NPTEL
NPTEL ONLINE CERTIFICATION COURSE

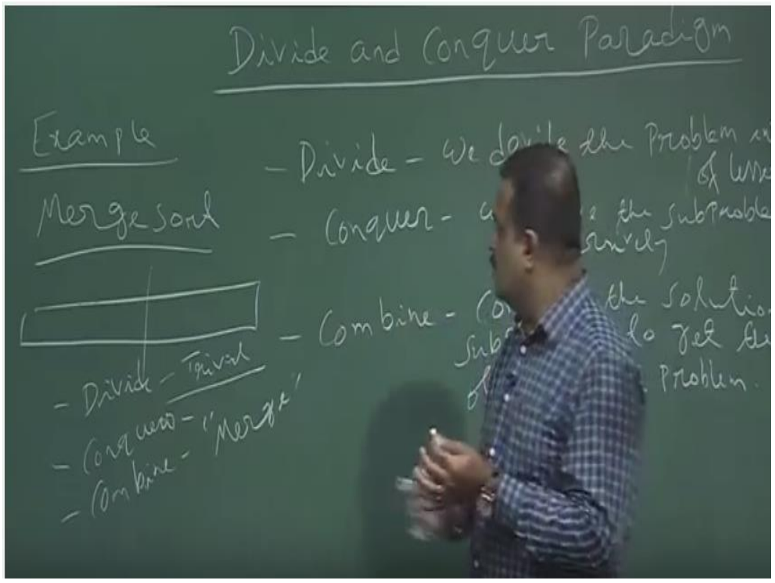
Course Name
Fundamental Algorithms:
Design and Analysis

By
Prof. Sourav Mukhopadhyay
Department of Mathematics
IIT Kharagpur

Lecture 03: Divide and Conquer Paradigm

Hi, so we talk about divide and conquer paradigm, so this is a divide technique, this is algorithm technique to solve the algorithm problem.

(Refer Slide Time: 00:33)



So it has basically three step, first one is divide step. So we have given a problem of size n, so what we do in this step we divide the problem into sub problems. We divide the problems into sub problems, problems are size. And next we conquer the sub problem by solving them recursively that is the conquer step. We solve the sub problems recursively. And the last step is the combine step, so in the combine step, so we have now solution of the sub problems.

Now to combine step we combine the solution of the sub problems to get the solution of the whole problems. So we combine the solutions of the sub problems to get the solution of the whole problem or the original problem, of the whole problem. Okay, so this is the basically the steps, so let us take an example, so we have seen the marge sort. So in marge sort what we are doing, we have given the array of size A and we need to sort it.

So this is the problem of size n, it is a sorting problem. So what we are doing, we are going to the middle of the array and we divide the array into two sub arrays. Now we recursively saw, so that is the divide step. Now we recursively solving this sub array and this sub array, that is the conquer step. Now once we have the solution for solution of this, once of these two sub arrays sorted.

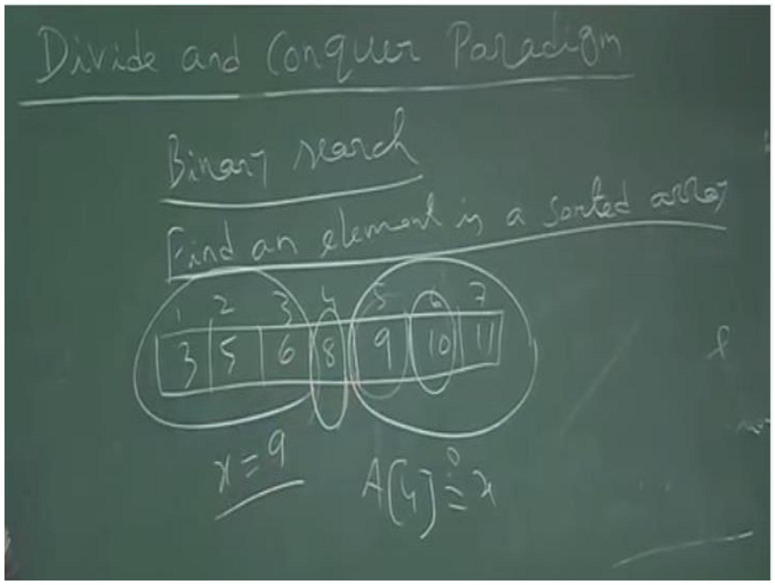
Then we are combining by calling the merge sub routine which is linear time algorithm to have a sorted array, total sorted array. So divide step is to where in the marge sort, conquer step is we are solving these two sub array recursively and then combine step is basically the merge sub routine. That is we have two sorted array, now we are calling the merge sub routine to sort the array.

