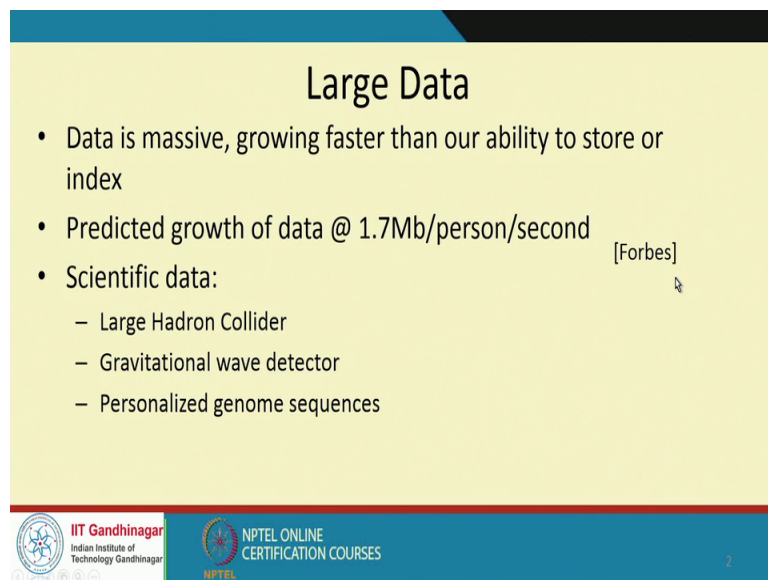


**Scalable Data Science**  
**Prof. Anirban Dasgupta**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Gandhinagar**

**Lecture - 08**  
**Streaming Model, Counting Distinct Elements**


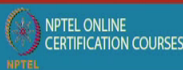
Welcome to the course on Scalable Data Science. Today's lecture is on a Streaming Models; on the streaming model and on the problem of counting distinct elements. My name is Anirban. And I am a faculty of computer science and engineering at IIT Gandhinagar.

(Refer Slide Time: 00:35)



**Large Data**

- Data is massive, growing faster than our ability to store or index
- Predicted growth of data @ 1.7Mb/person/second [Forbes]
- Scientific data:
  - Large Hadron Collider
  - Gravitational wave detector
  - Personalized genome sequences

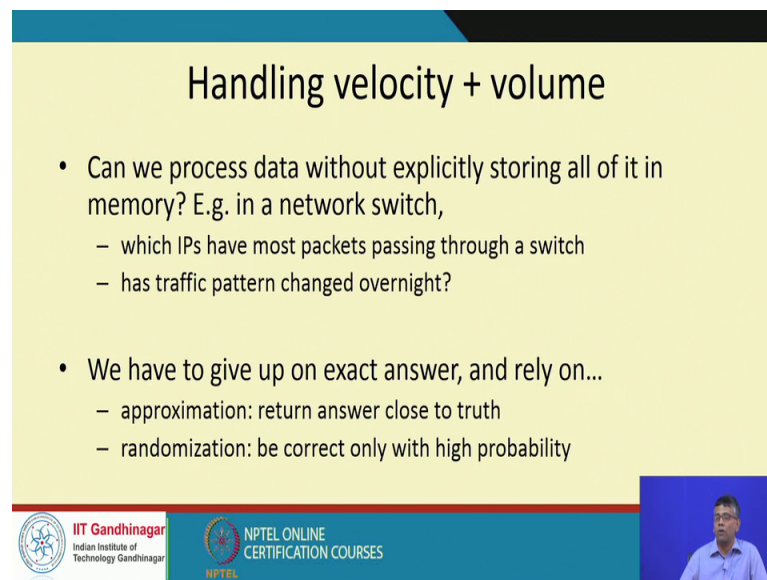
 

So, all of you have must of heard the phrase big data or large data, which is this course is all about of course. And I do not need to convince you if the data is massive it is going to faster than our ability to store or even index right. Fact without through any other equal numbers of it (Refer Time: 00:50) the predicted rate of growth of data given by Forbes is at 1.7 Mb per person per second right.

And this will create and this creates zeta bytes of data in about number in few hours or few seconds available right. Some of the more astonishing and very useful data sources that we really need to analyze then the, because we have spent millions and billions of dollars increasing this data, are for instance the large the data coming out of the large hadron collider or the data coming out of gravitational wave detectors.

Or the data that are going to come out of the personalized genome sequences right. Imagine if you could sequence a genome of obviously, individual in the in the planet how much data will that be. And we really need to be able to analyze this data at the speed that it is coming right.

