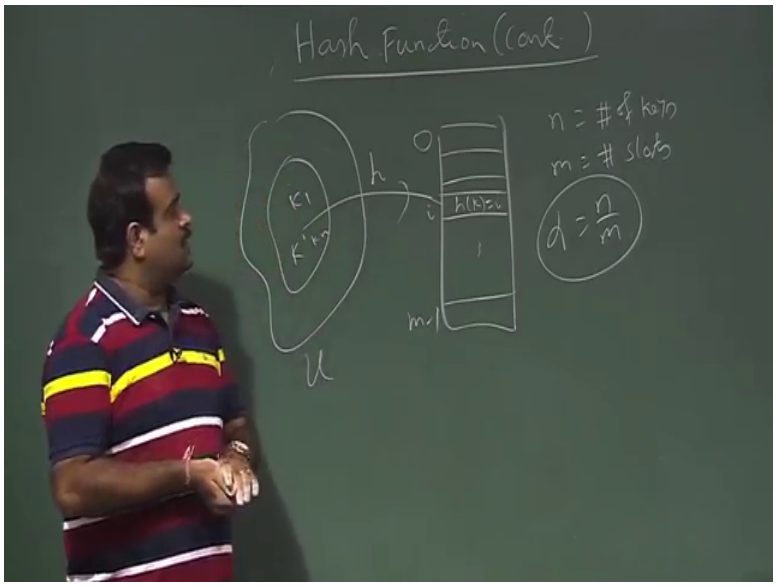
An Introduction to Algorithms
Prof. Sourav Mukhopadhyay
Department of Mathematics
Indian Institute of Technology, Kharagpur

Lecture - 22
Open Addressing

So we have seen the hash function, which is the function from set of universal of the key to a given slots.

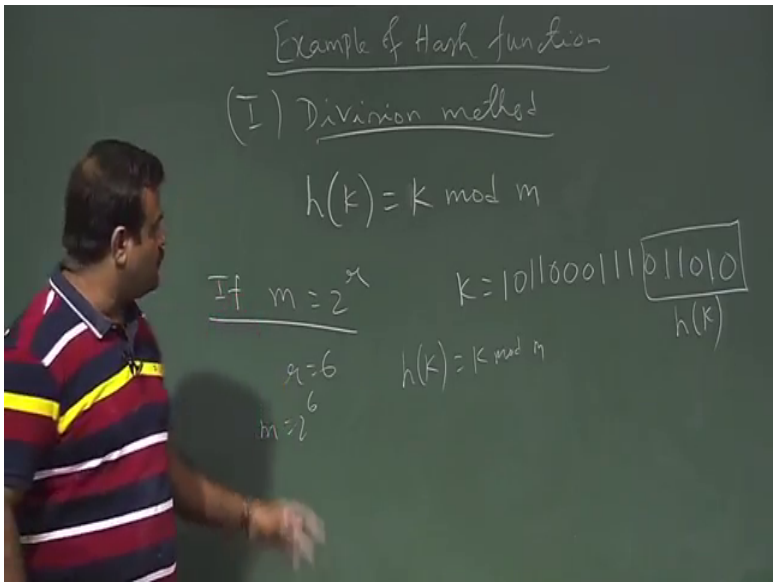
(Refer Slide Time: 00:25)



So, supposed to 0 to m minus 1 slots. So, it is a function from u 2. So, so you given a key if we apply this hash function it will be it will map to a particular slot. So, h of k is a i. So, now, we say that we know hash function is good hash function, if there are same keys n is the K 1, K 2, K n and there are there are m slot.

So, if the keys are distributed in uniformly over this slot. So, n is the number of keys and m is the number of slots, then alpha n by m which is basically expected now this is called load factor, which is basically expected number of keys per slot. And then we call this hash function is a good hash function, because it is distribution in the key uniformly over the slot. So, now, in this lecture we will construct few we will try to construct hash function and we will see whether this is a good hash function or a bad hash function. So, let us talk about some example of hash function, how we can choose a hash function.

(Refer Slide Time: 01:53)



So, example of hash function. So, first 1 is the division method. So, here we are assuming our keys are basically integer. So, we have integers all the keys are integer basically and we have a slot of size m which is also an integer. So, m minus 1 these are slot. So, we take an integer. So, we need to after applying the hash function it should be 1 of this slot. So, m minus 1. So, what is the function we can use for that what is the obvious function we can think of that modules function yes so; that means, h of k is basically a k mod m, were k is a integer m is also an integer. So, if you take the mod, mod mean if you divide this k by m k is key, key could be any big integer.

