1 plus we have this expansion, as you shown here. Now, let us take let p be s times 2 power t I can take all the 2 factors

in p, so this will be an odd number, so this is odd and t notice our p is given to be strictly less than n.

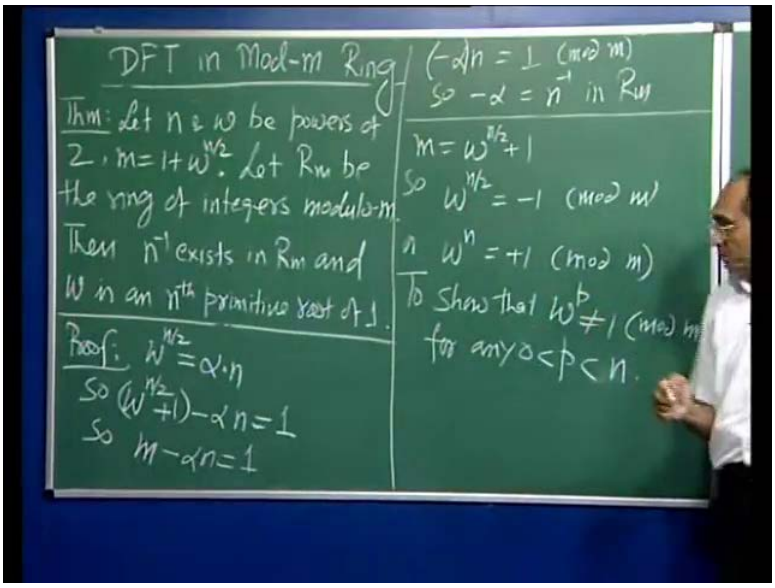
So, t cannot be equal to k here, n is 2 power n, so t can be any where from 0 to k minus 1 this is the possible range of s, it can be less than it yes, in the it cannot be equal to k, but it has to be, it can be any thing up to k minus 1. Now, let us try and plug this p in these terms, so the term in which we have to using for k minus t there will be a term with k minus t we will get.

(Refer Slide Time: 12:44)



So, we are going to find that a to the power s times 2 to the power t into 2 power k minus k minus 1 minus t, k minus 1 minus t, because the largest values is k minus 1. So, t it can at most be k minus 1. So, we will have a power s times 2 to the power k minus 1 plus 1 will be a factor of the left hand side. Now; that means, a power s times 2 power k minus 1 plus 1 divides, this sum 0 to n minus 1 and now, I am going to use this result notice that my s is in odd number and this result tells me, that a to the power 2 power k minus 1 plus 1 a to the power 2 power k minus 1 plus 1 divides a power s 2 power k minus 1 plus 1. Hence, a power 2 power k minus 1 plus 1 must also divide this. Now, over m is 1 plus a power n by 2 and this is also a power n by 2 a power n by 2 plus 1 divides this sum I going from 0 to n minus 1 implies, this sum is 0 modulo m because this is m correct.

(Refer Slide Time: 15:25)