(Refer Slide Time: 01:45)



The slide is titled "Handling velocity + volume" and contains two main bullet points. The first bullet point asks if data can be processed without explicitly storing it all in memory, using a network switch as an example, and lists two sub-points: "which IPs have most packets passing through a switch" and "has traffic pattern changed overnight?". The second bullet point states that we must give up on exact answers and rely on approximations or randomization, with sub-points: "approximation: return answer close to truth" and "randomization: be correct only with high probability". The slide footer includes the IIT Gandhinagar logo and name, the NPTEL logo, and the text "NPTEL ONLINE CERTIFICATION COURSES". A small video inset of a speaker is visible in the bottom right corner of the slide.

### Handling velocity + volume

- Can we process data without explicitly storing all of it in memory? E.g. in a network switch,
  - which IPs have most packets passing through a switch
  - has traffic pattern changed overnight?
- We have to give up on exact answer, and rely on...
  - approximation: return answer close to truth
  - randomization: be correct only with high probability

IIT Gandhinagar  
Indian Institute of  
Technology Gandhinagar

NPTEL ONLINE  
CERTIFICATION COURSES  
NPTEL

This I have is typically termed as problem of handling the combination the velocity and volume. In a large amount of data is coming at us at a tremendous high velocity and we have no option, but to try to process it without explicitly storing all of it in memory right.

So, imagine you are a network switch designer right. And as a designer you have to answer simple questions like this which IPs have the most packets passing through the switch right. Has the traffic pattern changed overnight dramatically? Because the answer to this questions might give you indication of whether there is a demand of define denials that is that happening by exploiting mechanism through a switch right. And you have better be able to sort of answer these questions fairly accurately right, without having 2 sort of stores the entire data in some offline cloud, and then doing some I mean processing on it and so on right. It was that takes very too much time right.

And has a network switch person you do not even have accessed to that much memory, it could potentially be megabytes or even smaller than that. So, what do we do? We have to give on trying to give an exact answer. This is the first thing that we do. We have to try to and we have to rely on only trying to deal with approximations.

That is will try to return answers that are close enough to prove it. And will also try to be correct only with high probability. We have already seen instances of this like in the bloom filter where we are giving up on the fact that we had to return in a correct times always we say it quite a bit on the space ok. We went from something like  $m \log u$  to  $n$  space and bits in the bloom filter. So, here again that is the strategy that will take that will return approximately answers and will be with returning wrong answers sometimes.

(Refer Slide Time: 04:02)

**Streaming model: sketches**

- Data is assumed to come as a stream of values
  - e.g. bytes seen when reading off a tape-drive
  - destination IPs seen by a network switch
- Size of universe/stream is much large compared to available memory
  - typically assume memory is  $\text{poly}(\log)$
  - Can make limited (possibly single) pass over data
  - Will create a “sketch” : a summary data structure used to answer queries at the end

IIT Gandhinagar  
Indian Institute of Technology Gandhinagar

NPTEL ONLINE  
CERTIFICATION COURSES

Page 1/1

So, data is assumed to come as a stream of values. For instance, it could be bytes when reading off a tape drive. Or it could also be the destination IPs that are seen by network switch right.

So, the size of the universe is assumed to be much large compared to the available memory. Typically, what we stay say that if the if the available if the size of the universe is  $u$ , then the available memory is at most  $\text{poly} \log$  of  $u$  not much more than that you also restrict algorithm. So, that it makes a limited number of passes over the data. And at best I mean you would really like you can make only one pass.

And in certain settings it is all we could do for instance if it is a network switch right the data is just flowing by  $u$  if it is wrong it is gone. And whatever you have to calculate you have to calculate only one possible of data. In certain other case for instance if data is really is in a tape drive somewhere you can potentially make a small number of passes over it is expensive, but you can still do it right.

So, what we want to do is that will create summary data structure ok. Let us call it a sketch although has been sort of define it later a sketch has certain. I mean a sketch is little more formulae define them we are sort of talking about here, but let us call it a sketch. So, will create these sketches out of the data and will use this sketches to try to answer the queries of the end ok. So, the question is; how do we design these sketches for non trivial questions.

(Refer Slide Time: 05:48)

Page 2/2

## Streaming problem: distinct count

- Universe is  $U$ , number of distinct elements =  $n$ , stream size is  $m$ 
  - Example:  $U$  = all IP addresses

10.1.21.10, 10.93.28.1, ..., 98.0.3.1, ..., 10.93.28.1, ...

- IPs can repeat
- Want to estimate the number of distinct elements in the stream

And here is a very simple looking question first. The problem is that of estimating the number of distinct elements that we have seen ok. So, what is this? Let us formulate define it suppose the element suppose universe is  $u$  the number of distinct elements that you have seen is  $n$  ok. This  $n$  is smaller than  $u$  of course, could be potentially much smaller and the stream size is  $m$  right which is bigger than  $n$ .

So, what does it mean it means? At the same element could potential appear multiple times. For instance, you see here instead of IPs is 20 dot 10 may be this particular IP has appeared here again and then there is some other IP and so on right. So, they say and may be this IP could come back here later and may be here later and so on. May be this IP could come back somewhere here later and so on. So, remember that we cannot really assume anything about the order in which this data comes right. Because of it is network switch the IP is that IP traffic is coming in a particular order ok. And we want to estimate the number of distinct elements. So, we have seen. So, what do we do right?