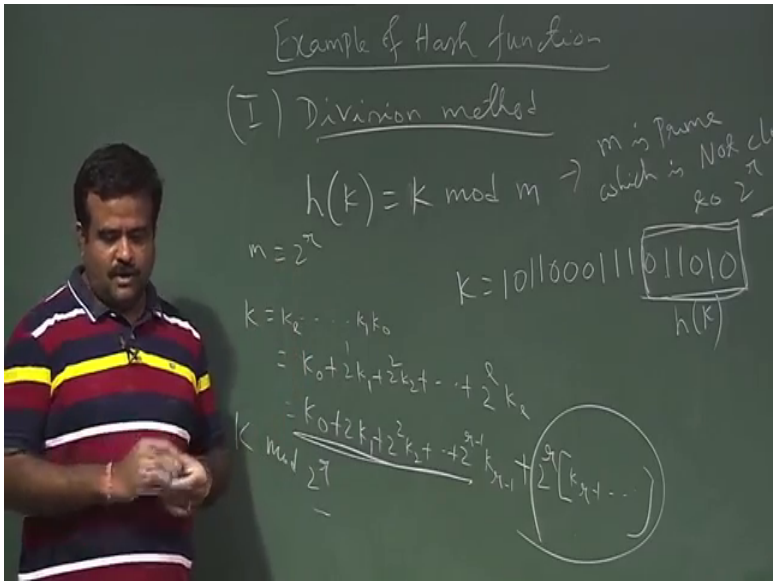
So, any big integer we take and we try to divide this by m and this is the reminder. This is the reminder and these reminder will be form 0 to m minus 1. So, this is the hash function, now we will talk about how this hash function is. So, whether this is the good hash function or bad hash function; that means, whether it is distributing the keys uniformly over the over this slots or there is huge number of collision for this so for that. So, if our m is a sum 2 to the power r, then we have a problem what is that problem. So, if n is 2 the power r.

So, then suppose we have key say k which is a integer and we convert in to binary say. So, suppose this is 1 0 1 1 0 0 0 1 1 1 0 1 1 0 1 0 suppose this our key, we have an integer we convert into binary. You know how to convert into binary we try all we divide by 2 and then again 2 like this. So, we will get all this component. So, this is basically the

binary representation of our integer. Now suppose k is basically 2 to the power r say r is 6 here say. So, for this r is a 6. So, m is basically 2 to the power 6. So, your if r is 6 then what is h of k? H of k basically k mod m. So, 1, 2, 3, 4, 6. So, this is basically our h of k.

If we choose r to be 2 to the power m to the 2 to the power r, I mean here r is 6 then this is basically h. So, does not matter what are the values are so; that means, if we take any bit differ over here. So, they all they are colliding to the same. So, if we fix this, why this is happening because if you take mod of this, mod of 2 to the power r then we take up 2.

(Refer Slide Time: 05:38)



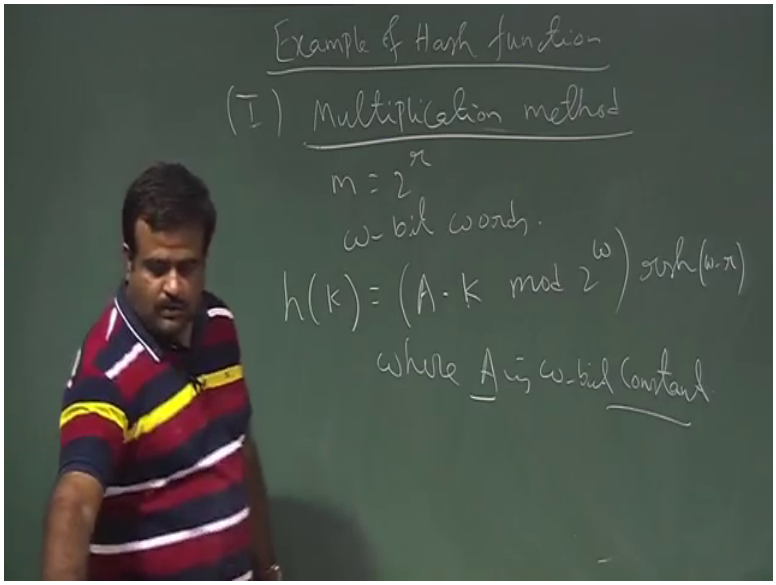
So, this is basically we can just try to write this. So, this is basically if k is say, k 0 k 1 dot dot dot k l. So, this is basically k 0 plus 2 k 1 plus 2 k 2 this is binary dot dot dot 2 to the power sorry 2 square.

