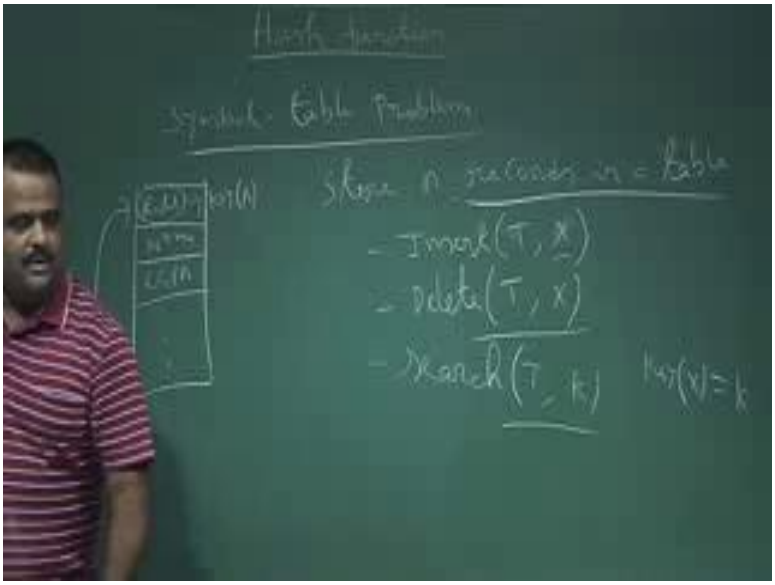
Internetwork Security
Prof. Sourav Mukhopadhyay
Department of Mathematics
Indian Institute of Technology, Kharagpur

Lecture – 40
Hash Function

So, we talk about Hash Function. So, hash function it is taking any bit input, a message long message and it has to give us the hash edges. So, let us talk about some background of this hash function where form it is coming.

(Refer Slide Time: 00:38)



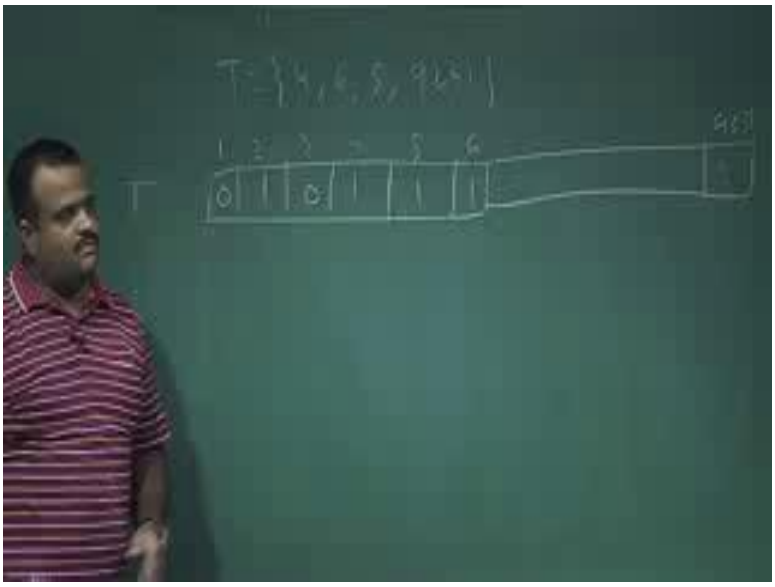
Basically it is to solve the problem which is called symbol table problem. So, in a symbol table problem what we have? We have few records it is basically database problem we have few records and we need to store this record in a table. So, basically each record has some entity like student record, your roll number, roll number of a student, your name. So, your CGPA, you address so and so. This is a record and if x is a pointer to this record. So among this field we choose one field as a key, so that should be unique to that records. So, it maybe roll number, for this purpose roll number can be the key. So, this is the key of x because key, roll number is unique to each student, so among this field each is a key.