(Refer Slide Time: 07:05)

Other applications

- Universe = set of all k-grams, stream is generated by document corpus
  - need number of distinct k-grams seen in corpus
- Universe = telephone call records, stream generated by tuples (caller, callee)
  - need number of phones that made > 0 calls

IIT Gandhinagar  
Indian Institute of Technology Gandhinagar

NPTEL ONLINE  
CERTIFICATION COURSES

So, before seeing what we do let us see let us see of a couple of applications. As we saw the network switch is one example here is another very common example. Suppose you are analyzing the document corpus right. And when you are doing modelling of this of the of this document corpus sometimes you need to create whereas, known as this k grams which are essentially k, I mean a bytes of you take k consecutive bytes that have appeared in your in your text data right. And you have to create a k gram out of it right. And your stream is generated by this by this document corpus that you are meeting.

So, this document corpus is used and you are reading it and you are sort of creating this k grams as you as you sort of pass by and in order to create your may be your topic model or your natural language model you need the number of distinct k grams the case seen in the corpus right it is some estimate of that right. And remember that the number of distinct k grams the potential number of distinct k grams could be the universe is huge right. Because there are having; let us say that there are 256 bytes and each of them and you are taking the value of k 256 to the power k which is huge even correspond the values of k.

Similarly, if we are analyzing let us say telephone call records as in who called whom right. And Vodafone could want to see what are the number of telephone calls that may what are the number of phones that may at least one call in this month right. And then from this tuple from the stream of this tuples you now want to see that I look at only the

first element of the tuple. And I want to see how many how many phone numbers distinct phone numbers appear as caller in at least one of the records ok.

(Refer Slide Time: 08:59)

The slide is titled "Solutions" and contains the following content:

- Naïve solution :  $O(n \log(U))$  space
  - store all the elements, sort and count distinct
  - store a hash map, insert only if not present in map
- Bit array:  $O(|U|)$  space
  - bits initialized to 1 only if element seen in stream
- Can we do this in less space? Not when exact solution needed!!

The slide footer includes the IIT Gandhinagar logo, the text "IIT Gandhinagar Indian Institute of Technology Gandhinagar", the NPTEL logo, and the text "NPTEL ONLINE CERTIFICATION COURSES". A small video inset of a speaker is visible in the bottom right corner.

So, again before we look at anything (Refer Time: 09:01) will try to look at what are the naive solutions. The first naive solution is what we know all I mean what we already have done I can immediately think about is that you store all the elements is sorted and you count the number of distinct elements. So, how much space does this take this takes a lot of space. Because in order to store all the store one element you need  $\log u$  bytes. I mean  $\log u$  bits and to store  $n$  of them you need  $n$  times  $\log u$  right.

So, therefore, you need that much  $n$  times  $\log u$  space, if you are doing a sorting. And then accounting it is not we have to see that and we already saying is that I mean if you stored a hash map instead right. And then they you do not sort, but you store we are keeping a hash map, and then and then you for every new elements that comes, first you check whether this element is already there in a hash map right. If it is not, then you insert it in the hash map and increment around and if it is there then it is not in this elements. Here you do not expect it is sorting.

So, it might be better you might think that it is better algorithm, but unfortunately not in terms of this space that you have done that you need even this needs  $n \log u$  space. The bit array solution that we saw before right although for a slightly different problem the same solution a similar solution works here, you could use a bit array of size of size  $u$

and you initialize only if the element is in the single stream this needs order  $u$  space. So, can we do it in much less space than this?

Unfortunately, no it is possible to show using information theoretical arguments, that if you need an exact solution, if you need an exact solution, even if you allow for randomness you cannot do much better than this. However, if you allow this combination if the of the 2 an approximate solution as well as randomness that is the possibility of making errors sometimes, then suddenly magic happens right.

(Refer Slide Time: 11:15)

Page 7/7

## Approximations

- $(\epsilon, \delta)$  –approximations
  - Algorithm will use random hash functions
  - Will return an answer  $\hat{n}$  such that
$$(1 - \epsilon)n \leq \hat{n} \leq (1 + \epsilon)n$$
  - This will happen with probability  $1 - \delta$  over the randomness of the algorithm

IIT Gandhinagar  
Indian Institute of Technology Gandhinagar

NPTEL ONLINE  
CERTIFICATION COURSES

What happens is that you get a much better solution in much less space. And we call this kind of solutions. So, these will be a common categories of these will see. And we will call these to be epsilon delta approximations right. So, what is an epsilon delta approximation?