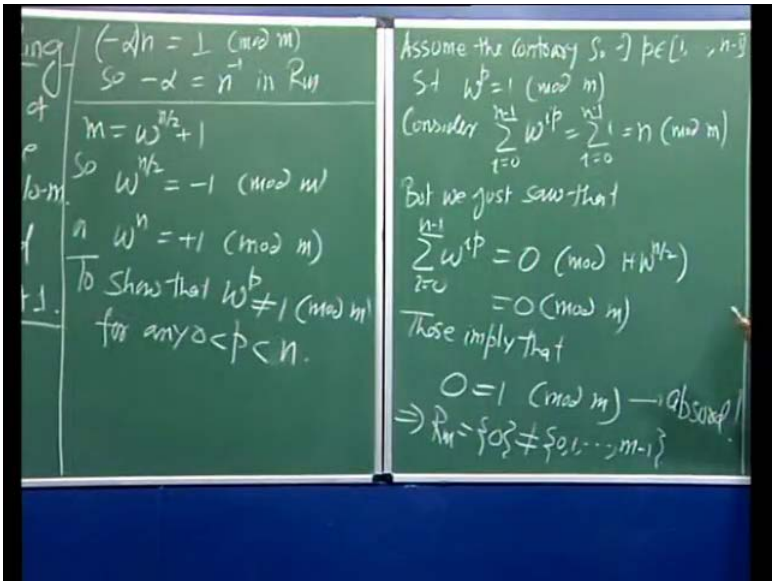
So, now, we are ready to describe the ring, in which we can compute DFT and that is comes from this result. The theorem says that let n and w be power of 2 of 2 m be 1 plus w power n by 2. Let R m be the ring of integers modulo m, then n inverse exits in R m and w is the an n'th primitive root of 1. So, these are the two properties we need, in the ring to be able to compute the DFT and we already start with, n to be a power of 2 to we can also use FFT to compute the discrete Fourier transform in this ring. So, let us try and show these two properties.

Now, first thing you notice is that n being a power of 2 and m is 1 plus power of 2, so w power n by 2 is n to the power sum alpha. So, I can actually when I do not need even power, to I can express this us some alpha times n, because this is a power of 2 this is a power of 2. So, I can boost this to this and multiplying a power of 2 and we are going to get w power n by 2 plus 1 minus alpha times n equal to just 1 and this is power m, so m minus alpha m is 1.

So, if I compute modular m of both sides I am going to get minus alpha n is 1 mod m. So, minus alpha is n inverse in R m this is multiplicative inverse of n, n times, so you can do this way. So, this n inverse exits. Now, lets take a look at w is it a primitive n s root of 1. So, first thing of course, we notice is that m being w power n by 2 plus 1. So, w power n by 2 is minus 1 mod m or if we square the 2 we are going to w power n to be plus 1 mod m. So, indeed it is an n f'th root. Now, the question is it a primitive n f'th root,

which means that w power now, you want to show that w power p is in not 1 mod m for any p between 0 and n. We would like to establish this and that would mean that w is a primitive n of root.

(Refer Slide Time: 20:20)



Now, suppose assume the contrary, so there exits a p in the range from 1 to n minus 1 such that, w power p is 1 mod m. So, let us consider the sum w power i p from i going from 0 to n minus 1, when you plug in i equal to 0 this is 1 subsequently this is still 1 because w power p is 1. So, this n d t is one for every value of i and that should become, sum 1 i going from 0 to n minus 1 which is n modulo m.

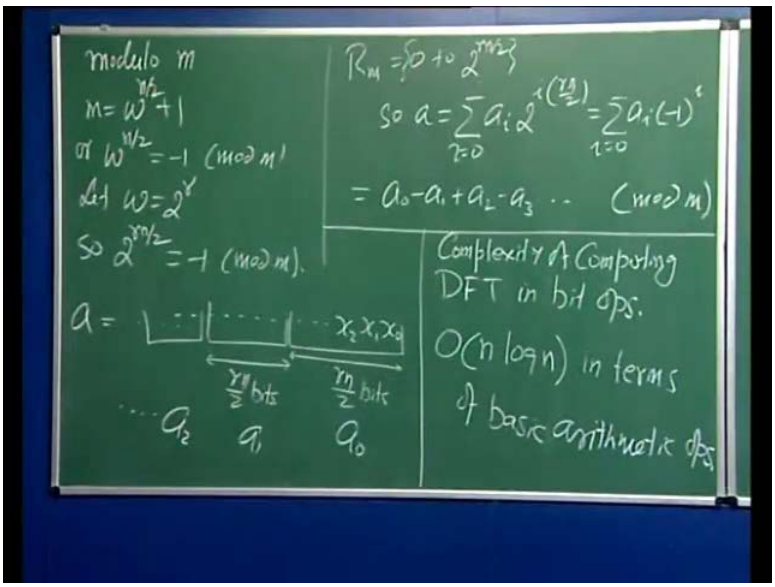
But, little while back we have shown, but we just saw that this sum w power i p, i going from 0 minus 1 is expression a 0 mod 1 plus a to the power n by 2 w power n by 2. So, we had just shown, where this was suppose to be an either. So, we have already shown this result, which is same as 0 mod m. So, these 2 together, these imply that 0 is equal to 1 modulo m, so in this ring 0 and 1 are equal. Now, if 0 and 1 are equal then in this ring you cant have any other term this would imply, that our entire ring is simply just 1 element we call it 0 or 1.

