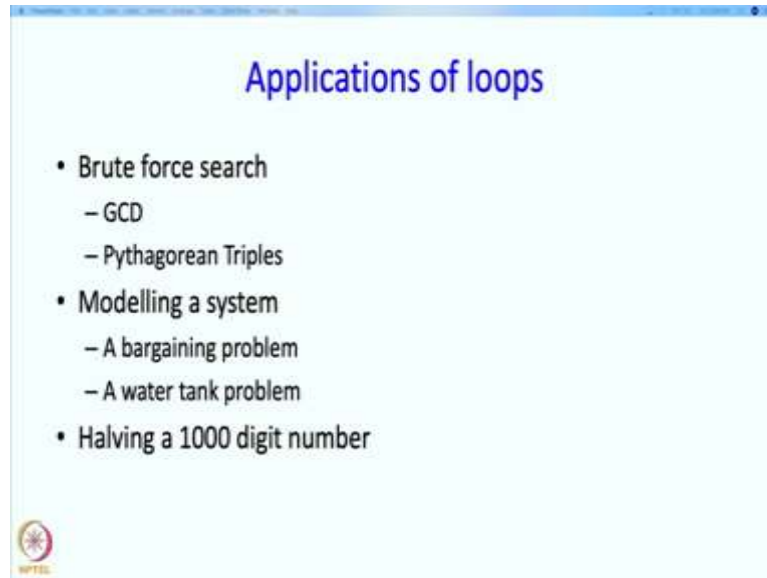


**An Introduction to Programming through C++**  
**Professor Abhiram G. Ranade**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology Bombay**  
**Lecture No. 09 Part - 1**  
**Loops in Various Applications**

(Refer Slide Time: 00:32)



Hello and welcome to the NPTEL course on an introduction to programming through C++. I am Abhiram Ranade, the topic for this lecture sequence is Loops in Various Applications. So we are going to study three themes today, one theme is brute force search. So often we find that we may not have any clear clever algorithm to solve a certain problem. However, we can identify potential solutions and we can just examine all of them. On a computer this can happen very fast even if the number of possible candidate solution is large, but finitely many. So this turn out to be an interesting way of solving problems. So we are going to see two examples of this.

Then we are going to talk about modeling a system. So, in this we will have some kind of a system which will respond to stimulation from the outside world. The first system we consider will be rather simple it will respond to the outside world. But it will not really have any state. So between responses it does not have to remember much.

The second system we talk about will be affected by what the world does to it and its responses will therefore, depend upon what has happened earlier. So modeling such systems is a very important use of computers and we will see that, just by using a single loop you can sort of do a lot. You can model systems very very nicely.

And then we will consider a problem relating to computer arithmetic. So, we will talk about arithmetic on integers which have many many digits. So as you know such integers cannot be easily processed. So we will see what kinds of things need to happen for us to be able to process such large integers.

(Refer slide Time: 02:43)

### Constraint satisfaction problems on finite domains

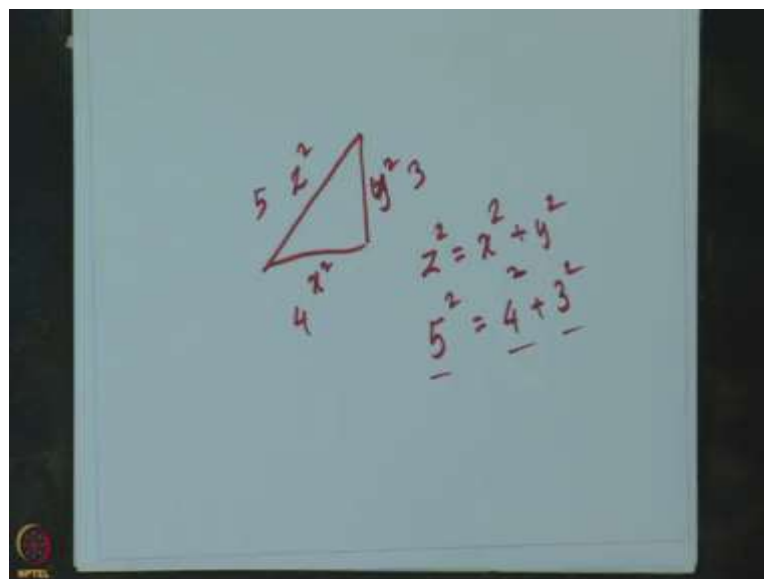
**Constraint satisfaction problems:**  
Find  $x, y, z, \dots$  such that they satisfy constraints ...

**Constraint satisfaction + optimization:**  
Find  $x, y, z, \dots$  such that they satisfy ... and ... is as small (large) as possible.

