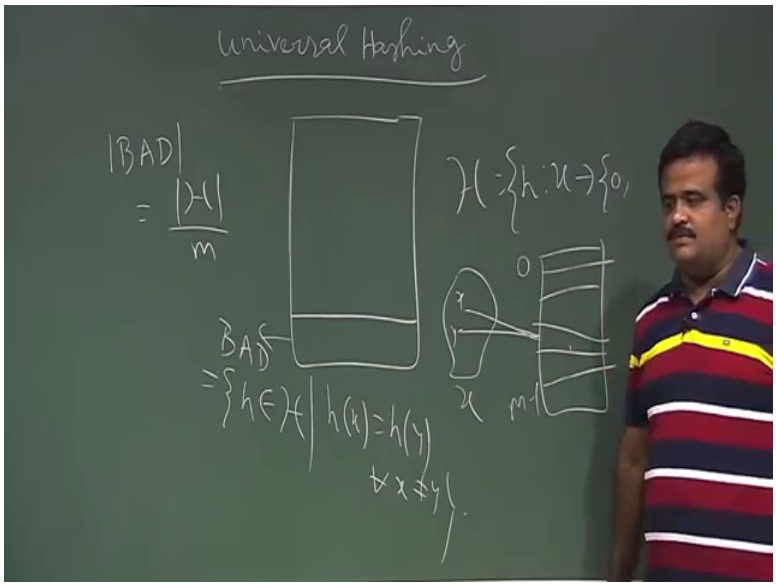
An Introduction to Algorithms
Prof. Sourav Mukhopadhyay
Department of Mathematics
Indian Institute of Technology, Kharagpur

Lecture - 24
Perfect Hashing

So, we talk about perfect hashing so before that let us in the last class we have discussed the universal hashing. So, today we will start with an, a construction of a universal hashing an example of a universal hashing. So, let us just recap what is universal hashing.

(Refer Slide Time: 00:36)

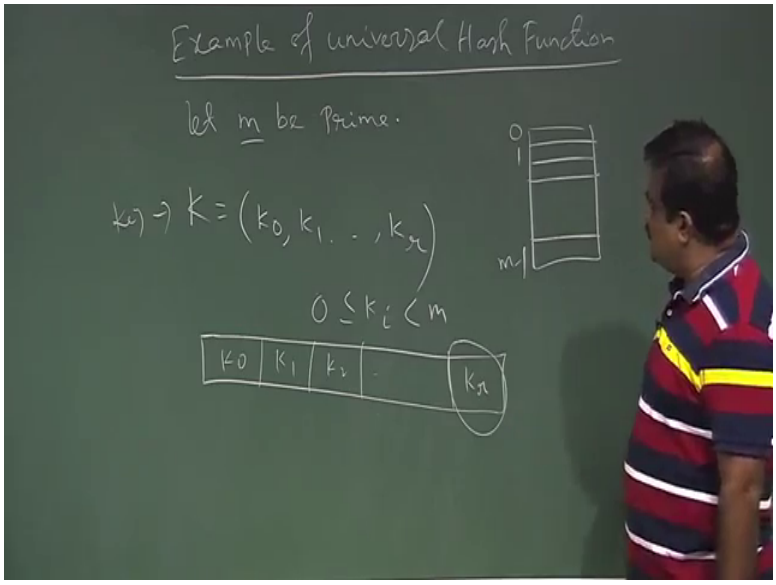


So, basically it is a collection of hash function H is basically hash function from u to 0 1 up to m minus 1. So, our table size is, our table size is 0 to m minus 1 this is our hash table size and we consider a collection of hash functions such that among these collection there is a bad portion and this bad portion is basically set of all functions set of all hash function such that given any 2 key x y they will collide for all x not equal to y if we choose any 2 key, if we choose any 2 key x y which they are not same then if we apply this hash function then they has to collide.

So, if we if our hash function is coming from this portion then there is a guarantee that there will be a collision. So, if you choose any 2 key then this has to this will collide. So, this is, this is that why it is called bad, bad portion and if the bad potion cardinality of

bad portion if, if the size of these set is basically 1 by m fraction of total set m is the table size basically, then this collection is called universal collection of hash functions and if you choose a hash function randomly from this collection, suppose we have such a collection and if you choose a hash function randomly from this collection then that is called universal hashing that hash function is called universal hashing. So, now, we talk about an example of how to construct such a universal hashing or such, such a collection. So, that is the, an example or we have to construct such hash function or such collection so construction of or example of universal hashing function.