Now, we have such n records and we need to store the record into a table in a dynamic this is a dynamic set. So, we need to store the records, they say they are at some point of

time there are n records in a table, such that we should perform some dynamic operation like insert, we should able to insert a record in a table, which is not there so we should able to insert a record, we should able to delete a record from the table; a record is there we can delete a record suppose student pass student pass out, so we should delete that record from the table and we should able to this 2 are dynamic operation and then main important thing is we should able to search record.

So, we should give a key value; k is the k. So, we should be able to, so corresponding that record we should be able to get. So, this is the search; we should able to search whether that record is that key value is there or not. So, these are the operation we can do. So, we need to have a data structure for this. So, first we can think of a tally of doing this; tally data structure for doing this.

(Refer Slide Time: 03:09)

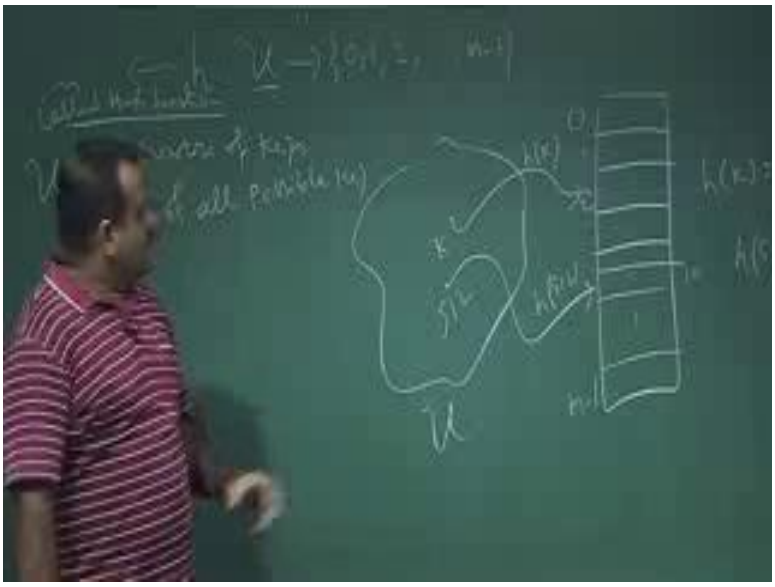


Suppose we have a say record say we have some records, whose key value. So, this is our dynamic set S, this is the record. So, this is T, suppose we are just having say 4, 6, 9, 2. So, these are the key value corresponding to the record we have at some point of time.

So, what we can do? We can have a say 6 this is a 5. So, we can have an array of size 6; 1, 2, 3, 4, 5, 6; we can have a array of size 6 and then if a record is present we will put it 1 otherwise we will put it 0. So, this is just a direct access table or just a bit vector. So, this is very simple and, but it is a very powerful data structure, but there is a problem with this problem is if we have a key whose value is bigger, suppose we have instead of

2 suppose we have a record whose key value is this; all number of record is less, but the value of the key of a particular record is big. So, in that case we need to maintain the array of this much size, so that is unnecessary uses of memory, that is the drawback of this simple array representation.

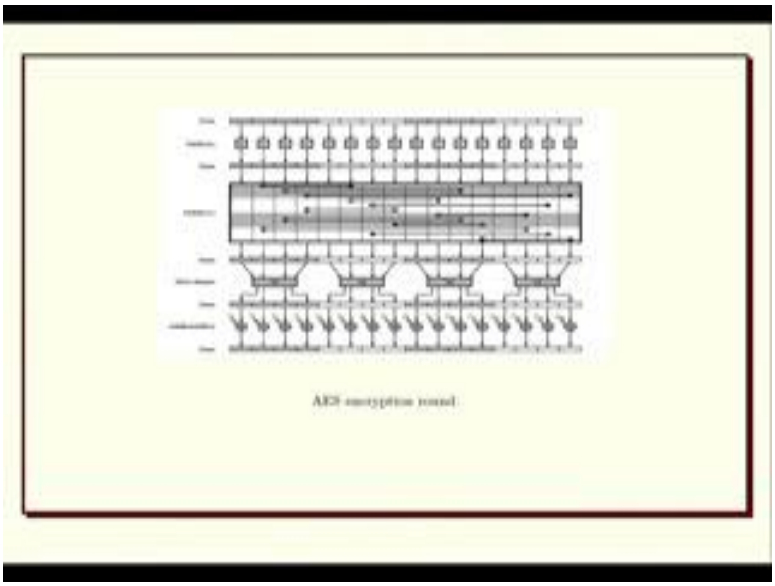
(Refer Slide Time: 04:48)