(Refer Slide Time: 03:59)

So now we have got the recurrence of the marge sort which is basically T(n)=2T(n/2)+θ(n). So this 2 is basically denoting the number of sub problems, number of sub problems. And this n/2 is denoting the size of the sub problems and this is the cost for divide and combining, both for the divide and combine step, okay.

And now we know how to solve this by master method, this is basically θ(nlogn). Okay, so now we talk about some other divide and conquer algorithm. So the next one is binary search.

(Refer Slide Time: 05:15)



So in binary search what is the problem, problem is we have given a sorted array and we need to find out an element in the array whether if it is there we will return that and otherwise we will say no.

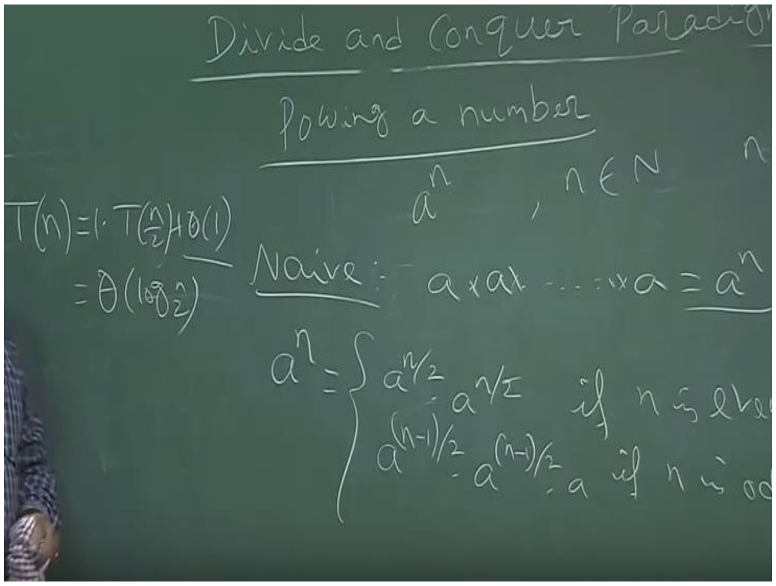
So we need to find an element in a sorted array. This is the problem. So how we can use the divide and conquer technique here, so we have a sorted array, say for example 3, 5, 6, 8, 9, 10, 11, suppose this is our given array. Now suppose we are going to find whether 9 is there or not, that is the our x, we are trying to see, we are trying to search the element 9 in this sorted array, so what we do, so we first check in the middle, so this is 1, 2, 3, 4, 5, 6, 7, so we check in the middle so middle is basically this one so A[4], so we check with A[4] whether A[4] is matching with x or not, if not we will check, if it is not matching so you are not lucky.

So we check whether 9 is greater than this middle or not, if it is greater than so we will look at the rights of array, we will again search in the rights of array. Now if it is less than we will look at the lefts of array, so that is the divide step, and in the conquered step we will be the recursively looking at the that sub array and then so 9 so we will go for here, say then we will check with this it is less than so we will look at the left sub array of this sub array so it is 9 so matching, done.