(Refer Slide Time: 03:03)

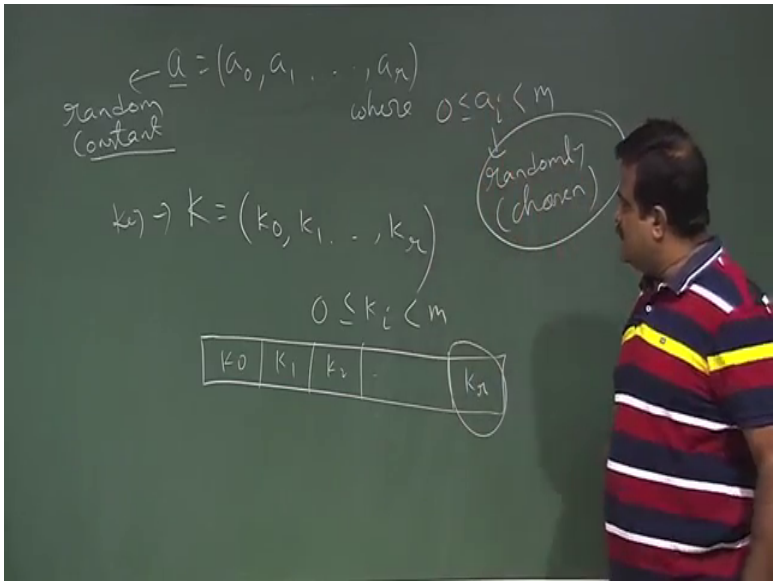


So, here we so, we have given key suppose keys are basically, so this is a random choice. So, suppose let m be a prim which is the table size, so our table size is m so we have a 0 1 up to a m minus 1 this a table size and this is a prime number, in a prime number; that means, a number which is divisible by only itself or 1. So, there is no factor, factor of that number this is an integer. So, now, we have a key K we decompose this key into r r bits, r digits.

So, k 0 k 1 I mean r plus 1 digits. So, k r where each of this k's are coming from this so, the value is maximum value is m. So, if we have a given key, key usually very long so what we are doing, we are dividing this key into digit like k 0 this is k 0 k 1 k 2 like this. So, last one is k r and each of this is basically bounded by each of this value is less than m. So, this is the decomposition on we are doing on the key. So, this is a key ok.

So, now what is the random strategy here? So, now, this is the key now we are choosing a constant a vector.

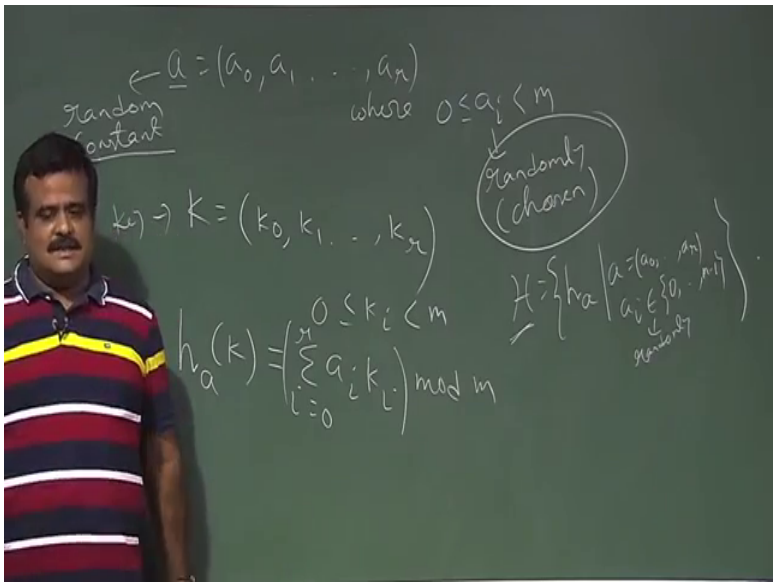
(Refer Slide Time: 05:13)



So, we are choosing a constant a which is also a 0, a 1, a 2 a r where a i is a i is also form 0 to m and a i are chosen randomly, a i is chosen randomly. So, this is the random choice of a i. So, this this vector this is basically a vector a is a vector which is a basically random numbers and each digit of these. So, these are all random digits and these are coming from the value is coming from 0 to m.

So, this is con, this is a constant which is chosen randomly from this is a random constant, random constant. So, we are choosing a a i randomly from this then we construct this we have this a, then how to define this hash function then hash function h of. So, hash function will depend on this constant

(Refer Slide Time: 06:30)



So, it is denoted by h of a, a is this random vector h of a on k. So, k is the vector it is basically the inner product of this and this. So, it is basically the summation of a i k i, i is equal to 0 to r then mod m. So, this is basically our hash function. So, it is basically a 0 k 0 multiplication.