So, here is our filterpically look at that supposed the true the true solution is  $n$ . Suppose, the true solution is  $n$  which is the actual number of distinct counts. Suppose the algorithm is returning some  $\hat{n}$ . The algorithm is returning  $\hat{n}$ , and epsilon we call it epsilon delta approximation. If it satisfies the property that the estimate  $\hat{n}$  lies within  $1 - \epsilon$  of  $n$  and  $1 + \epsilon$  of  $n$  with probability  $1 - \delta$ .

So, with probability  $1 - \delta$  we can say that the solution that have returned is within a  $1 - \epsilon$  to  $1 + \epsilon$  multiplicative factor of the original solution

ok. And so, what is this probability over this probability over the randomness of algorithm in our specific case, in the case of the distinct counts the randomness will be the choice of random hash functions.

So, the probability will be over the choice of the random hash functions ok.

(Refer Slide Time: 12:50)

**First effort**

- Stream length:  $m$ , distinct elements:  $n$   $m > n$
- Proposed algo: Given space  $S$ , sample  $S$  items from the stream
  - Find the number of distinct elements in this set:  $\hat{n}$
  - return  $\hat{n} \times \frac{m}{S}$
- Not a constant factor approximation
  - $1, 1, 1, 1, \dots, 1, 2, 3, 4, \dots, n-1$
  - $m - n + 1$
  - $m > n$
  - $m = n^2$
  - $S \approx n$
  - $\hat{n} = 1$  okp

Handwritten diagram:  $a_1, a_2, \dots, a_m$  with arrows pointing to 1, 2, 1, 2. Below it:  $\hat{n} = 2$ ,  $2 \times \frac{m}{4}$ .

So, again before, we do anything complicated let us try something simple. Let us look at this case setting that the stream that the universe size is  $n$  the stream length is  $m$  and  $m$  is much bigger than  $n$ . So, do remember that we are using  $m$  here for the length of the stream, because in our previous lectures we have used it as the size of the memory or the size of the data sets.

So, we have removed the notations here just to just to caution you propose here is the proposed algorithm. A proposed algorithm is that suppose we are given space limit of  $s$  right; that you can store only store  $s$  items what can we do let us sample  $s$ ,  $s$  items from the stream right. Let us sample  $s$  positions from the stream that may be if the stream is like ok, a 1 a 2 an a  $m$ , we let us say sample the position 1 we sample the position 2 we sample the position 3 and we sample position  $m$  right.

So, we have sampled  $m$  positions. Let us find out the number of distinct elements in this set. So, may be a one is really the element 1 a 2 a  $m$  minus 1 is also the element 1 a 2 element 2 and  $a_m$  is element 2. So, the number of distinct elements  $\hat{n}$  here is 2. And



then what am proposing is that you return  $\hat{n}$  times  $m$  by  $s$ . So, for instance here will return 2 times 4 2 times  $m$  which is known divided by  $s$  which is 4 ok. That is the estimated method will return here ok. Since natural right, but this is a good thing to do. Unfortunately, not, and it is not very hard to see why. So, imagine this particular setting right.

So, in this particular setting we have only the elements from one to  $n$  minus 1, let us say and the first whole lot of the array the whole lot of the of the of the input sequence. In fact,  $m$  minus  $n$   $m$  minus  $n$  plus 1 of the of the stream positions is really one element, one the same element occurring multiple times. And then you get  $n$  minus 1 distinct elements or rather  $n$  minus 2 distinct elements here ok. So now, supposing if  $m$  is much bigger than  $n$  suppose if  $m$  equal to let us say  $n$  square.

So, in that case and suppose in  $s$  is like,  $s$  is like I do not know  $n$ . So, if you sample  $n$  positions out of  $n$  square right with very, very high probability, you will only get the elements here right. And therefore, you will return you are your estimate  $\hat{n}$  will be equal to 1 with high probability ok.



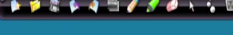
So, it does not work. So, what do we do?

(Refer Slide Time: 16:15)

Page 17/12

## Linear Counting

- Bit array  $B$  of size  $m$ , initialized to all zero
- Hash function  $h: [x] \rightarrow [m]$
- When seeing item  $x$ , set  $B[h(x)] = 1$
- $z_m =$  fraction of zero entries
- Return estimate  $-m \log\left(\frac{z_m}{m}\right)$

And in fact, it does not work because see what we what may happens is that, if you are sampling only this one particular element you cannot distinguish between the 2 cases.

(Refer Slide Time: 16:30)

Page: 8/11

## First effort

- Stream length:  $m$ , distinct elements:  $n$
- Proposed algo: Given space  $S$ , sample  $S$  items from the stream
  - Find the number of distinct elements in this set:  $\hat{n}$
  - return  $\hat{n} \times \frac{m}{S}$
- Not a constant factor approximation
  - $1, 1, 1, 1, \dots, 1, 2, 3, 4, \dots, n-1$   
 $\underbrace{\hspace{1.5cm}}_{m-n+1}$

IIT Gandhinagar  
Indian Institute of  
Technology Gandhinagar

NPTEL ONLINE  
CERTIFICATION COURSES

When we have when we have 1 1 1 1 1, when we have 1 1 1 1 one and 2 3 4 n minus 1 we have 1 1 1 1 1. And then again more ones right. These 2 cases we return the same answers right. So therefore, this particular sampling algorithm cannot work.