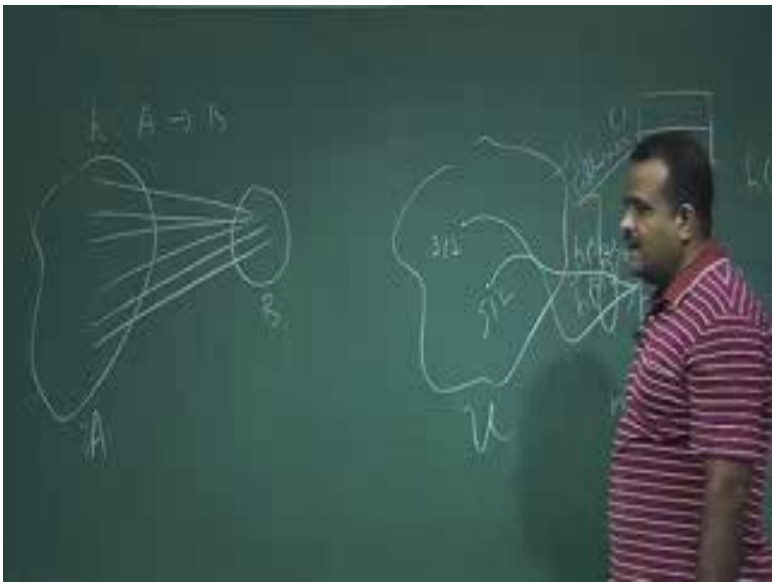
So, to solve this problem what we have? We have the function which is called hash function. So, hash function is a function form, this is a set of all records I set of all keys. So, this is universe of keys; that means, set of all possible keys; this is this set. So, this is all possible keys we have and we have a fixed size, table size fixed. So, we have a table size of fixed length say set table sizes 0 to m minus 1. So, we have total m slot. So, this is basically an array, but table size is fixed array size is fixed. So, it is a m size array 0 to m minus 1, in C language we use 0 to m minus 1.

So, then any function h, which is from set of keys to this is called a hash function. So, any function forms it taking a key and it is applying the hash value hash of h of this and it is putting into some slot. So, this is the thing. So, h of k is 2 like this. So, say you have a key 5 1 2. So, it is applying the hash function on 5 1 2 say it is storing into the 10 slot. So, h of 5 1 2 is 10 something like that. So, it is basically a function which is taking a key of key could be any arbitrary length and this could be a very fixed size smaller size. So, it is just a mapping from this key to the slot, it is hitting a slot there. So, this is called hash function.

(Refer Slide Time: 07:01)



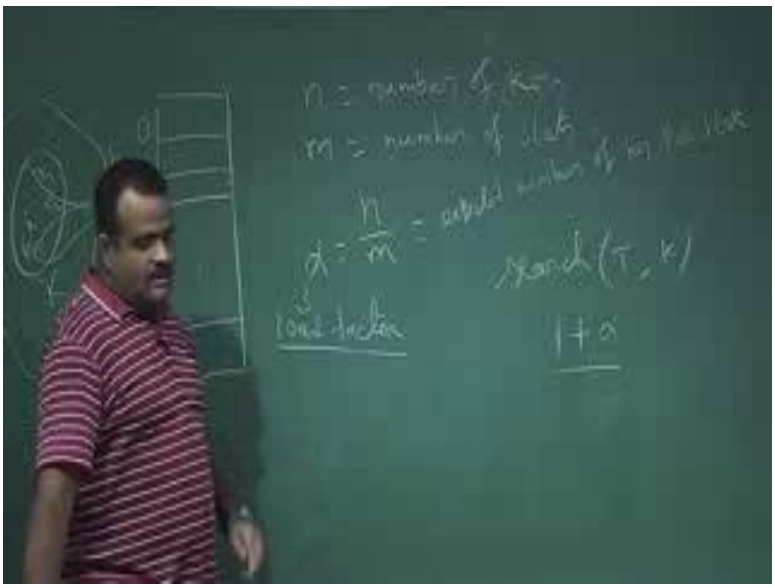
(Refer Slide Time: 07:11)