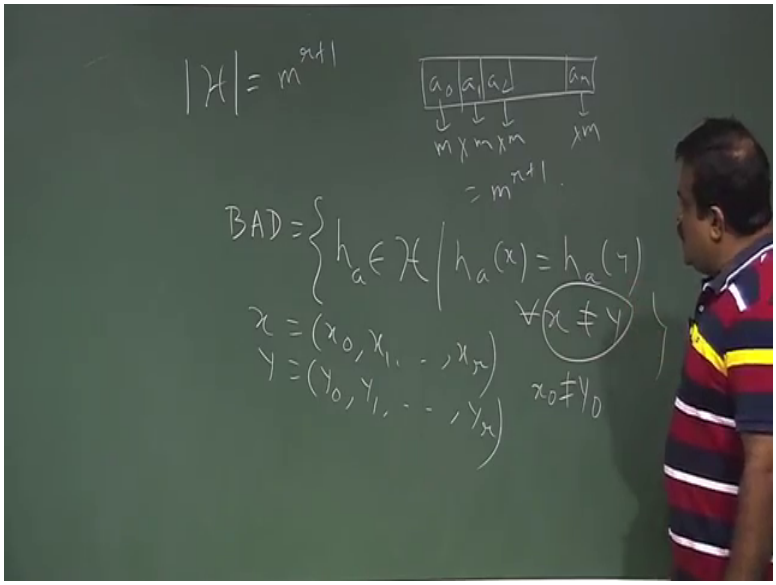
These are all integer so, a 0 k 0 plus a 1 k 1 dot, dot, dot plus a r k r and then this inner product may be more than in then we have to take the mod m to fit in to the table. So, this is the hash function. So, this is called this hash function is depending on this choice of a. So, if we choose a randomly then we have a. So, for different values of for different, different a, we have different different hash function. So, this a, we are going to choose at the run time. So, now, so if we considered this collection like H h is basically this collection a where a is coming from a is basically a 0, a 1, a a r and a i are basically coming from this set.

This, this is 0 one up to m minus one and this choice is randomly so this collection, this collection is a hash function collection. So, where we are choosing this a i is randomly from this set 0 to m minus 1 and then once we choose a i randomly we have this a vector random vector a then use that random vector a we define the h of a by using this formula in our product formula. So, this our hash function and this is our collection. So, we want to know whether this collection is a universal collection of hashing so whether this collection is a universal collection. So, to prove that we have to see the bad portion I

mean what is size of the bad portions. So, let us just have the cardinality of H first and this, the random strategy because we are choosing the hash function randomly at the run time because we do not know which a i we are going to choose this a i we are going to choose at the run time.

So, once we got the a i then we have the hash function using the formula ok.

(Refer Slide Time: 09:23)



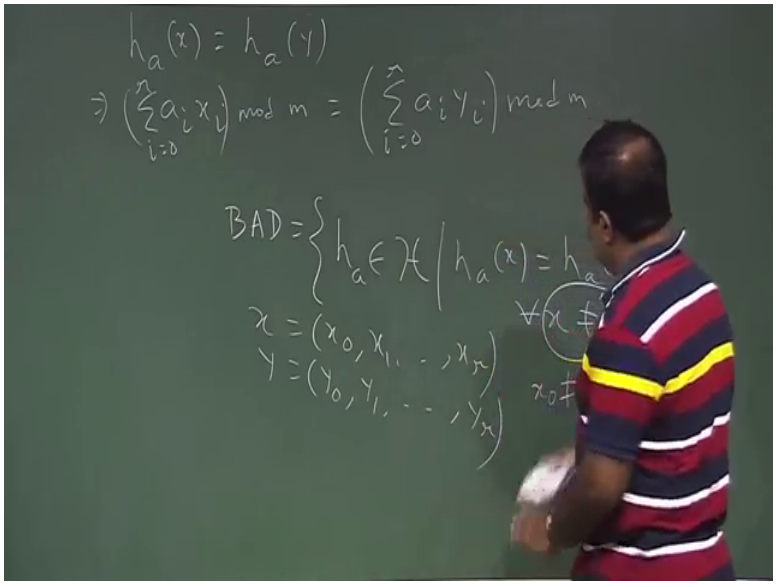
So, what is the cardinality of this set? So, what is the size of this? So, this is basically wearing this a a digit. So, then how many there are r plus one a a i and each of then take value m. So, size of this is m to the power r plus 1 because this is basically the choice of a is so a 0 a one a 2 like a r. So, each of a i can be chosen m ways this is m ways, this is m ways, this is m ways. So, the total possibility is m to the power r plus 1.

So, this is the cardinality of, cardinality of this collection H now we want to see whether this collection is a universal collection or not. So, for that we need to consider the cardinality of the bad portion. So, let us just talk about cardinality of bad portion so for the bad portion what we have, we know the bad portion is such that this is the set of all function h from this such that h of x equal to h of a y where x y not equal. So, if, if given any 2 key and this is to for all x y, given any 2 distinct key if they are colliding and if this is true for all such keys, such pair of keys then that collection is called bad portion.

