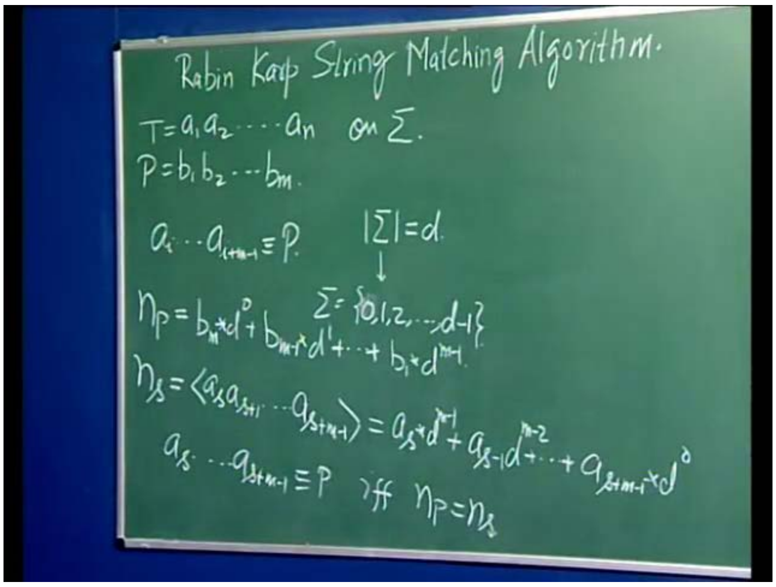
Computer Algorithms-2
Prof. Dr. Shashank K. Mehta
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture - 15
Rabin Karp Algorithm

Hello, today we will discuss yet another algorithm for string matching called Rabin Karp string matching algorithm.

(Refer Slide Time: 00:20)



Rabin Karp String Matching Algorithm.

$T = a_1 a_2 \cdots a_n$ on Σ.

$P = b_1 b_2 \cdots b_m$.

$a_i \cdots a_{i+m-1} \equiv P$. $|\Sigma| = d$. $\downarrow$ $\Sigma = \{0, 1, 2, \ldots, d-1\}$

$n_P = b_m * d^0 + b_{m-1} * d^1 + \cdots + b_1 * d^{m-1}$.

$n_s = \langle a_s a_{s+1} \cdots a_{s+m-1} \rangle = a_s * d^{m-1} + a_{s-1} d^{m-2} + \cdots + a_{s+m-1} * d^0$

$a_s \cdots a_{s+m-1} \equiv P$ iff $n_P = n_s$

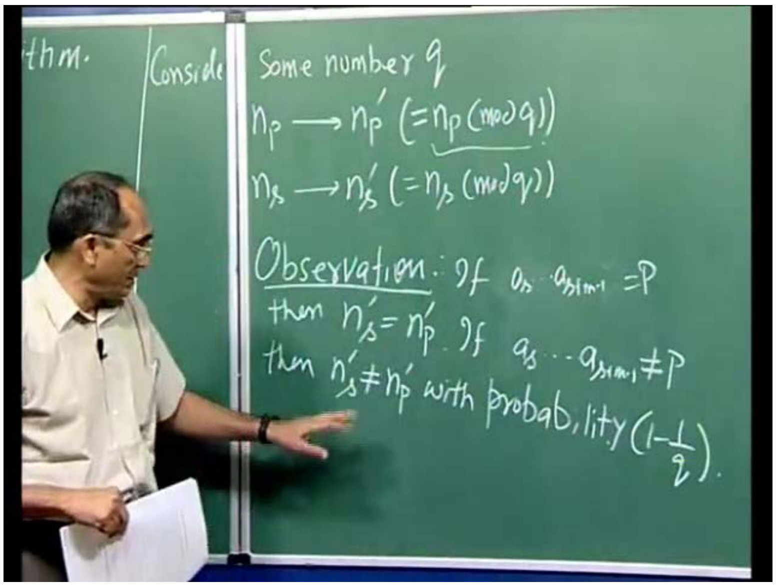
Let me remind you the problem was that you are given a text string over some symbol on set sigma. We have another string, we call pattern string and we are trying to find out, if there is a position a i such that a i to a i plus m minus 1 is same as P. Now, in this approach, we are going to transform these strings into numbers and then will compare numbers.

So, let us define a number n p corresponding to string P in the following fashion, let suppose the number of symbols are d, then we can treat each of this symbol, so we can think of sigma as 0, 1, 2 to d minus 1. So, in a base d system, we have these d basic symbols, and then we can define a number associated with this string as b m into d power 0 plus b m minus 1 into d power 1 up to b 1 into d power m minus 1.