2 square k yeah 2 to the power l k l. So, now, this can be written as k 0 plus 2 k 1, plus 2 square k 2, plus 2 to the power r minus 1 k, k r minus 1 plus 2 to the power r then we take the common here then kr plus dot dot dot like this. Now if you take if our m is 2 to the power r now if you take this (Refer Time: 06:42) or 2 to the power r then; that means, this portion will be vanish and only these values will come. So, that is why if r is 6 this is the this is the key mod 2 to the power 6 does not matter what are the values these are. So, if we change these values any one of these values, by fixing this values all are colliding into the single slots.

So, there are huge number of collision is happening. So, it is not distributing uniformly over the slots. So, this choice of m is not a good choice. So, this is not a good hash function in that sense if we choose m to be like this. So, usually for this we choose m to be prime is a prime number, which is not close to close to some 2 to the power r or 10 to the power r; because we may consider into the decimal system also. So, that is the choice of m 1 should be considered if we are having this division method. So, division method have this issue that we should not choose m to be 2 to the power r then we have a huge number of collision ok.

The second example is the multiplication method. So, there we are allowing m to be 2 to the power r. So, multiplication method this is another example of hash function ok.

(Refer Slide Time: 08:30)

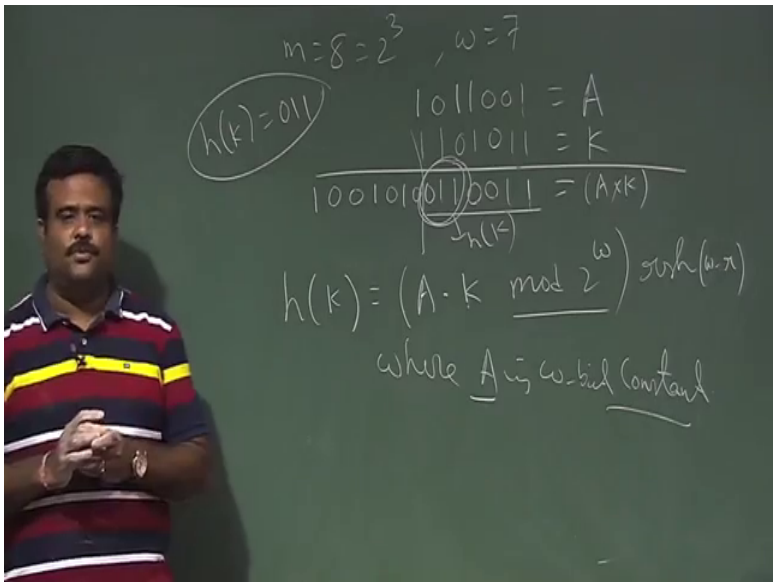


So, what we are doing here. So, basically we have here we are allowed to choose m to be 2 to the power r and here we are our numbers are said all are say w bits. So, our word at w bit word. So, our key is w bits. So, our computer is w bits. So, 64 bit computer nowadays. So, 60 w could be 64 ok.

So, now we. So, our key is w bit. So, h of k is basically a into k will. So, a is the constant again up w bit we have to choose that a, mod 2 to the power w this is into and then left right shift how many time w minus r times, right shift w minus r time. So, where A is a w bit constant. So, if our computer is 64 bit, we choose a 64 bit constant A and then we choose a key which is again a 64 bit, w is 64 say then we multiply these 2 then it will 2 w

bit now we want to make it again w bit. So, that why we take mode 2 to the power w. So, it will again become a w bit. So, the last w bit and the we will do a left shift I sorry right shift w minus at times. So, this is basically called the multiplication method, we will take an example for this suppose.

(Refer Slide Time: 10:42)



So, let us take an example of this suppose say m is 8, 2 to the power 3. So, we have 8 slot. So, 0 to 7, and our w is say 7 bit suppose our computer is 7 bit computer I mean whole computer ok.