Now we want to we want have to cardinality of this bad portion, to have the cardinality of the bad portion let us take x is a key so, x is also in this form x r and y is a another key y 0, y 1, y r and since x is not equal to y. So, any 2 digit of this at list any 2 digit of this will be not equal to y. So, for the simplicity we are taking x 0 is not equal to y 0, without loss of generative for the simplicity we can assume because we want x is not equal to y.

So, now we take this is equal so we actually want to find out the cardinality of this. So, h of a x equal to h of a y.

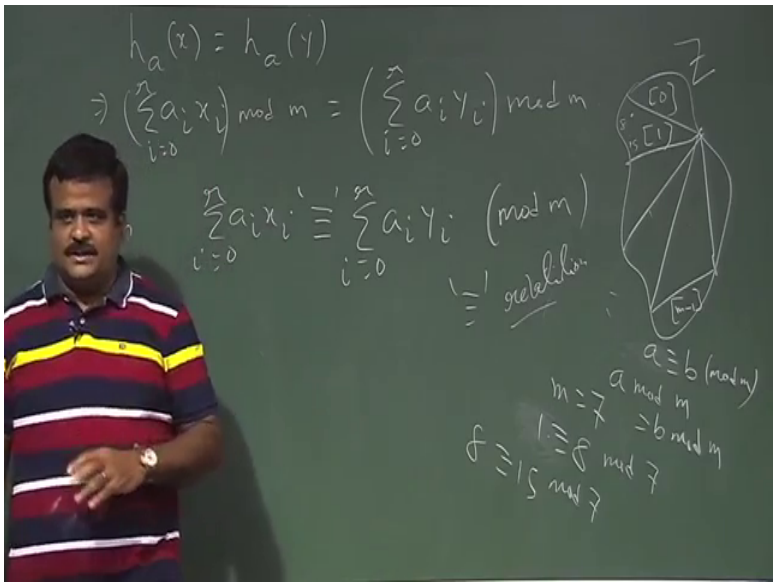
(Refer Slide Time: 11:55)



So, this means, this implies summation of a I, x i is equal to summation of a I, y I so, mod m and this is also mod m so, this is also mod m. So, this means summation of so, i is from 0 to r, mod m is equal to summation of a i, y I, 0 to r mod m. So, now, this is the scenario now this is basically so this 2 expression is equal under mod m so; that means, once we have 2 expressions equal under mod m.

Then we can say these 2 are. So, under the congruence relation.

(Refer Slide Time: 13:10)



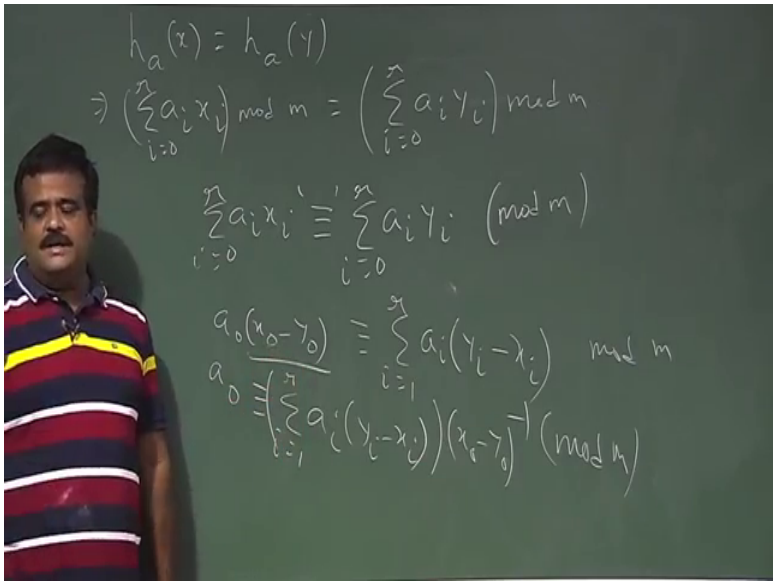
So, we can write this as or summation of a i xi, i is equal to 0 to r is congruence to summation of a i y i i is equal to 0 to r mod m. So, this congruence is a relation is a relation. So, how this is coming it is coming basically suppose this is a equivalent relation suppose we have a set of integer, this is a integer set now we take a m, now we just take a any integer and we divide we try to divide this by m. So, then what are the reminder? Reminder will be 0 to m minus 1 if you take a any integer if you divided by m then we are the reminder 0 to m minus 1.

So, that will be basically the equivalence by the cross section. So, this is 0 class, this is 1, class or dot dot this is m minus 1 class. Now, we say 2 integer say x y, we said 2 integer a b will be in the same class or they are related. So, a b a is congruence to b mod m, when you say this if a mod m is same as b mod m so; that means, if we divide a by m the reminder will same as if we divide b by m, suppose m is 7 set then 1, then 8, 1 and 8 they are congruence under 7 1 is congruence to 8 mod 7; mod 7 or 8 is congruence to 15 mod 7, because if we divide 8 by 7 the reminder is 1, if you divide 15 by 7 then reminder will be one so; that means, 8 and 15 will be in the same class this class.

