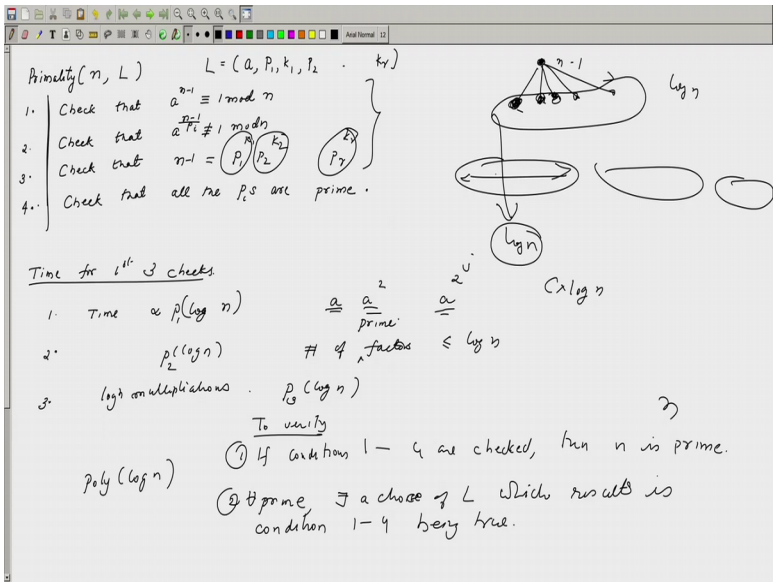
Randomized Algorithms
Prof. Benny George Kenkireth
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati

Lecture – 23
Pratt's Certificate

This lecture we will do the complete algorithm; non-deterministic algorithm to check for primality.

(Refer Slide Time: 00:34)



So, the algorithm will have two inputs; the first is the number and the second we will think of it as a list, ok. So, this list essentially has many components first would be the number a and then followed by a prime factorization of n minus 1. So, those we will think of it as P 1, comma K 1, P 2 comma K r, ok. So, we have a list containing 2 r plus 1 integers. So, we can think of this is a regular algorithm where the input is n and L is the certificate. So, from L 1 can in some sense deduce that n is prime. So, what are the checks that one has to do?

So, the first check would be check that a raise to n minus 1 is congruent to 1 mod n that is the first check the second check would be check that a raise to n minus 1 divided by P i is not congruent to 1 mod n. The third check would be check that n minus 1 check that n minus 1 is equal to P 1 to the K 1, P 2 to the K 2 P r to the power K r. The fourth check would be check that all the P i s are prime, ok. So, this fourth check we will not bother

about it for the time being , but let us say we first do these three checks and then make a recursive call to the same procedure,.

So, let us first focus on the first three. If you look at the first three checks how much time will we take for the first three checks now we need to compute a raise to n minus 1 that can be done by our repeated multiplication, ok. So, first exponentiation would take let us say time will be proportional to log n, because a raise to n minus 1 can be computed as a a square a raise to 2 raise to j and so on and then multiply amongst themselves.

So, the time taken is going to be something like a polynomial in log n because each terms here can be computed by multiplying it to the previous term ok. So, a square into a square gives you a raise to 4 and so on. So, we can compute all these terms by performing at most something like C times log and multiplications and once all these terms are obtained from the binary representation for n minus 1, we can compute a raise to n minus 1. So, the total time is bounded by some polynomial in log n, ok. So, I will just write it as polynomial in log n.

The second step is again a to the power n minus 1 by P i. So, n minus 1 by P i is going to be some number smaller than n minus one. So, that is also going to be some polynomial in log n , but we might have many such i's, but the maximum number of i's is going to be at most log n, ok. So, since the number of prime factors are small this is going to be less than log n. So, number of P i's is going to be at most log n.

Well, the input list could when we just look at L it could have more than log n numbers in which case we can just right away say that the check has failed because any list with more than log n numbers would not really be a proper certificate, ok. But, even if it is less than log n it could be the case that in each of the individual P i's may not be prime that we will check later.

And, we will additionally check the third part and the third part would take it is again repeated multiplications P i to the K i. So, it is going to be r multiplications. So, log n multiplications and each of those terms so, this is one term another term and there K r terms the number of r terms and the total number of terms is bounded by log n. So, if we compute all these then you are on good grounds and each the individual terms the K r's are at most n or at most log n.

And therefore, again the total cost of step 3 is bounded by some polynomial in log n and once all these are done we need to now check whether the P i's themselves are prime? You can imagine that each of the P i's is at most half of the original ones and if the original number was n, its factor is at most half of the original one; original number and we can just recurs. We can represent this by a tree; at the first stage we are factorizing n minus 1 and then there are recursive calls you can unroll the recursion tree and do it in a non-recursive fashion.

But, at each of these nodes the cost is at most half of the original one and the total number of these nodes at any level is at most log n, ok. So, since each node is no more than half of the original one the total depth of this is going to be bounded by log n, ok. So, here is a tree whose depth is bounded by log n and therefore, you can do the computations and ensure if you look at all these recursive calls together there that is not going to be a huge tree.

