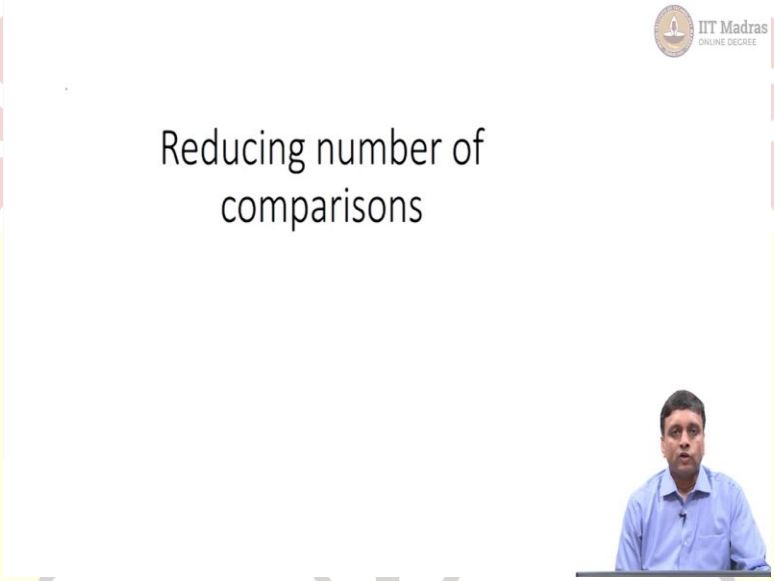
IIT Madras

ONLINE DEGREE

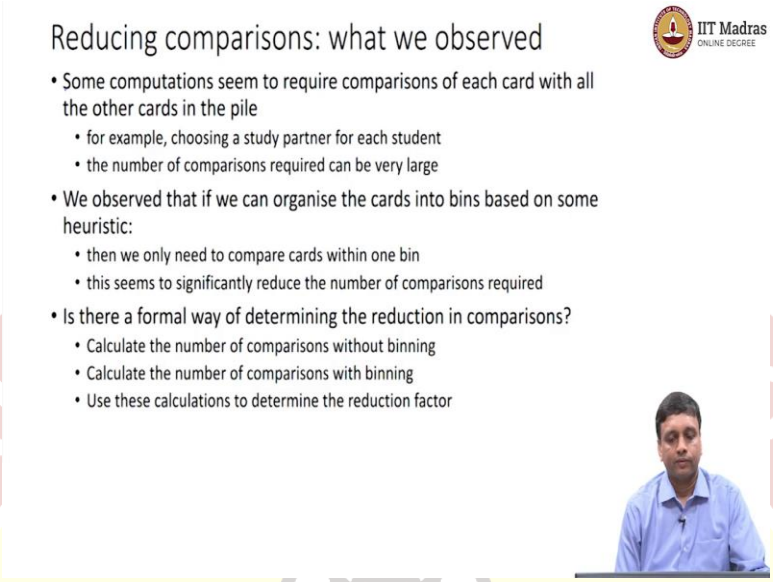
Computational Thinking
Professor Madhavan Mukund
Department of Computer Science
Chennai Mathematical Institute
Professor G Venkatesh
Indian Institute of Madras
Importance of binning to reduce number of comparisons in nested iterations

(Refer Slide Time: 00:15)



We saw some examples where we had to do what are called nested iterations. For instance, when we were trying to check whether two children in the have the same birthday, what we did was we took each child's card and look through all the other cards to see if the birth days match or not. So in this process we have to compare cards with each other and we gave some informal arguments about how we can reduce the number of comparisons. So let us try to formally work it out, so there is no confusion about the precise nature of what we have been doing.

