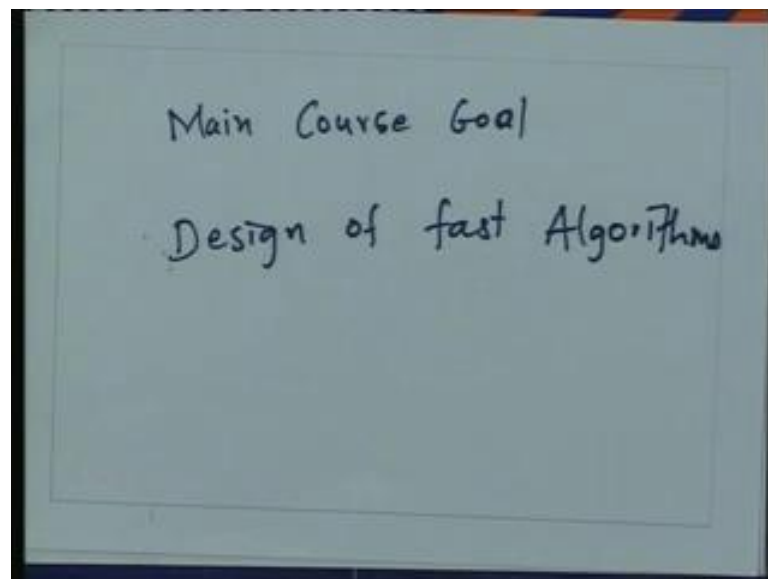


Design and Analysis of Algorithms
Prof. Abhiram Ranade
Computer Science Engineering Department
Indian Institute of Technology, Bombay

Lecture - 1
Overview of the course

Welcome to the course on design and analysis of the algorithms. Today's lecture is going to be an overview of the course and I want to introduce you to the main problems in the course and also give also convey a spirit of the course. Let us start with the fundamental question. Given a certain problem how do you solve it on a computer? For many problems it is relatively easy to design algorithms that will somehow solve them. However, quite some cleverness is needed and designing algorithms that are also fast that is algorithms which give answers very quickly. This will be the major challenge in the course.

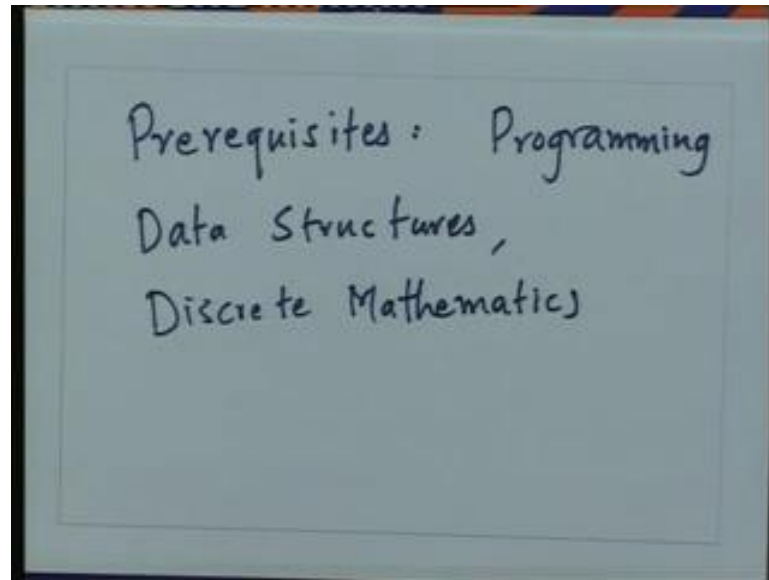
(Refer Slide Time: 01:34)



Our main course goal as said is to design algorithms which are fast. Design of fast algorithms. As you may realize designing anything be it computers, be it cars, be it clothes is an art. So, in some sense you have to be creative and it cannot be taught, but in some other sense there are also some very well defined design techniques which have evolved for this purpose. And the goal of this course is to study these techniques we will also have lots of exercises in which you can apply these ideas. And our hope is at the end

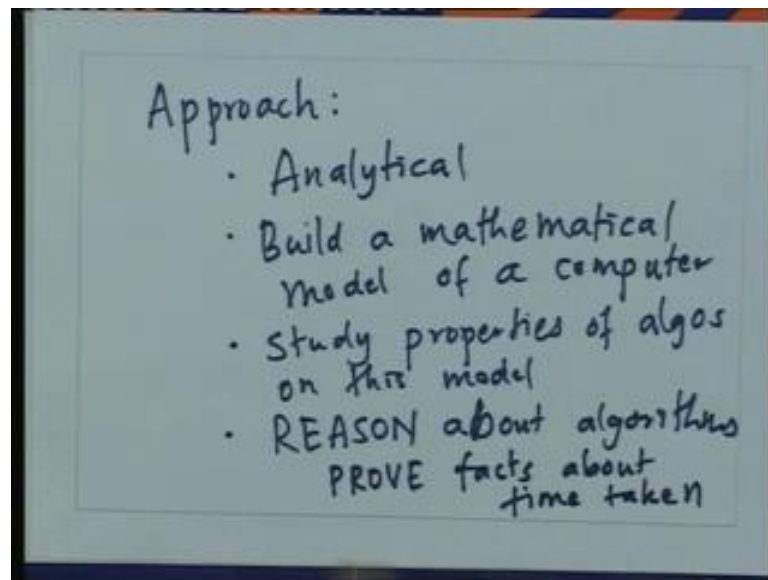
of the course you will be able to solve algorithm design problems that may that you may encounter later on in your life. There are some prerequisites for this course and let me just state them.