So, here if any 7 so 8 and 15 will be seating here. So, this is the equivalent classes and this relation is equivalent relations and it will form a partition on this set and this is the equivalent classes. So, basically since they are under mod there equal; that means, there in same class so; that means, there are related by this relation this is called congruence

relation congruence module m. So, anyway this is basically giving us this relation now we want to take this this side.

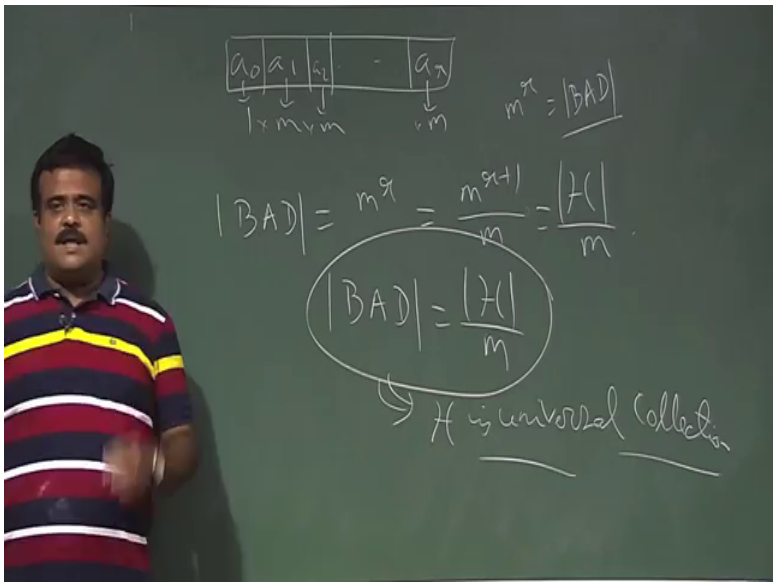
(Refer Slide Time: 16:04)



So, this basically so, we want to take a 0 x 0 minus a 0 y 0 1 side and then remaining are other side. So, summation of a so a i will take common a i, y i minus so x i will come this side, so this this from 1 to r because we have taken a 0 x 0 here so we can take common of a 0 x 0 minus y, so this is under mod m so; that means, what now we are assuming x and y are not are same. So, we are assume x 0 is not equal to y 0, if x 0 is not equal to y 0; that means, this is non 0 x 0 minus y 0 is not equal to 0 if x 0 minus y 0 is not equal to 0.

Then they have a inverse under mod m. So, basically a 0 is basically we equal to. So, summation of a I, y i minus x i and this i is from 1 to r multiply with x 0 minus y 0 inverse under mod m so; that means, we are not having choice of a 0 a 0 will be determine by this and this is possible because x 0 minus y 0 is not a is a non 0 quantity. So, it as a inverse we can multiply the inverse both sides so; a 0 is basically coming from this expressions. So, then, that means what? That means, we have a choice for this a i's other than a 0, if you just write this, basically the a i's. So, this is a 0 a 1 a r.

(Refer Slide Time: 17:56)

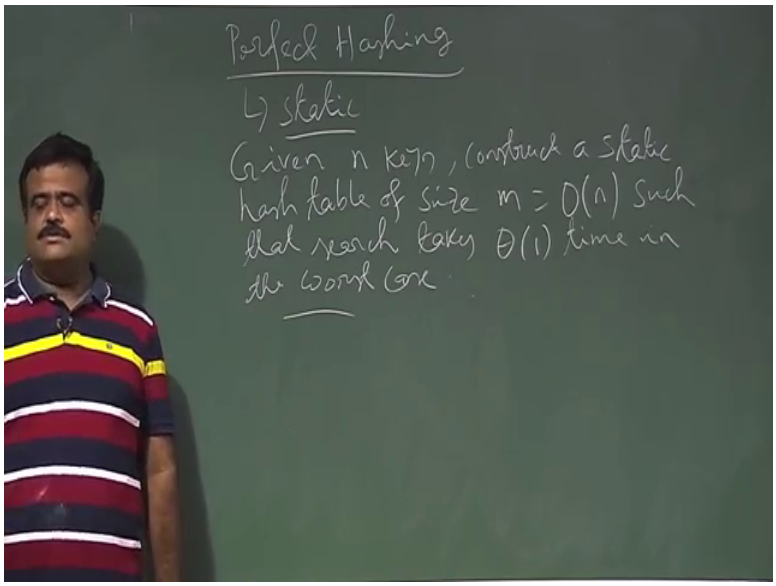


So, this can be chosen m ways this is a 2 this can be chosen m ways and this can be m ways all this, but we do not have choice for a 0 because is coming from this expression. So, a 0 can be chosen only one way because this is coming from this expression so; that means, what is the cardinality of bad portion is basically multiply of this, so this is basically m to the power r. So, this is the cardinality of the bad set. So, bad set means where there coming to be equal. So, this is basically m to the power r and m to the power r is nothing, but what? So, m to the power, so this is the cardinality of bad set m to the power r which can be written as m to the power r plus 1 by m and m to the power r plus 1 is the cardinality of h by m.