(Refer Slide Time: 00:45)



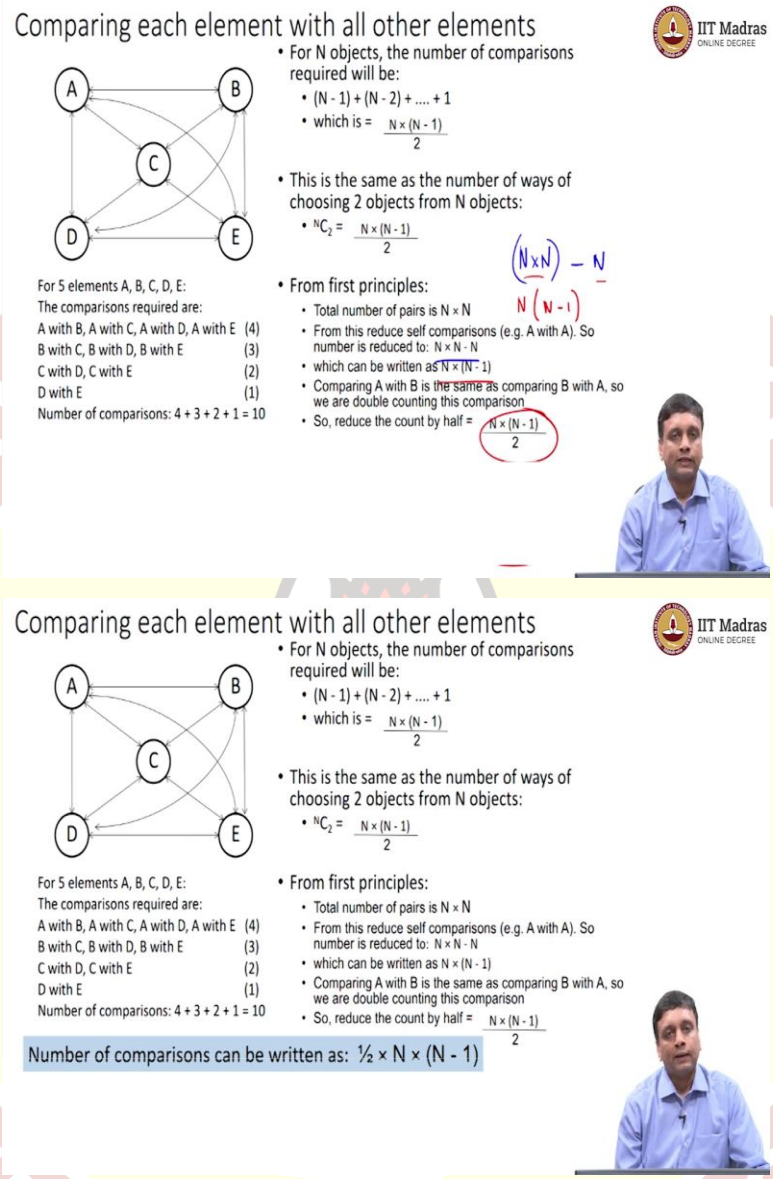
So what we observed is that when we are doing these processes like checking whether two children have the same birth day, we have to compare each card with every other card in the pile. So, this is also for checking study partners maths, you know marks we wanted to check if the marks were comparable and so on. And one thing we saw was, that if we do this blindly checking everything against everything, then the number of comparisons can be very large.

So then what we did was, we group them, for instance in the birthday case, we said that we can first group them by the month and then we know that if two children have the same birth day, they must be in the same month, so we only need to compare across the cards from the same month, so this bin of January will be different from the bin of February and so on.

So we never need to compare cards across bins. Similarly, when we did the batching up of students in pairs we looked for people who had similar totals and then within similar totals we group them into those who had higher physics marks and maths marks and higher maths marks and physics marks and then we merged across the two batches. So we binned and then within the bin again we created two bins and then we compared across these two bins.

So in all this we argued that this binning process reduces the number of comparisons. So what we are trying to do in this lecture is to formalize this notion, so how many comparisons would we do if we did not bin, how may comparisons would we do if we do bin and so by what factor are we reducing the work and why is this important?

(Refer Slide Time: 02:22)



So let us look at a simple case where we have 5 elements; A B C D E and we want to compare each of them with every other one. So let us start with A, so A has to be compared with all the others, so not itself. So there are 4 comparisons required, we have to compare A with B, A with C, A with D and A with E.

Having done this now we move to B. So when we look at B we compare it with C D and E, that is 3 comparisons but notice that we do not need to compare it with A because A has already been compared with B before, so when we come to B, we do not have to go back and do this comparison the second time.