(Refer Slide Time: 02:36)



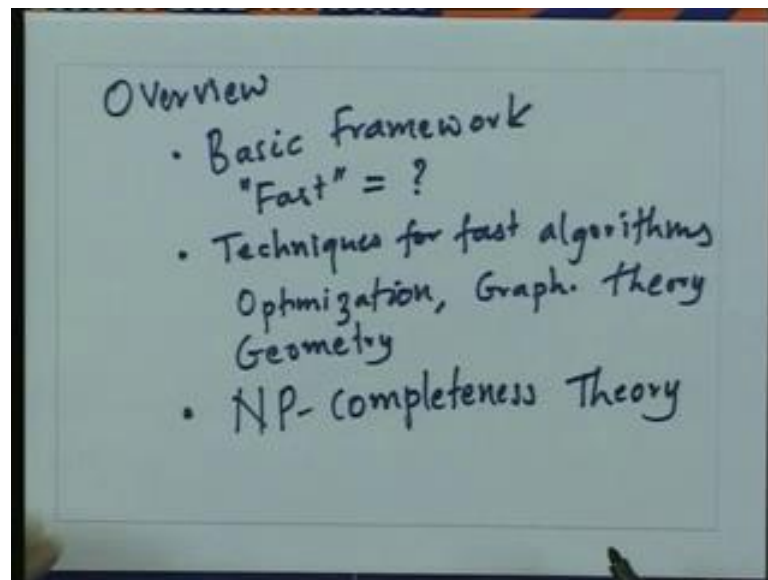
The prerequisites are that you should be familiar with some programming. You should have done some amount of programming in some common language say c or scheme or basic. You should also have done a course on data structures and you should have some discrete mathematics background. As you may see this is not an elementary course and this background is going to be necessary for us.

(Refer Slide Time: 03:27)



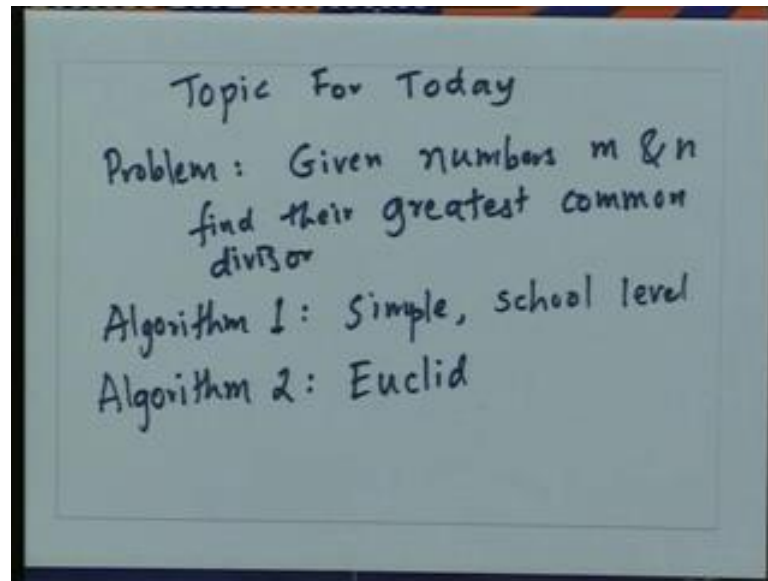
Our approach is this course is going to be analytical. In the sense that will build the model of a computer analytical will build a model of a computer that is mathematical build a mathematical model. And on this mathematical model we will be designing algorithms. So, we design algorithms and we study their properties of algorithms on this model. The whole point is to reason about algorithms. So, this is very important if we want to mathematically prove properties and we want to prove properties the emphasis on this course is proof. Prove properties about speed say prove facts about time taken. All this so there is going to be a number of things that we have to do, and of course there is a whole course for this.

(Refer Slide Time: 05:04)



So, I want to give an overview of the course. In the next few lectures we will develop the basic framework what I mean by this is that we will define the mathematical model. We will say what fast means when I say fast algorithm what do I mean? This is what we will say. Then we will embark on a fairly long stretch which involves techniques for designing fast algorithms while doing this we will also be surveying many problems. So, we will look at problems from optimization graph theory some problems from geometry also some others. It will turn out that there are some problems which do not really quite respond to our algorithm design techniques. Of course, we will see many problems for which we can design really god algorithms our techniques work beautifully, but there are also some problems where our techniques do not work so well. And for these problems a fairly intricate theory has developed over the last over the last several year's 10 20 30 years and this theory is the, so called theory of N P completeness. So, we will be studying this theory as well.

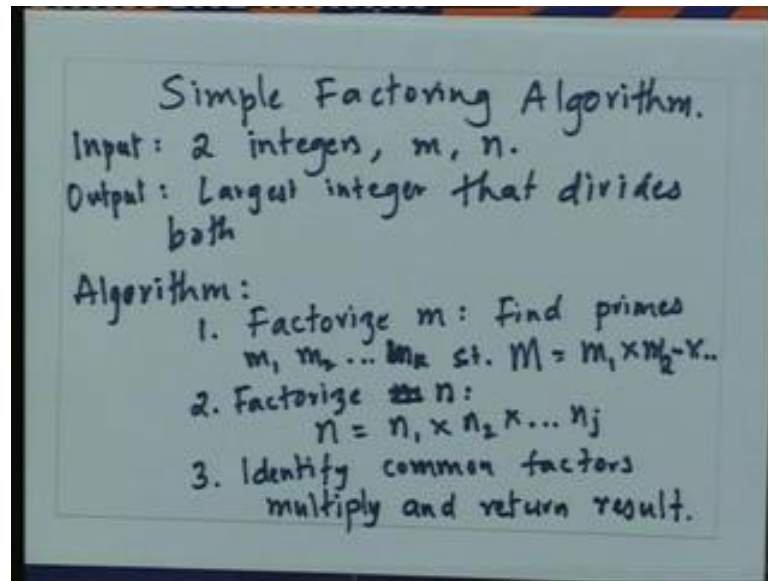
(Refer Slide Time: 06:59)



Let me now, come to the topic for today. The main topic for today. So, today I do not want to be too formal, but I nevertheless want to convey the spirit of the course to you. So, here is what we are going to do? We are going to take a fairly simple problem which everybody will have no should have no trouble in understanding and let me state that problem right away. The problem is given 2 numbers m and n find their greatest common divisor. So, everybody understands this problem. You are going to see 2 algorithms for this. One is a very simple algorithm which you probably have learnt in school. It is probably the algorithm that you were taught in first standard or fourth standard or something like that and which probably most of us used to solve this to find the greatest common divisor where we need to such as when simplifying fractions.