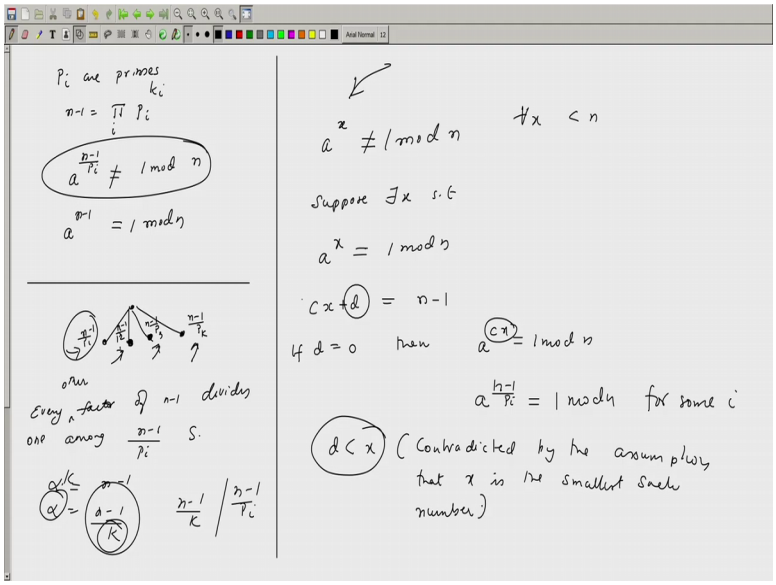
So, if all the factors were distinct you would get at most log n different factors and each of these factors there is at most half of the original one. So, the branching factor is bounded by log n, and the total depth is again bounded by log n and let us look at any intermediary stage, I mean the total number of small instances these prime numbers, they should not multiply out and give something say larger than n minus 1. So, again the width at no stage is greater than log n and therefore, the total tree bound the size of the tree is fairly small. Right now argued is that the total computation is going to be some poly log n.

Now, we need to argue that if these conditions have been verified then the number is indeed prime, ok, that is one thing. So, to verify, if conditions 1 to 4 are checked then n is prime and the second thing is for every prime there exists a choice of L which results in condition 1 to 4 being true, ok.

So, why is it that any prime number has one such L, that is true from the theorem that we had earlier which said that for any prime number if and only if there exists any is, it a raise to n minus 1 is congruent to 1 mod n and for all these smaller numbers a raise to x will not be equal to 1 mod L since such an a is guaranteed we know you can construct an l which passes these tests.

Now, why is that any a which passes these tests I mean if you have one such a which passes these tests then how can we say that the number is prime?

(Refer Slide Time: 10:45)



Well so, we will assume that the P i's are prime, because of our check 4 and then we will assume that n minus 1 is equal to product over i P i to the K i, ok. These conditions we have checked, and the next condition that we have checked is a raise to n minus i by P i is not equal to 1 mod n and we know that a raise to n minus 1 is congruent to 1 mod n, ok.

What we need additionally is a raise to x is not equal to 1 mod n, for all x less than n, ok. How do we show this? Well, suppose there is one such x such that a raise to x is congruent to 1 mod n. We can look at dividing n minus 1 by x. So, Cx plus d is equal to n minus 1, and if d is 0 then what we have is a raise to Cx is equal to 1 mod n and Cx divides n minus 1. If you look at n minus 1; n minus 1 has all these factors n minus 1 by P 1, n minus 1 by P 2, n minus 1 by P 3, n minus 1 by P k. So, if P 1 to P k were the factors of n minus 1, then these are the largest factors of n minus 1; largest in the sense if you look at the Hasse diagram of these of these factors and these are in some sense the largest, ok.

Every other factor of n minus 1 will have to divide one of these, ok. So, we can write that every factor we write every other factor of n minus 1 divides one among the n minus 1 by P i's. So, suppose you have a factor let us say alpha; alpha into K is equal to n minus

1. So, alpha is equal to n minus 1 by K, and we have looked at all K s which are primes and therefore, this is going to be a smaller number, ok. So, we can conclude that one of these I mean if you have alpha equals n minus 1 by K n minus 1 by K divides n minus 1 by for some P i, ok.

Since we have we have this condition we can conclude that if a to the power C x is equal to 1, then a to the power n minus 1 by P i is equal to 1 mod n for some i, ok. So, that would contradict what we have already checked and therefore, d cannot be equal to 0. If d is not equal to 0 then d is smaller than x, ok; in that case if we had started off this particular assumption by assuming that x is the smallest such integer then d less than x would contradict our assumption.

So, we can say that this is contradicted by the assumption that x is the smallest such. So, if it passes these four tests, then it is certainly prime and for any prime number there is a choice of L which will result in these four tests resulting the value true.

So, we will stop here and look at randomized algorithms for checking primality. Now, what we have seen as a non-deterministic algorithm, we will see that there is a randomized algorithm as well for checking primality.