So, the cardinality of the bad set is basically 1 by m fraction of the total. So, this implies this collection is h is universal collection universal hash function, universal collection. So, this is a now h is universal collection now if you choose a hash function from this collection then that hash function will give us the that hash function is called universal hashing. So, this is one example of universal hashing. So, it is also it is basically the random choice. So, if we choose hash function random depending on the value of this vector a so; that means, nobody come with some key the input set where it is colliding because we do not know which way we are going to choose at the wrong time.

So, this is a random choice. So, this is one example of universal hashing.

(Refer Slide Time: 20:23)

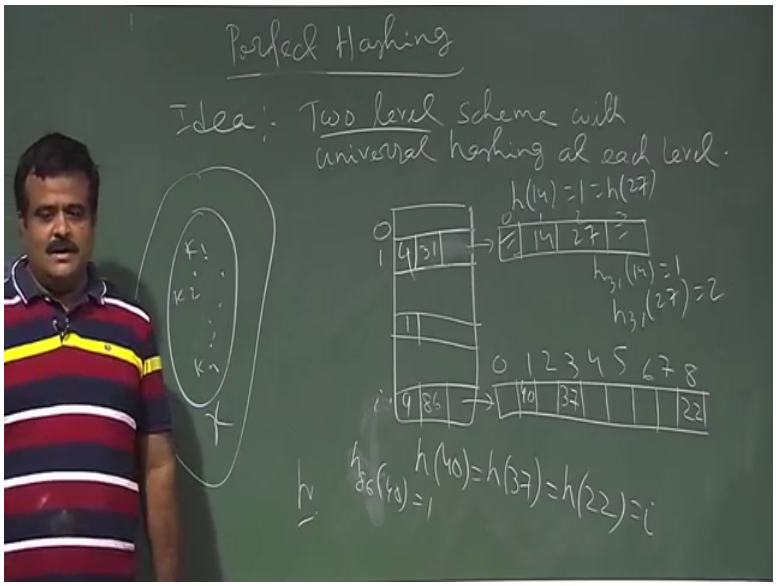


So, next we will talk about perfect hashing, perfect hashing so, it is basically a static arrangement. So, it is a static arrangement, so what we are doing here we are basically we have given, problem is we have given n keys and we need to construct a static hash table.

That means, we are not allowing anybody to join or anybody to delete from this table. So, the dynamicity we are not allowing here this set is static set, so given this we need to construct a static hash table of size m which is also order of n such that search will be taking in constant time, such that search takes theta 1 time in the worst case in the worst case. So, this is our problem, so, our problem is we are given n keys and we have given a, we need to construct a table and this set is static.

That means, we are not allowing anybody to joint in a set or anybody to delete from this set and this set is static. So, given way we need to construct a hash table and the table size is of order m such that the search will be faster ok.

(Refer Slide Time: 22:28)



So, this will be done, this will be done using the 2 level hashing 2 level hash table so, the idea is to, idea is to 2 level 2 levels scheme 2 level hash function with universal hashing at each level, with universal hashing at each level. So, we have basically 2 level hash function.

2 level hashing so, what is the idea? So, we have given we have a hash, function hash table 0 to m where m is of size m and we have given our key set where are basically k 1, k 2 up to k n we have n keys. So, what we have doing we are basically apply the first level hash function so, we have a hash function h which is first level hash function. So, we will apply this hash and this is also universal hash function this is also coming from universal hash hashing. So, every level so, we just apply the hash function and it will store it will mapped into this slots and there will be come collision, if there are collision suppose here some n i suppose so, we are n n keys we are mapping into this table.