So, let us look at another simple algorithm. This will not be very efficient, this will not be the most efficient algorithm that will be that will see, but it will be fairly good it is it is actually fairely useful for lot of real settings. And this is known as linear counting. So, what do we have we have a bit array  $b$  of size  $m$  again initialized to all 0s, will choose  $m$  the value of  $m$  later. We again have a hash function  $b$  right that maps from that maps from the unversed to  $m$ .

So, when seeing an item  $x$  right, we do not I mean we only do one simple thing we go to the position that the hash function in bits and then we set that bit to be one right. So, this should really be u ok. So, what we do when we are trying to estimate. So, at the end we look at the number of non 0 entries.

(Refer Slide Time: 18:10)

Linear Counting Analysis

- $\Pr[\text{position remaining } 0] = \left(1 - \frac{1}{m}\right)^n \approx e^{-\frac{n}{m}}$
- Expected number of <sup>u</sup>positions at zero =  $E[z_m] = me^{-n/m}$

==

- Using tail inequalities we can show this is concentrated
- Typically useful only for  $m = \Theta(n)$ , often useful in practice

IIT Gandhinagar Indian Institute of Technology Gandhinagar NPTEL ONLINE CERTIFICATION COURSES

We look at.

(Refer Slide Time: 18:31)

Linear Counting

- Bit array  $B$  of size  $m$ , initialized to all zero
- Hash function  $h: [n] \rightarrow [m]$
- When seeing item  $x$ , set  $B[h(x)] = 1$
- $z_m =$  <sup>number</sup> ~~fraction~~ of zero entries
- Return estimate  $-m \log\left(\frac{z_m}{m}\right)$

IIT Gandhinagar Indian Institute of Technology Gandhinagar NPTEL ONLINE CERTIFICATION COURSES

So, we look at  $z_m$  to be the number of non 0 entries of. So, the number of 0 entries right. And then we return an estimate like this minus of  $m \log$  of  $z_m$  by  $m$  ok. So, why does this makes sense? Let us try to do the analysis.

So, we return this as our estimate  $\hat{n}$  ok. Fair enough. Why does it makes sense? Let us try to analyze. So, just is the sanity check see that it is returning at least I mean at least positive number right. Because the  $z_m$  by  $m$  is the fraction and therefore, the log of that

is the negative number. So, it is return with at least something that is a positive number fair enough. Let us see.

So, to see by that makes sense, let us look at the posi I mean again the probability that a particular position remains at 0. We have looked at such a probability when analyzing the bloom filters right. Then it is very similar analysis, that because we have done  $n$  insertions the probability that particular bit positions a particular bit position remain 0 is again the probability that all the insertions going to other  $n$  minus 1 bit positions. And the probability of that is  $1 - \frac{1}{m}$  to the power  $n$  right. And this let us approximate by  $e$  to the power minus  $n$  by  $m$ .

So, therefore, that the expected number of positions that are still at 0. So, remember  $z_m$  is the number of positions that are counting at 0. What is the expectation of  $z_m$ ? So, the expectation of  $z_m$  is  $m e^{-n/m}$  right. Because the probability that each of that a single one remains 0 is  $e^{-n/m}$  therefore.

So, we define a indicated random variable and you just count the number of positions that are 0. And they turns out to be  $m e^{-n/m}$ .

(Refer Slide Time: 20:34)

The slide is titled "Linear Counting Analysis" and contains the following content:

- $\Pr[\text{position remaining } 0] = \left(1 - \frac{1}{m}\right)^n \approx e^{-\frac{n}{m}}$
- Expected number of positions at zero =  $E[z_m] = m e^{-n/m}$

Handwritten notes on the slide include:

$$\hat{z}_m \approx m e^{-n/m} \quad \ln\left(\frac{\hat{z}_m}{m}\right) \approx -\frac{n}{m} \Rightarrow \hat{n} = -m \ln\left(\frac{\hat{z}_m}{m}\right)$$

- Using tail inequalities we can show this is concentrated
- Typically useful only for  $m = \Theta(n)$ , often useful in practice

The slide footer includes logos for IIT Gandhinagar and NPTEL ONLINE CERTIFICATION COURSES, along with a small video inset of a speaker.

