INDIAN INSTITUTE OF TECHNOLOGY MADRAS

सिद्धिर्भवति कर्मजा

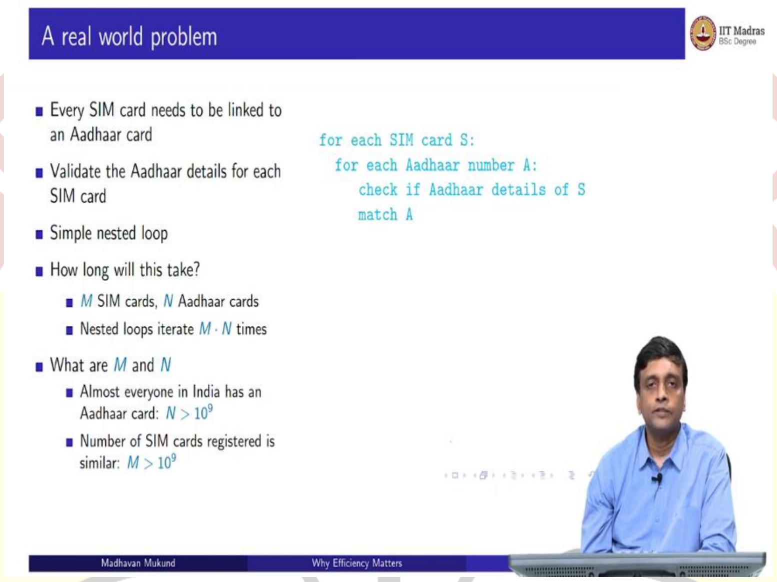
IIT Madras

ONLINE DEGREE

Programming, Data Structures and Algorithm using Python
Professor Madhavan Mukund
Why Efficiency Matters

So, to wrap up this introductory week of the course, let us look at an example to illustrate why we are interested in efficiency and why we are interested in both algorithms and data structures? What is the kind of benefit that we hope to get out of this?

(Refer Slide Time: 00:26)



A real world problem

- Every SIM card needs to be linked to an Aadhaar card
- Validate the Aadhaar details for each SIM card
- Simple nested loop
- How long will this take?
  - M SIM cards, N Aadhaar cards
  - Nested loops iterate $M \cdot N$ times
- What are M and N
  - Almost everyone in India has an Aadhaar card: $N > 10^9$
  - Number of SIM cards registered is similar: $M > 10^9$

```
for each SIM card S:
  for each Aadhaar number A:
     check if Aadhaar details of S
     match A
```

Madhavan Mukund Why Efficiency Matters

So, let us look at a kind of real-world problem that you might imagine. So, as you probably know, when you go to get a SIM card, you have to give your Aadhaar details. So, every SIM card needs to be linked to an Aadhaar card. Now, if you have an unscrupulous dealer, handing out SIM cards, it could well be that they are putting fake Aadhaar details just to sell you the card.

So, the government would like to check that the Aadhaar details on each SIM card actually match with what the Aadhaar number says. So, A the Aadhaar number must exist and whatever details are there about the person who holds the SIM card must match what the Aadhaar number says about that same person. So, this is the problem, every SIM card needs to be linked to a valid Aadhaar card. And we want to validate that the Aadhaar numbers which have been, so let us assume that every SIM card has been actually provided with some Aadhaar details, we want to now check whether the Aadhaar details in all these SIM cards are actually correct or not.

So, this is something that we saw a long time ago in computational thinking. This is basically a simple nested loop, for every SIM card, we need to check whether its Aadhaar details are valid or not. So, the only way sensible way to do this is to go through all the Aadhaar cards and look for a match. So, we take each SIM card. And then for that SIM card, we run through all the Aadhaar cards till we find a match.

And we find a match saying that this Aadhaar number is Aadhaar number which is mentioned in this card, then we will check that the details match. Of course, if there is no valid Aadhaar number for the card, that is also a mismatch. So, at the end of this, either this card will match or it will not match.

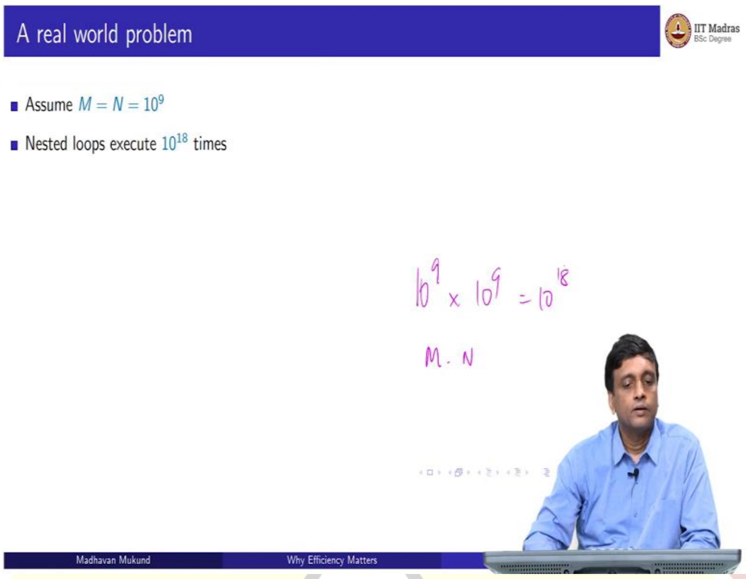
So, I can separately now take this and create a list of all the SIM cards which are problematic. But fundamentally, I have to run this nested loop, which compares every SIM card against every Aadhaar card. So, the question is, how long will this take? So, supposing there are M SIM cards and N Aadhaar cards, then it is easy to see that the outer loop will run M times. And each time the outer loop runs, the inner loop will run in general N times if I go through all the Aadhaar cards.

So, let us assume that all the SIM cards have invalid Aadhaar numbers, then I will have to scan through all the Aadhaar cards to discover this. So, in the worst case, I will end up with M times N. So, this is a kind of complexity of this naive solution. So, what in practice are M and N? This is why we are looking at this example to get an idea about what happens in the real world when we deal with data. Because data is what we are going to deal with a lot in machine learning. And these kinds of scales are typically what we have to deal with not the toy examples that we construct by hand.

So, here we have Aadhaar cards and SIM cards. So, how many Aadhaar cards and how many SIM cards are there in India. So, by now almost everybody in India has Aadhaar card. And as you know the number of people in India is more than 1 billion, which is 100 crores, which is in scientific notation 10 to the power 9. So, at the least there are 10 to the power 9 Aadhaar cards, it might surprise you to know that there are as many SIM cards. It is not true, of course that everybody has a SIM card.

Babies do not have SIM cards, small children may have Aadhaar numbers, but they will not have SIM cards. But many people have more than one SIM card. So, it all kind of balances out. And if you actually count the number of SIM cards, which are there in the country, it is roughly equal to the number of people in the country. So, both M and N are actually of this order 1 billion, 10 to the power 9. So, that is the problem that we are dealing.