So this is the divide and conquer algorithm, so now what is the recurrence here, the recurrence is basically we are we are dividing the problem into, basically we have one sub problem we are not looking into both the sub arrays we are looking into one sub array, so this is basically one T(n/2) + the cost we are spending for dividend combining, so that is basically θ(10 constant time here we are spending.

So again by master method we can see this is basically θn so this a logarithmic algorithm for finding the, finding an element in a sorted arrays. Now the next problem is powering a number.

(Refer Slide Time: 08:31)


Divide and Conquer Paradig
Powing a number
a^n, $n \in N$
$T(n) = 1 \cdot T(\frac{n}{2}) + \theta(1)$
$= \theta(\log_2 n)$
Naive: $a \times a \times \cdots \times a = a^n$
$a^n = \begin{cases} a^{n/2} \cdot a^{n/2} & \text{if } n \text{ is eve} \\ a^{(n-1)/2} \cdot a^{(n-1)/2} \cdot a & \text{if } n \text{ is od} \end{cases}$

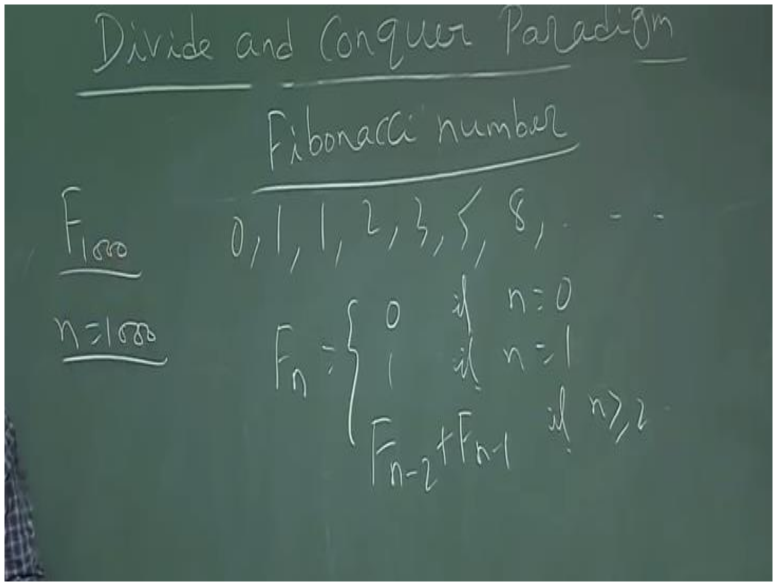
Okay, so we want to find out a^n where n is a natural number, say it could be 1000, n could be 100 something like that, so we want to find a^n so this is the problem a and n are given so you want to have, we want to find a^n so how we can do so many approaches, so we can multiply a, a…..a so n-1 times.

So that will give us a^n when the time complexity of that is basically order of n because we are doing n multiplication, now we want to see whether we can use the divide and conquer algorithm to solve this problem, okay, so for that we need to write the recursive form of this a^n can be written $a^{n/2}$ x $a^{n/2}$ if n is even, otherwise $a^{n/2 -1/2}$ x $a^{n/2 -1/2}$ x a if n is odd, so this is basically our divide step.

So we are, we have a problem of size a, we are going to find out $a^{n,}$ we divide the problem into two sub problems basically we calculate this once so $a^{n/2}$ once we have the solution of this sub problem that we will do in the conquer step then we multiply it twice we get the solution of the whole problem, so what is the recurrence, recurrence is, so we are just computing $a^{n/2}$ or this once and then in the combine step we are just multiplying if it is n is odd we are multiplying this again with n, so this is basically, so this is the size of the sub problems, we are only computing one sub problem one $a^{n/2}$ and then this the cost for divide and conquer, I mean conquer, I mean sorry divide and combine, so combine phase is basically we are multiplying two numbers.

So that is the same as binary search recurrence so the solution is this again by the master method of case 2, okay. So the next problem is finding the in a Fibonacci number.