So, we are treating this as a number on base d now, we can also define a number n s, which is the number due to symbols a s, a s plus 1 through a s plus n minus 1. And that, as we just did we will have a s into d power m minus 1, a s minus 1 d power m minus 2, a s plus m minus 1 to d power naught. Now obviously, the string a s through a s plus m minus 1 can match P if and only if, n p is n s.

So, we have all the information about the string in the corresponding number hence, this can certainly be a possible method to perform string matching. But, in general, the numbers we are dealing with may be very large, a single arithmetic operation cannot be considered a unit operation or a constant time operation, if we are dealing with the very large numbers. Because, they may not even be representable in our system, in which we are trying to program such an algorithms. We have to remedy that problem and for that, what we will do is, we will do the hashing on these numbers.

(Refer Slide Time: 04:44)


thm.

Conside

Some number q

$n_p \longrightarrow n_p' \; (= \underbrace{n_p \,(\bmod q)})$

$n_s \longrightarrow n_s' \; (= n_s \,(\bmod q))$

<u>Observation</u>: If $a_s \cdots a_{s+m-1} = P$ then $n_s' = n_p'$. If $a_s \cdots a_{s+m-1} \neq P$ then $n_s' \neq n_p'$ with probability $(1 - \frac{1}{q})$.

Let suppose, consider some number q, we have a number q and we are going to map every number say, n p to n p prime, which is n p mod q, by this we simply mean, the reminder of n p when divided by q. Similarly, we will have n s map to n s prime, which is same as n s modulo q now, if we take certainly, we have lost certain information here. Once that, we have replaced a original number by only the reminder with respect to our cube.

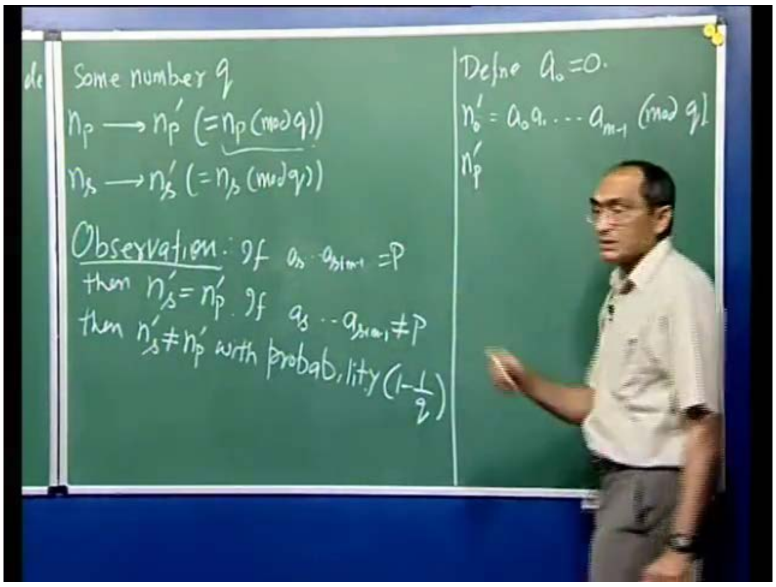
So, if the two numbers are equal that is to say, if n s is equal to n p then we can be sure that, these two will also be equal, there is no doubt about it. But, what happens when they are not equal, if they are not equal then still there is possibility that these two may be equal. Hence, we cannot for sure say that, whenever this n s and n p meet they match, they are equal, that would mean that we have found the pattern, we cannot say.

So, what is it, that we can certainly say so here is one observation that, if n s is equal to rather, if pattern a s through a s plus m minus 1 is equal to p then we know that n s prime is equal to n p prime. But, if this pattern a s through a or rather I would not put any sign around it but if the pattern a s through a s plus m minus 1 is not equal to P then what is the chance that these two still will be equal. Now, since we have only few possible values for a reminder so, if spread over all possible values, there is only one chance in q that even when these two are, this thing or these two will meet.

So, assume that, this is some random string different from P then we can say, then n s prime is not equal to n p prime with probability 1 minus 1 over 2, only in few chances there, that this still may be equal otherwise, they will not be. So, if I am going to use this test then I cannot not be sure that, indeed the two patterns are equal or not but with a very high probability provided q is large, I will generally be right but with the small probability, there may be a false alarm.