(Refer Slide Time: 04:10)



A real world problem

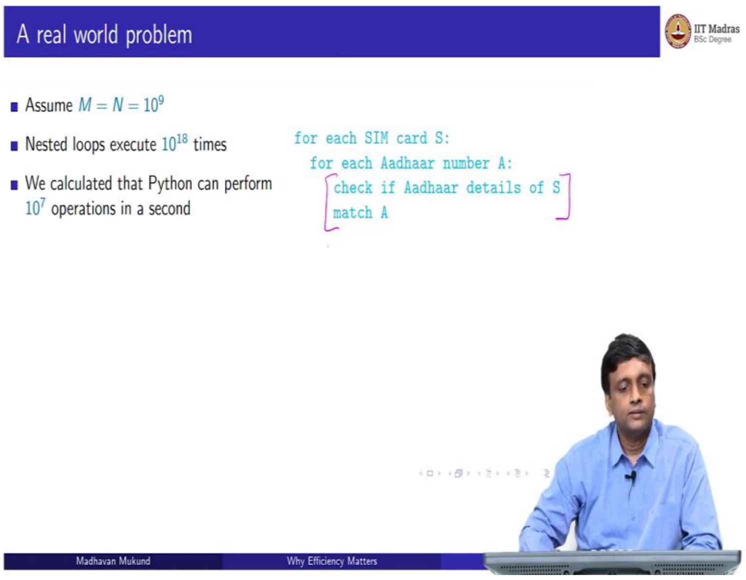
- Assume $M = N = 10^9$
- Nested loops execute 10^{18} times

$10^9 \times 10^9 = 10^{18}$

$M \cdot N$

So, let us assume that, for simplicity, that they are actually equal to, they are actually bigger than this, but let us assume they are just exactly 1 billion. So, there are exactly 1 billion Aadhaar cards exactly 1 billion SIM cards. So, now let us do the math. So, the nested loop, if you remember, runs M times N times, so 10 to the N times 10 to the 9 multiplied by 10 to the 9, this is M times N, and this gives us 10 to the 18. So, I have this naive search through a nested loop is going to execute 10 to the 18 times.

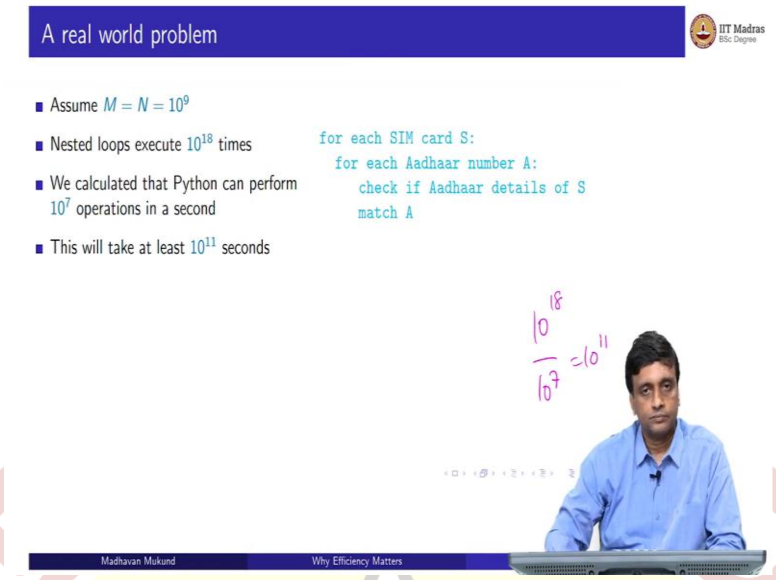
(Refer Slide Time: 04:43)



A real world problem

- Assume $M = N = 10^9$
- Nested loops execute 10^{18} times
- We calculated that Python can perform 10^7 operations in a second

```
for each SIM card S:
  for each Aadhaar number A:
    check if Aadhaar details of S
    match A
```



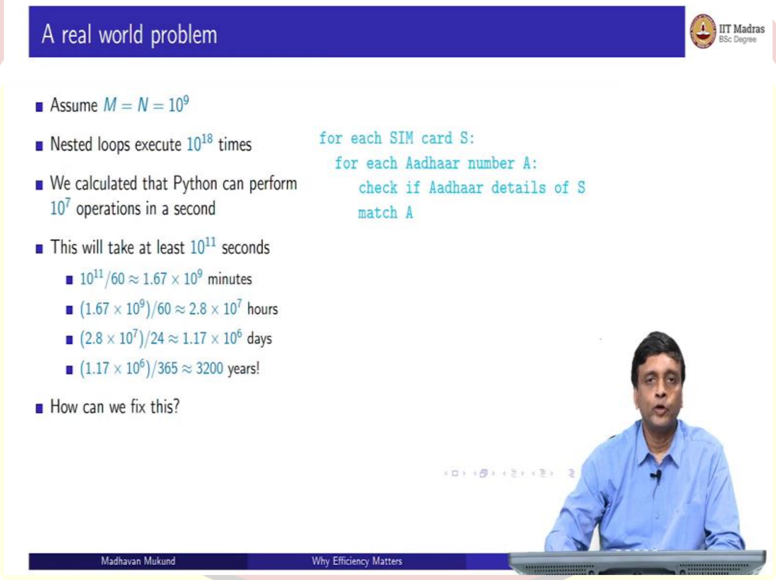
A real world problem

- Assume $M = N = 10^9$
- Nested loops execute 10^{18} times
- We calculated that Python can perform 10^7 operations in a second
- This will take at least 10^{11} seconds

```
for each SIM card S:
  for each Aadhaar number A:
     check if Aadhaar details of S
     match A
```

$\frac{10^{18}}{10^7} = 10^{11}$

Madhavan Mukund Why Efficiency Matters



A real world problem

- Assume $M = N = 10^9$
- Nested loops execute 10^{18} times
- We calculated that Python can perform 10^7 operations in a second
- This will take at least 10^{11} seconds
  - $10^{11}/60 \approx 1.67 \times 10^9$ minutes
  - $(1.67 \times 10^9)/60 \approx 2.8 \times 10^7$ hours
  - $(2.8 \times 10^7)/24 \approx 1.17 \times 10^6$ days
  - $(1.17 \times 10^6)/365 \approx 3200$ years!
- How can we fix this?

```
for each SIM card S:
  for each Aadhaar number A:
     check if Aadhaar details of S
     match A
```

Madhavan Mukund Why Efficiency Matters

So, we have already calculated that using the timer and all that, that Python actually can run about 10 to the 7 operations in a second. So, even if we assume that every time, we run this loop all this checking happens in some unit operations, this is a very simple thing that we have to do here. Even if there is just a basic step here. We have to do this step 10 to the 18 times, and each of this thing is going to take one operation.

So, 10 to the 7 operations per second means that I am going to use 10 to the 11 seconds. This is just 10 to the 18, divided by the number I can do in 1 second, which is 10 to the 11. So, 10 to the 11 seconds, is my estimate for how long, it is a very conservative estimate for how long this code is going to take to run. So, what is 10 to the 11 seconds. So, 60 seconds, make a minute. So, if I divide 10 to the 11 by 60, I get something like this, I get something like 1.67 times 10 to the 9 minutes. So, this is still a little abstract to understand.