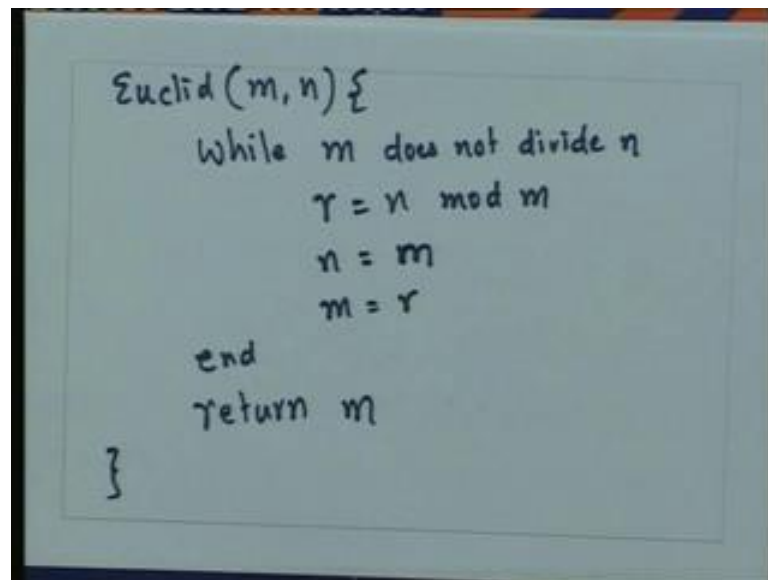
Then we will study another algorithm and this is one of the earliest algorithms which was ever invented. And this was invented by the mathematician Euclid who you might know from plane geometry. Yes Euclid did invent this algorithm even though there were no computers in his time. So, we will study Euclid's algorithm and from this we will get a sense of what a fast algorithm is you will see that Euclid's algorithm. Even though we have not defined what fast is you will intuitively understand that Euclid's algorithm must be much faster will be certainly much faster than the simple school level algorithm that we are talking about. And Euclid's algorithm is also clever and that makes it more exciting and that is also again the spirit of this course.

(Refer Slide Time: 09:31)



So, let me begin with the simple school level algorithm for factoring. So, everybody knows this, but let me state it anyway. So, basically there are 2 or 3 steps. Let me write down the input it is 2 integers m and n . What we need is the greatest common divisor which is also the largest integer that divides both and when I say divides I mean without leaving a remainder. So, here is what the algorithm will look like. So, step 1 we are going to factorize m . What does it mean? It means finding primes that we call them m_1, m_2, \dots, m_k such that m is m_1 times m_2 times all of this. The next step is to factorize n . What does that mean? Again find break n into its factors. So, write n as n_1 time n_2 times till some n_j . Note that the same factor may appear several times and of course, we are here to write it separately. The next step is to identify common factors and then multiply and return the result. We are going to take an example of this and in a minute and we will see we will see what exactly these steps do.

(Refer Slide Time: 12:19)



```
Euclid(m, n) {  
  while m does not divide n  
    r = n mod m  
    n = m  
    m = r  
  end  
  return m  
}
```

Let me now, state Euclid's algorithm. So, I am going to write this as a procedure. So, Euclid is a procedure which will take 2 arguments m and n . And I am going to invent a pseudo language as I go along and that is again going to be in the spirit of the course we are not going to be too picky about how we write down algorithms? So, long as what I mean is clear to you perfect everything is fine. So, you could express your algorithms in the most in the most suitable nicest syntax. So, that it is easy for you to get your meaning across. We will come to all this in somewhat more detail in the next lecture or, so what does Euclid do? So, the first is a check. So, we are going to check whether m divides n while m does not divide n we are going to do the following.

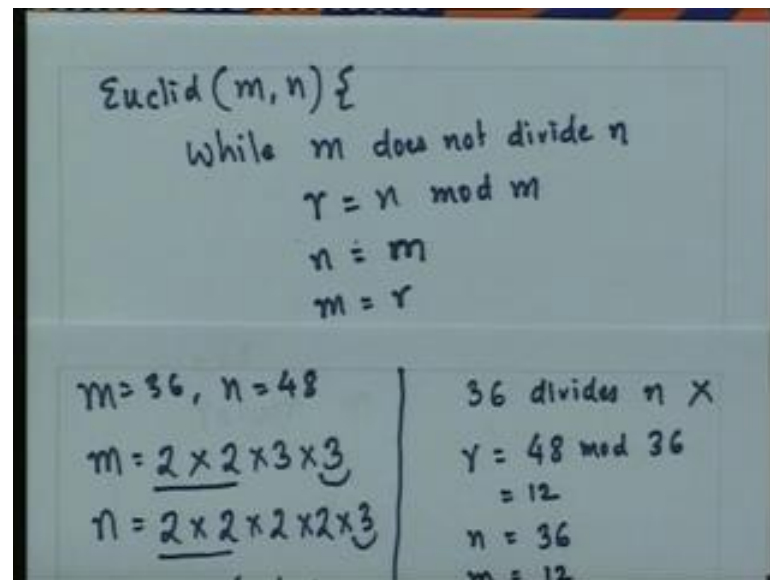
So, first we are going to calculate the remainder. So, we are going to calculate r of $n \bmod m$. Then we are going to set n equals m and then we are going to set m to be r . All these steps are going to be done inside the loop. So, that is the end of the loop over here we will prove soon that eventually this loop will have to terminate. And after it terminates all that Euclid's algorithm does is to return the value that m has at that point and that is it that is all there is to the algorithm. It is not clear when you first look at this algorithm that in fact, this algorithm works. It seems to be doing some divisions and taking some remainders, but it is not actually factorizing any of the numbers that you want who is greatest common divisors you want. So, let us now, take some examples and see whether or not this algorithm works.

(Refer Slide Time: 14:19)

$$\begin{aligned} m &= 36, n = 48 \\ m &= \underline{2 \times 2} \times 3 \times 3 \\ n &= \underline{2 \times 2} \times 2 \times 2 \times 3 \\ \text{Common factors} & \\ & 2, 2, 3 \\ \text{GCD} &= 2 \times 2 \times 3 \\ &= 12 \end{aligned}$$

So, let us first take let us first take a small example where we have m equals say 36 n equals 48. How will you do it using our simple algorithm. Well we factor m is equal to a product of prime factors. So, the idea is going to be that we are going to test numbers one after another. So, we start with say 2 and yes 2 is a factor. So, we have to write it as 2. So, then we divides 36 by 2 we get 18. So. In fact, now, you have to factorize 18 well 2 is again a factor. So, then what remains is 9. Now, 2 is not a factor. So, you go out to the next prime number which is 3 which is a factor and then only the factor 3 remains. So, we have factorized m what about n? Again we do the same thing we start with 2 yes 2 is a factor. So, that gives us 24. Again 2 is a factor that gives us 12. Again 2 is a factor. So, that leaves us 6 again 2 is a factor and then that leaves us 3. So, then we want to identify what factors are common well these 2 are common and then 1 3 is also common. So, common the common factors are 2 2 and 3 and, so G C D is equal to 2 into 2 into 3 or 12 nothing terribly difficult in this. This is all of course, school level stuff. So, let us contrast this with what Euclid's algorithm does. So, let me bring Euclid's algorithm back. So, we are going to start with m equals 36, n equals 48.