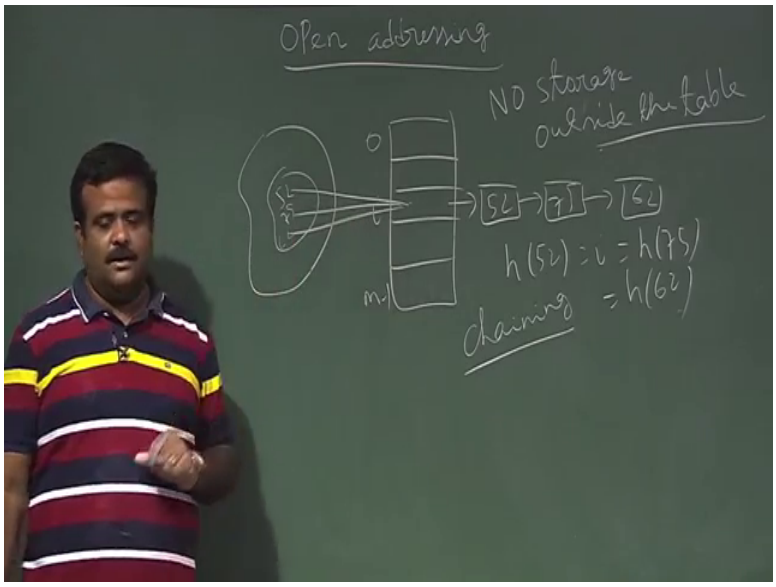
So, now we choose a, a is also 7 bit we have to choose a say a is constant we choose 1 0 0 1. So, this is our A set. Now we choose a e which is also say which is also 7 bit and we need to apply the hash function by this formula. So, we let us take a key. So, 1 1 0 1 0 1 1. So, this is our k set. So, now, we apply this we multiply this A into K, if we multiply this this will be this is 7 this is w bit, this is this w bit, then the result will be 2 w bit this is 7 bit this is 7 bit. So, result will be 14 bits. So, if you do that it will be like this 1 0 0 1 0 1 0.

Then 0 1 1 0 0 1 1. So, this is again fourteen bit now we have to take mod of 2 to the power w, mod of 2 the power. So, this is up to 7 8 1 2 3 4 5 6 7. So, this is the this is the mod of 2 to the power w, and then we will do the right shift of w minus r times. W is 4, r is 3 where w is 7 r is 3. So, fourth time if we sweep these 4 times, then what then this is

gone this is gone, this is gone. So, this is our basically h of k this 3 bit. So, h of k is basically here 0 1 1 this is our h of k. So, this is the multiplication method.

And here we are allowed to take m to be 2 the power r and this is and we have seen that people have seen this is sort of distributing the key uniformly over the over the slots. So, One can have some more example of hash function, but this is the way we choose the hash we have discussed 2 method one is division, another one multiplication method ok.

(Refer Slide Time: 13:48)



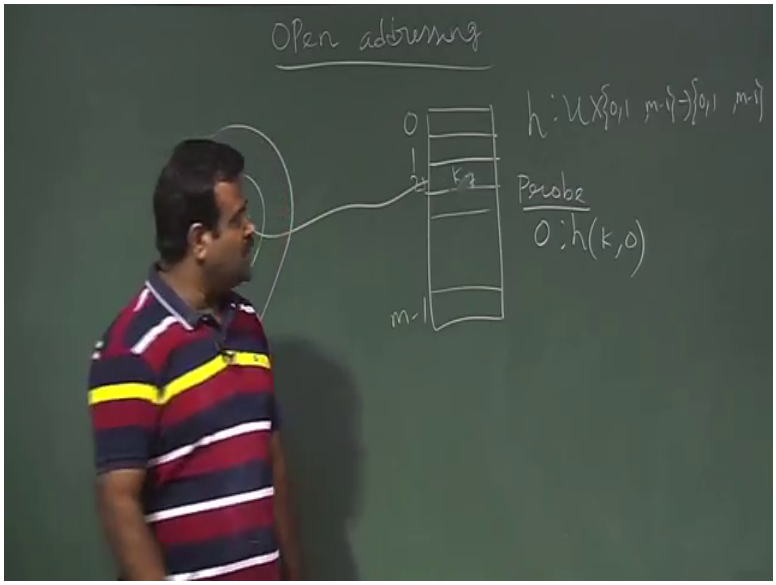
So, now we will talk about a strategy to handle the collision which is called open addressing. We have seen a strategy for handling the collision that was basically chaining. So, now, we discuss another way we can handle the collision open addressing ok.