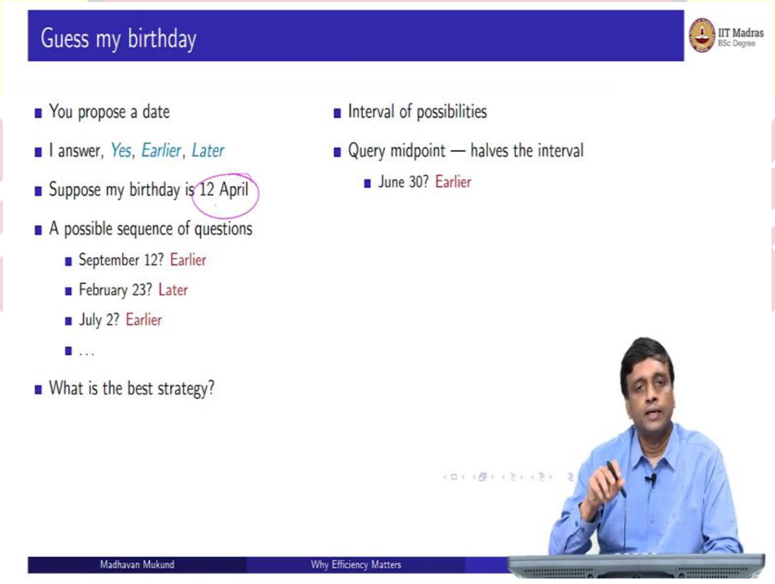
So, 60 minutes makes an hour, so if I divided 60 again and I get something like 2.8 into 10 to the 7, so 2.8 into 10 to the 7 is 28 into 10 to the 6, so 10 to the 6 is 1 million. So, this is 28 million hours. Still not very clear. So, let me divide by 24 to get days. So, if I divide 2.8 times 10 to the 7 by 24, I get the total number of days, so I get more than a million days.

So, what is a million days? Well, let us take a year to be 365 days. So, if I divide this by 365, I get 3200 years. Now, this is something we can understand. So, 3200 years is a long time, it is probably going to be much beyond the lifetime of the planet as we are currently running it. And we certainly cannot wait so long to validate whether every SIM card has a valid Aadhaar number.

So, this is just to illustrate that, if you just take a problem at face value and address it at the way it is presented to you without doing any thinking, you might well end up with a problem that is unsolvable, simply because it is physically not possible to solve that problem given the resources and constraints that you have. So, we need a different way to do this. So, how do we solve this problem?

So, clearly, this cannot be the only way to solve this. Otherwise, there is no point in associating Aadhaar numbers with SIM cards because nobody is ever going to be able to check that they are valid or not.

(Refer Slide Time: 07:19)

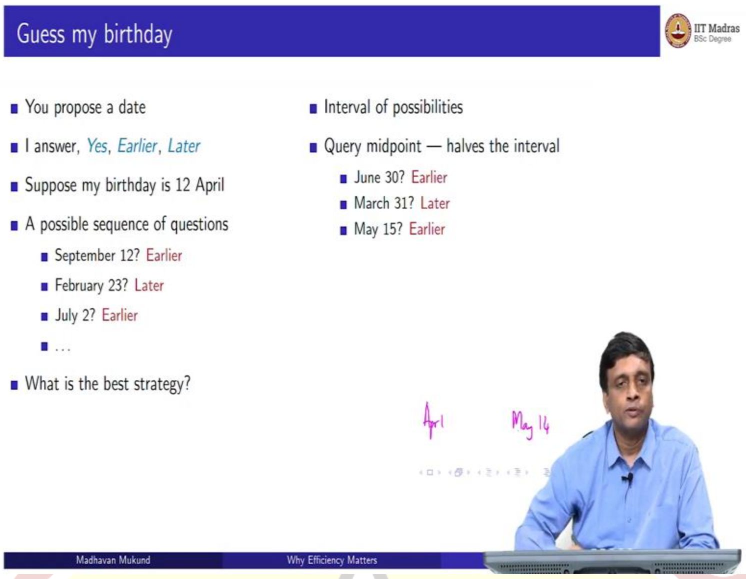


Guess my birthday

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*
- Suppose my birthday is 12 April
- A possible sequence of questions
  - September 12? Earlier
  - February 23? Later
  - July 2? Earlier
  - ...
- What is the best strategy?

- Interval of possibilities
- Query midpoint — halves the interval
  - June 30? Earlier

Madhavan Mukund   Why Efficiency Matters



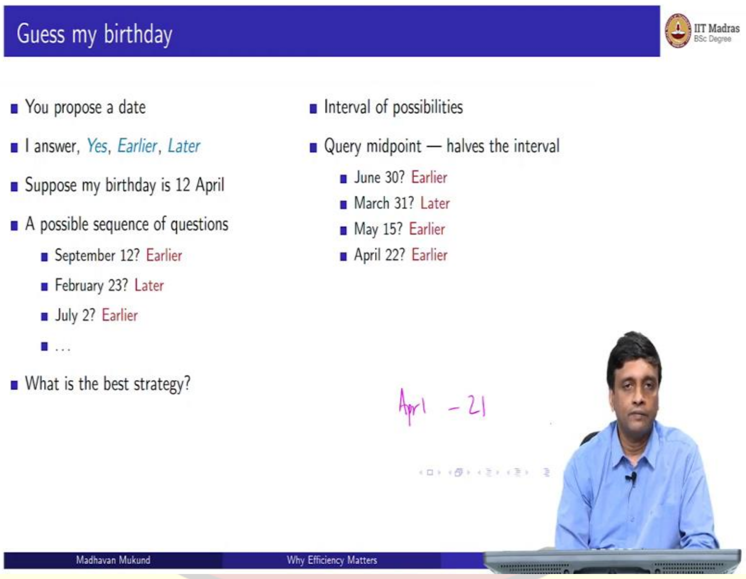
Guess my birthday

IIT Madras BSc Degree

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*
- Suppose my birthday is 12 April
- A possible sequence of questions
  - September 12? Earlier
  - February 23? Later
  - July 2? Earlier
  - ...
- What is the best strategy?
- Interval of possibilities
- Query midpoint — halves the interval
  - June 30? Earlier
  - March 31? Later
  - May 15? Earlier