So, there will be some collision suppose in this slot there are say 2 keys which are colliding, so, these 2 keys are say 14 and 27, 2 keys are colliding in this slot so, what we do we cannot fit it here. So, we will have a second level hash table, but in the second level hash table what we are doing. So, since 2 keys are colliding in this slot we are having 4 slot for the second level. So, second level table size is 4 for 2 keys so, if it is 3 keys it will be 9 if it is n i key it will be a ni square in the second level. So, second level so, this is 14 and 27. So, this is the second level so, we have to apply the second level

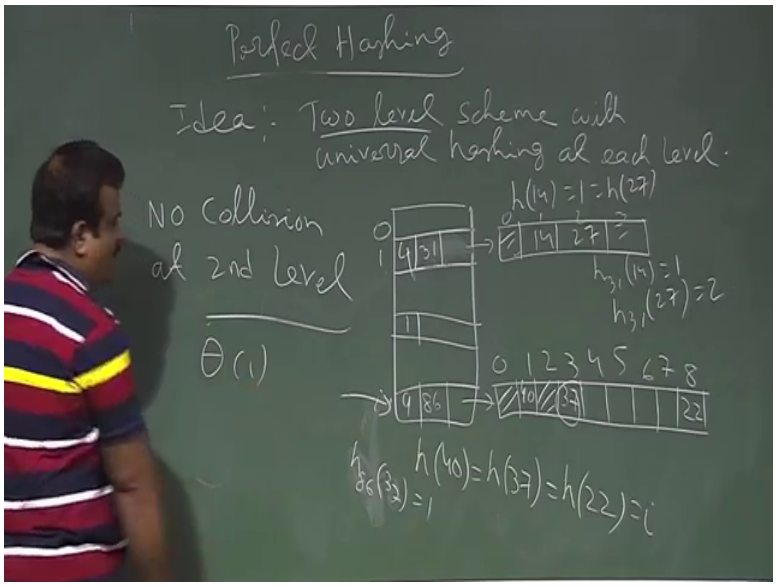
hash function. So, for that again this is coming from universal hashing. So, we have to choose a, a to have this hash function.

So, this is a, a means that vector h of a is our hash function. So, this is the first level hashing and this is the second level hashing so; that means, what? So, this is say 1 so, h of h is the first level hashing h of 14 equal to 1 is equal to h of 27. So, under first level hash function they are mapping to slot 1 and since they are colliding and there are only 2 keys are colliding. So, we need to have a second level hash table and i say in the second level will have the table size square of that so, 2 key; that means, 4 table size so; that means, and for that we need to have a hash function that hash function again we are choosing from universal hashing for that we need to have a so; that means, h of 31, 14 equal to so, this is 0 1 2 3 is equal to 1 h of 31 27 equal to 2 so this is the way.

So, similarly if there are only 1 is here so then we do not need to have any second level, but there are say 3 keys are colliding here so second level. So, this will indicate second level how many what is the table size and this is the, this is the second level hash table and the second level since 3 keys are colliding in the second level we have we need to have 9 is the table size. So, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 so 9 means up to 0 to 8 so, this is the second level and suppose these are the keys are colliding. So, say 40 say 37 and say 22. So, we know in the first level these are colliding so; that means, under h of 40 h of 30 there colliding into the same slot and this is a ith slot. So, there are 3 keys colliding in the i th slot in the first level hashing.

So, for the second level we need to have 3 keys of 3 square so 3 square means 9, 9 slot we need to have and for the second level we need to use a different hash function so that hash function we have getting from h of a. So, a is 86 8 of h of. So, basically h of 86 of 40 is 1. So, h of 86 of this is this, so this is the way so this is the 2 level hashing. So, now, we have to, so since we are taking a in the second level we do not want any collision. So, the second level hashing should be collision free and to guaranty that we have taking the size of the table is square of that.

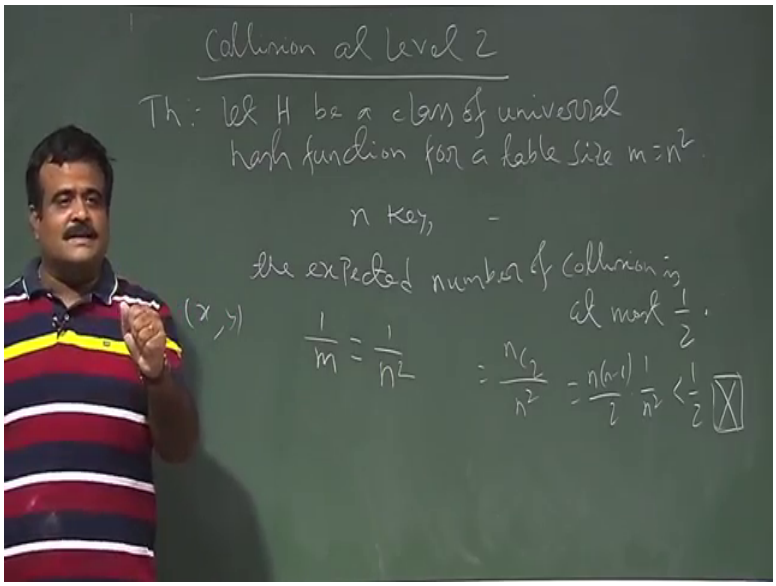
(Refer Slide Time: 28:04)



So, no collision at second level so, this we have to ensure and for that reason we have taking the table size for the second level is double. So, we have to analysis that, so before that how to search how the search is in constant time suppose we want to search 37, so what we do? We apply the first level hashing so it will map to here, so we know the there is the second level hashing and we need to apply the second level hash function. So, where from we can get the sec second level hash function that information has to store here. So, if a is given we know the hash function, so this a is given in this table, so we know h of a, so we know the hash function. So, we apply h of that on 37 we will reach here and we got 37. So, searching is just 2 hash function. So, searching is theta of 1 time because just a 2 hash so theta 1 time is the search time.