Because comparing A with B is the same as comparing B with A. So exploiting this the number of comparisons left for B reduces from 4 to 3. In the way when we move on to C, we have to compare C with D and C with E, but we do not have to go back and compare C with either A or B because those two have already been done before. So for C we have only two comparisons and working forward to D, now D has already been compared to A B and C, so the only comparison which you are still to do with D is with E.

And we do not have to do anything new with E because E has been compared in each of the previous 4 steps with all its neighbors. So the total number of comparisons for this 5 elements is 4 plus 3 plus 2 plus 1 which is 10. So in general we saw that the first step with 5 elements gave us 4 comparisons, so we had N elements, the first step would give us N minus 1 comparisons, the first one with all the others.

The second step would give us N minus 2, because you have to compare the second one with all the others except the first one, so it is N minus 1 minus 1, so N minus 2. So third one will be N minus 3 and so on, so we add the sum N minus 1 plus N minus 2 plus up to 1 and you may have studied this in school but this is a well-known formula that the sum of any such sequence from 1 to N minus 1 is actually N into N minus 1 by 2.

There is a different way of thinking about it, which is that what we are doing is we are taking every pair of elements from this set and comparing them. So taking the pair of elements is exactly what happens when you do this commentarial thing called choosing a pair, so N choose 2. So we choose 2 objects from N objects, so we have this formula for N choose 2 which also works out because it is the same thing really as to N into N minus 1 by 2.

So these are two different things which you may have studied before, which will give you the same answer, which explains exactly what is happening when we compare every element in the set to every other element in the set. Now if you do not know this or if you are confused about remembering it, there is a very easy way to remember it from first principles.

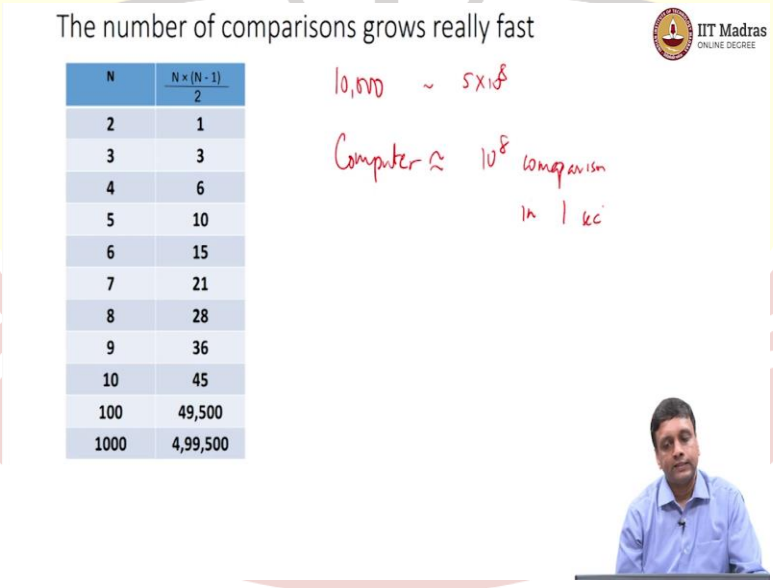
So the total number of pairs that you can do is N cross N, you can take everything in the first and compare it with everything in the second. Now of course some of these comparisons are useless because we do not want to compare A with A and B with B. So you remove for instance those N comparisons, so you take N cross N, this is our total number of comparisons and you remove the

N self-comparisons. Now if you write this as an expression, this is the same if you factor out 1 N, it is N into N minus 1 because N goes into this N times and N goes into these ones.

So N times N minus N is the same as N into N minus 1. And now we exploit the fact that when we do N into N minus 1 comparisons, we are comparing A with B but we are also comparing B with A. Because we have counted every comparison except the self-comparisons. So then it includes all this symmetric comparisons, where an element is compared whether it is the first part or the second part, so you remove the symmetric comparisons by keeping only one of each pair.

So you divide by 2 and you get N into N minus 1 by 2. So here are 3 different ways of thinking about why this answer should be N into N minus 1 by 2, so if you want you can just remember it but it is not very difficult to derive, either because you have learned this before in terms of the summation or in terms of N choose 2 or just by this simple operation of counting how many pairs you actually into count. So now what we want to do is explore what happens when we do a binning.