Apr1 May 14

Madhavan Mukund Why Efficiency Matters



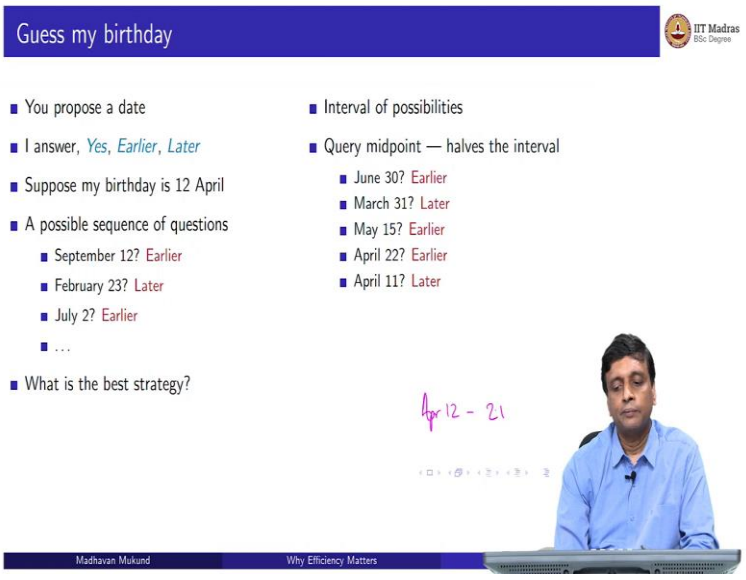
Guess my birthday

IIT Madras BSc Degree

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*
- Suppose my birthday is 12 April
- A possible sequence of questions
  - September 12? Earlier
  - February 23? Later
  - July 2? Earlier
  - ...
- What is the best strategy?
- Interval of possibilities
- Query midpoint — halves the interval
  - June 30? Earlier
  - March 31? Later
  - May 15? Earlier
  - April 22? Earlier

Apr1 – 21

Madhavan Mukund Why Efficiency Matters



Guess my birthday

IIT Madras BSc Degree

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*
- Suppose my birthday is 12 April
- A possible sequence of questions
  - September 12? Earlier
  - February 23? Later
  - July 2? Earlier
  - ...
- What is the best strategy?
- Interval of possibilities
- Query midpoint — halves the interval
  - June 30? Earlier
  - March 31? Later
  - May 15? Earlier
  - April 22? Earlier
  - April 11? Later

Apr 12 – 21

Madhavan Mukund Why Efficiency Matters

So, let us switch to a different problem. So, supposing I want you to guess my birthday. And the way it works is that you are allowed to make a guess. And I will either say yes, you got it. Or I will tell you no, it is wrong, because my birthday is earlier than the day you presume. And or it is later. So, this is just a date.

So, I am just asking for a date like September 7 or October 24th. I am not asking you to guess the year at this point, just the date within the year. So, there are 365 or if you count leap years 366 possibilities, I want you to figure out which one of them is my birthday. So, let us suppose that my birthday is 12th of April. So, you might ask questions for instance, you might ask, is it the 12th of September?

And I would say no, it is not my birthday is earlier than that, then you would logically ask a date. I mean, after I tell you it is before September 12. Presumably, you will not ask me a date like November 15. Because you know, it is before September 12. So, you might ask me something at the beginning of the year. So, you say is it February 23rd? I say no, it is not February 23rd in fact it is later.

So, now what have you done? You know, it is between February 24th. Because it is not February 23rd. And it is latest between February 24th and September 11th. So, you will have something in that range. So, maybe you might ask July 2nd. Now, I will say no, it is not July 2 it is earlier. So, now you know it is between February 24 and July 1st. So, our question is, there a systematic way to ask these questions? What is the best sequence of questions to ask in order to guess my birthday? This is the rule that I can say yes. Or I can say it is before or I can say is after.

So, as we saw, what we are doing is we are actually taking an interval of possibilities and narrowing it down. So, initially, I have all 365 days after I asked one question, the interval narrows. So, the best way to narrow it is to take a question in the midpoint. So, if I take the midpoint, then the next time I am going to look at half the interval. Now see, September 12. If it had been after September 12, I would have got lucky because I would have narrowed it down to the last three and a half months of the year.

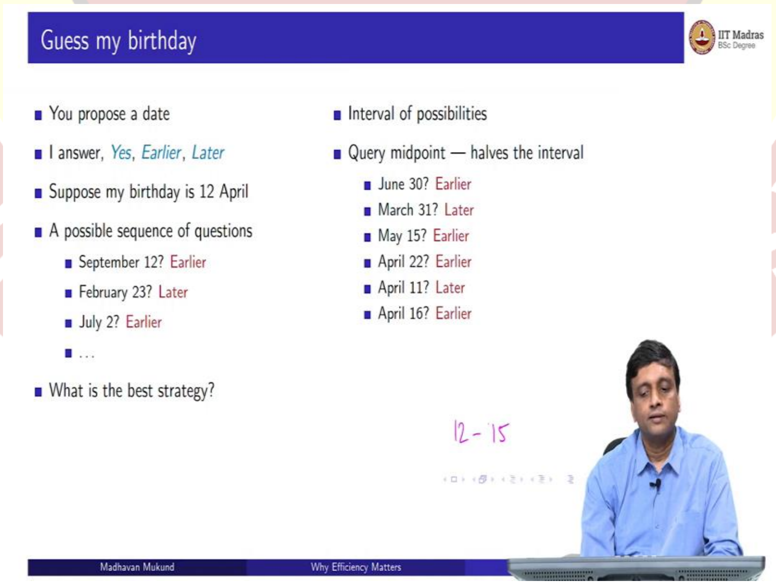
But because it was not after but before and still left with 8 and a half months, which is not great. So, instead of having this imbalance with 8 and a half and 3 and a half, it will be better to have something in the midpoint so that whichever way the answer goes, I am guaranteed that I am making an equal amount of progress. The midpoint is best.

So, if I take the middle month, end of June or beginning of July, so let us say I take June 30th. So, the answer is still the same. I am still asking questions about 12th of April. So, if you ask me June 30th. No, I will say it is earlier. Now, you have range from January 1st to June 29th. So, January, February, March, April, May, June. So, at the 3 months point so, you will ask say March 31st.

So, now that you say March 31st, I will say it later. Now, you have narrowed it down to from 6 months to 3 months. So, it is now in April, May or June between April 1st and June 29th. So, now you take the midpoint in that the middle month is May. So, you take the midpoint say 15th of May. So, you will say is it May 15th? And I will say no, it is earlier. So, now my range is between April 1 and May 14.

So, this is like 45 days roughly. I have cut it down from 12 months to 6 months to 3 months to one and a half months 45 days, so 45 days, so, about 22 days is the midpoint. So, I may task 22 days from April 1st, so, I may say April 22nd. Now, if you ask April 22nd I say it is earlier, so, now, my range has come to April 1 to 21 because it is before April 22 and it is after March 31. So, this is a 21-day range, so I might ask 10 to 11 days into this range. So, I maybe I will ask April 11th. So, now, it says later, so now it is April 12 to 21 this is my range.