So, therefore, we say that suppose in we believe that  $z_m$  hat is closed to expectation. Therefore,  $z_m$  hat equals  $m e^{-n/m}$ . And therefore,  $z_m$  hat by  $m \ln$

of that equals minus  $n$  by  $m$ . And that gives you the estimate that  $\hat{n}$  equals minus  $m \ln z$  by  $m$  ok. And now you see how that mysterious formula came to be.

So, this equality why does it hold; this in order to see why this equality is justified we have to go back to some other tail inequalities that we have taught you. And we will not sort of do that analysis  $m$  because we look at smarter algorithms, but it is not very hard to show that is fairly concentrated. So, this is actually justifiably gives you a nice estimate. Although for that right you have to use the value of  $m$  to be some constant factor times  $n$ , which is the number of elements that you are that you are looking to count.

But this is often useful in practice, that is if you are not sort of interested in coding some more complicated algorithms this is our a quick. And dirty if you are reasonable amount of we have been a whole lot of memory always in our memory that you dispose ok.

(Refer Slide Time: 22:07)

Page 9/17

## Flajolet Martin Sketch

- Components
  - "random" hash function  $h: U \rightarrow 2^\ell$  for some large  $\ell$
  - $h(x)$  is a  $\ell$ -length bit string
  - initially assume it is completely random, can relax
- $\text{zero}(v)$  = position of rightmost 1 in bit representation of  $v$ 

$$= \max\{i, 2^i \text{ divides } v\}$$
  - $\text{zeros}(10110) = 1$ ,  $\text{zeros}(110101000) = 3$

4 3 2 1 0  
 $\text{zeros}(10110) = 1$ ,  $\text{zeros}(110101000) = 3$ 
3

So, here is a very fun algorithm right. And this was a very rich tree by martin, when was actually dealing with pdp computers in which you literally had an equilibrates of memory right. And then but he was an engineer and he could not really analyze it and it was analyzed by this brilliant mathematician Philip Flajolet please look him up on your Wikipedia if you can, to provide this and further sort of improved beyond what martin was doing to be rather is very excellent algorithm, which is still in use this is known as commonly known as the Flajolet martin sketch or the FM sketch.

So, here is what we have. Again the secret ingredient is the hash function. And here it is a hash function that maps every element of the universe to a bit stream of size  $l$  ok. Here I have written it as  $2$  to the  $l$ , but think of it as a bit stream of size  $l$  a bits stream of size  $l$  ok. So, initially let us just assume that  $h$  of  $x$  is completely random right. Although, remember that we have talked about the fact that completely random hash functions are not practical because they cannot really be stored.

So, we have to use something like  $2$  universal or  $k$  universal hash functions. And ah, but it is just easier to the analysis with completely random hash functions and then see how we can change it for  $k$  universal or for more realistic hash functions. So, here is what we need to define, very mysterious quantity. We need to define given any bit stream  $v$ . So, think of bit streams and integers as interchangeable quantities right. For any bit stream  $v$  define  $0s$   $0$  of  $v$   $0$  of  $v$  to be the position of that right most one

So, think of this positions  $0$  index which is position  $0$  position  $1$  position  $2$  position  $3$  position  $4$  right. So, I say that  $0$  of  $1\ 0\ 1\ 1\ 0$  is  $1$ . Because the right most one is at position  $1$  here the right most one is at position  $0\ 1\ 2\ 3$  at position  $3$  right. So, note that this also means that this is the maximum  $I$  such that  $2$  to the,  $I$  divides the integer  $v$  right. Because this quantity  $I$  mean you have  $I$  minus  $1$   $0s$   $m$  is that it is divisible by  $2$  to the.  $I$  ok.

(Refer Slide Time: 24:55)

Page 12/20

## Flajolet Martin Sketch

Initialize:

- Choose a "random" hash function  $h: U \rightarrow 2^l$
- $z \leftarrow 0$

Process(x)

- if  $\text{zeros}(h(x)) > z$ ,  $z \leftarrow \text{zeros}(h(x))$

Estimate:

- return  $2^{z+1/2}$

maxim rtimeat posh of 1

IIT Gandhinagar Indian Institute of Technology Gandhinagar NPTEL ONLINE CERTIFICATION COURSES

So, here is the very simple algorithm. The algorithms just says that we choose a random hash function  $h$  that maps every element of  $u$  in to a  $l$  length bit stream. Then we keep a

counter  $z$ , right. That is initially said to be 0. So, this is initialization that is it. Now we go we are going over all the elements of the stream right. And at every element when you when you get the element  $x$  you call the function process of  $x$ , when we call process of  $x$  this is what happens, right. It just says that.

It counts the it calculates  $h$  of  $x$ , then it sees that let us look at the number of zeros let us look let us calculate the zeros function  $n$   $h$  of  $x$  which means that let look at the position of the right most one. Is that bigger than the values showed in  $z$ ? If that is so then you update  $z$  with this  $y$  that is it. So, basically  $z$  contains always contains the maximum position the maximum of the right most bit elements right most bits of the right most one bits ok.