(Refer Slide Time: 17:11)



So, let me write it down as we go along. So, does m divide n ? So, 36 divides n no, so when, so therefore we enter the loop. So, now we calculate r equals $n \bmod m$. So, in other words r is equal to $48 \bmod 36$ that is equal to 12 then we calculate n is equal to m . Let me write down over here that well we know that m is equal to 36 and n is 48. So, n takes the value that m originally had. So, n is now, going to become 36 and m takes the value which r has, so r is going to become 12. So, at this point the iteration has ended. So, at this point our iteration has ended and we have left with m equals 36 and n equals 36 and m equals 12. But now, we have to go and check out the loop again, because well that is what the while loop says. So, this time n equals 36 and m equals 12 and again we had to check whether m does or does not divide 12 m r does not divide n .

(Refer Slide Time: 18:43)

$m = 36, n = 48$ $m = 2 \times 2 \times 3 \times 3$ $n = 2 \times 2 \times 2 \times 2 \times 3$ Common factors 2, 2, 3 $GCD = 2 \times 2 \times 3$ $= 12$ 9 divisions	$36 \text{ divides } 48 \times$ $y = 48 \bmod 36$ $= 12$ $n = 36$ $m = 12$ $12 \text{ divides } 36 \checkmark$ return 12 2 divisions
---	--

So, does 12 divides 36. This time the answer is yes and therefore, we are going to exit the loop. And in fact, at the end of it we are going to return the current value of m which is equal to 12, so we will be returning 12. So, let us now, compare these 2 algorithms. So, here you can see roughly what the work done by the 2 algorithms is. The simple algorithm had to do factorizations then it had to collect common factors and it had to do it had to multiply the factors together and return the answer. Euclid's algorithm on the other hand did 1 division. So, first time it divided 36. It divided it checked if 36 divides 48 which is the value of n that we had. And when it found that answer was false the division was not possible then it took the remainder. Then it simply exchanged the numbers basically and then again it did 1 more division.

So, I can summarize the work over here by saying that this did roughly 2 divisions. How much work does this do? Well it calculated these, so many factors. So, it calculated 4 factors, so it had to at least do 3 divisions to get each factor. Here it calculated 5 factors. So, it had to at least do 4 divisions. So, it at least did 9 divisions probably it did more. You can see in a quick example immediately where it will have to do many more than these. As you can see in this school level very simple factoring algorithm we use 9 divisions whereas, in this somewhat sophisticated algorithm we use really 2 divisions. So, clearly you have done less work. And although i have not proved yet that Euclid's algorithm in fact, works you can see that it is returning the correct answer. So, but we will show later on that. In fact, Euclid's algorithm does do does work correctly.

(Refer Slide Time: 21:17)

$$\begin{aligned}m &= 434, n = 966 \\m &= 2 \times 7 \times 31 \\n &= 2 \times 7 \times 139 \\ \text{Common factors} &: 2, 7 \\ \text{GCD} &= 14.\end{aligned}$$

So, let me take one more example m equals 434, n equals 966. So, these are somewhat large numbers and let us see what will happen over here. So, suppose you want to factorize these numbers first of all it will take some work, but let us see what the answer is going to be. So, m can be written as 2 times 7 times 31, n can be factorized as 2 times 7 times 139 what is the answer? The common factors are 2 and 7 and the G C D therefore, is 14 . Let us now, come back to Euclid's algorithm.

(Refer Slide Time: 22:31)

$$\begin{aligned}\text{Euclid}(m, n) \{ \\ \text{While } m \text{ does not} \\ \quad r = n \bmod m \\ \quad n = m \\ \quad m = r \\ \text{end} \\ \text{return } m \dots\end{aligned}$$
$$\begin{aligned}m &= 434, n = 966 \\ r &= 966 \bmod 434 \\ &= 98 \quad 966 = 2 \times 434 + 98 \\ n &= 434 \\ m &= 98\end{aligned}$$

So, let me write down again m is equal to 434 and n is equal to 966. The first step is to compute the remainder of $n \bmod m$. So, then r is going to be according to this step r is going to be $966 \bmod 434$. Well 966 is equal to 2 times 434 which is 868 plus 98 under 4 I can write down r as equal to 98. After this we just had to exchange values basically. So, now, n is going to take the old value of m . So, n is going to be 434 and m is going to take the value of r , so which is 98. So, at the end of one iteration of Euclid's algorithm we have new values of m and n and these are the new values. So, we started off with the values 434 and 966. After 1 iteration we have the values 434 98 and 434 in that order m and n . So, now, we just have to repeat the same thing with these new values, so this is the end of the first iteration. And in the second iteration we are again going to calculate r equals $n \bmod m$.

(Refer Slide Time: 24:26)

Euclid(m, n)

while

end
return

}

$n = 434$
 $m = 98$

$r = 434 \bmod 98$
 $434 = 98 \times 4 + 42$
 392

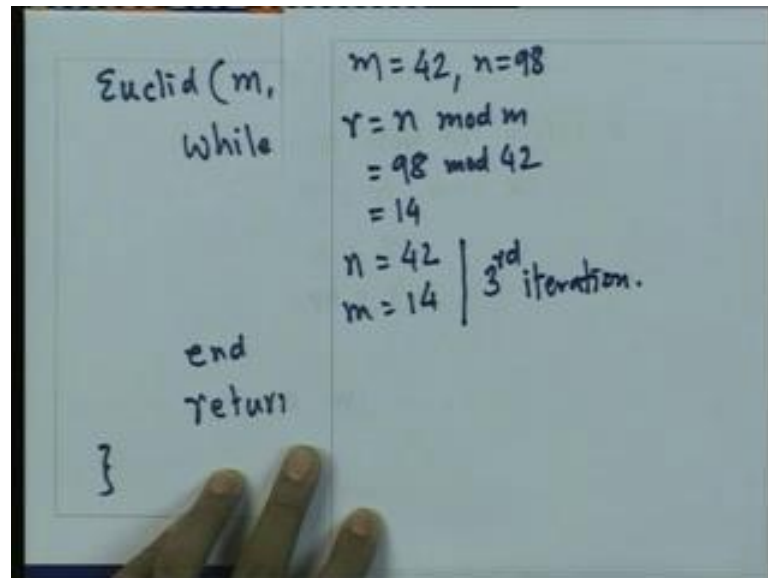
$= 42$

$n = 98$?
 $m = 42$ }

So, let us do that. So, if you calculate r equals $n \bmod m$ we are going to get the new value of r to be equal to $434 \bmod 98$. So, you have to do the division over here and which is 434 can be written as 98 times 4, so this is 392, so plus we are going to get 42. So, r is going to be equal to 42, but of course, when we began this iteration we had to check whether m does or does not divide n or whether 98 does or does not divide 434, but clearly it is leaving the remainder and therefore, we will enter this iteration. After that we are going to set n equal to m , so m the value of n is 98 and m will be equal to r which is 42. So, at the end of the second iteration these are the values that we have. So, n