But, that is not what we have, we have actually this not equal to 0 1 to m minus 1 hence, in this ring 0 and 1 cannot be equal. So, this was absurd, hence this assumption is not correct indeed w power p is in not 1 or any exponent before n, indeed it is 1 for 0. So, we now have shown both these property, which were required for defining DFT are held in

this ring of R m and we also have a nice property that n is a power of 2. So, we are in a position to define discrete Fourier transforms in this ring, another interesting property about this is, that we can choose orbit ally large n the reason is that m is nothing but 1 w power n by 2.

So, 1 you decide what ever n is, you can chose orbit ally big w and once you chosen w you have chosen here m. So, in many computation when you want to perform exact results, when you want to compute exact result in integers, you can use this rings, but if your m is very large then you will never cross that domain and you I will exact results. We will see in an example, in today and tomorrows lectures. Now, I am going to discuss the complexity of computing DFT in this ring in terms of bit operation, the reason is we are dealing with now purely integers. And hence, we should be able to compute the complexity in terms of bit operation, but before that I would like to point out, one more intrusting in very useful property of this ring R m and notice that.

(Refer Slide Time: 25:22)



We will offend have to compute modulo m of an integer well we are computing, performing computation in R m we will have to repeatedly perform modulo m. Now, m is w power n by 2 plus 1 or w power n by 2 it is minus 1 modulo m. Now, let w be 2 power r notice, we had assume that w is a power of 2. So, we have 2 power r n over 2 is minus 1 modulo m now, lets consider you have a number, some number a that number is say x 0, x 1, x 2, so this is the binary expression for number a.

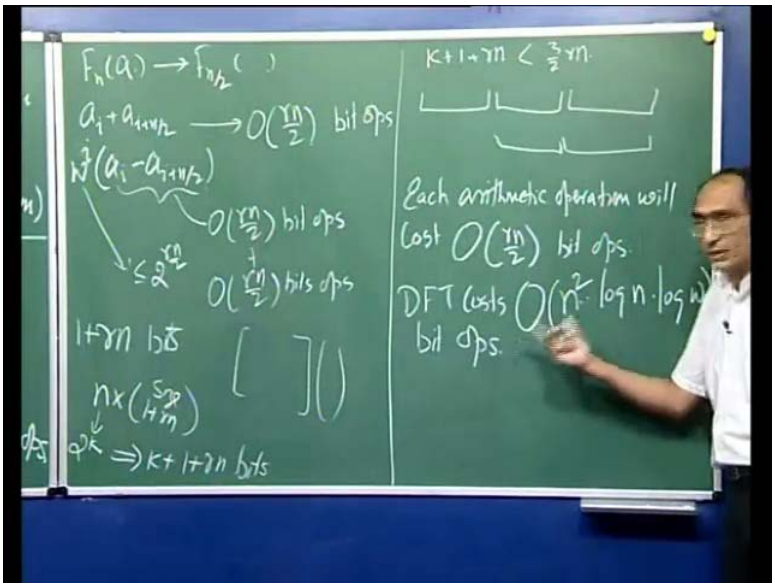
So, what we will do, it is split this into blocks each blocks, has a r n by 2 bits and so on. Now, in capital r m the numbers that we have, go from 0 to 2 power r n by 2 because our m is 2 power r n by 2 plus 1. So, this is the total span of, so this size of the numbers, in terms of number of bits can be up to r n by 2 as big as this. Now, let suppose the number expressed by these bits is a 0 the number expressed by these r n by 2 bits is a 1 and so on a 2 etcetera.