Now, there is a problem with this method what is called collision; suppose we have this situation, now we want to apply this 3 1 2 and suppose we apply 3 1 2 and 3 1 2 is also hitting here. So, h of 3 1 2 is also 10; so 10 which is same as h of 3 1 2. So, this is possible very much because h is a function from A to B. So, this is A set this is A B set of universe A and this is B set, ball very small set because number of slot is very small. So, there are this there has to be collision, because it cannot be a by injective mapping, it cannot be injective mapping because this is 2 sets this is a very smaller set than this one.

So, there has to be many keys which are mapping to the same slot and this situation is called collision and there has to be collision, you cannot avoid that because this domain is much bigger than the co domain. So, now, the question is how we can handle this collision, because if 2 points are mapping to a same slot, we cannot fit this 2 keys there because each slot is a room for capable to store only 1 key. So, what we do? We can have a chain. So, here 2 keys are colliding 5 1 2, 3 1 2 in the tenth slot. So, what we can have? We can have a chain or linguist; this is called chaining method chaining.

(Refer Slide Time: 08:47)



So, we can have a chain, which is just if in a slot if many if more than 1 key are colliding, we will put in a linguist. So, this is the hash function, this way we will store our keys.

So suppose we have a this is a set of universe of key, suppose we have this is our key set we are interested and suppose there are n keys k 1, k 2, k n and we have this hash table. So, now we apply the hash function and we put it into the table, so each of this. Now in the worst case, it may happen that all the keys are going to the same slot. So, all the keys are going to the same slot that may happened, see if that is the case that is called worst case in that case, what is the search time? If you have to search a key and if it is hitting to that particular slot, we have to read the whole list, so that will take a linear time, so that is the worst case.

But if we assume our hash function is good hash function, in a sense that it will distribute the key uniformly along the slot. So, some sort of uniformity is there; uniform distribution of the keys among the slots then we called our hash function is good hash function; that means, we take a particular key then it will go to the any of the same slot with probability 1 by m, so it is equally likely. So, keys are distributing very much uniformly over the slot so; that means, probability that k i is going to the j slot is basically 1 by m. So, each slot are equally likely for to fit a key. So, other way also, it is just a uniform distribution of the key among the slot.

So, in that case if there are m slots 0 to m minus 1 and if there are n keys. So, if n is the number of keys at some point of time and m is the number of slots, then we define n by m which is basically alpha which is basically. So, we have n keys we have m slot, so n by m is the expected number of keys in each slot. So, if we have say10 slot and if you have 100 keys anyways our hash function is a good hash function in that sense, then it is uniform the distributor of the key is among the slot. So, each slot will be having roughly n by 10, this is a sorry n by m this is basically 100 by 10000 keys. So, this is called this is the expected number of keys per slot. If our hash functions is good hash function in a sense that it distributing the key uniformly among the slot, expected number of the keys per slot and this is called load factor.

So, now, if we have these if our hash function is good hash function, then how much time we will take for searching a key how to search a key? So, we want search a key k. So, what we do? we have to first apply the hash function, so then one hash function is required then we reach to the slot and in that slot we are expecting alpha number of we have a chaining method, because alpha is the number of keys colliding there. So, we just have to read that, scan that list that is that chain. So, that will take this many times. So, this is the time for searching if our hash function is distributed in the key uniformly. So, this is the analysis of the chaining method.

Now, we will talk about how we can construct a hash function, some example of hash function.

(Refer Slide Time: 14:28)

So, first method we will discuss is called division method. So, suppose we have our keys are integer, keys are integer and with our table size we choose m, is only we choose m to be prime. Now so keys are integer, so hash function on this key is basically mod m. So, this is a set of integer, so we have a fixed size hash table 0 to m minus 1.