(Refer Slide Time: 11:26)


Divide and Conquer Paradigm
Fibonacci number
F_{1000}
$n=1000$
0, 1, 1, 2, 3, 5, 8, . . .
$F_n = \begin{cases} 0 & \text{if } n=0 \\ 1 & \text{if } n=1 \\ F_{n-2}+F_{n-1} & \text{if } n\geq 2 \end{cases}$

So finding in a Fibonacci number, so what is in a, what is the first Fibonacci number we start with 0, 1 then we add these two, 1 , 2, 3, 5, 8, like this so this is we can say 0 Fibonacci number or yeah so the form is this Fn = 0 if n is 0, we can say we are starting with F0, 1 if n is 1 and the other than this we have n-2 + this is a recursive form of the Fibonacci number, okay.

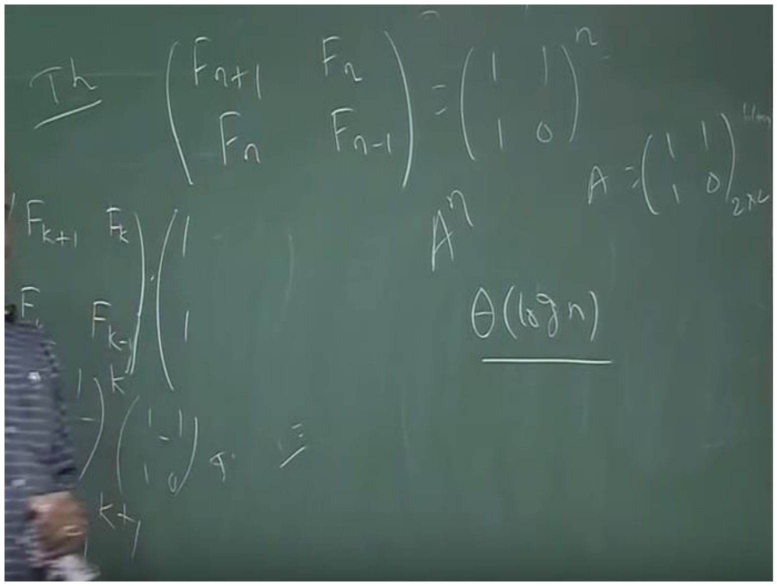
So now the question is how we can find the, in a Fibonacci number, n is another n, n is the say we want to find 1000 Fibonacci number $F_{1000's}$, so how we can do that, that is the problem, okay. So one way is data projects what, we can just keep on calculating the Fibonacci numbers.

So just like this and ultimately we will go to the F_n like this, so we will keep on calculating the Fibonacci number and you store the last two Fibonacci number and then we get the by adding these two and we get the next Fibonacci number, so what is the time complexity for that, is basically O(n), so now we have to see whether we can use some divide and conquer approach to solve this problem to find any Fibonacci number.

Okay so for that we know that in a Fibonacci number can be written as $\phi^n/\sqrt{5}$ where 5 is the golden rescue which is basically $(1+ \sqrt{5}) /2$ so one root of that, so that is golden rescue, this is also called golden rescue, now if we can compute ϕ^n then we can get that in a Fibonacci number, but this is looks like a polynomial number but here we are dealing with the real numbers. So we have to see that how we can do the error correctly, error rounding the error up.

So all those issues will come, so this is not realistic approach to get a log in time complexity. So we will look at a different method which is coming from this formula.