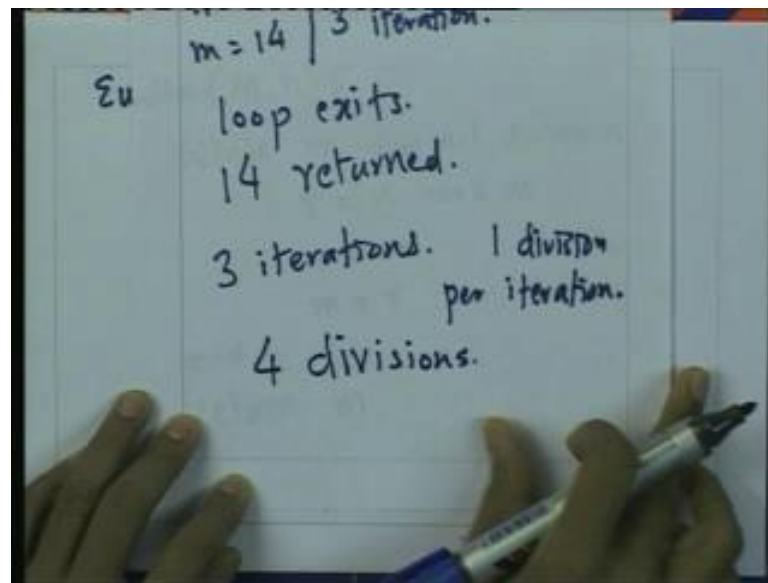
is equal to 98 and m is equal to 42. So, we will have to go into the third iteration for the beginning of the third iteration, let me just write down the values.

(Refer Slide Time: 25:56)



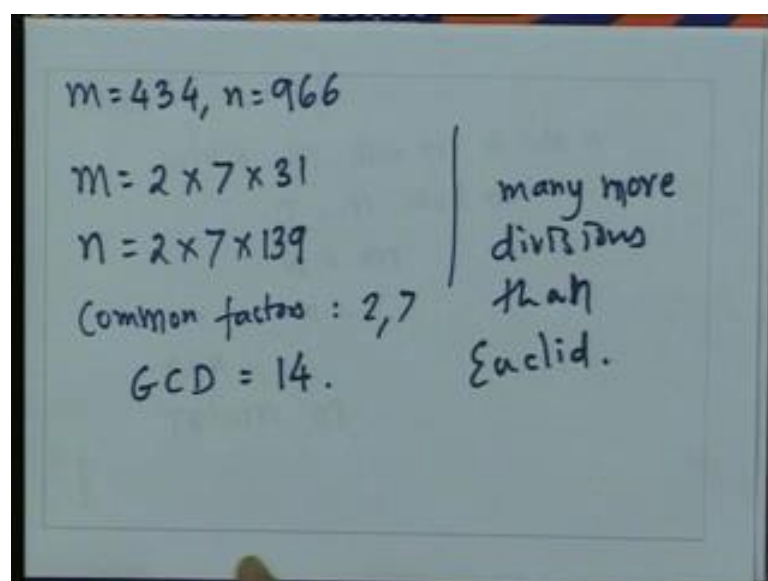
So, we have m equals 42 n equals 98 again we are going to check does m divide n. It does not and, so we will enter the loop when we enter the loop we have to find out r equals n mod m. So, that is equal to 98 mod 42. 42 times 2 is 84, so 98 mod 42 is 14. Then there is just a matter of setting n and m correctly. So, n will now, equal the old value of m. So, this is 42 and m will equal the value of r we just calculated. So, it is going to be 14. So, these are the values at the end of the third iteration. So, now, again we are going to come back and execute this loop and again we are going to check does m divide n or not. And this time we will see that 14, in fact does divide 42 and at this point the loop will be exited.

(Refer Slide Time: 27:20)



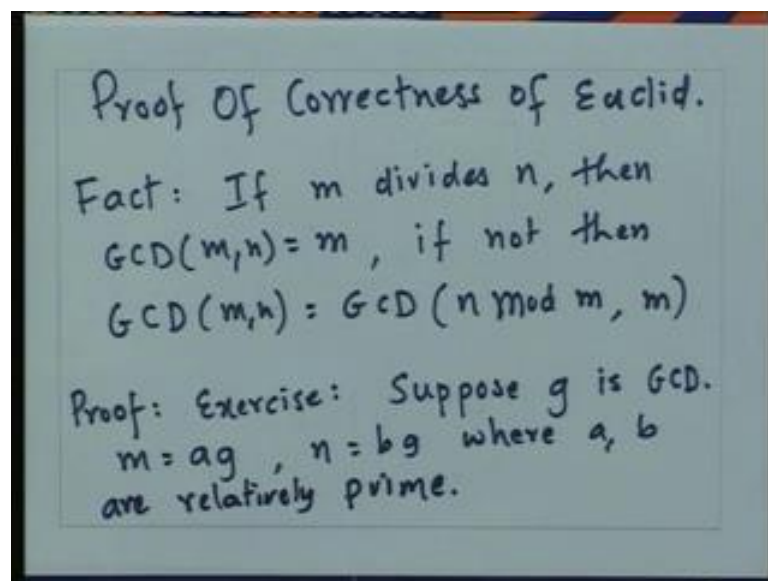
And at the end we are going to return the result as m . The value of m right now is 14 and therefore, 14 will be returned. So, what has happened? We took 3 iterations. So, summary of this all this is we took 3 iterations what happened in each iteration well we did 1 division per iteration. And before quitting from the loop we have to do one more iteration one more division and therefore, that took 4 divisions essentially. So, Euclid's algorithm for this somewhat complicated problem m equals 434 n equals 966 where we are asking to find the greatest common divisor of m equals 434 and n equals 966 took 4 divisions. Let us now, check what happens with the simple algorithm.