(Refer Slide Time: 11:25)



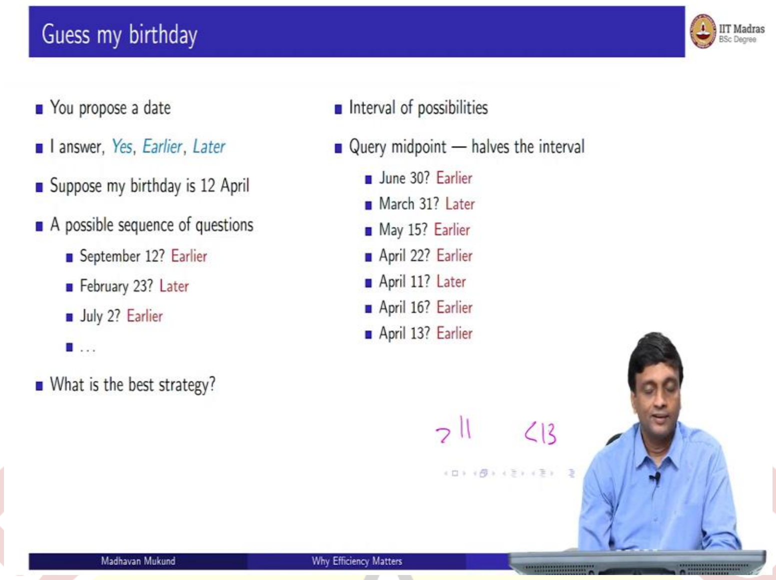
Guess my birthday

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*
- Suppose my birthday is 12 April
- A possible sequence of questions
  - September 12? Earlier
  - February 23? Later
  - July 2? Earlier
  - ...
- What is the best strategy?

- Interval of possibilities
- Query midpoint — halves the interval
  - June 30? Earlier
  - March 31? Later
  - May 15? Earlier
  - April 22? Earlier
  - April 11? Later
  - April 16? Earlier

12 - 15

Madhavan Mukund Why Efficiency Matters

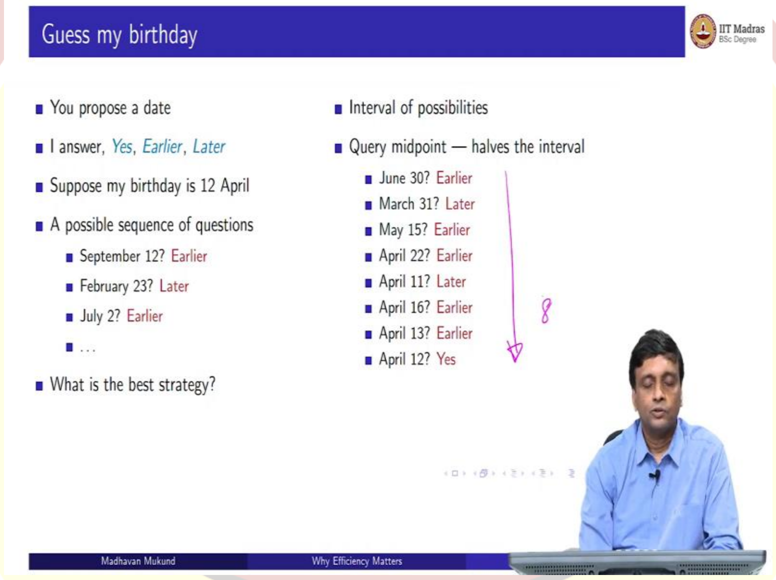


Guess my birthday

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*
- Suppose my birthday is 12 April
- A possible sequence of questions
  - September 12? Earlier
  - February 23? Later
  - July 2? Earlier
  - . . .
- What is the best strategy?
- Interval of possibilities
- Query midpoint — halves the interval
  - June 30? Earlier
  - March 31? Later
  - May 15? Earlier
  - April 22? Earlier
  - April 11? Later
  - April 16? Earlier
  - April 13? Earlier

$>11 \quad <13$

Madhavan Mukund Why Efficiency Matters



Guess my birthday

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*
- Suppose my birthday is 12 April
- A possible sequence of questions
  - September 12? Earlier
  - February 23? Later
  - July 2? Earlier
  - . . .
- What is the best strategy?
- Interval of possibilities
- Query midpoint — halves the interval
  - June 30? Earlier
  - March 31? Later
  - May 15? Earlier
  - April 22? Earlier
  - April 11? Later
  - April 16? Earlier
  - April 13? Earlier
  - April 12? Yes

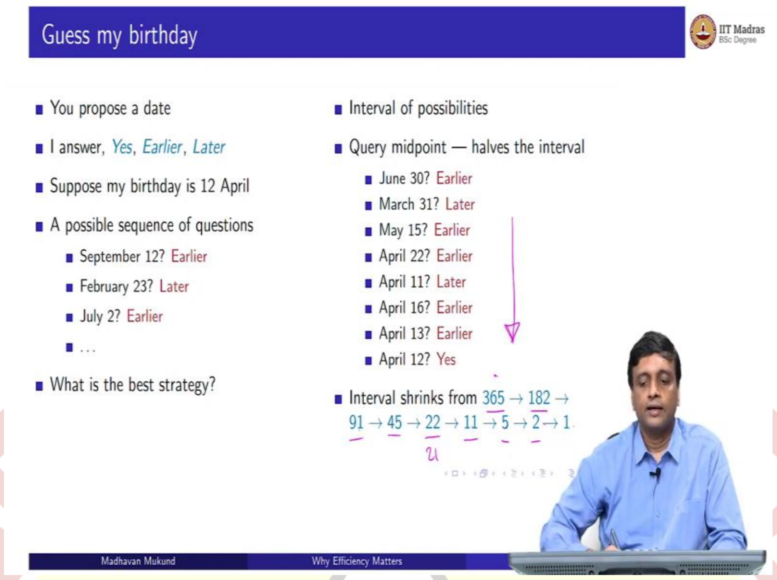
8

Madhavan Mukund Why Efficiency Matters

So, I take the midpoint of this. So, I may say 16th, it is the 16th or 17th I mean the midpoint sometimes could be one of two values, so I ask 16th, is earlier, so, now it is between 12 and 15th, it is definitely after 11th before 16th. So, 12, 13, 14, 15. So, I may be ask 13. And this is earlier, so now I know it is after 11.

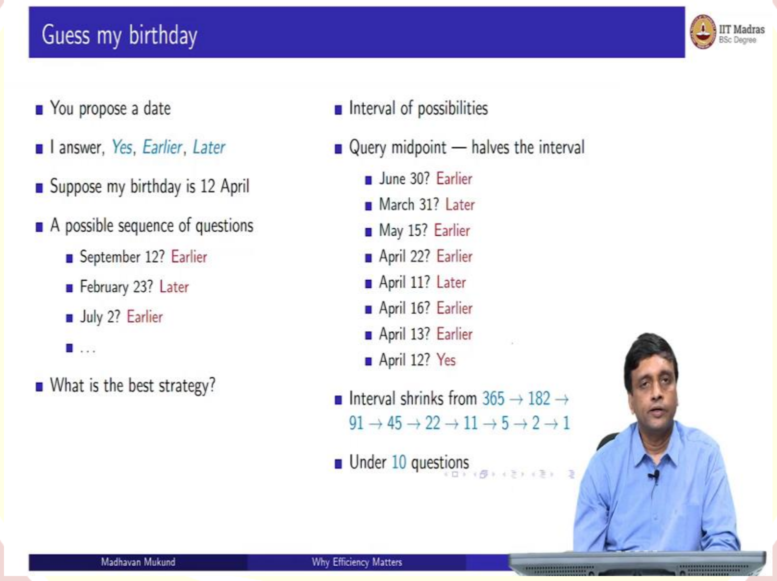
And it is less than 13. So, I have no choice. I mean, unless I have been lying, and I am not born at all this year, I have narrowed it down to an interval of 1. So, it must be April 12th. So, I have come down this list and answered your question. And in this particular case in 8 questions.