So, our number a is nothing but sum a i 2 power i times r n by 2, i going from 0 to sum number what ever with highest exponent here number of blocks, but 2 power r n by 2 is minus 1. So, this becomes sum a i minus 1 power at 0 onwards, so this becomes simply a is 0 minus a 1 plus a 2 minus a 3 in the ring, that is modulo m. So, to compute modulo m the fast method is just to cut this into these blocks and just take this numbers and subtract this one from this and then add this to this resultance.

So, just perform this operation and that is a fast way to compute modulo m, incase this result is still larger than this many place, which means this result is still outside the rings domain, then you repeat that process. So, we can always weakly compute the modulo m and this is important because this will be an additional cost that we will have to account for when we determine the, bit wise operation cause of computing DFT. Now, let us go back to the complexity of computing DFT in bit operations.

Let us go back and recall, that the fast Fourier transform algorithm or computing DFT was costing s order n log n in terms of basics arithmetic operations, not bit operation, but adding two numbers, multiplying two numbers and etcetera, that was the cost. So, now lets go back and those operations and from that we will find an upper bond for each of these operations in terms of bit operation, that should give us the bound for DFT.

(Refer Slide Time: 31:06)



So, recall that actually, that DFT that is F n of a vector was reduce to F n of F n by 2 of half size by 2 and in doing. So, what we were doing, is we where first computing a i plus a i plus n by 2 this gave us one of the half size vector, the other one was w power i a i minus a i plus n by 2. So, in third process we where computing new to new vectors, each of n by 2 sizes starting from the vector of size n, which was given to us.

So, each of these operations that we are doing, is simply in operations which is linear, in the sizes of the number here, we are just adding two numbers remember here, we are subtracting and this multiplication is nothing but shifting because this is a power of 2 w being a power of 2 this does not really have to be multiplied we have to only shift. So, the number of operation here, in terms of bit operation is going to be at most or r n by 2 bit operation.

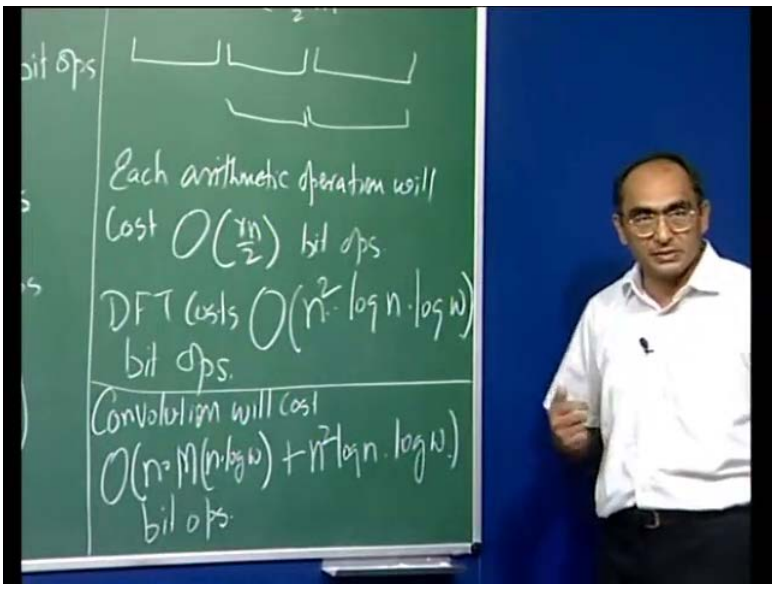
Now, over here again the these operations, are these are order r n by 2 bit operations and this is, in at most this is going to be this is less than equal to 2 to the power r n by 2. So, once again I am going to perform order r n by 2 bit operations, which are nothing but shifting that many shifts. After we are perform this, the number that will result, can have at most and that is in the second case, can have r n bits because here adding this and may be 1 plus this 1 plus r n bits that can result because here 1 plus r n bit and here let say 1 plus r n two bits plus r n by two bits.