In case there is a false alarm, what we should be doing is, we should actually go ahead whenever we find that there is a matching here, we should go and test, whether indeed the two strings are equal or not, because for sure, we have a match or not, that can be decided only by matching the pattern. But, the advantage is that, with a very small probability, it will be a false alarms, hence we will not be testing this to off hence, as a result, we may be able to perform well in this approach in finding a pattern.

(Refer Slide Time: 10:17)



Some number q

$n_p \longrightarrow n_p' \; (= \underbrace{n_p \pmod q}\,)$

$n_s \longrightarrow n_s' \; (= n_s \pmod q)$

Observation: If $a_s \cdot\cdot a_{s+m-1} = P$ then $n_s' = n_p'$. If $a_s \cdot\cdot a_{s+m-1} \neq P$ then $n_s' \neq n_p'$ with probability $(1 - \frac{1}{q})$

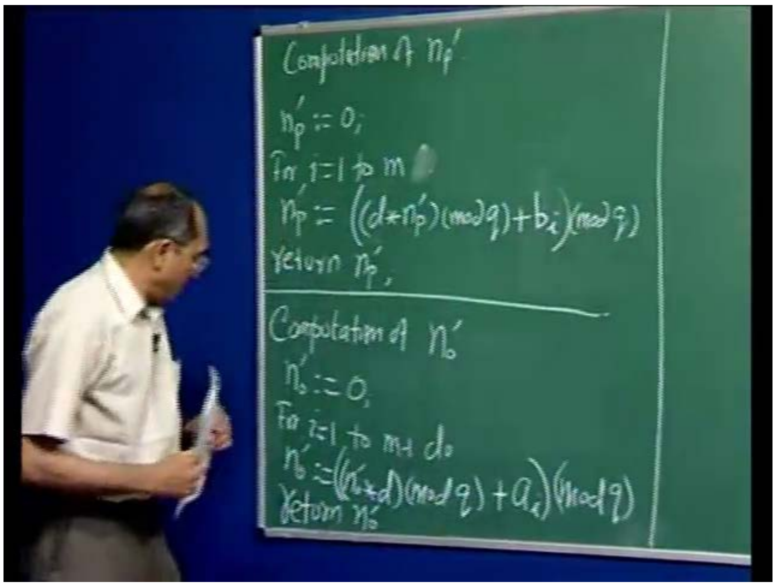
Define $a_0 = 0$.

$n_0' = a_0 a_1 \ldots a_{m-1} \pmod q$

n_p'

So now, we will define a number, let us first define for a convenience, define symbol a naught to be 0 and define n p prime equal to a naught a 1 a m minus 1 sorry this will be n prime 0, modulo q. Now, we will first try to compute n 0 prime and n p prime, which we have already described so, let us compute n p prime.

(Refer Slide Time: 11:11)



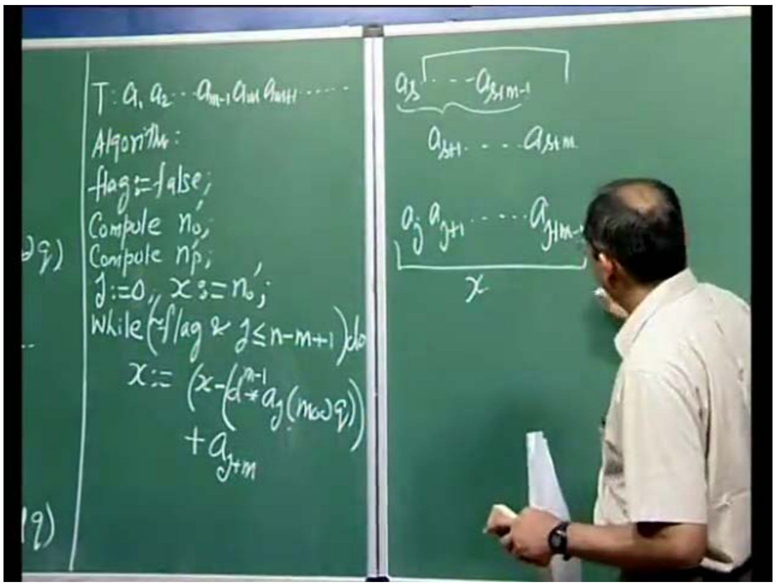
```
Computation of $n_p'$
$n_p' := 0;$
For $i = 1$ to $m$
    $n_p' := ((d * n_p') \pmod q) + b_i) \pmod q)$
return $n_p'$,

Computation of $n_0'$
$n_0' := 0;$
For $i = 1$ to $m-1$ do
    $n_0' := ((n_0' * d) \pmod q) + a_i) \pmod q)$
return $n_0'$
```

And then for i equal to 1 to m, n p prime will be a d star n p prime mod q plus a i so, this actually going to raise the power of d for the current number and at the next position, it is going to m. So, i equal to 1 to m, and so we each time we are going to multiply the current number raise it is position from unit to d th level by multiplying by d adding a i. And then in each step, though we are going to compute modulo d to ensure that, in each arithmetic operation, the numbers that we are dealing with are within cube.