So,  $z$  contains the maximum right most position of one. So, when we trying to estimate, we take; I mean at the end of the stream we are supposed to call the function estimate. And we calculate  $2$  to the power  $z$  plus half one and we just return that. That is our estimate ok.

So now, why this is happening? Why does it work? Let us before do the analysis let us run through a simple example for that right.

(Refer Slide Time: 27:05)

Page 13/21

### Example

$m=2$   $n=4$

1 1 1 1 2 2 2 2 2 2 2 2

$z=0$

$z=2$

$2^{2+1/2} \approx 2^{2.5}$

	$h(.)$
●	0110101
●	1011010
●	1000100
●	1111010

$\leftarrow$  zeros = 1

$\uparrow$

$\downarrow$

17 135 14:02

So, imagine these are the elements right. So, the color is the particular element id. So, here we have stream of length say 1 2 3 4 5 6 7 8 9 10 11 12 12. Let us say stream of length is 12 we have  $n$  equal to 12 and the true  $n$  equal to 4 on the 4 elements in here.

So, first  $z$  is initialized to 0. Ok now we get the blue element right here zeros the functions zeros return 0. So, here  $z$  is still 0 right sorry. Now first we get the red element first, we get the red element and here zeros  $z$  is 1. So, therefore, we get. So, therefore,  $z$  is initialized to be 1 right. Then we get the blue, right zeros; 0 therefore,  $z$  is the maximum of 1 and 0.

Therefore, it remains  $z$  1, right. Then we get that red again does not matter remains at one. Are you noticing something one again, it remains at one do you notice that if thus if, if an element appears the second time or the third time it does not change the value of  $z$  right that is very important criteria of the statistics that we are keeping. Now we get the turquoise the turquoise blue for this one  $z$  is 0 1 2. So now,  $z$  becomes 2. And once it is 2 here again remains a 2 again remains a 2 again remains a 2 remains a 2 again turquoise does not change remains a, 2 now we are getting a new element, but that has said to be that has zeros said to be one right, because right most one is at position 1.

So, therefore, 2 does not change this statistic that we are keeping that does not change and here it does not change. So, therefore, at the end of the stream we get  $z$  equal to 2 and therefore, we did we return estimate 2 to the power 2 plus half, which is something like 2 to the power 2.5. See this is not entirely accurate of course, because this problem was very simple with this switches to dry run right. And this is how this is as simple how simple the algorithm is, but now the major task behind us is to show why is this correct ok

So, how can we do that?



(Refer Slide Time: 29:45)

The slide is titled "Space usage" and contains the following text:

- We need  $\ell \geq C \log(n)$  for some  $C \geq 3$ , say
  - by birthday paradox analysis, no collisions with high prob
- Sketch :  $z$ , needs to have only  $O(\log \log n)$  bits !!!
- Total space usage =  $O(\log n + \log \log n)$

There is a handwritten note in blue ink that says " $\ell > 3 \log n$ " with a box around it and lines pointing to the right. The slide also features logos for IIT Gandhinagar and NPTEL ONLINE CERTIFICATION COURSES at the bottom.

So, before we try to analyze why this is correct ah, let us look a little bit into the space usage of this algorithms. So, first of all what we need to think about is how big do we need these bits stream this string of bits to be right. That if you are mapping every element to a stream of bits how big is the stream of bits it should how big the stream of bits should d.

So, using something known as a birthday paradox, that you should definitely look up, you can what you can say is that if the size of bits stream is at least let us say  $3 \log m$  right. Then with very high probability every element gets a unique bit stream. That is, we do not get collisions ok. And this happens with very high probability as long as the size of the stream the size of this bit stream is bigger than  $3 \log n$ . Because, in some sense if the size of the bit stream is bigger than  $3 \log n$  then the number of such possible bit streams is like  $2$  to the power  $2$  to the power  $3 \log n$  which is  $m$  cube right.

(Refer Slide Time: 31:03)

Page 15/22

## Space usage

- We need  $\ell \geq C \log(n)$  for some  $C \geq 3$ , say
  - by birthday paradox analysis, no collisions with high prob
- Sketch :  $z$ , needs to have only  $O(\log \log n)$  bits !!!
- Total space usage =  $O(\log n + \log \log n)$

$2^{3 \log n} \sim n^3$

$0 - 2^{3 \log n - 1}$

$\log_2(2^{3 \log n}) \approx 3 \log n$

So, refer the number of possible if you are mapping  $n$  thing is randomly into a universe of size  $m$  cube number of collisions is a basically 0 right with very high probability ok.