(Refer Slide Time: 06:42)



So why do we want to do this first of all? The first reason is that, this number actually grows very fast, so here is the small table telling us how N into N minus 1 by 2 grows as we go from 1 to 1000. So at 1000 we have already reached something like 5 lakhs. Now you can imagine that

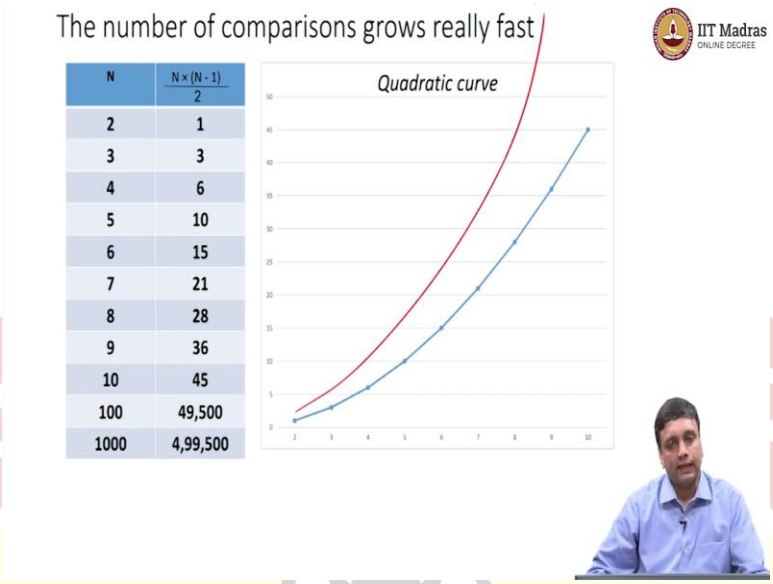
if you go to 10000, then you will actually go to something of the order of 5 into 10 to the power of 8 or maybe little more than that.

Now why is this the problem, because you might say we are not going to this by hand like we have been doing with the card, we are going to use a computer and after all a computer is very fast, so why does not it make any difference? Well, actually it turns out that roughly a computer can do something like 10 to the 8 comparisons in 1 second. So if I have to do 5 times 10 to the 8, it will already take 5 seconds. Now 5 seconds is a long time, let me just pause for a 5 seconds and you can see.

So now if one step in your program took that long, then each time you do, run your program, it will take a very long time and if you move this is just for 10000, now 10000 is a very small number, supposing you want to this for the number of say people in a midsize town, that would already be 1 lakh. If you want to do it for a big town, that will be in 10s of lakhs.

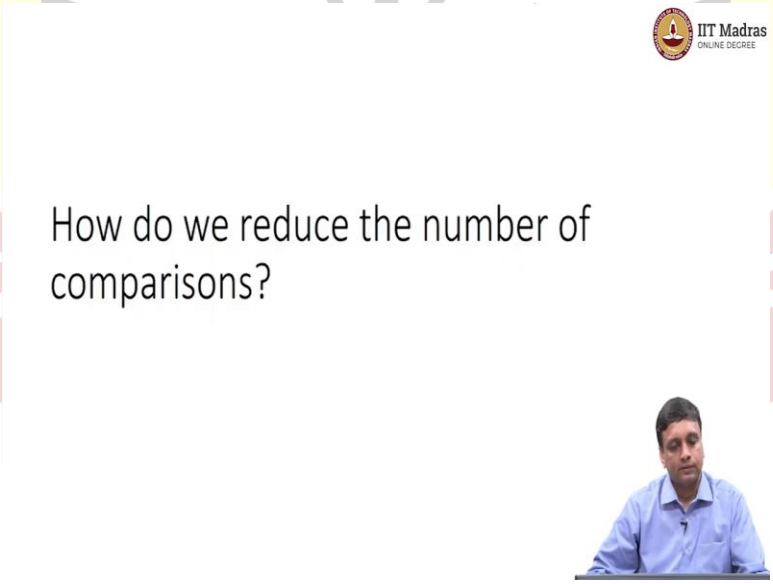
So if you want to do some comparisons which you wants to check for instance supposing you want to check whether the same phone number is registered with two different people in a city, then you will have to check across all the people in the city whether they have the same phone number which is essentially the same as checking whether across the entire city, two people of the same birth day and this is going to take you forever. So this is why we want to avoid this N squared.