(Refer Slide Time: 14:17)


Th
$\begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n$
$A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}_{2\times 2}$
A^n
$\Theta(\log n)$

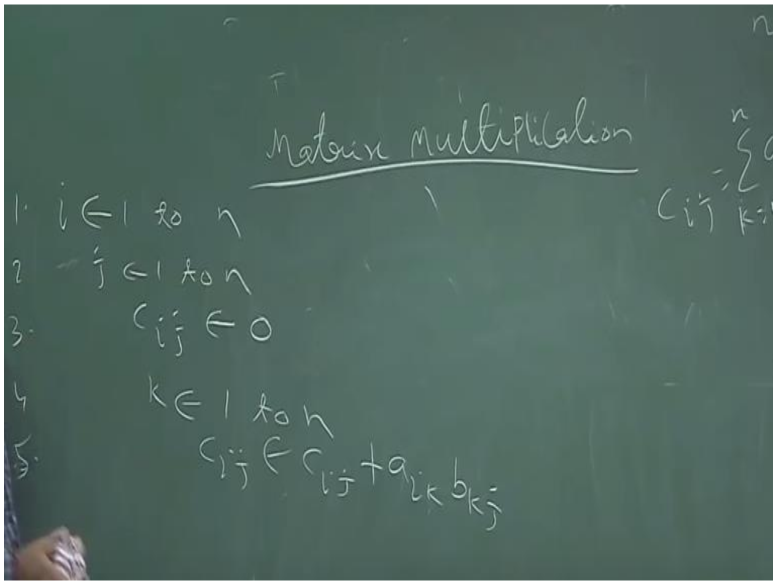
So f_n + 1, $f_{n,}$ f_n, f_n-1 these can be written as, so this can be written as 1, 1, 1, 0 to the power n so this is a theorem while f_n is denoting by the n^{th} Fibonacci number, so we can easily prove this theorem by induction so best case is, n = 1.

So for n = 1 this is second $F_{1,}$ F_0 so this is nothing but 1,1,1,0 so this is true for n =1. Now we assume this is true for n= k, so that means if k+1, f_k, $f_{k,}$ f_{k-1} this is basically 1,1,1,0 to the power k, now this is our induction hypothesis. Now we have to prove that this is true for n= k+1, okay. So for that we can just use this so k+1 so k+2, f_{k+1}, $f_{k,}$ these can be written as (f_{k+1}, $f_{k,}$ $f_{k,}$ f_{k-1}) x (1110) this we can easily check.

Now for this we are using the induction hypothesis so this is 1110 to the power k and this is (1110), so this is basically $(1110)^{k+1}$. So the result is true for n = k+1, so by the method of induction we can say that result is true for all n. So this theorem is true, now once this theorem is true now to find the in a Fibonacci number basically we have compute this a^n where a is this matrix.

So this is just a 2 b y 2 matrix so this is same as powering a number, powering a matrix but this matrix is a constant matrix so we can easily use that recurrence for this power similar to powering a number and we can solve it by log n time. So this is the divide and conquer technique for this problem, okay. So next we will talk about the matrix multiplication problem and we can see how we can use the divide and conquer approach for that.

(Refer Slide Time: 17:23)



Matrix multiplication

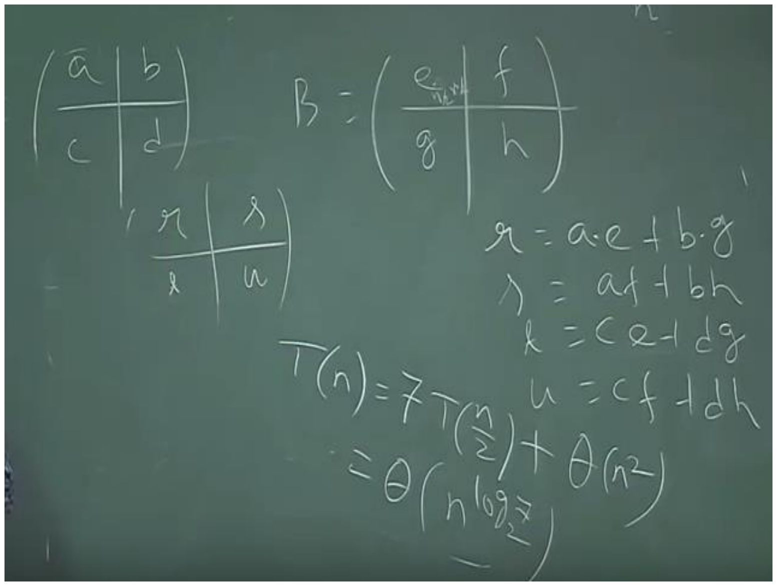
```
1. i ← 1 to n
2.   j ← 1 to n
3.     $c_{ij} ← 0$
4.     k ← 1 to n
5.       $c_{ij} ← c_{ij} + a_{ik} b_{kj}$
```