(Refer Slide Time: 12:00)


Guess my birthday

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*
- Suppose my birthday is 12 April
- A possible sequence of questions
  - September 12? Earlier
  - February 23? Later
  - July 2? Earlier
  - ...
- What is the best strategy?
- Interval of possibilities
- Query midpoint — halves the interval
  - June 30? Earlier
  - March 31? Later
  - May 15? Earlier
  - April 22? Earlier
  - April 11? Later
  - April 16? Earlier
  - April 13? Earlier
  - April 12? Yes
- Interval shrinks from $365 \to 182 \to 91 \to 45 \to 22 \to 11 \to 5 \to 2 \to 1$

Madhavan Mukund Why Efficiency Matters


Guess my birthday

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*
- Suppose my birthday is 12 April
- A possible sequence of questions
  - September 12? Earlier
  - February 23? Later
  - July 2? Earlier
  - ...
- What is the best strategy?
- Interval of possibilities
- Query midpoint — halves the interval
  - June 30? Earlier
  - March 31? Later
  - May 15? Earlier
  - April 22? Earlier
  - April 11? Later
  - April 16? Earlier
  - April 13? Earlier
  - April 12? Yes
- Interval shrinks from $365 \to 182 \to 91 \to 45 \to 22 \to 11 \to 5 \to 2 \to 1$
- Under 10 questions

Madhavan Mukund Why Efficiency Matters

So, how does this work in general? Well, the interval shrinks, so initially, we are looking at 365 days, then we look at half of that, because one number gets knocked off. So, 364 divided by 2, so 182 days, then 182, one number gets knocked off, I have 90 or 91. So, in the worst case, I will have 91, and so on. So, 91 becomes 45, 45 becomes 22.

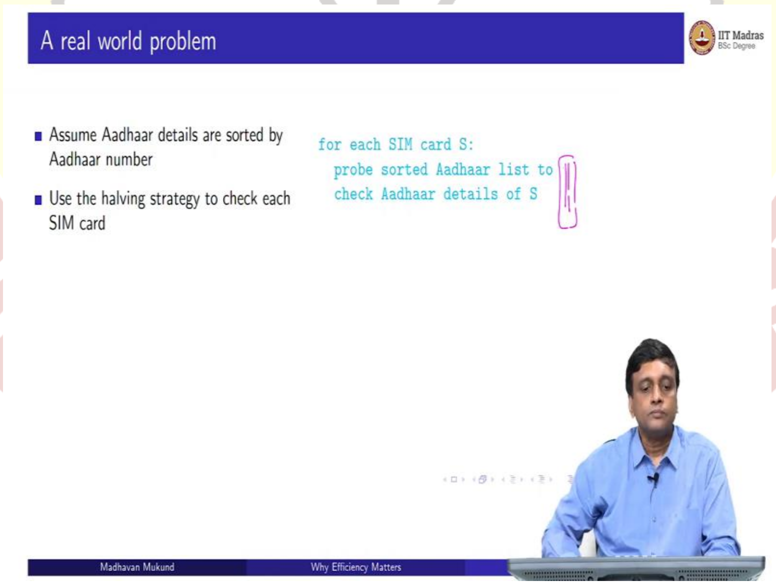
So, each time I am subtracting 1 and halving and getting roughly that number, if it is not even I am taking the bigger of the 2, so 22 becomes 21, because it is not that day. So, I have 10 on one side, 11 on the other side. So, 11 is the bigger of the 2. So, the worst case I might get 11, 11 one will go so I will get 5 and 5.

Similarly, 5 will go so, I will get 2 and 2. And then when I ask an interval of 2, then I will basically be asking the midpoint and either I will get it right or it will be bigger or smaller than the number I ask. So, I will finally end up with 1, so you can count 1, 2, 3, 4, 5, 6, 7, 8, 9 in general, here, I asked 8 questions and got to this interval. So, certainly it is under 10 questions.

So, this is the guessing the birthday thing. So, what is the property that we are using, we are using the property that I can say earlier or later. So, the birth dates are arranged in a sequence. And I am asking you about a particular date. And I know whether to go backwards or forwards. So, this order is important. I have got a sequence of dates. And I am asking you and you can use the same principle to look for, say, you know, words in the dictionary or whatever we are quite used to this.

So, when you have a sorted sequence, searching for something and going backwards and forwards, you would not find a page number in a book. It is very easy, supposing somebody says open this book to page 784, you will first open it at random, and you will see whether the page you have opened is before 784 or after 784. And then you will go forwards or backwards and you will quickly arrive at the page that you need, you will not flip through the book one page at a time to reach 784. Now, how do we apply this to our earlier problem of SIM cards and Aadhaar cards?

(Refer Slide Time: 13:53)



A real world problem

- Assume Aadhaar details are sorted by Aadhaar number
- Use the halving strategy to check each SIM card

```
for each SIM card S:
  probe sorted Aadhaar list to
  check Aadhaar details of S
```

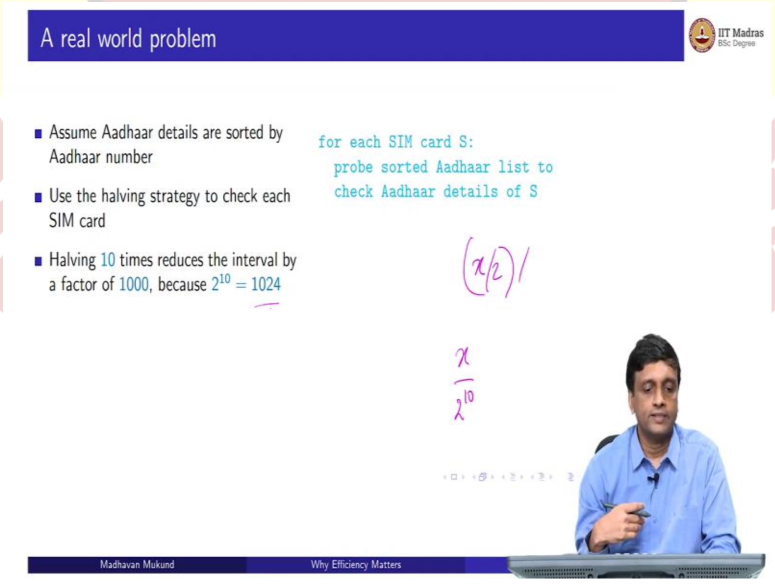
So, let us assume that this Aadhaar database is actually sorted. So, we can talk about Aadhaar numbers and go backwards and forwards. If I look for Aadhaar number, I can say whether the Aadhaar number I am looking for is before or after the current one I am searching for it, so I can probe this Aadhaar list just like a probe this birthday list, I can say look at the middle number in my Aadhaar list and then decide whether the number I am looking for is before or after, that is more or less what we did in the birthday, I asked you a date in the middle of the year. And I am told with this before or after by you. So in this case, by looking at the Aadhaar number I can tell if I look at the middle Aadhaar number in my list, I know whether the Aadhaar number I have is smaller or bigger.