Now, similarly so, return n p prime, computation of n 0 prime once again we will start with 0 and for equal to 1 to m minus 1, we will do the same thing. We will compute n 0 prime as n 0 prime time's d modulo q and then note that we are dealing with the symbols of P, so this should have been b i, this is b i and this is a i mod q return n 0 prime. So, actually n 0 prime is the number generated by first m minus 1 symbols.

(Refer Slide Time: 15:03)



```
T: $a_1 a_2 \cdots a_{m-1} a_m a_{m+1} \cdots$
Algorithm:
flag := false;
Compute $n_0'$;
Compute $n_p'$;
$j := 0$, $x := n_0'$;
While ($\neg$flag & $j \le n-m+1$) do
    $x := (x - (d^{m-1} * a_j (\bmod q)) + a_{j+m}$
```

$\underbrace{a_s \cdots a_{s+m-1}}$
$a_{s+1} \cdots a_{s+m}$
$\underbrace{a_j a_{j+1} \cdots a_{j+m-1}}_{x}$

Now, we have to move from, let us just take a look at the string so, we have a 1, a 2, a m minus 1, a m, a m plus 1 so on. So, suppose at some point we are here, at a s through a s plus m minus 1 suppose, our current variable give us the number associated with these m symbols. And then we are interested in finding out the number corresponding to the next m consecutive symbols, which would be due to a s plus 1 through a s plus m. So, the advantage we have in converting this into a number is that, the number that we have here we can remove the contribution of a s, which is a s into d to the power m minus 1.

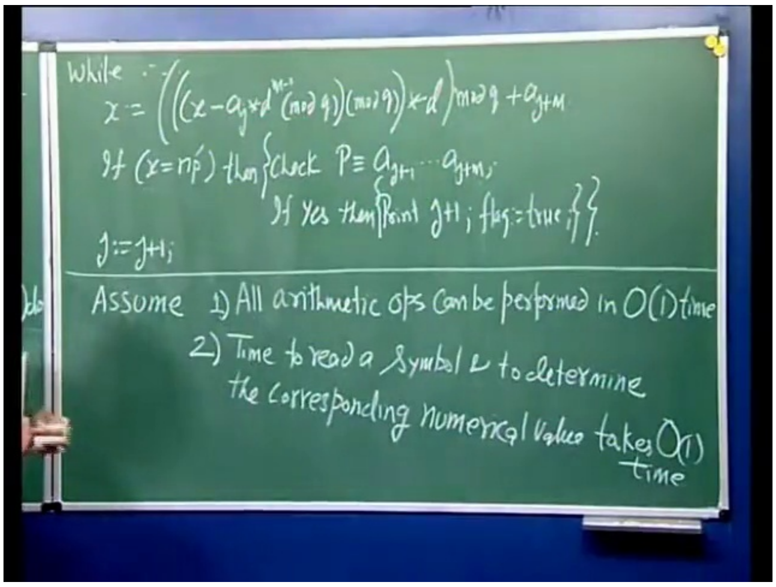
Subtracting that, will give me the number corresponding to this, I will raise that by d and then add a s plus m, that will immediately give me the number associated with this and this requires only two operations. Hence, in a constant amount of time, we can move from this number to the next number, so this is what? We are going to explore it. So, let us say, we have a flag, which indicates that we have found a match, which is false to begin with.

We have already described how to compute, compute n 0 prime and compute n p prime, which is corresponding to the pattern string. And now, we will proceed as long as we do not find the flag to be prove or we have reach the limit after which, you do not have m consecutive symbols left in the text string. So, we will say while so, we will use a counter and let us say, we use a while loop, as long as flag is false and j is less than or equal to n minus m plus 1.