(Refer Slide Time: 28:36)



So, here where the factors that we found. So, how many divisions did it take? At first glance you might see you might think that it is only going to take 2 divisions to calculate m and 2 more divisions to calculate n , so which is 4 divisions, but that is not quite correct. To check that there is no factor between 2 and 7 we would require checking for 2 3 for 3 five as well. Now, after dividing by 7 we had to check whether 139 is a prime. So, that would also involve checking all the numbers until 39. So, that would require substantially many divisions. So, here again we require many divisions many more than Euclid's algorithm. In fact, you will see that if we take bigger and bigger numbers factoring them becomes much harder. You will have to do a lot more work whereas, in Euclid's case we will just do divisions and we are we will. In fact, show now that as we do the divisions then numbers will become smaller and smaller and the algorithm will terminate. So, the next topic that we are going to get into is I want to argue that Euclid's algorithm actually works.

(Refer Slide Time: 30:10)

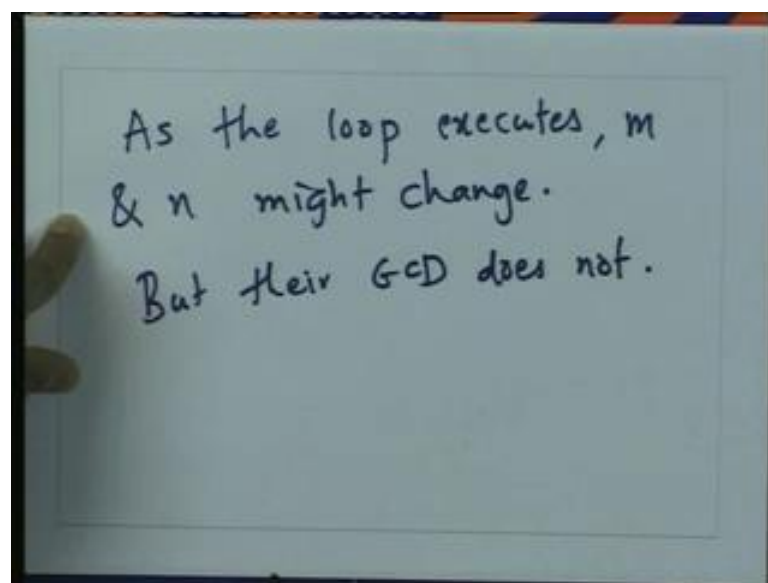


I am not going to give a fairly very detail proof I just want to indicate the main idea. The main idea is really a fact about divisibility and remainder and things like that, so let me indicate that. So, this idea says that if m divides n then G C D of m and n must; obviously, be equal to m , because clearly m will be the largest number which divides both m and n . If not then we can write G C D of m and n is the same as G C D of n it is the same as the G C D of $n \bmod m$ and m how do you prove this? I will leave it as an exercise, but let me just mention the main idea. The main idea is suppose g is the G C D

then we can write m as a times g and n as b times g where a and b are relatively prime having written them in this manner you should be able to just substitute into what we have in the fact and then you should be able to get the answer. Actually the fact itself is very similar to what we had in Euclid's algorithm in the fact that we have written down. So, let me just show you the iteration the loop part of Euclid's algorithm. So, the fact says that if you want to calculate the G C D of m and n then you might as well calculate d G C D of $n \bmod m$ and m what does our loop do?

It wants to calculate we want to calculate d G C D of m and n . It first checks what the remainder is and then it calculates the fact is essentially $n \bmod m$. So, it is essentially this term over here and we need and then it sets m to be equal to r . So, the first argument is set to r and the second argument is set to the old value of m , but this is precisely what the fact says. The fact says that if you want to calculate G C D of m and n instead calculate G C D of $n \bmod m$ and m . In fact, once we have once we are given this fact the proof of Euclid's algorithm the correctness of Euclid's algorithm is at least partially done. Because what we have accomplished is that we are sure that as we go through iterations we will never be we will always maintain integer's m and n and who's G C D will be the G C D of the original values of m and n . So, this is the invariant that we are going to maintain. So, in the specific value of m and n might change, but their G C D is never going to change. So, let me write this down.

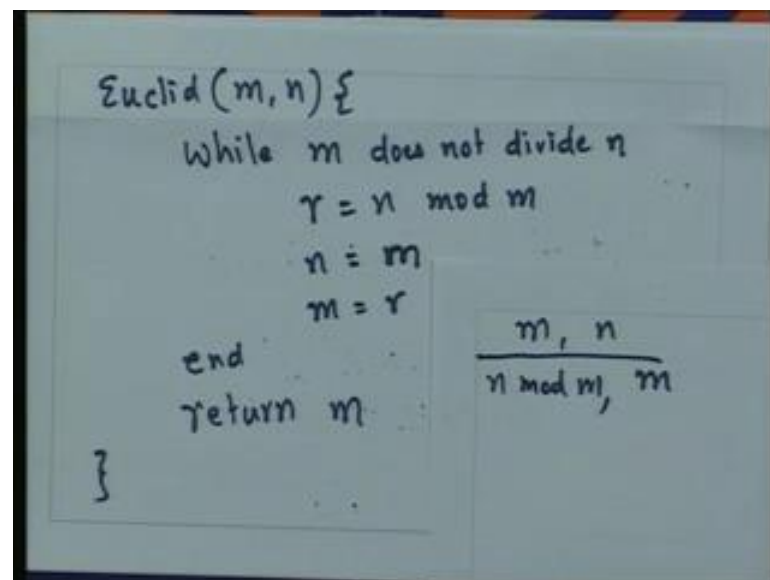
(Refer Slide Time: 34:45)



So, as the loop executes m and n might change, but their G C D does not. And hence what we have is that we will preserve the G C D and eventually when we exit the loop will exit with the same G C D. We just established that as the loop executes m and n might change, but curiously enough their G C D continues to remain the same. As a result if we ever get out of the loop then that will be, because m divides n , but notice that even at this point the G C D will still be the same as the old G C D the G C D of the old m and n .

But if m divides n then the G C D will simply be n and that is what you will return and therefore, in fact, we have established that if the loop in fact, terminates then we will be returning the correct value. Now, you want to argue why the loop should terminate? Why we need to exit? Why we will always exit from the loop? This is actually fairly straight forward. Let us examine Euclid's algorithm again and see what happens in each iteration. So, long as the loop is executing. So, initially the values of m and n might be taking some values. What are the values of m and n after one iteration? Well, the first step is calculating r , but notice that immediately afterwards we are setting n to have the value m .