So, I can use the same halving strategy that we did for guessing the birthday in order to keep halving the interval to search for the Aadhaar card. So, in other words for each SIM card S I use this halving strategy to find the Aadhaar card matching it. Now in the birthday case, I always found the birthday because that birthday existed.

Now, in the Aadhaar case, there is a situation where the Aadhaar card may not exist. So, what will happen is you will come down to an interval of 1 and that one card you only have one number left to look at and that is not the number that you have, then you will say it does not exist. Because if it were there, it should be this number, but I have because I have narrowed down this interval down to one number, either this number is the one I am looking for, or there is no such number in this list. That is something I can say.

So, we will look at this more formally. But that is essentially the idea that I keep halving it. And when I finally come down to this interval of one, either I found it or it does not exist. So, how long does this this probing take? How long does this take?

(Refer Slide Time: 15:30)


A real world problem

- Assume Aadhaar details are sorted by Aadhaar number
- Use the halving strategy to check each SIM card
- Halving 10 times reduces the interval by a factor of 1000, because $2^{10} = 1024$

```
for each SIM card S:
  probe sorted Aadhaar list to
  check Aadhaar details of S
```

$(x/2)/$

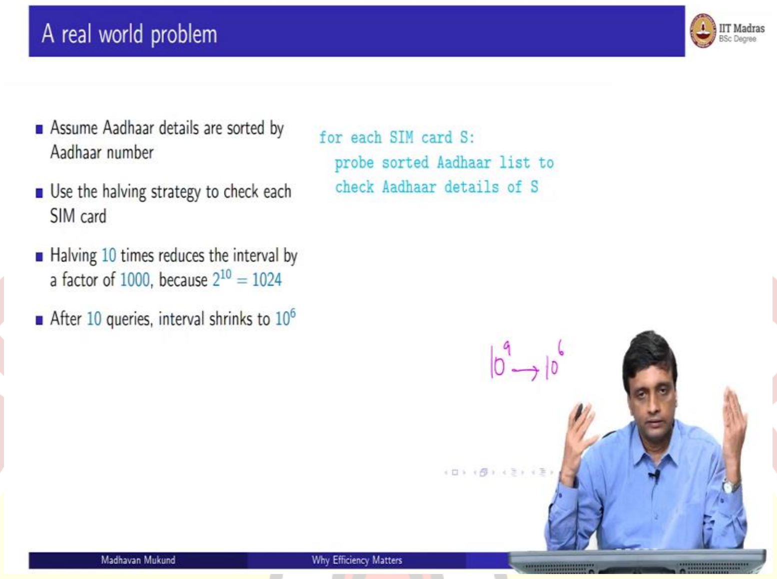
$\frac{x}{2^{10}}$

Madhavan Mukund | Why Efficiency Matters

Well, if I halve many times, then I am dividing by 2. So, if I halve 10, times, if I start with an x, and I divide by 2 and then divide by 2, and so on, then if I divide 10 times, then I divide by

2 to the 10. And 2 to the 10, is roughly 1000, 2 to the 10 precisely is 1024. So, if I halve something 10 times I reduce it to 1000th of its size.

(Refer Slide Time: 15:58)



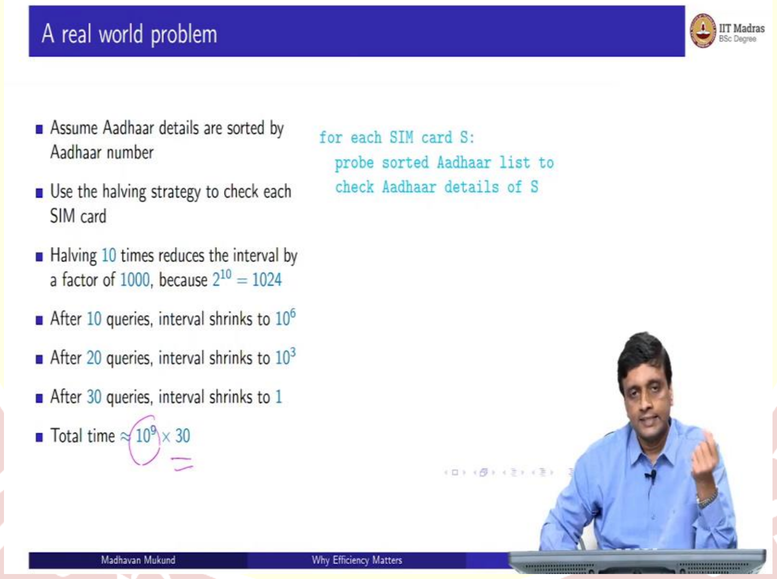
A real world problem

- Assume Aadhaar details are sorted by Aadhaar number
- Use the halving strategy to check each SIM card
- Halving 10 times reduces the interval by a factor of 1000, because $2^{10} = 1024$
- After 10 queries, interval shrinks to 10^6

```
for each SIM card S:
  probe sorted Aadhaar list to
  check Aadhaar details of S
```

$10^9 \rightarrow 10^6$

Madhavan Mukund   Why Efficiency Matters

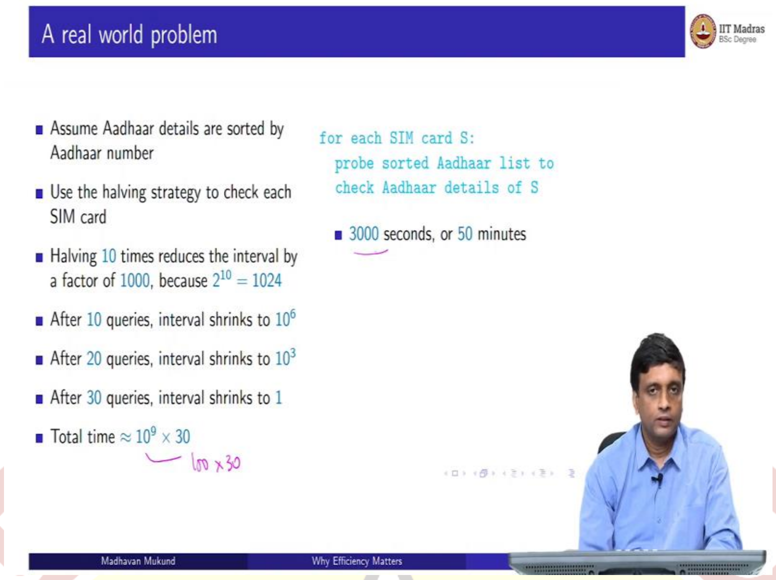


A real world problem

- Assume Aadhaar details are sorted by Aadhaar number
- Use the halving strategy to check each SIM card
- Halving 10 times reduces the interval by a factor of 1000, because $2^{10} = 1024$
- After 10 queries, interval shrinks to 10^6
- After 20 queries, interval shrinks to 10^3
- After 30 queries, interval shrinks to 1
- Total time $\approx 10^9 \times \underline{30}$

```
for each SIM card S:
  probe sorted Aadhaar list to
  check Aadhaar details of S
```