So, in the chaining what we did? We did what. So, if we have a if we have this slots 0 to m minus 1 and if we have some keys and if some keys are colliding in a slots, then we have a chain say 75, 6. So, this all this h of 52 is equal to i is equal to h of 75 is equal to h of 62. So, these are all that keys which are colliding into the. So, 52 75 62. So, these are the keys all are mapping to the same slots ith slots, and then then to this is the collision and we handle this collision by having a chain outside the table. So, this is a extra storage. So, we have to maintain a linked list outside the table.

So, suppose this is not allowed, suppose no storage no storage outside the table suppose this is our restriction, then we cannot do the chaining? This is chaining. So, if you are allowed to do any extra storage outside the table, then we cannot have a linked list outside the store outside the table. So, the chaining cannot be applied there are. So, in those extra storage is available outside the table. So, everything we have to do in the table. So, we have area we have table or area of size m, 0 to m minus 1 and everything we have to do there. So, that problem we have to address by open addressing. So, no extra storage.

So, chaining is not allowed for chaining we need to have a linked list outside the storage outside the table, but that is not allowed here ok.

(Refer Slide Time: 16:16)



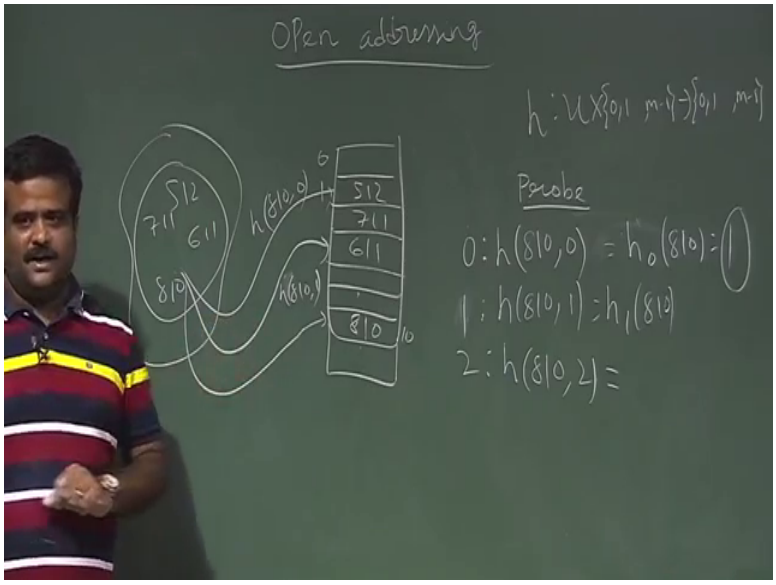
So, what is the idea? Idea is. So, basically what we have we have a table 0 1 2 m minus 1. So, we have a key say we have key, we are mapping over here. So, now, suppose it is heating to a empty slot and there is no issue, but suppose it is heating to a. So, suppose this is a k 10, suppose it is heating to a slot which is already occupied then what we have to do we have to search for a next slot, because earlier method we have a had a chain, but here chaining is not allowed here no outside storage is available.

So, linked list is not allowed in the outside the table. So, then we have to find a slot which is free. So, to find that slot we cannot apply the same hash function. If you apply the same hash function, then it will again heat here so; that means, we have to apply a

second hash function and then again if that second hash function heating some occupied slot then we have to apply a third hash function. So, this is the probe sequence. So, you have to apply sequence of hash function or sequence of probing. So, we have probe sequence. So, basically here our hash function is for open addressing it is basically function for u cross 0 1 to m minus 1 to 0 1 to m minus 1 ok.

So, basically we first. So, this is the probe sequence probe sequence. So, we first zeroth probe. So, we apply the 0th hash function and suppose it is heating to. So, so suppose it is heating to some occupied slot over here.

(Refer Slide Time: 18:30)



So, this is our k. So, suppose. So, let us draw some example it will be more clear, suppose this is the scenario. So, somebody is seating here 5 1 2 6 1 1 say 7 1 1 like this suppose this is this are the key we have where 7 1 1 6 1 1, suppose we have a key say 81 0.