So, k is any integer of arbitrary length. So, this could be say n could be say some fixed length 3 bits so; that means, maximum value of n is 2 to the power 3 8 and this could be any size 5 1 2 bit, any length integer. So, we will just map into that fixed size. So, anyways this is the operation mode operation, so if you take the mod m, it will ultimately give us a remainder will be the integer from 0 to m minus 1. So, it will hit it slot, this is the division method.

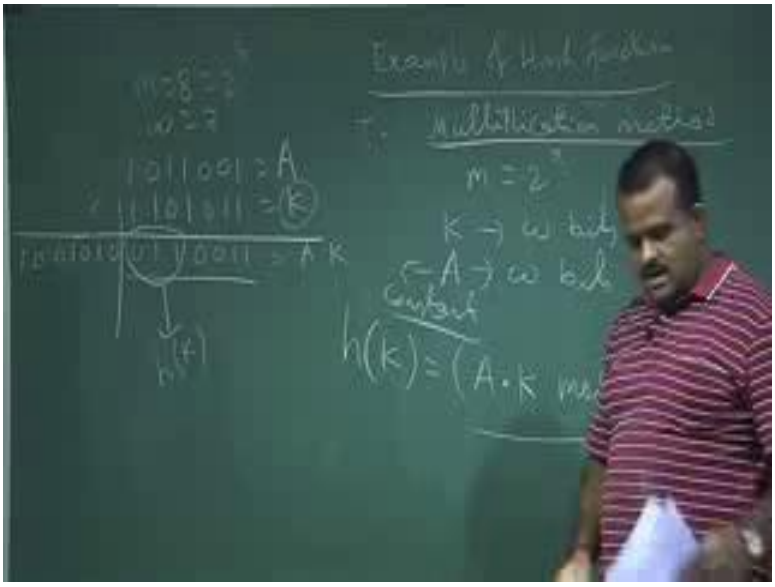
Now, in this method instead of choosing m this, if we choose m is equal to 2 to the power n, 2 to the power some r then there is a serious problem with this method, if we choose this suppose we have a key say like 1 1 1 1 0 1 1 0 suppose key say 2 to the power 7 bit, key is 7 bit say, so 1 2 3 4 5 6 7 8 9 10 11 12; suppose key is 12 bits h k. So, this is a particular key.

Now, if we choose m is equal to say 2 to the power 5; 2 to the power 5 means 1 2 3 4 5 this one. So, in that case, what is the k mod m? K mod m is basically this one. So, these are the high significant bit, so high significant means have no contribution with this hash value; that means, if we change any of this position, it will give us the same hash value if

we fix these bits. So, there are lots of collision there are. So, any change over here, there are how many keys 1 2 3 4 5 6 7. So, 2 to the power 7 bits are colliding, 2 to the power 7 keys are colliding into a particular slot, which is this and this is a huge number of collision is happening over here; that means, this hash function is not distributed in the key uniformly among the slots. So, this is not a good choice; m is 2 to the power r is not a good choice. So, we avoid this in order to having more collisions.

So, we all our hash function should distribute the key in alpha way load factor way. So, that is not happening over here. So, usually for this method we choose m to be prime, which is not close to some 2 to the power r or if you are in decimal system at 10 to the power r or if you are in octal system or hexadecimal or like this. So, this should not be close to this. So, to avoid this problem we have the second method, which is called multiplication method, this is one example of hash function.

(Refer Slide Time: 18:57)



So, another example is using the multiplication method. So, in this case we are allowed to choose m to be 2 to the power r. So, we are taking m is 2 to the power r, and suppose our computer words are w bit. So, our key is w bit binary bits.