And in the computation of DFT, we are actually just performing linear operation we are taking the dot product here. So, each of these terms is being multiplying to this, an at the end of the day we are adding them all. So, we have n such terms that we are adding, so the number that we will have is n times a number of size 1 plus r n bits because we will have n such numbers and this is a 2 power k. So, that could mean that k plus 1 plus r n bit size, would be the resulting the number that will get for each of these locations, for each of the components will have that when we bits at most.

Now, that this k plus 1 plus r n is a less than 3 by 2 r n. So, when we compute the modulo m of this numbers, we will have at most 3 blocks and we will be computing this minus this plus this and the result is still greater than r n by 2 bits that then we will have to compute one more of these operation. Hence, we have at most 1 2 and 3 additional operation of summation of these.

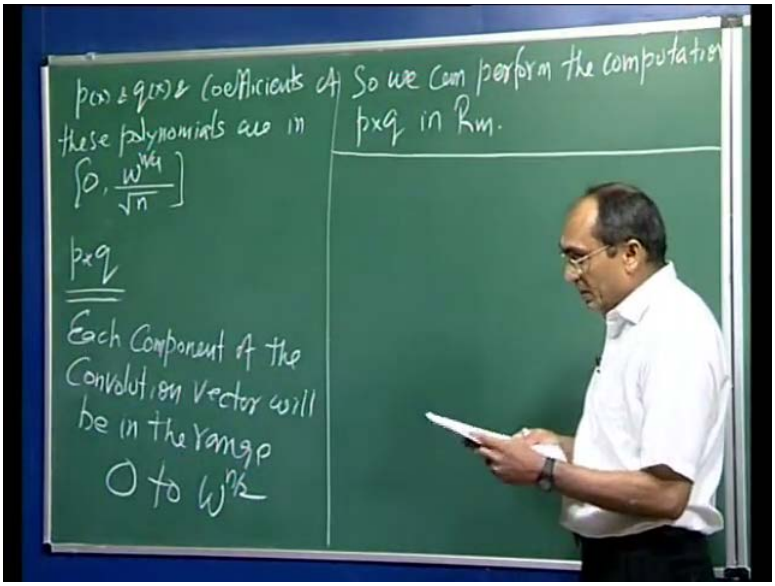
So, in the end the entire, that is the each arithmetic operation will cost order r n by 2 bit operation, you can perform the entire DFT the performing n log n operation each taking this many bit operation. So, the DFT costs order n there are 2 n's n square times log of n times n r is nothing but log of w, so which is log of w, these many bit operation, the consequence of this is now, we can also determine the cost of computing a convolution of two vectors.

(Refer Slide Time: 37:22)



So, the convolution will cost order now, in addition to computing DFT is up two vectors in case of convolution we also have to compute, point to point multiplication and then compute inverse DFT. Now, inverse DFT in DFT case the same. So, this will be the cost of all the 3 DFT operations. Now, in additions to that we have to add the cost of m multiplication, of numbers, of size r n by 2 bits. So, that will be n times M r n is n log w plus this will account for those computation and this term n square log n log w. So, convolution in r n will got cost this many bit operation, in case we are computing in this ring, the multiplication of two polynomials, then it will cost us this many bit operation because polynomial multiplication is nothing but the computation of multiplication.