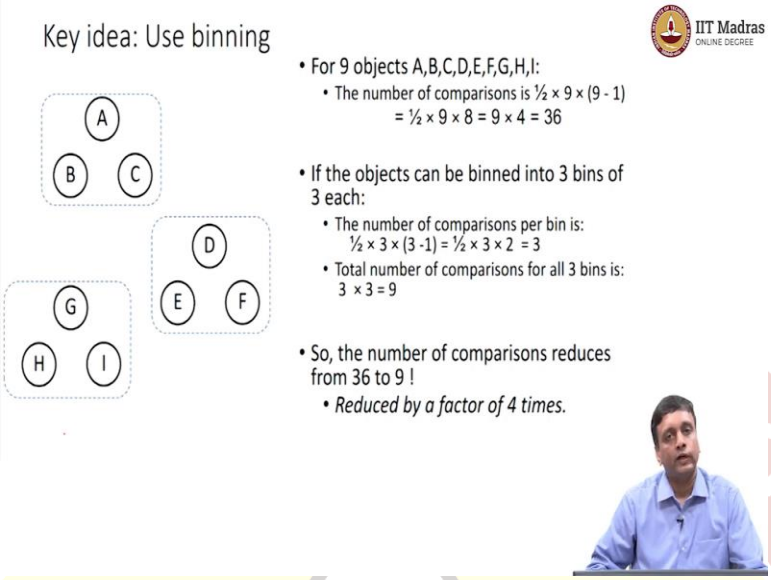
(Refer Slide Time: 08:38)



Another way of looking at N squared from a mathematical point of view is that this is growing as a quadratic, so this curve becomes steeper and steeper, it goes up very fast as N grows. So which is why we want to cut it down and which is why we do our binning to reduce the number of comparisons.

(Refer Slide Time: 08:53)



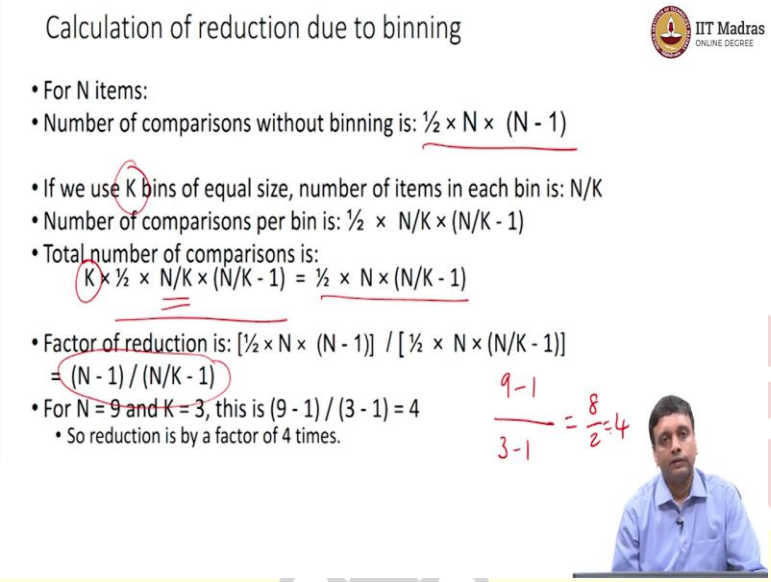


So how did we reduce it? Well, we grouped these elements into smaller groups such that we only have to compare within each group. So for instance, if we have 9 objects by our previous calculation, we need 9 times 9 minus 1 which is 8 divided by 2. So 72 divided by 2 is 36 comparisons, if we wanted to explicitly compare every one of these 9 elements with every other one without doing any duplicate counting.

Now suppose we realize that actually these 9 objects fit into 3 groups of 3 each and no comparison is needed across the groups, we do not need to compare A with anything other than B and C, we do not need to compare E with other than D and F and I only has to be compared with G and H. So we do these 3 bins and now in each bin we have to do comparisons between those elements in the bin, so we have 3 into 2 divided by 2, so 3 comparisons.