**Examples:**

- GCD of integers  $A, B$ : find  $x$  such that  $x$  divides  $A, B$  and is as large as possible.
- Pythagorean triples: Find  $x, y, z$  all distinct s.t.  $x^2 + y^2 = z^2$  and  $z \leq 20$ .

- A simple way to find GCD:
  - $x$  must be at least 1
  - $x$  cannot be larger than  $\min(A, B)$
  - Domain of  $x$ :  $\{1, 2, \dots, \min(A, B)\}$
  - "Try out each integer between 1 and  $\min(A, B)$ . Pick the largest one that divides both"
- A way to find Pythagorean triples
  - $x, y, z$  must be at least 1, at most 20.
  - Domain of each:  $\{1, 2, \dots, 20\}$
  - "Try out all possible triples"
  - At most  $20 \times 20 \times 20 = 8000$



So, let me begin with this idea of brute force search. So to tell you that, I need to tell you about constraint satisfaction problems on finite domains. So, a constraint satisfaction problem looks like this, it asks find  $x, y, z$  such that they satisfy some set of constraints. You can have a constraint satisfaction together with optimization. So, this problem looks like find  $x, y, z$  such that they satisfy these constraints and in addition something should be as small as possible, or may be as large as possible.

So, here are a few examples, so we have seen the GCD problem earlier. So, the problem is given two integers A and B, we to find their greatest common divisor. Now, we can think of this as a constraint satisfaction problem. How? Well, x is what we want to find and the constraints that x has to satisfy are that x must divide A and B both. So it must be a common divisor, it must divide A and B both without leaving a remainder. And there is also an optimization angle. We want x to be as large as possible, satisfying these constraints.

So earlier we have seen algorithms for solving this problem. Now, those algorithms in particular Euclid's algorithm was a very clever algorithm. What is we are going to do today is, solve this using a very simple algorithm. But it will take a long time, but that is okay. Suppose, we are not clever enough and suppose, or suppose in some problem we are not able to have any clever algorithms. Then the approach that we are going to talk about today, will still be useful.

So, that is the point of discussing GCD so we want to, we have formulated GCD as a constraint satisfaction and optimization problem, and then we can use the idea of brute force search as we will see soon.

The second problem is that of discovering or that of listing out Pythagorean triples. So specifically we want to find x, y, z integers such that,  $x^2 + y^2 = z^2$ , and we want these numbers to be distinct. And we have put down a condition, additional condition that these numbers should be smaller than 20. Why are these called Pythagorean triples? Well, you know the theorem of Pythagoras. So it says that, if you have a right angle triangle then, the sum of the squares of the two sides. So, let us say x and y are side lengths and the squares are  $x^2$  and  $y^2$ . Then, the sum of the squares is equal to the square of the hypotenuse. So  $z^2 = x^2 + y^2$ . You know that you can have a Pythagoras and a right angle triangle, of side length 5, 4 and 3. Because 5 squared is equal to 4 squared plus 3 square. And because of the theorem of Pythagoras, these numbers 5, 4, 3 constitute what is called a Pythagorean triplet.

So, people have curious about such numbers and what we are doing over here, is we are going to look for them. We will not do anything clever, but still will be able to find such numbers reasonably easily. So here is, the simple way of finding GCD that, I was talking about so, so let us make the first observation which is that the GCD itself, the greatest common divisor of any two numbers must be at least 1. And the common divisor cannot be larger than the smaller of the two numbers than the minimum of the two numbers. So, what is its mean? So this means that x can potentially only be one of these numbers. 1, 2

all the way till minimum of A and B, it cannot be a number beyond this, it cannot be number smaller than this.