Now, we choose a constant A which is also w bit. So, this is a constant we are choosing. So, now, what is hash function h of k what we are doing? We are multiplying A with K. So, if you multiply A with K. So, this will be A is w bit, this is w bit. So, this will be 2 w bit; now we want to make it again w bit, so we will do the mod 2 to the power w. So,

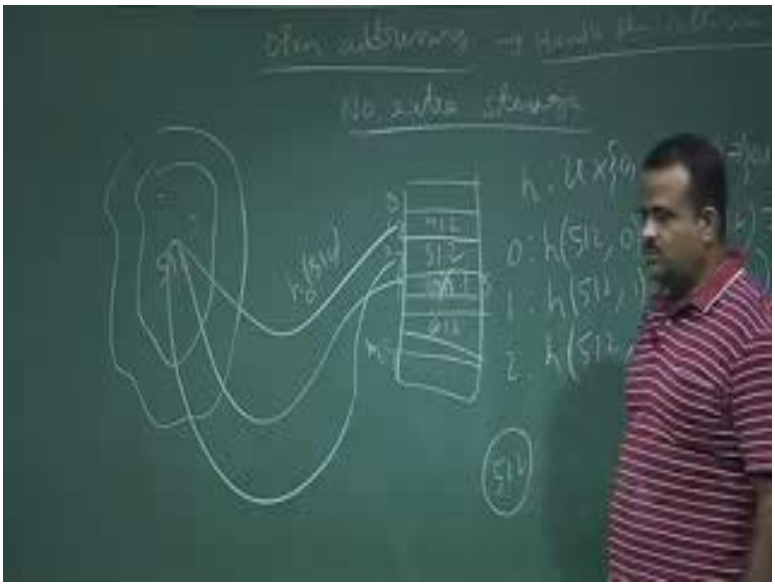
then it will give us a w bit, but now we want this to be r bit, because m is 2 to the power r. So, then we will do the right shift by w minus r bits. So, this is the hash function h of h h of k. So, we will take an example how it is give the values.

So, let us take an example for this suppose m is 8, which is basically 2 to the power 3 and our w is 7; now we take A, suppose A is 1 0 1 1 0 0 1 this is A, we have to choose a constant of 7 bit. So, this is A we are choosing. Now we have to choose a k for on which you have to apply this hash function. So, let us take a k 1 1 0 1 0 1 1 this is our key k. So, now, we have to apply h of k is by this formula, so we have to do the A into K. So, if you do this then we have we will be getting a 2 w bit; that means, 14 bits; so 1 0 0 1 0 1 0 then 0 1 1 0 0 1 1. So, this is basically A into K.

So, this is just a 14 bit, now if we want to make it 7 bit again, so we are taking mod 2 to the power 7. So, this will give us only this, this is mod 2 to the power. Now this we want to apply the left shift of w minus r; w minus r means w is 4 times we will do the left shift sorry right shift. So, if we see if this is gone, this is gone, this is gone, this is gone only this one is left. So, this is basically our h of k for this k. So, this is the multiplication method and here we are allowing the h to be m to be 2 to the power r. So, now, we will talk about there are other example of hash function, we can discuss later on.

So, now we will talk about how to handle the collision, without having a extra storage without having extra I mean we are not allowed to do the chaining because we are not having extra memory, so that will be handled by what is called open addressing.

(Refer Slide Time: 22:53)



Open addressing this is to handle the collision; you cannot have say there will be no collision in the expansion, collision will be there because domain size is bigger.

So, one way of handling the collision we will know that is the chaining method, but in the chaining method we need to have a chain in the outside the table, extra storage is required. So, in this suppose we have no extra memory, no extra storage. So, everything you need to do inside the table, only everything we need to do inside the table. So how we can handle that? We are not allowing to make the chain, we are not allowing to have the linguist is outside the table, everything we need to do into the table, that is the thing, that is addressed by the open addressing, so that is called open addressing method.