Because, j equal to n minus m plus 1 will still give me m symbols so, let us see now, first thing we do is, we extended the number to the next position. So, we are notice that, we had already chosen a number associated with m minus 1 symbols, will move onto the next symbol and that is what we are going to do. So, let us take x, I need to set my variable x to n 0 prime, there will compute x to the x minus d to the power m minus 1 into a j.

Note that, currently we are looking at the… So, we are going to remove the contribution of a 0, this we do well. Let us first compute the mod of this, mod of this and to that, we will add a j plus m that will add the new symbol. So, we have say currently, a j, a j plus 1, a j plus m minus 1, a j plus m, this is the number corresponding to, this is the x to start with here, we remove the contribution of this, multiply this by the and then we add this. So now, we have the new value, of which is actually n j plus 1 once again, I will compute mod q. At this point, we have to raise this by d, this whole thing times d and plus this and then mod q. So, maybe I will just rewrite this; let us just write down here.

(Refer Slide Time: 22:05)



```
While ...
    $x := (((x - a_j * d^{m-1} (\mathrm{mod}\ q))(\mathrm{mod}\ q)) * d) \mathrm{mod}\ q + a_{j+m}$
    If $(x = n p')$ then {check $P \equiv a_{j+1} \cdots a_{j+m}$;
                          If yes then {Print $j+1$; flag := true;}}.
    $j := j+1$;
```

Assume 1) All arithmetic ops can be performed in $O(1)$ time

2) Time to read a symbol & to determine the corresponding numerical value takes $O(1)$ time

So, we have this while loop and x is x minus a j times d power m minus 1 mod q, mod q d and then again, we are going to compute modulo q and at the end, we are going to add a j plus m. Now, we have the number so, we are checking, if x is equal to n p prime now, this is the place where, we are uncertain. If they are equal then we will explicitly go and check, whether the two patterns are equal then check whether P is identical with the string a j plus 1 through a j plus m.

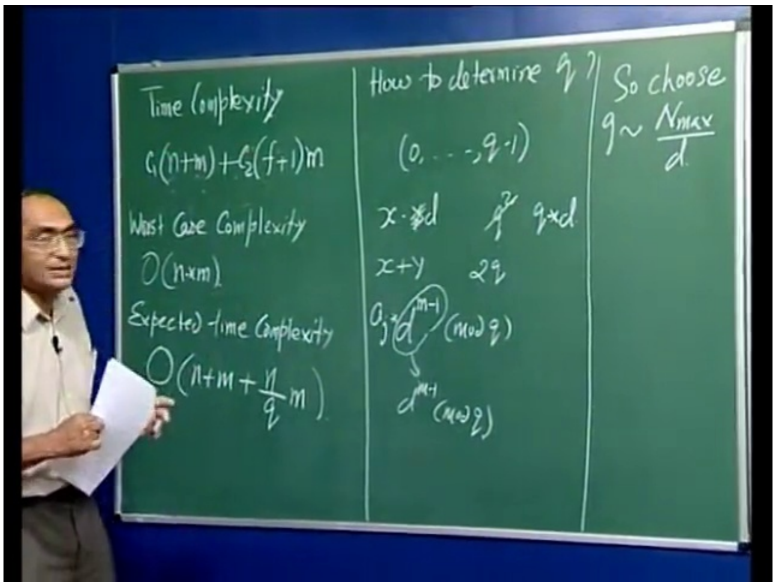
If yes then print j plus 1 now, that is the position from where, the text string matches with the pattern string otherwise, we have to just move on and in this case, we are going to update the index j to j plus plus. Now, one more thing we have to do since we are using the flag, we are going to also set the flag to be true. Now, this is all there is, there is nothing note to do, we will continue to check and now, the time is to look at the two issues here.

First what is the complexity, second is how do we decide the value of q, let me make some assumptions. We assume 1, that all and we will have to justify this later on and that all arithmetic operations can be performed in order 1 time. Now, that is possible if we ensure that, every number occurs in the computation is within the representation range of the given machine. Is there another assumption we will make and that is the following that, when we are reading, see you are given a text string in some unknown symbol set, which is not necessarily associated with any kind of integers.

Hence, we will have to make a map from each symbol to the corresponding digit in base d now of course, there is one remark that, I go to make here is that, that base d or computation could be done in any other base. Normally, we do our computation in either base 10, base 2, base 8, base 16 that does not matter but the time to compute to read a symbol and determine the corresponding numerical value, I assume that takes order 1 time.