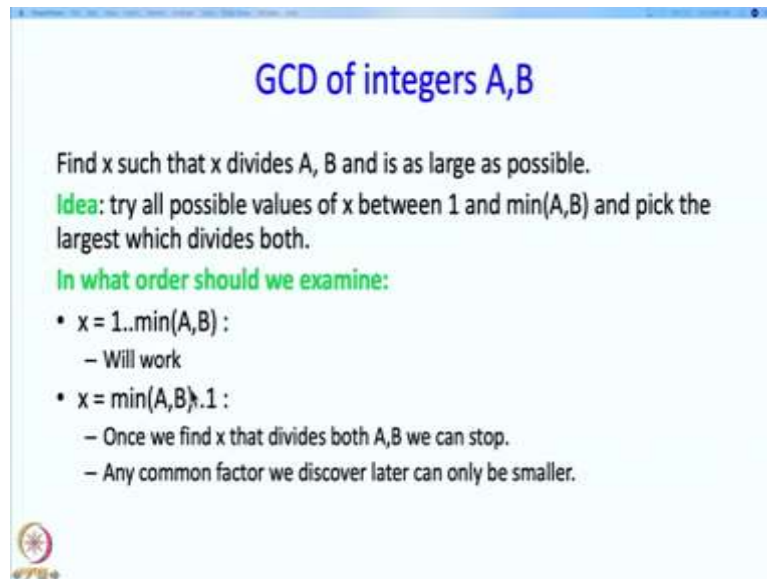
So, here is the strategy, so we are going to tryout each integer between 1 and then A, B and we are going to check, does that integer perfectly divide both A and B, if it does, of those integers we will pick the one that is the largest. Simple enough? Notice that strategy is much much simpler when Euclid's algorithm for which you need at a fair amount of cleverness. This on the other hand is very simple, it will take more time, than Euclid's algorithm. It will require more arithmetic operations. But maybe you do not care, or maybe you did not, you are not clever enough to discover a good algorithm for finding GCD. Of course there is the high school algorithm as well. So, there are several choices and this is just one way of doing things and as I said, I am showing you this, so that you get a feel for this notion of constraint satisfaction and searching through all, through a set of possible candidates.

Now, here is a way to find Pythagorean triples again it is a very similar way so, we can observe that  $x, y, z$  all the three numbers must be at least 1 and at most 20. Well, we said that  $z$  must be smaller than or equal to 20. So clearly these numbers must be at most 20. And why should  $y$  be at most 1? Well, if they are 0, and we want they have to be integers okay. If they are 0, then what happens? Then we will get a trivial solution. That  $y^2$  equal to  $z^2$  okay, so we do not want such trivial solutions. We want these numbers to be nonzero and therefore, they are expected to be distinct as well, it follows that they have to be distinct.

Well, if the numbers lie between 1 and 20, or another way of saying that is that the domain of each has to be the set mentioned over here, the numbers 1 through 20. So if this is the domain for each number each  $x, y$  and  $z$  then what can we say about all possible triples? Well there are just 20 times, 20 times, 20 - 8000 triples and we know that, so we need to know which one of these triples satisfy the Pythagorean property.

So, the algorithm naturally is, that we are going to try out, we are going to generate and tryout all possible triples. And for each triple we are going to check does it satisfy this equation? If it does, we will print it out. So, again we are being very clever over here, we are just doing something really brute force, we are trying out everything. But computers are fast and so therefore, sometimes it is okay, if we want to tryout everything, computers may do it fast enough.

(Refer Slide Time: 11:06)



## GCD of integers A,B

Find  $x$  such that  $x$  divides  $A$ ,  $B$  and is as large as possible.

**Idea:** try all possible values of  $x$  between 1 and  $\min(A,B)$  and pick the largest which divides both.

**In what order should we examine:**

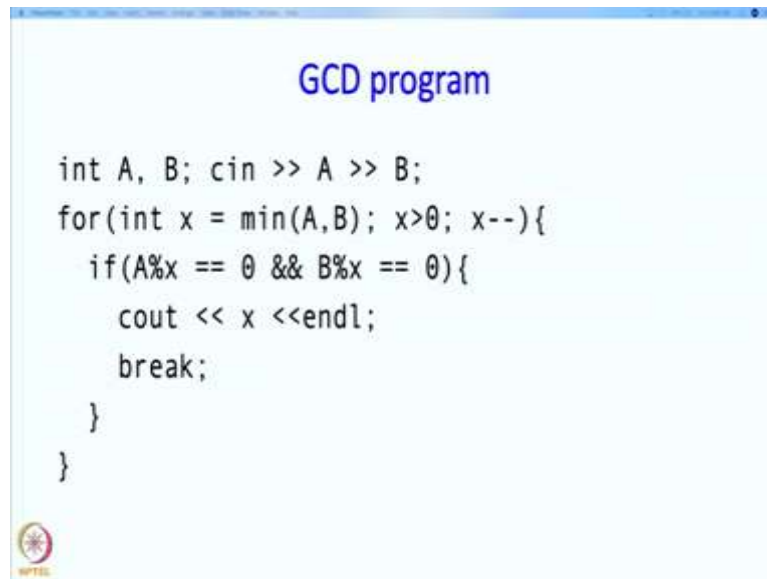
- $x = 1..\min(A,B)$  :
  - Will work
- $x = \min(A,B)..1$  :
  - Once we find  $x$  that divides both  $A,B$  we can stop.
  - Any common factor we discover later can only be smaller.