Now, suppose you want to insert 8 1 0 here. So, what we do we apply the. So, we try for the zeroth probe we apply the hash function on 8 1 0 ,suppose this is heating here 0 1 suppose this is heating here. So, this is this is say 1 which is not empty slot. So, then we have to go for a next probe. So, this is basically the hash function which is giving us the value 1 which is not available which is not available. So, this is the first hash our original hash function maybe then we go for the next probe, 8 1 0 this is basically h 1 of 8 1 as if we have sequence of hash function. So, if we have (Refer Time: 19:47) because if you

because if you say. So, same hash function it will be here. So, we use. So, suppose it is heating some slot which is. So, this is basically h of 8 1 0. So, this is h of 8 1 0 1.

Suppose again it is heating some occupied slot, then we have to go for the next probe and suppose it is heating some slot which is available. So, this is the ten slot then we put 8 1 0 over here. So, this is the idea. So, we are basically sequence of hash function. So, we have sequence of probe sequence, we first attempt the zeroth probe or maybe original hash function if it is heating to some occupied slot then we have to basically the idea is to search for a empty slot by this probing probe sequence, then we will go for the next hash function or the first next probe and if still heating a some occupied slot, then we will go for the next probe like this. So, this is the way we insert the keys into the table ok.

And basically. So, then our hash function is basically this u to the set of universe of the key and this is the probe sequence and we will do up to m minus 1 times, and this is the slots available this this is the slots 0 to m minus 1. So, this is our function basically we have sequence of hash function k 0, k, k 0, k 1 like this. So, if one function is given to some empty slot a some occupied slot we use the second hash function with the hope that we will get a empty slot and then until we continue like this we continue this probing until we reach to a empty slot once we reach to a empty slot we will put that. So, this is the way we insert an key ok.

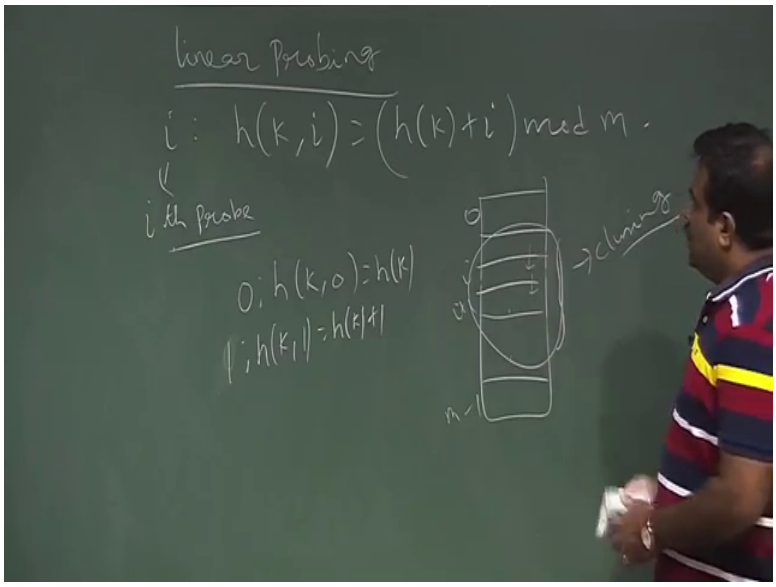
Now, how to search? So, searching is basically. So, suppose we want to search say 8 1 0. So, what we do? So, suppose you are searching 8t 1 0. So, what we do? So, we need to follow the same probe sequence as they have inserted. So, we first apply the zeroth probe. So, it will heat here. So, we will see whether these values 8 1 0 or not it is not. So, we will go for the second probe then we will heat here. So, we will see whether this is 8 1 0 or not it is not, then we will go for the next probes then we will see it is the 8 1 0. So, we got the value. So, this is the way we search. Now deletion is little has to be tricky here because suppose we delete this 6 one 6 1 one suppose and then after the deletion suppose we are searching 8 1 0 then what will happen.

So, we will do this probe sequence in the zeroth probe we will heat here. So, we check 5 1 2 is not is is not matching with 8 1 0 then we will go for the next probe, it is heating to a empty slot then we stop, but this is that correct because we it is not originally empty

somebody was there, but it got deleted. So, that that information has to kept somewhere here in a single bit. So, that we can go further otherwise if we see a empty slot will stop. So, deletion has to be in the in the deletion we have to take care this. So, if somebody if we delete somebody then we have to put a tak over here, that this room was not originally empty somebody was seating there and it got deleted. Otherwise we cannot find 8 1 0 if we found this is the empty the we stop, but it is not the correct one.