So, now, searching is now let us have a quick analysis of this why this is, 2 types of analysis we need to do why there is no collision in the second level and we have to bother about storage also because we are using intuitively it is where we are maybe using lot of storage, but that also we need to analyse.

(Refer Slide Time: 29:33)



So, let us analyse the collision at level 2, so why it will ensure that there will be no collision is level 2. So, this is coming from this theorem let h with the be a class of because every level we are using the universal hashing universal hash function and in the second level the hash function table size, universal hash function for a table size.

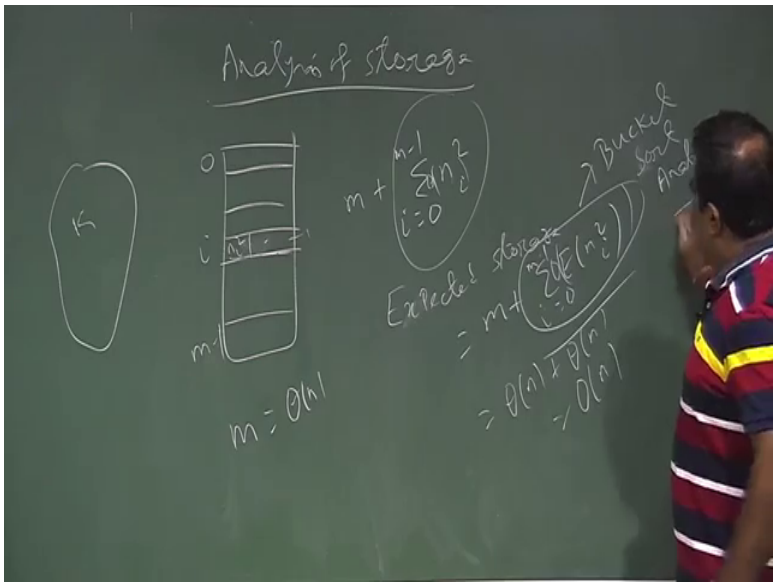
So, table size we are using that square so, if there are m keys and we are using if there are n i keys we are using n i square. So, suppose there are n keys in the second level and so the, our table is n square for the n keys. So, n is the keys and the table size is n r square. So, now, the theorem is telling the expected the expected number of collision number of collision is at most half; that means, we cannot expect even 1 collision also. So, how to prove this? So, prove this, so if we take 2 x y 2 keys so what is the probability that they will collide is basically 1 by m m is the table size it is basically 1 by n square because we are choosing the universal hashing now how many pairs are there.

So, expected, so there are n c 2 so, expected number of collision is basically n c 2 by n square. So, it is basically m into n minus by 2 into one by n square this is basically less than half, so this is the proof. So, even we cannot expect more than 1 collision this number of collision is at most half so; that means, because we have say we have 3 keys and we have 9 slots. So, it is quite obvious now the expected collision will there will be no collision because there are 9 slot and 3 keys has to be fitted and our hash function is

good hash function it will distribute the key uniformly over the slots. So, the chances of collision is less, now let us do a quick storage analysis.

So, this is the second level that is why second level there will be no collision because number of size of the slot we are taking n square.

(Refer Slide Time: 32:11)



So, analysis of storage, analysis of storage, so what is table size? So first level we had this is the first level we had m is order of m we have m tables now suppose we have n keys and suppose at the i th slot there are n I, n i keys are colliding in the slot in the first level. So, so in the second level table size is n r square now. So, what is the size in the second level table second level table size is n i square summation of n i square i is equal to m to n minus 1 and. So, what is the total size?

So, m plus this this is the total storage so this is the storage for the second level where n i is the number of keys colliding in the ith slot after the first level now if you take the expectation. So, expected storage is basically m plus summation of expected of n i square or it is order of order of n i square. So, this is order of n i square 0 to m minus 1. So, these expression we have seen for the bucket slot if you remember we have we have some keys and we have throwing in to the bucket and this we have prof there it is order of n. So, this order n this is also order of a n, so this is the this is coming from bucket slot analysis, bucket slot analysis. This storage is also order of n. So, we are not using really extra storage although its look like that n i is the number of keys colliding. So, in an

square it look like we are using new storage, but this is the analysis that we are not really using new storage now storage total storage is order of m.

Thank you.