Lecture Transcript
Bisection Method

Hi and welcome to this next lecture on Data Structures and Algorithms. Starting today, we'll discuss some algorithms to find roots of equations. So we are looking at, at the equation of the form f(x)=0. What we'll discuss today is a method call the Bisection Method to find root of such an equation that is solutions to f(x)=0. Now, we are typically interested in a nonlinear equation. For the linear equation, f(x)=0, so solution should be very straight forward. Basically, we're looking for the x coordinate corresponding to the y intercept of 0. So, it is very trivial for linear equations. It is nothing but intercept on x-axis. Pictorially this is what we mean, if this value here, equation f(x)=0, this intercept x' would correspond to f(x') = 0. However, what you do when we you have nonlinear equations and here is a depiction for a nonlinear equation. The binary search method basically is an interactive method. Instead of a one shot solution such as in the case of the x intercept. You start with some two points, let's say point here and point here and we'll canonically call these points a and b. So given a function f(x), we'll assume that it's continuous on this interval ab. Let's say the initial guesses are a and b, we'll also assume that the function actually has at least one root between a and b.

So, we might initialize a and b to be sufficiently far apart to ensure that the function does have a root between a and b and of course a sufficient condition for such a continuous function to have a root between a and b is that they have opposite signs. So, sufficient condition is that the sign(f(a)) is minus of the sign (f(b)) will this basically means that the product of f(a) and f(b) is negative. Now, this is a sufficient condition. Not a necessary condition. So it is possible that f(a) into f(b) is positive and it still has a root and classic case for us will going to be the squared function or x raised to an even power so consider function like this. This could be f(x) = x is raised to 4. So, this indeed has a root at x=0. However f(a) into f(b) is always going to be positive. No matter what your choice of a and b is. So, therefore this is not a necessary condition. So, only a sufficient condition and unfortunately binary search method operates under this condition. Finally, it guarantees finding only one root. So, it is possible that function which has this.

Let's considers this function, now it's possible that your initial choice of a and b are such that they cover two roots. But only one root is guaranteed to be found using the y section method and we will see why. By virtue of its reduction of the search space by half it will either omit this root the root on the left or the root or the right only one of them will be found. An example of such a curve is a sine curve, sine also locate the cosine functions. So, here is the algorithm for bisection method. You begin with points a and b such that f(a) into f(b) is less than 0. Now, all the such algorithms operate on certain tolerance parameters, in fact, here we have two tolerance parameters one is a tolerance limit on the function itself. So, which means that you will stop when f(c) is less than tolerance value. But there is also another implicit tolerance parameter which is number of iterations this is tolerance based on argument. So for example, you might have a curve which gets to zero very gradually. So, let's consider the following curve, now curve like this gets to zero very gradually and it is possible that even after lots of iteration you are hovering around zero but not exactly a zero. So, until you have this maximum number of iterations and tolerances reached you do the following, you basically, set the upper and lower limit for further division.

So, imagine that a and b were the original pro points, you look at the midpoint and let say orbit point is somewhere here the c was here. You then check if a into c is negative 'a' and f(a) and f(c) have opposite signs. If yes, then you know that a new search space can be here. So, this becomes your new search space. How do you do that? You do that by setting, pushing b to c as what you have done here. On the other hand, it is possible that c is on the other side. So in that event, c were on this side you would set a to c would push a, see and search for the interval c, b. So, you to give track of the iteration number and then further look at the midpoint of the new interval. Please note that the algorithm tops or terminates when both these conditions are satisfied.

So, by design it is guaranteed to run for the n max number of iterations. This method is guaranteed to converge by design because of the both the tolerance parameter and because there is an increment in every iteration. Further, yours search interval is halved for every iteration which means there is a size of the interval after k iterations assuming unit initial interval will be 1/2 raised to k. So, as you can see there is an exponential decay or exponential shrinkage in the interval size. Despite all this the converges is slow. The question is there for, can we be wiser in our splitting of interval? Could we do better than half? And, is it necessary that you need to go on for n max number of iterations? Is it possible that you stop earlier? Is it possible that the algorithm becomes adaptive? Can the shrinkage be adaptive as you get closer to convergence?

So, we leave certain questions. Question one, can we reduce interval for search by factor less than half, can we adapt shrinkage and so on? In fact, if the initial guess is closer to the root it will take more iteration to converge primarily because the shrinkage is relative. So, you will like to detect, if you are closer enough to the root. And, most importantly the algorithm has severe limitations in what it can deal with. So, reminding ourselves of one function for which we said the algorithm won't function is the quadratic function f(x) equals x squared or equivalent consider x raised to four.

Now, there is no way you can find a and be with opposite signs for the function. On the other hand, we know that there is a root and we know that if you were about to find a and b, and be able to half the interval. You would actually find diminishing values of the function. How you do that? Also, this won't work if root is at infinity. So, 1/x is a classy example. So, here is the equation f(x) equals x squared which cannot be solved using this method. The equation f(x) = 1/x has no root but it's interesting that it just sign. So, just because you changed the sign as you go from say a potential point a here to potential point b here, you're going actually search for a midpoint shrink your interval, but you realize that unless you move away from the interval a, b you won't get to the root.

So, the fundamental problem with the assumption that if a and b have their signs changed, functions changes signs between a and b then the root will be between a and b. But recall, our assumption was subject to the assumption that f is continuous between a and b. So note that f was to be continuous in the interval close interval a, b which is not the case here. So, only under this condition was the change of sign of sufficient condition. So, what do you do if these conditions don't hold? More about this in the following lecture.

Thank you!