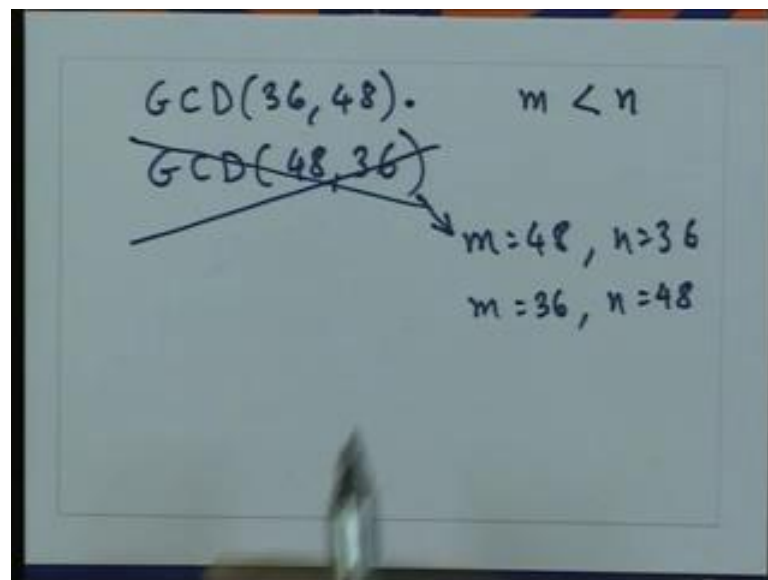
(Refer Slide Time: 36:42)



So, if the original values are m and n then at the end of 1 iteration n will have the value m and what will the value of m be m will have the value r . So, it will be $n \bmod m$. So, what has happened is that the values m and n have now, changed to $n \bmod m$ and m , but $n \bmod m$ anything $\bmod m$ is actually going to be smaller than m itself. So, notice that the

first argument is always continuously is going to decrease in every iteration of the loop. The value of m is always going to decrease. How long can it keep on decreasing? Well, it has to remain positive. It cannot even become zero and it might decrease. It will have to decrease at least by one in each iteration. As a result of this we can conclude that the loop has to exit at some point and we know that if the loop exits then the correct value is returned. So, that proves that Euclid's algorithm is correct. Our next step is to argue that Euclid's algorithm in fact, runs reasonably quick. So, the basic idea we want to prove is going to be something like this. Well before that we are going to assume we are going to we are going to prove something about cost Euclid's algorithm and in doing this we are going to assume that the first argument that we sent to Euclid is always smaller than the second.

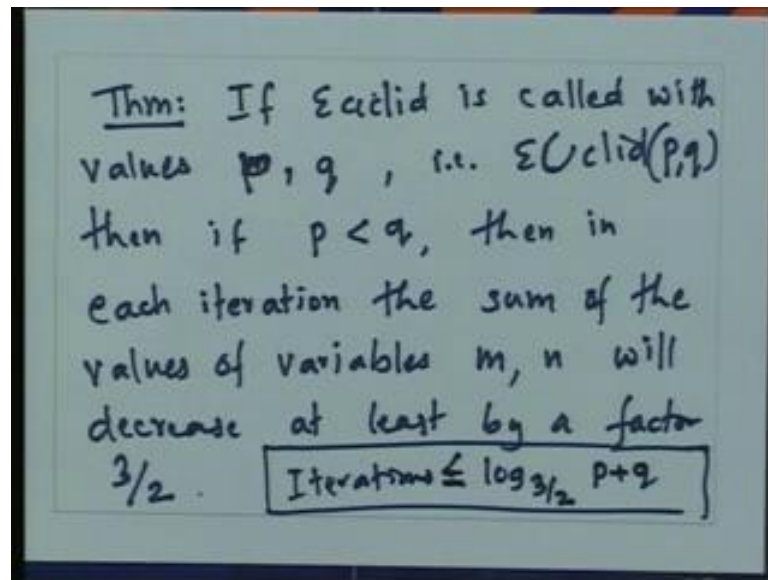
(Refer Slide Time: 38:21)



So, this means that we will always call $GCD(36, 48)$ and not $GCD(48, 36)$. This is only for the purpose of analysis. Actually the algorithm as we wrote will work fine if we make this call as well. If we make this call you will see that internally we will start with m equals 48 n equals 36, but in one iteration we will be exchanging these values. The first iteration will only be spent for exchanging these values. So, m will become 36 and n will become 48. So, then we can as good as assume that in fact, we will start off our execution in this manner. So, we will assume in fact, that we are not analyzing this because that only adds 1 iteration. So, our assumption is that we are analyzing calls of this kind where m is less than n well of course, if m is equal to n then there is no

question. So, the loop wills the Euclid will immediately return . So, let me now state the main result that we want to prove. The main result that we want to prove is something like this.

(Refer Slide Time: 39:44)



So, if Euclid is called with values m with values p and q that is Euclid of p, q is called I want to make a distinction between the variables m, n and their values. So, we will think of p, q as values and m, n will be variables. So, you are going to call Euclid with values p and q then if p is less than q then in each iteration the sum of the values of variables m and n will decrease at least by a factor 3 halves. So, initially if the sum is 6 then in one iteration the sum must go below 4 become 4 or go below 4. If it is 16 in 1 iteration it must go below 40 or stay or become 40. What is the implication of this theorem? This theorem establishes that the sum will drop very fast. And in fact, the number of iterations this establishes that the number of iterations is equal to log to the base 3 halves of p plus q . Well, it is less than equal to, so this theorem will put a good upper bound on the number of iterations that Euclid is going to execute. And therefore, all that we need to do is to prove this theorem and we have a bound on how many iterations Euclid takes. So, our goal now, is to estimate what happens to the sum of the values taken by m and n in each iteration. So, let us say that we start off the whole process and at the beginning of the iteration m and n take values p and q .