Madhavan Mukund   Why Efficiency Matters



A real world problem

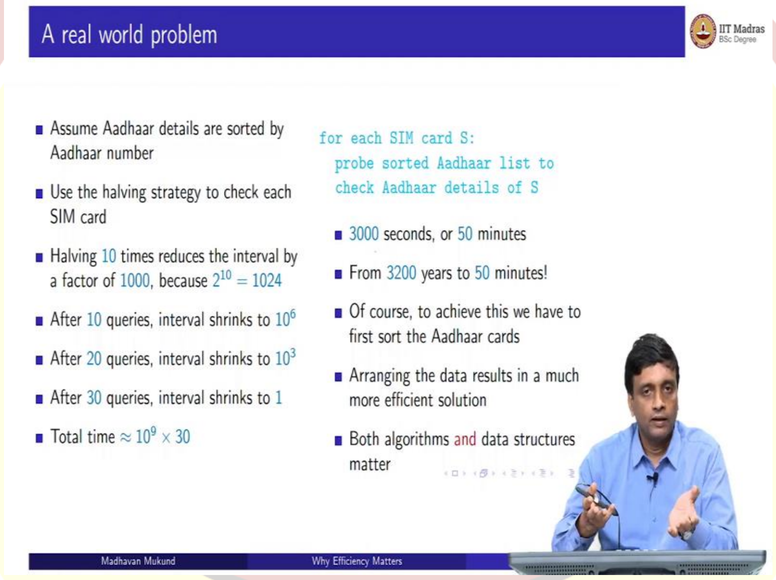
- Assume Aadhaar details are sorted by Aadhaar number
- Use the halving strategy to check each SIM card
- Halving 10 times reduces the interval by a factor of 1000, because $2^{10} = 1024$
- After 10 queries, interval shrinks to 10^6
- After 20 queries, interval shrinks to 10^3
- After 30 queries, interval shrinks to 1
- Total time $\approx 10^9 \times 30$

$10^9 \times 30$

```
for each SIM card S:
  probe sorted Aadhaar list to
  check Aadhaar details of S
```

- 3000 seconds, or 50 minutes

Madhavan Mukund   Why Efficiency Matters



A real world problem

- Assume Aadhaar details are sorted by Aadhaar number
- Use the halving strategy to check each SIM card
- Halving 10 times reduces the interval by a factor of 1000, because $2^{10} = 1024$
- After 10 queries, interval shrinks to 10^6
- After 20 queries, interval shrinks to 10^3
- After 30 queries, interval shrinks to 1
- Total time $\approx 10^9 \times 30$

```
for each SIM card S:
  probe sorted Aadhaar list to
  check Aadhaar details of S
```

- 3000 seconds, or 50 minutes
- From 3200 years to 50 minutes!
- Of course, to achieve this we have to first sort the Aadhaar cards
- Arranging the data results in a much more efficient solution
- Both algorithms and data structures matter

Madhavan Mukund   Why Efficiency Matters

So, therefore, after 10 queries, remember my Aadhaar interval, there are 10 to the 9 possible Aadhaar cards from the smallest one to the biggest one, if I have now halve this thing, 10 times, I have brought this down by a factor of 1000 to an interval of size 10 to the power 6. Now, I keep halving. So, this halving applies at every step, so this 10 to the 6 after 10 more queries will come down to 10 to the 3.

So, I am now have come from 1 billion to 1 million after 10 queries. From 1 million I have now a range of 1000 after 10 more queries, and now 1000 range in 10 more queries, I will narrow down to one because 1000 by 1000 is 1. So, after 10 queries, I had 1 million after 20 queries, I had 1000 after 30 queries, I have narrowed down to a single card either this is the Aadhaar card I want to check all this is not the Aadhaar card, there is no Aadhaar card of my SIM card in this. So, in 30 queries, I have completed the search for each SIM card.

So, therefore overall, there are 10 to the 9 SIM cards and each of them I might spend 30 steps looking for the thing. And then let us assume that you do some basic operations. So, this is my estimated work. So, earlier, if for each SIM card, I had to scan through 10 to the 9, so it is 10 to 9 times 10 to the 9, I have converted that second 10 to the 9 work, instead of scanning through the entire list by probing this sorted Aadhaar list, I brought it down to 30.

So, remember that this is going to take 100 seconds, because 10 to the 7 is what I can do. So, 100 times 30 is the number I am looking at. So, I have 3000 seconds. If I divide 3000 by 60, then I get 50. So, therefore, this process is going to take me something relate off the order of 50 minutes. Now, when it was not sorted, we had something which would take 3200 years.

And by just the simple fact that we have arranged the Aadhaar cards in a nice way, we are able to use this birthday probing, halving the interval and bring it down by enormous number of amount of time from 30 to 100 years to 50 minutes. So, 50 minutes is not fast, but at least you can start this thing and go and have lunch and come back and it will get done. So, it is something which will happen in an hour's work.

So, it is something that you can start off and go and finish and come back. So, it is not, and of course, this is in Python, if we do it in another language, it will actually happen in less than a minute. So, we have taken something which is wildly infeasible and made it extremely efficient.

So, of course, there is a step which we have to do, which is to sort the Aadhaar cards. Because every time somebody creates a new Aadhaar card, this list will it is not a onetime operation. So, this Aadhaar card list has to periodically be resorted because new Aadhaar cards keep getting added to this list. So, we need an efficient way to sort the Aadhaar cards which we are not discussing now.

So, assuming that this Aadhaar card sorting does not take 3000 years, if it took 3000 years to sort the Aadhaar cards, then we are anyway doomed. But assuming we can also sort the Aadhaar cards in some reasonable amount of time. Then we can apply this halving trick to solve this problem in 50 minutes.

So, the main purpose of this story is to show you that we were able to achieve this efficiency by assuming something about how the data is arranged. So, we had to first put the data into this sorted form and only then we could apply this halving strategy to get to our answer. So, both were required I could not, so when we did GCD for example, and we came up with

Euclid’s algorithm, we did not exploit anything about how the numbers were given to us M and N is given to us, but we exploited some number theoretic properties of the remainder in order to convert it to a simpler problem.

So, that was an algorithm which did not represent depend on how the data was presented to us. Here, we are actually changing the way the data is presented to us we are assuming that the Aadhaar cards are sorted and if the Aadhaar are sorted then we can do this efficiently. So, the moral of the story is that both are important.

So, this is why this course is called data structures and algorithms. So, it is important sometimes how the data is organized. So, in the very least thing we have the organization in this case, so, it is not a structure as such, but it is just saying that the arrangement is in a particular way. We will also see examples where the format in which the data is presented makes it easier to look up things.

So, those are data structures. So, if you have good data structures, then you can think of cleverer algorithms. So, both data structures and algorithms are important. And it can make a problem which is immensely useful to solve and bring it down from something which is wildly impractical to something which is highly efficient.