So, we say that, the time to read a symbol and to determine the corresponding numerical value takes order 1 time so, these are the two assumptions. In case, you do not agree with this assumption then obviously, this is going to take order log of the number of symbols that we have. Because, it will take us to determine, what symbol is this and then converting that into the corresponding number so, that is not a problem. So, subject to these assumptions, it will take us a unit time to compute the next value of x.

(Refer Slide Time: 28:54)



Time Complexity

$c_1(n+m)+c_2(f+1)m$

Worst Case Complexity

$O(n \times m)$

Expected time Complexity

$O\left(n+m+\frac{n}{q}m\right)$

How to determine q?

$(0, \ldots, q-1)$

$x \cdot d$ $\quad$ q $\quad$ $q \times d$

$x+y$ $\quad$ $2q$

$a_j \cdot d^{m-1} \pmod q$

$d^{m-1} \pmod q$

So choose

$q \sim \frac{N_{max}}{d}$

And the time complexity will be 1 unit of time for computation of each successive value of x so, we will we spending n plus m unit of time, m for initial computation of n p prime and n 0 prime. Subsequently, for every positions accurately, n minus m plus 1 but then we have a additional cast to pay. Remember, every time there is a false alarm, we will be doing a complete check of the two patterns, and finally when we have a correct signal then again, we will doing a complete check.

So, suppose, there are f false alarms and at most one correct indication so, that many times you are going to perform a complete comparison of P with the current symbol sequence and that takes order m time. So, we have, this is the price we have to pay so, the worst case time complexity, worst case complexity will be one in which f happens order n times. So, we will end up with order n times m, but what is the expected time complexity.

Recall that, we have notice that, if our pattern is random or not particularly biased then there is only one in q chances that, we will have a false alarm. So, the number of times that we will have false alarms will be only one over q times m so, that says that, the expected time complexity will be order n plus m plus, n over q times m. So, it is very clear that, other objective is to work with as large q as possible, so the next question is that, how to determine q.

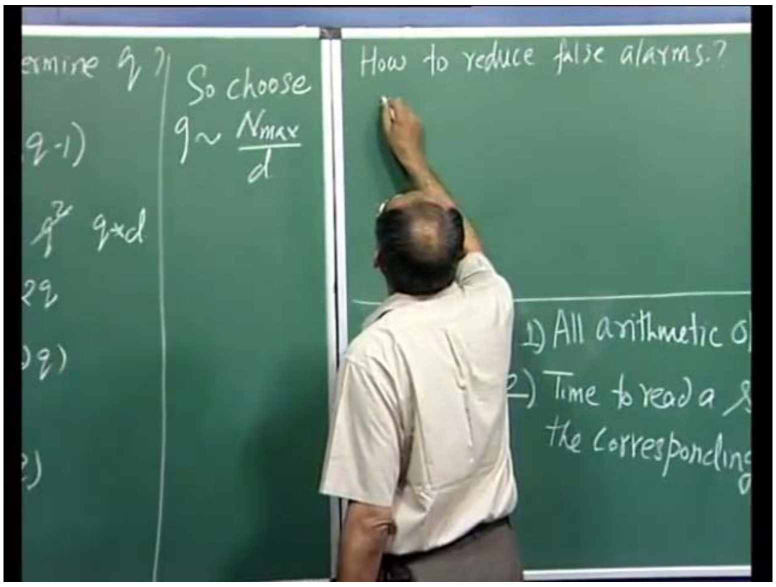
Now, what you notice is in our algorithm, after every arithmetic operation we were reducing the given number within the range of 0 to q minus 1 by doing modulo q of computation. After each arithmetic operation, we were doing modulo q so, every time a number that we were dealing with was in this range. But, computations that we were doing apart from one special case, which I will point out later is that suppose, we have a number x and a number y both in this range then tentatively we are going to generate the number of the order of q square.

If you are doing a sum or subtraction, we are going to generate the number of the size 2 q and that will be the largest position, up to which it is going to rise. There was one place where, we were actually computing d to the power m minus 1, we had actually some a j times this and then mod q. Now, notice that, such a number can be pre computed, we can compute d power m minus 1 mod q (( )). Hence, without any problem, we can assume this is also a number within cube, so the numbers that we are going to be dealing with will be…

But, one more thing that one observes is that, the only place that we are going to multiplication is, where we are having the second number to be d. So, this is actually not going to be q square but it will be q times d, d is smaller than q in case it is but it cannot exceed q. Because, even d will be reduced so, the largest number that we have in the process is q times d.