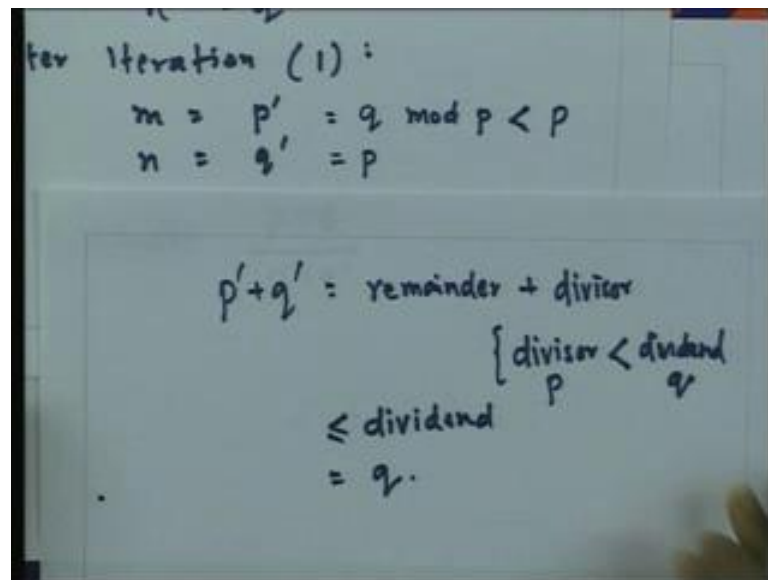
(Refer Slide Time: 42:27)

Beginning of iteration:
 $m = p$
 $n = q$
After iteration (1):
 $m = p' = q \bmod p$
 $n = q' = p$
Estimate $\frac{p+q}{p'+q'}$

So, at the beginning of iteration m is equal to p and n is equal to q . After the iteration and I mean just one iteration let us say m takes the value p prime n takes that value q prime. So, what is our goal we want to estimate by how much the whole thing drops. So, you want to estimate p plus q upon p prime plus q prime. So, if we prove that this ratio is at most 3 halves then we are done. So, all that remains now, is to express p prime and q prime in terms of p and q . So, for that will need our Euclid procedure again.

So, what does the Euclid procedure do if it does not terminate and that is the case that we are looking at currently? Then it computes r equals $n \bmod m$ and then set n equals m . So, in terms of this the new value of n is the old value of m . So, going back to this the new value of n must be the old value of m . So, therefore we get q prime must be equal to p . The new value of m is the value of r which is the old value of $m \bmod$ old value of $n \bmod$ old value of m . The old value of n is simply $q \bmod m$. The old value of m is p and therefore, this whole thing p prime is $q \bmod p$ we need one more fact in addition to all this which concerns p prime plus q prime.

(Refer Slide Time: 45:26)



Handwritten mathematical derivation on a whiteboard:

Iteration (1):

$$m = p' = q \bmod p < p$$
$$n = q' = p$$

$$p' + q' = \text{remainder} + \text{divisor}$$
$$\leq \text{dividend}$$
$$= q$$

Side note: $\begin{cases} \text{divisor} < \text{dividend} \\ p < q \end{cases}$

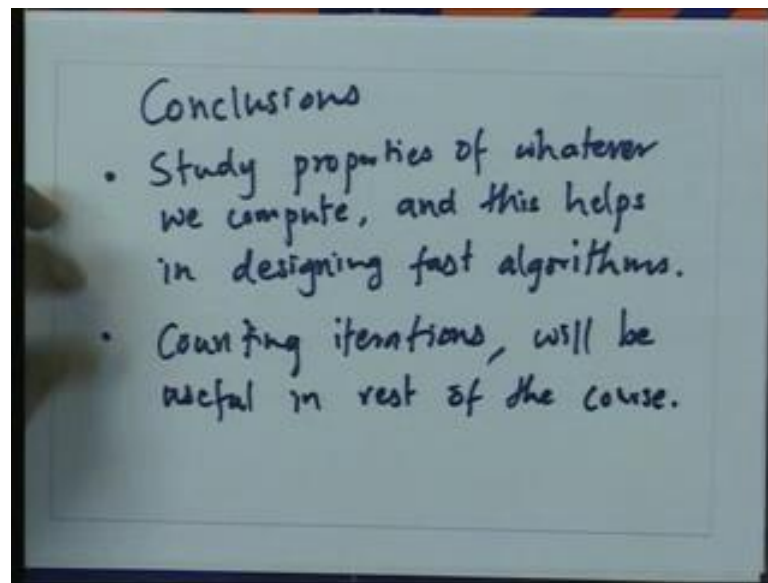
Well, what is p prime plus q prime? P prime is the remainder when q is divided by p and q prime is p itself. So, this is remainder plus divisor and we know that the divisor is strictly less than the dividend. Divisor in our case is p and the dividend is q and we assume that p is in fact, less than q . And therefore, we can conclude that this whole thing has to be less than or equal to the dividend. In other words this is q , so here are the 3 facts that we wanted p prime plus q prime is less than q . Let me align it q prime is equal to p and p prime is less than $q \bmod p$. If p prime is less than $q \bmod p$ then I can conclude that p prime has to be less than both. In fact, it has to be less than p . So, now, it is just a matter of algebra we are going to add this and this and this last inequality, but just to get the terms right we will multiply this last inequality by 2 times. So, if we do that let me adjust this.

(Refer Slide Time: 47:50)

$$\begin{aligned}
 p' + q' &= \text{remainder} + \text{divisor} \\
 &\leq \text{dividend} \\
 &= q. \\
 p' + q' + 2(p' + q') &< p + p + 2q \\
 3(p' + q') &< 2(p + q) \\
 p' + q' &< \frac{2(p + q)}{3}
 \end{aligned}$$

So, if we do that addition we are going to get p prime plus q prime plus p prime plus q prime this comes from over here. The whole thing is less than this p plus this p plus this q, but let us take this 2 times therefore, we will get 2 q. So, now it is just a matter of simplifying this. So, what is this equal to this is nothing but 3 times p prime plus q prime and we have shown that it is less than 2 times p plus q and this is exactly what we wanted. So, we have been we have showed that the original value of the sum has reduced by a factor of two-thirds. So, p prime plus q prime is less than p plus q upon 3 halves that concludes the analysis of the algorithm. And we have been able to show that in fact, the algorithm will execute a fairly small number of steps.

(Refer Slide Time: 49:19)



So, let me now conclude this lecture and highlight the main points. The very first point that I want to make is that is the difference between the 2 algorithms. The school level algorithm basically uses the definition. Euclid's algorithm uses some more interesting deeper properties deeper mathematical properties of the quantities that we are going to calculate. So, we study properties of whatever we computing and this helps in designing fast algorithms. There is also one more point that I want to make which is that the basic way in which we did this analysis counting iterations will be useful in the rest of the course and the precise details of all of this we will cover in the subsequent lectures. So, that marks the end of this lecture.