(Refer Slide Time: 39:02)

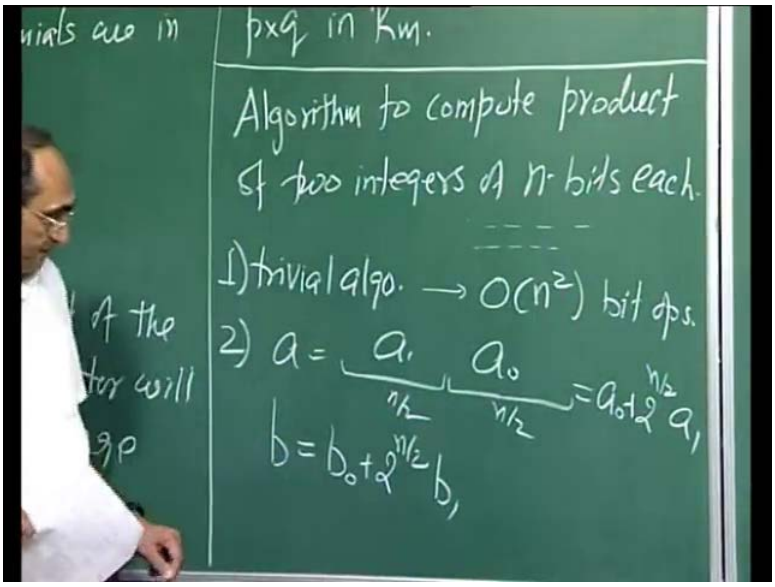


To finally, when no rework in case we have a polynomial suppose we have, p of x and q of x and the coefficients of these polynomials are in the range 0 to w power n by 4 divided by square root of n. Let consider a situation, where the coefficients are restricted to this, in that case the largest number that will result, when we perform the multiplication of p and q, p times q in this computation we are going to perform effectively, the convolution of these coefficient vectors.

And in doing so what we are going to do is we are going to multiply two numbers in this rings and then add those n such numbers. So, each component of the convolution vector will be in the range 0 to the square of this tomes n and that will be w power n by 2. Now, this indicates that if might coefficients are limited to this, then I can perform the same

computation in r n. So, we can perform the computation p time q in R m the reason is in R m these values will not change, they will remain as it is. So, we can get exact value of these 2 of the product of these 2 polynomial when we perform computations in R m. So, these are the observations about the complexity and of course, one remark about, how far we can go without loosing any information.

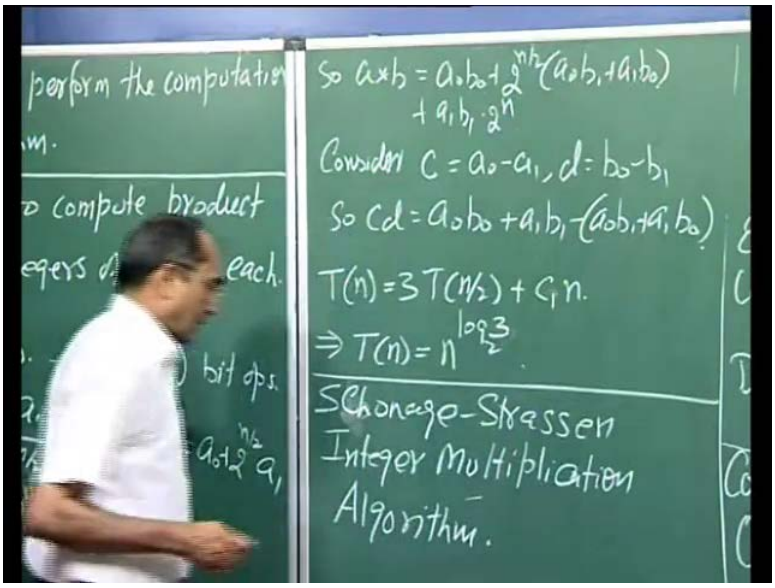
(Refer Slide Time: 42:15)



Now, we are going to just give some initial remarks about any interesting algorithm for computing algorithm to compute product of 2 integers of n bits each. Now, first of all lets recollect that the trivial algorithm, that is the first 1 is a trivial algorithm, the long head algorithm in which all you do is, you take each bit and multiply to the n bits of the this things. So, we really end up adding n numbers of size n and adding n numbers of n builds, we are going to take order n square bit ops.