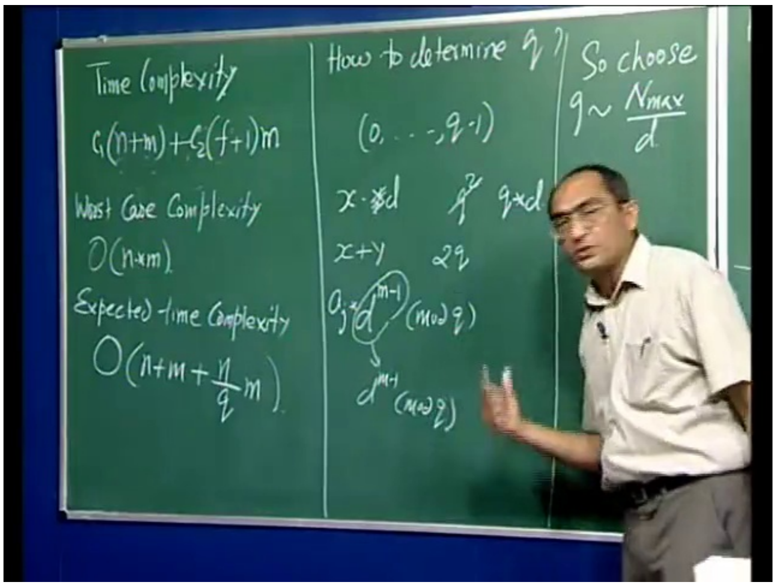
So, we should choose q, q be approximately n max divided by d, the largest number representable in your system divided by d would be the largest number, that we can take as q. Because, at that time, the number is that will emerge will be within the representation range. So, that completes our basic algorithm, the only thing now, I will do is, I will give you one method by which we can improve the possibility or reduce the possibility of false alarm.

(Refer Slide Time: 35:55)



So, how to reduce false alarms we know that, the best way to do that is by increasing the value of q.

(Refer Slide Time: 36:13)


Time Complexity

$c_1(n+m)+c_2(f+1)m$

Worst Case Complexity

$O(n \times m)$

Expected time Complexity

$O(n+m+\frac{n}{q}m)$

How to determine q?

$(0, \ldots, q-1)$

$x \cdot d \quad q^2 \quad q \times d$

$x+y \quad 2q$

$a_j d^{m-1} (\bmod q)$

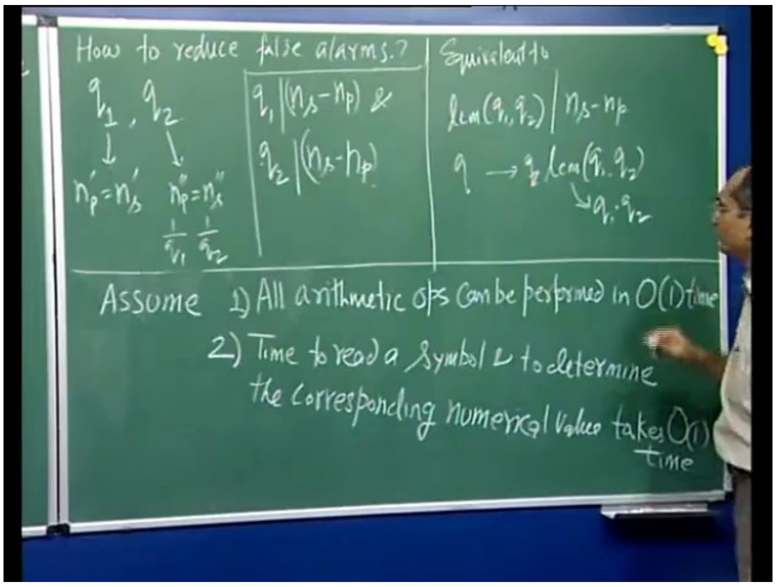
$d^{m-1} (\bmod q)$

So choose

$q \sim \frac{N_{max}}{d}$

But, we have limitations, we have already seen that, we cannot take q to be bigger than this. Because, all other computations must be done in such a way that, all number that emerge in the during the computation are within the representation, so what we could do is the following.