So, we if we see this bit it is wrong; that means, somebody was there then we go for the next probes and we will found 8 1 0. So, deletion 1 has to be careful in the deletion. So, now, this is the oh this is called open addressing. So, basically we have sequence of probes and by there we will find the empty we will try to find the empty slot in the table ok.

(Refer Slide Time: 24:37)



Now, we take some example of this probing there are basically we will talk about 2 example 1 is linear probing. So, 2 example of open addressing linear probing. So, in the linear probing the h of k comma i. So, this is the ith probe is basically h of k plus I mode m ok

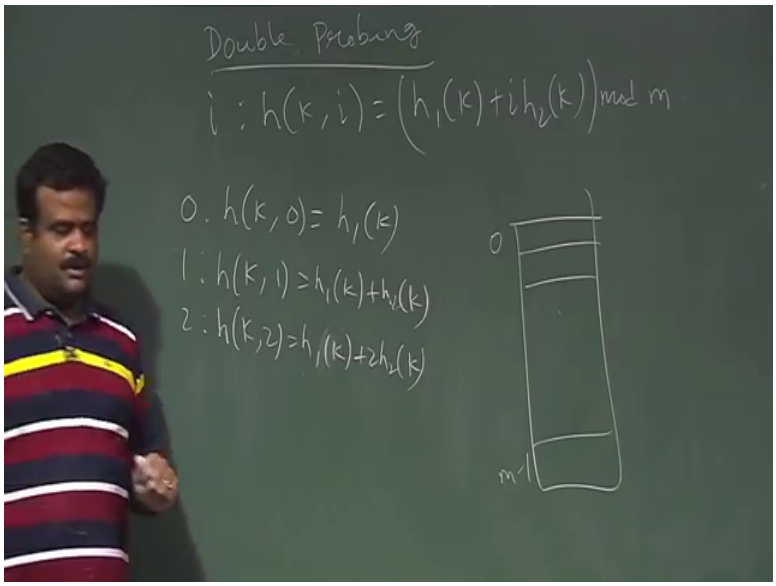
. So, this is the ith probe ith probe. So, this very simple I mean basically if we this is our slot this our slot m minus 1 0 to m minus one. So, suppose it is heating some occupied slot then this basically is looking the next 1 where next 1 is available or not. So, if it is available they are putting there. So, this is because we have to everything we have to fix

into the table we are not allowed to have the chain. So, we just try to see the next 1 is available or not this is the way we we see whether the next 1 is available or not so; that means, we the zeroth probe is basically our h of x h of k. So, we will go there and if it is not available then we go for the first probe.

so; that means, h of. So, means 1 this is basically h of k plus one. So, we take we check suppose it is heating the ith slot then we check the ith plus 1 slot is empty or not. So, like this. So, we have to take a mod because it means then we have to come back here. So, this is the linear probing, but these has a drawback because it is sort of clustering if it is heating somebody here and if this is empty then it is basically clustering in some portion. So, clustering. So, it is not uniformly distributing the keys over the slots it is sort of clustering if it is heating some occupied slot then after that it is basically making a cluster it is not distributing over this.

So, to avoid that there is another example of open addressing is called double probing. So,

(Refer Slide Time: 27:02)



So, in double probing basically we use 2 hash function h 1 and h two. So, in the ith probes is basically. So, h 1 k plus I h 2 k mod m. So, basically this is our table. So, in the zeroth probe what we do. So, h of 0 is basically h 1 of k. So, we say h is our original hash function. So, we apply the hash function if it is heating to some occupied slot then we go for the next probe then it is basically h 1 k plus h 2 k.

So, we have another hash function we just add it and; obviously, mod m check whether again anything a occupied slot or empty slot or not if it is not available then we go for the next flow which is basically h 2 of h 1 of k plus 2 of h 2 of k since we have already calculated h 1 h 2 we can use this value to calculate this we did need to calculate again h 1 h 2 like this. So, we check this; obviously, mod m we check this whether we are getting a occupied orders. So, this way depending on the is h 1 and h 2 are good hash function distributing then this this this is shown to be a good hash function over the I mean its distribute the keys in the uniformly over the slot. So, this is slightly better than the double hashing or 1 can think of it (( )) triple hashing or some quantity hashing over here.

So, in the next class we will analyze the open addressing method and we will see how good or how bad this is.

Thank you.