Because, each summation takes order n time and the actually n minus 1 summation because there are n numbers, but we can do some thing better without much of an effort and this idea that we have going to present is very similar to the idea we had shown, coming from subtraction in matrix multiplication. So, let us take the number a which is n bit number. So, let us actually splits this into 2 n by 2 bit numbers and this I call us number a 0 and a 1 which simply means a 0 plus 2 to power n by 2 a 1. Similarly, we will have b 0 plus 2 power n by 2 b 1 both all these numbers a 0 a 1 b 0 and b 1 are n by 2 bit numbers.

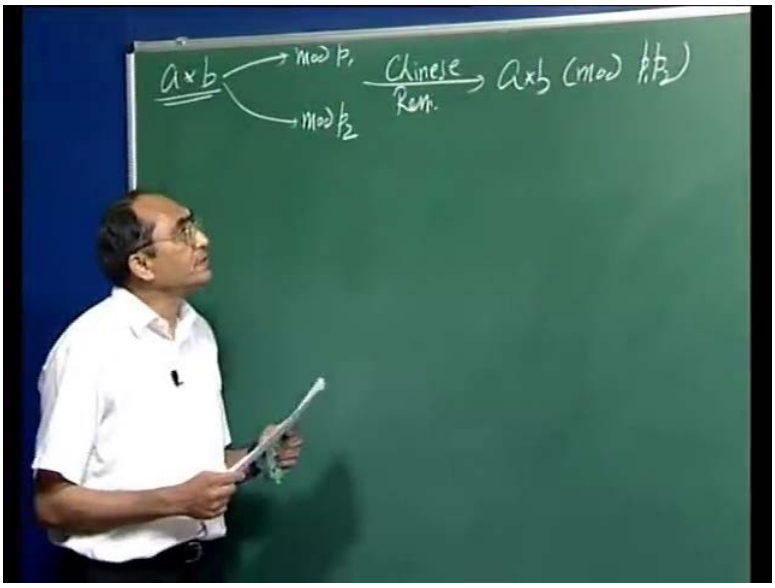
(Refer Slide Time: 44:31)



Then a times b is a 0 b 0 plus 2 power n by 2 a 0 b 1 plus a 1 b 0 plus a 1 b 1 times 2 power n. Now, consider C equal to a 0 minus a 1 and d equal to b 0 minus b 1. So, these are also only n bit numbers, n by 2 bit numbers. So, c d is nothing but a 0 b 0 plus a 1 b 1 and minus a 0 b 1 plus a 1 b 0. So, what you notice is, if I compute this product, this product and this product I get this immediately and you plug in these 3 values here, we are getting the entire product of a and b.

So, the total time complicity of these operation is, sum constant c 1 times n because we have certain linear operation in time now, we are just subtracting and adding an all. So, we are performing 3 operation of size n by 2 and some linear operations. Hence, we will get T of n to the n to the power log base 2 of 3. So, we can very quickly prove this exponent, from 2 down to log 3 of 2, but this trick does not take you much further than that. And now, we are going to discuss in this and the next lecture, something called Schonage Strassen integer multiplication algorithm. In this algorithm we are going to use two key ideas that we have discussed earlier, the first one is Chinese remaindering and second is of course, use of DFT in a modulo m ring.

(Refer Slide Time: 48:16)



In this algorithm what we will do is will compute the product that is to say, we have a times b to compute, this quantity will compute in modulo p 1. And in modulo p 2 rings which we hope to do more efficiently then directly computing this and then we are going to use Chinese remaindering, to get a times b modulo p 1 p 2 will chose p 1 p 2 which are co prime. So, this will be the chore of our idea and we will choose our p 1 p 2 in such a way that we can use this R m ring that we have discussed. So, that would be in the next lecture, so I hope to finish that lecture, this algorithm in the next lecture.