(Refer Slide Time: 36:38)


How to reduce false alarms?

q_1, q_2

$n'_p = n'_s \quad n''_p = n''_s$

$\frac{1}{q_1} \quad \frac{1}{q_2}$

$q_1 | (n_s - n_p)$ &

$q_2 | (n_s - n_p)$

Equivalent to

$lcm(q_1, q_2) | n_s - n_p$

$q \rightarrow lcm(q_1, q_2) \rightarrow q_1 \cdot q_2$

Assume 1) All arithmetic ops can be performed in O(1) time

2) Time to read a symbol & to determine the corresponding numerical value takes O(1) time

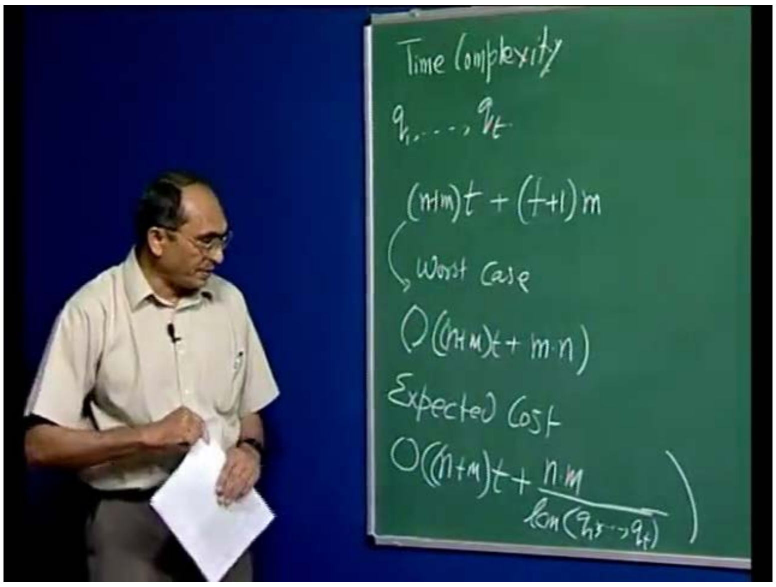
Let us say, we perform the entire computation, so we do entire computation with respect to of q 1, we also perform the computation with respect another number q 2. Instead of taking one q, we are going to take two such numbers, we will compute everything with respect to q 1, we will do the same thing with respect to q 2 at only when, we have a match here as well as here. So, we have over here n p prime equal to n s prime, this is with respect to q and we also have n p double prime equal to n s double prime.

Note that, these are computed using q 2, when both happen then only, I am going to perform the explicit check, whether the patterns swing matches with the text string starting at position s. That should improve the situation because what is the chance that this and this both happen, on the face of it we might say, we have only one over q 1 chance here. And if the number is, q 2 is also chosen in such a way, that there is no correlation then we have further reduce the probability of a collision.

But, to be precise, how many numbers are both divisible by both q 1 and q 2 so, we happened to have see, what is it that we are trying to avoid, we are saying that, whenever q 1 divides n s minus n p, we are going to have a false alarm, when this is not a 0 number. But, with two such numbers, we want this and this, this indicates that q 1 divides this number, this says q 2 divides this number, so both these should divide then only, we will have n p prime equal to n s prime.

So, the possibility that will happen is, when this number is divisible by the LCM of the two numbers so, this is equivalent to the fact that, LCM of q 1 and q 2 divides n s minus n p. In case, you chose them to be co prime that is, their GCD is 1 then this is nothing but q 1 times q 2, what has happened. Effectively by doing this work with two numbers, I am upgrading my q to LCM of q 1 and q 2, which could by a wise choice of the two numbers, which could be q 1 times q 2. Remember that, we were not in a position to chose large number, because of our restrictions but this technique can effectively give the same advantage, as a number as large as this.

(Refer Slide Time: 41:10)



Time Complexity

$q_1, \ldots, q_t$

$(n+m)t + (t+1)m$

Worst Case

$O((n+m)t + m \cdot n)$

Expected Cost

$O\left((n+m)t + \frac{n \cdot m}{lcm(q_1, \ldots, q_t)}\right)$

So, we generalize our method by taking not one but say, t of these numbers q 1 through q t, we will be computing every number with respect to each of these numbers. So, the time for every successive numbers, computation will be n plus m, times t plus whatever is our number of false alarms, that into m. Because, this is the cause for the explicit comparison of the two patterns, so this is our new cause, which in the worst case again, is order m plus m times t plus m n.

Because, we need not take t too large so that, this exceeds this but on the other hand, we look at the expectation value, expected cost will be order n plus m t plus n times m by LCM of q 1 q t. So, one has to carefully chose t such that this is not too large, the two terms are comparable effectively, that will give us the best result so, that completes the discussion of Rabin Karp’s algorithm.