So, here what we have? We have a table of size n, sorry m minus 1 and say we have k many hash function k sorry we have n keys and here our hash function is basically it is we are the family of hash function basically, we have familiar hash function. So, we first apply that, so this is the probe number. So, this is first probe of the zeroth probe. So, we apply, suppose you want to insert a key which is 5 1 2. So, suppose we it has some keys over here 3 1 1, 6 1 2 like this. Suppose we insert 5 1 2 you have some other keys, now we first we try for the first probe. So, this is basically h 0 of 5 1 2, we are many hash function. So, suppose it is hitting here, h 0 5 1 2 means 1. So, this is not empty, we cannot put it there, it is occupied. So, then we have to go for the next line, we have to go for the next hash function, that is the next probe. So, 5 1 2 1, this is the next hash

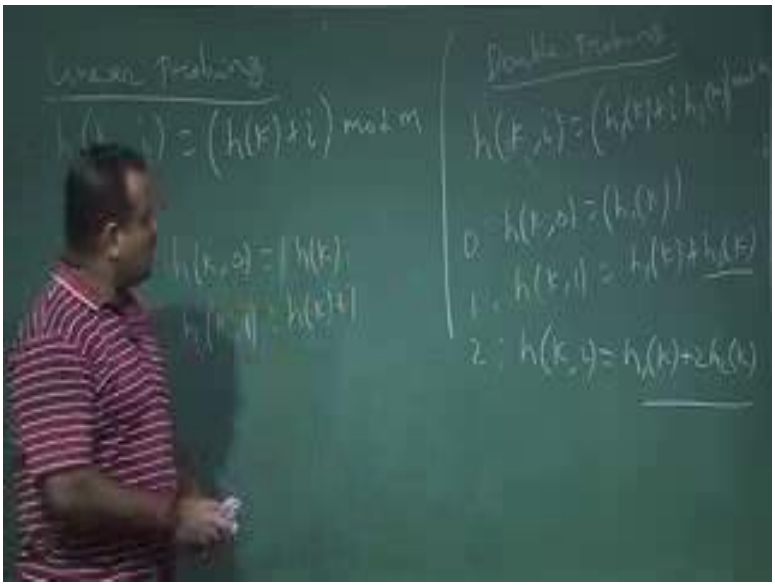
function suppose again. So, this is first probe or the fastest again suppose this is hitting here. So, this is say fifth slot and which is occupied by somebody else. So, this is not empty, so next suppose we next we have to go for the next probe 5 1 2, 2 this is another hash function.

Now, suppose it is giving us some empty slot. So, this is the second slot 5 1 2. So, this is we fit it there, this is the way. So, we have a collection of hash function, we will keep on searching the empty slot until we succeed. So, every time we this is the probe sequence you have to follow. So, we first apply the h 0 the first I mean the. So, we have collection of hash function. So, first probe zeroth probe will apply that this one, if it is hitting a occupied slot we will try next one, next probe next probe like this until we reach to a until you hit to a empty slot. Once we hit to a empty slot we will store it there.

Now deletion is a problem with this because suppose, so it is hitting first it is hitting here, next it is hitting, suppose we deleted this 3 1 2. So, 3 1 sorry 3 1 1 or something, we deleted this and we make it this slot empty. So, what we will do? Then after that suppose we are finding 5 1 2, you want to find 5 1 2. So, we will follow this we will apply the first probe or zeroth probe, it will hit to here then we apply next probe it will hit to here, but we see this is the empty slot then we may report that 5 1 2 is not there, but that is wrong. So, somehow we need to have some information over here this room was originally not empty, somebody was sitting here he or she has died. So, this room has some, so some sort of that we can need to put that some keys are there this room is not from the beginning empty somebody was there. So, we put a tag over there so that we can go for further probe to get that 5 1 2. So, that is the deletion we need to handle.

So, now let us analyze this probing strategy I sorry open at the same stage with strategy and then we will take some example of a probing. So, first example is linear probing.