So let us see more detail, how we solve this GCD problem. So the problem again is, find  $x$  such that  $x$  divides  $A$  and  $B$  and is as large as possible. So the idea we mention was tryout all possible values of  $x$  between 1 and minimum of  $A$  and  $B$  and pick the largest which divides both. Now, we have to examine the values between 1 and minimum of  $A$  and  $B$ , but we could, we have a choice of examining these numbers in whatever order we want. So, let us consider the natural order first so, we first try out  $x$  equal to 1, see if divides? Of course it divides  $x$  equal to 2, see if it divides,  $x$  equal to 3 and so on. And then we list out all such, all whether or not, any of this numbers divides  $A$  and  $B$  perfectly. So we consider  $x$  in this order 1 through minimum of  $A$  and  $B$  and we remember which number, which  $x$  is dividing both? And now from those we pick out the largest. So, that is one way of doing things.

So this will work, but here is another order so in this we are going to start with the minimum of  $A$  and  $B$ , we are going to consider the largest possible candidate. We check if that divides both  $A$  and  $B$ . What if it divides  $A$  and  $B$ ? Do we need to check the next smaller candidate? Well we do not because the next smaller candidate will be smaller and we want the largest solution. So this is just going to be a better way of doing things. So as we go down this list, once we find an  $x$  that divides both  $A$  and  $B$ , we can stop. So indeed we should consider these numbers in this order.

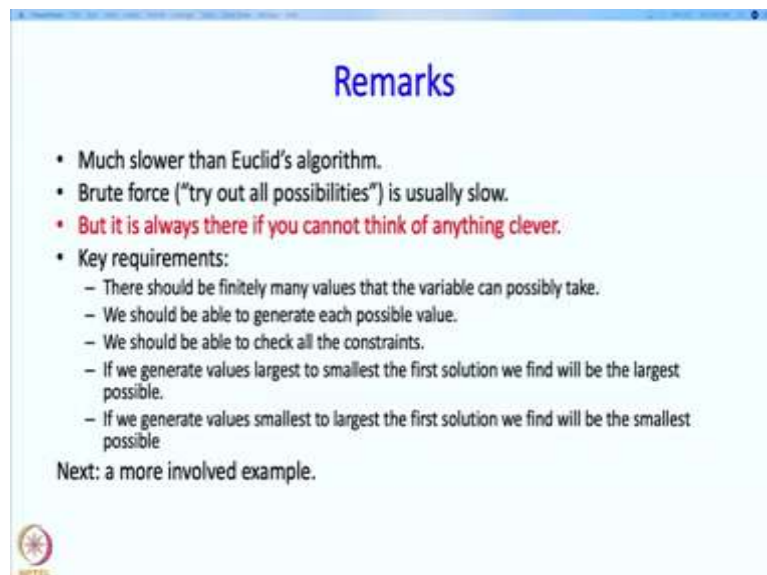
So any common factor that we discover later can only be smaller than common factors which we have discovered earlier. So therefore, we go in this order and stop as soon as we find an  $x$  which is a common factor of both  $A$  and  $B$ .

(Refer Slide Time: 13:40)



So, now the program is immediate, so here is the program fragment. We will use variables A and B and into it we will read the numbers whose GCD we want to find. Then we are going to start with x equals minimum of A and B and we are going to keep on decrementing it, so long as it is bigger than 0. And what do we need in it iteration? Well, we check whether A is divisible by x perfectly, and B is divisible by x perfectly. If it is, then we are going to print it out. But, we can also stop the loop right here, so we are going to break. That is it, so that is a program.

(Refer Slide Time: 14:29)



So, some remarks, this turns out to be much slower than Euclid's algorithm. I mentioned that earlier and in general brute force algorithms are slower than clever algorithm. That should not be a surprise, but the point is that you should consider brute force if you cannot think of

anything else. You should not give up because computers are fast enough, and if and if you have to examine a large number of candidates. You may be surprised how quickly computers can go through all such choices.

What are the requirements? Well, the candidates that you want to search over the possible values that you want search over must be finitely many, they could be a large number but they have to be finitely many. And we should be able to generate each possible value, in this case the generation was very easy. We just have to count okay, count down or count up it does not any matter. And given a candidate we should be able to check the constraints.

Now, if we generate the candidates or if we generate the values largest to smallest then the first solution we find, will be the largest possible. If we generate values smallest to largest if the candidates we generate are in the order smallest to largest then the first solution we find will be the smallest possible. So, depending upon whether you want the smallest possible solution or the largest possible solution, you should pick one of these orders.

We are going to take a break now, but when we come back, we will have a more involved example, the Pythagorean triples which we will again solve using brute force search. Thank you.