$c_{ij} = \sum_{k=}^{n}$

So okay, so suppose we have given two matrix $(a_{ij})_{n \times n}$ both are n x n and we have a C matrix which is basically their multiplication, so C is also a n x n matrix C_{ij} now we know c, C_{ij} is the summation of a_{ij}, a_{ik}, $b_{kj.}$ K = 1 to n, this is basically then we take the higher throw of a j^{th} column of b and then we do the inner product. So, so how to get $c_{ij,}$ we can have the standard c to put for this, so we are starting from i 1 to n, then j 1 to n, okay, and then we take this 0 and then we can have k 1 to n, c_{ij} is basically, $c_{ij}+a_{ik}$ x b_{kj} that is it. So this is the code, this is in, this is the standard code for matrix multiplication and what is the time complexity for this code? We have 3, 4 of our loops each of size n so the time is order of n^3.

So this is the time for problematic standard matrix multiplication formulae, matrix multiplication code. Now the question is whether we can do some divide and conquer algorithm here or not to improve this time complexity.

(Refer Slide Time: 19:32)



$B = \begin{pmatrix} e & f \\ g & h \end{pmatrix}$
$r = a \cdot e + b \cdot g$
$s = af + bh$
$t = ce + dg$
$u = cf + dh$
$T(n) = 7T(\frac{n}{2}) + \Theta(n^2)$
$= \Theta(n^{\log_2 7})$

So for that what we do, we just divide the matrix into sub matrices a,b,c,d and B e,f,g,h so this is each of this is just in by 2xn/2, this is just the sub matrices, okay. Now C we denote r,s,t,u, this is also sub matrices, so basically R is basically what, r is basically ae+bg, a is basically af+bh, t is basically ce+bg and u is basically this into this, so cf+bh, okay.

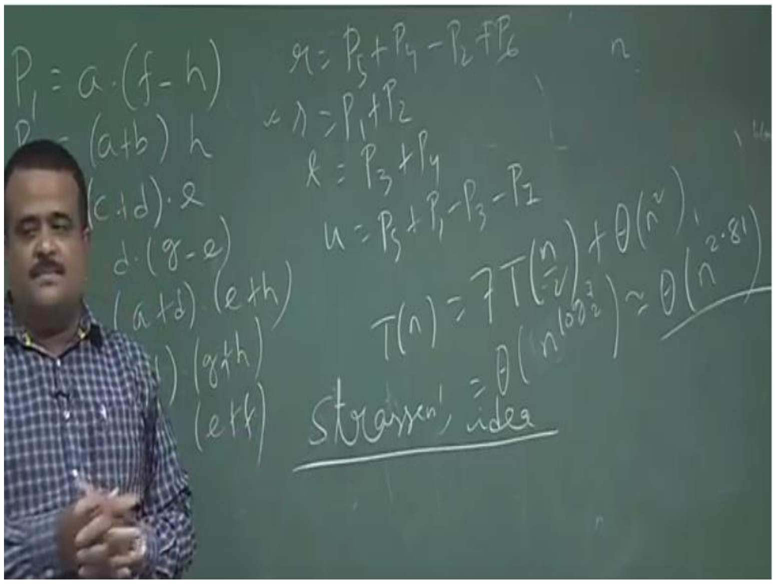
So this is the divide step, so we divide the problem into sub problems because these matrix multiplication of size, lesser size, earlier it was in cross matrix, now this is n/2 x n/2 matrix so we reduce the problem into sub problems, here size is n/2 x n/2. Now how many matrix multiplication on there, so there are1, 2, 3, 4, 5, 6, 7, 8, matrix multiplication and how many addition is there, four addition. So what is the time complexity, now in the conquered step we will recursively solve this sub problems and then once we have solution this sub problems in the combine step we will just do the addition.