So we do 3 comparisons here, 3 comparisons here and 3 comparisons here, so totally we do 9 comparisons, and instead of 36 comparisons. So already even with a small number of elements that is 9 and small number of bins we see that we have a dramatic improvement in the number of (bins), in the number of comparisons we have to do after binning compare to doing it without binning.

(Refer Slide Time: 10:16)

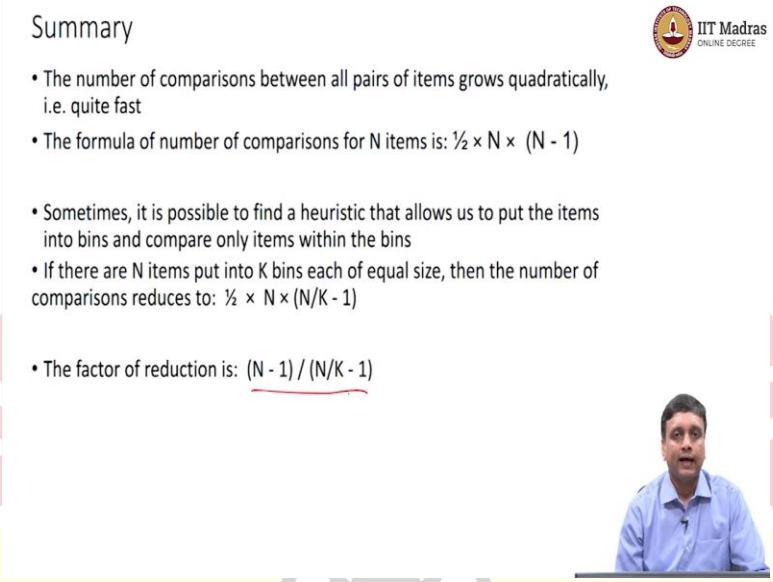


So if you want to do an explicit calculation, so we have this formula N into N minus 1 by 2 as a total number of comparisons if we just do a brute force calculation across every pair in the entire set. Now if we have K bins of equal size, then what happens is that each bin is of size N by K. So if we had 3 bins for 9 then we have 9 by 3, if we had say for example 16 elements and we (com) put it into 4 bins then each bin will have 4.

So each bin now becomes half into the size of that bin and the size of that bin minus 1, so half into 1 by K into N by K minus 1, that is just applying the same formula to each bin and how many bins are there, there are K bins. So if you now multiply this out, you get half into N into N by K minus 1. So this gives us the fact that we get a factor of reduction if you do this division of N minus 1 divided by N by K minus 1.

So for instance in our case N minus 1 is, N was 9, so N minus 1 is 8 and N by K was 3, so 3 minus 1 is 2, so we get 8 by 2 which is equal to 4, so we get a 4 4 reduction. So if we do this for a big class for example and we do this for birth day type of calculation, then we would have something which we will divide by something like a roughly a factor of 12. So it becomes a fairly important calculation reduction in terms of the number of comparisons we have to do.

(Refer Slide Time: 11:49)



So hopefully this will give you a clearer picture of what happens when we do binning, so when we do binning, we reduce the number of comparisons from this N into N minus 1 by 2 to a factor which is replaced by N by K, where K is a number of bins. So the more we can bin, the more bins we can put them in and reduce the comparison across bins, the better the speed up that we get in terms of the fewer comparisons that we have to do.

So the factor of reduction if explicitly is N minus 1 upon N by K minus 1 and remember that a quadratic computation blows up very fast, so if we have to do comparisons across say datasets which involve say large numbers of people or large numbers of objects, very often an N squared calculation does not work and we have to do binning. Now, given this one should remember that binning does not always work because you have to find the right bins.

Now if you have bins where everybody goes into the same bin for instance, if all the students in the class are born in the same month, then binning by month does not change your problem at all because you have to still compare everybody with everybody. But then maybe you should bin by some other quantity by date of birth, not the month of birth. You know that then there are 31 dates, so you will have 31 bins instead of 12 bins, so you will have a different way of binning.

So you have to look at your data and decide what is a reasonable strategy for binning and hope that the bins will give you a smaller quantities within which you have to do the comparisons and this will give you a dramatic speed up in the number of comparisons overall.