(Refer Slide Time: 28:29)



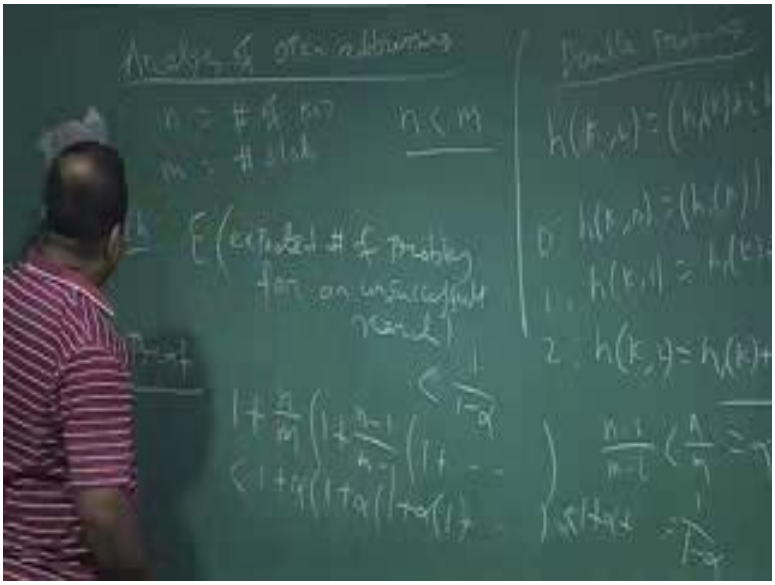
So, in linear probing it is basically ith probe is; so h of k plus i mod m. So, it is basically, we have the hash function h will apply h and if it is. So, we have a key, we apply h on that, so zeroth probe. So, this is the 1 probe. So, for further the first probe or the zeroth probe what we are doing, we are just applying h of k mod m. So, it is hitting somewhere if it is not free if it is not empty then we will go for 1 probe, 1 probe is basically done in first next probe h of k plus 1 a mod m.

So, we will look at the next. So, this is hitting, in that we look at the next slot and if it is not empty then we look at the next slot that is k 2 like this. So, we are just looking at that serially like this. So, next slot next slot whether it is empty or not. So, these are the problem of clustering; if it is hitting an occupied slot it will just put it into the next. So, there is a clustering with these slots. So, this is not uniform distribution along the slot that has a problem with this.

So, next is to avoid this problem we will apply what is called double probing. So, in the double probing this is the jth probe, we have 2 hash function mod m. So, what we are doing? So, zeroth probe h of 0 is basically we are just calculating h of k if it is hitting to a occupied slot then we go for the next probe, which is basically h of k h 1 of k plus h 2 of k. So, we have another hash function we are adding then mod m; obviously, then again if it is hitting so we will go for the next probe, but we already calculated these 2

value so we just do this mod m, this way I will check. So, this is better than the linear probing to avoid the clustering.

(Refer Slide Time: 31:12)



So, let us just quick analyze this open addressing. So, now, if you have a open addressing method to solve the collision, then suppose there are n keys n is the number of keys and n is the number of slots. So, for open addressing; obviously, n must be less than m otherwise we cannot handle everything in into the table, we are not allowing any extra storage. So, this is a theorem it is telling us the expected number of probes for an unsuccessful search is equal to is less than equal to 1 by 1 minus alpha, alpha is the load factor. So, we need to probe this, how to probe this? So, suppose there are n key; one probe is mandatory, we need to apply the hash function one probe is mandatory; then one is go for second probe if it is hitting to a occupied slot. So, what is the probability that it will lead to occupied slot? Is basically n by m because n is the total number of slot that is the total number of cases and n is the total number of keys which is occupied in the slot. So, it will hit to a slot which is not empty means, if the just the classical definition and then again then will go for the next probe.

Now, again when you go for the next to next probe, if it is hitting again occupies slot. So, that probability is this. So, these one we will continue. So, this is basically less than, we can use this n minus I is less than basically 1 by m. So, if you do this and this is alpha; we can just write this less than equal to 1 plus alpha into 1 plus alpha into 1 plus alpha so

on 1 plus so on. So, this is basically less than equal to 1, less than 1 plus alpha plus alpha square so on. So, this is 1 by 1 minus alpha. So, this is 1 by, now if our table is half full; that means, if alpha is 50 percent, alpha is half then what is 1 by 1 minus alpha? It is basically 2.

(Refer Slide Time: 33:47)

So that means, if the table is half full, we need only 2 probes for an unsuccessful search, but if the table is 90 percent full, 0.9 then we need we need 10 probes to get unsuccessful search.

Thank you.