So this is basically, we have 8 matrix multiplication plus the cost of the addition, we are having four addition, but each addition takes how much time so this is basically in cross n/2 x n/2 matrix P matrix plus the Q matrix, so n/2 + n/2 so this will take order of n^2 time, because basically this is a vector, so we are doing the position addition order of n^2, so this is basically order of n^2. Okay, so this is basically if we apply the master method this will give us order of n^3 which is not better than the standard matrix multiplication.

So the question is how we can improve this, so idea is, so this idea is from stressions, idea is so here we are doing 8 multiplication, instead of 8 multiplication if we can reduce just one multiplication, just seven with a more addition, so addition cost will be taken care by $\Theta(n^2)$, suppose we are instead of four addition if you are doing 50 addition, 20 addition that will be coming under $\Theta(n^2)$. So that, then in that case the time complexity will be reduced like log7 n^2, which is better than n^3.

So we will, we will discuss the stressions idea, so how stression is able to reduce the multiplication, then addition, so for that stressions is computing some intermediate terms.

(Refer Slide Time: 23:22)



$P_1 = a \cdot (f - h)$
$(a+b) h$
$(c+d) \cdot e$
$d \cdot (g - e)$
$(a+d) (e+h)$
$(g+h)$
$(e+f)$
$r = P_5 + P_4 - P_2 + P_6$
$s = P_1 + P_2$
$t = P_3 + P_4$
$u = P_5 + P_1 - P_3 - P_7$
$T(n) = 7T(\frac{n}{2}) + \Theta(n^2)$
$= \Theta(n^{\log_2 7}) \approx \Theta(n^{2.81})$
Strassen's idea

So P_2 is basically several terms, P_4 is (b x g) + (g – c) P_5 is (a+ d) x (e + h) $P_{6\ is}$ (b – d) (g + h) P_7 is (a – c) x (e + f), now r is basically $P_5 + P_4 – P_2 + P_6$, this is $P_1 + P_2$, t is $P_3 + P_{4,}$ q is $P_5 + P_1 – P_3 – P_{7,}$ okay, so this is the intermediate term so we have to we can verify, so we can verify this r here then it can remaining thing you can check whether this is okay or not.

So what is P1 of r is basically it is telling P1 + P2 now what is P1, P1 is basically this one so af – ah + P2 is ah + bh so this is basically af + bh which is basically af + bh which is basically yes, so you can just verify this is working fine, so remaining term you can just verify, so this is the idea, so look at here we are just doing the 7 multiplication to get this value P1 P2 up to P7 but we are having more addition, here we are all addition subtraction is basically addition.

So how many addition 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 this is 15 16 17 18 so we are having 18 addition but we could able to reduce the multiplication by from 8 to 7 so that is the idea so that way, so what is the time complexity, so time complexity is basically the recurrence we got here is T(n) = so 7 multiplication we reduce the problem into sub problems and how many sub problems 7 sub problems, now we recurrence only solve the sub problems and then we have this.

Combined step to get the solution of the whole problem, so this is basically so 18 addition but that will be taken care by this part so this basically by master method this is the case one, so this log of $n^{\log}$ of 7/2 which is $n^{2.81}$ which is much more better than $n^{3.}$ So this is the idea of stressions so this is the stressions idea, okay, so this is another example of divide and conquer algorithm, so we can use, we can solve this matrix multiplication problem.

With the help of divide and conquer algorithm. So in the next class we will be talking about another different conquer algorithm which is quick sort and we will see comparison of the quick sort, so we stop now, thank you.