So, then, but that means, is that in order to store the bits stream you also meet the algorithm meets store  $c \log n$  right. Because when it gets an element it calculates the bits stream for that. So, you keep  $c \log m$  space. The only or the space that it requires is this counter  $z$  that it is storing right; what is the size of that right: see the maximum value of  $z$  only needs to be  $3 \log n$  right. Because remember what  $z$  is store storing,  $z$  is storing the position of the right most one. So, the position because the bit stream because this stream is of length  $l$  the position of the right most one also varies from 0 to  $l$  minus 1. So, therefore,  $z$  only needs to store values from 0 to  $l$  minus 1 which means that it needs to store values from 0 to  $3 \log n$  minus 1.

Therefore, the size of  $z$  only needs to be  $\log$  base of  $3 \log n$  right. Which is let us say  $\log n$ . That is quite astonishing right. We only need to store a counter of size  $\log \log n$ . So, therefore, the total space usage is really  $\log n$  plus  $\log \log n$  right. It is very interesting, but is it any good, we have developed an algorithm to that uses magically uses very less space, but does it provide a good estimate and we do not know that.

(Refer Slide Time: 32:58)

**Intuition**

- Assume hash values are uniformly distributed
- The probability that a uniform bit-string
  - is divisible by 2 is  $\frac{1}{2}$
  - is divisible by 4 is  $\frac{1}{4}$
  - ...
  - is divisible by  $2^k$  is  $\frac{1}{2^k}$
- We don't expect any of them to be divisible by  $2^{\log_2(n)+1}$

Handwritten notes on slide:  
– is divisible by 2 is  $\frac{1}{2}$  (with "10" written above)  
– is divisible by 4 is  $\frac{1}{4}$  (with "00" written below)  
– ...  
– is divisible by  $2^k$  is  $\frac{1}{2^k}$  (with "0...000" written below and "k ~ log<sub>2</sub>(n+1)" written to the right)  
– We don't expect any of them to be divisible by  $2^{\log_2(n)+1}$  (with  $\frac{1}{2^{k+1}} \approx \frac{1}{2^k} \cdot \frac{1}{2}$  written to the right)

So, what is intuition?

So, will do the intuition? In this in this in this lecture and we will postpone the actual prove to the next lecture for this. So, the intuition for this is as is very simple. It says suppose assumed to the hash values are uniformly distributed among these 2 to the 1 possible bit streams. That is for every element right it gets a uniform at random chosen ha a bits stream out of this 2 to the 1 possible bits streams.

So, because it is a uniform bits stream, the probability for a fixed element the probability that the bit stream is divisible by 2 is half. Because about half the bit streams bit streams are divisible by 2. About one-fourth of the bits streams are divisible by 4. About 1 by 2 to the k of the bit streams are divisible by 2 to the k. So, what does it mean for a, but stream to be divisible by 2 only.

(Refer Slide Time: 34:01)

Formalizing intuition

- $S$  = set of elements that appeared in stream
- For any  $r \in [\ell], j \in U$ ,  $X_{r,j}$  = indicator of  $\text{zeros}(h(j)) \geq r$
- $Y_r$  = number of  $j \in U$  such that  $\text{zeros}(h(j)) \geq r$

$$Y_r = \sum_{j \in S} X_{r,j}$$

- Let  $\hat{z}$  be final value of  $z$  after algo has seen all data

IIT Gandhinagar  
Indian Institute of Technology Gandhinagar

NPTEL ONLINE  
CERTIFICATION COURSES

Page 10/24

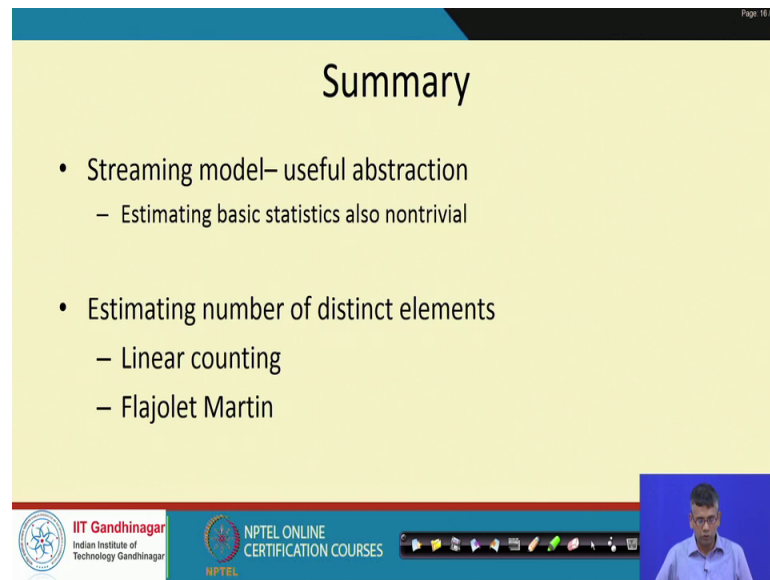
It means that it has, if a bit stream is divisible only by 2, it means that it has somewhere it has one, and then a 0 right. It has at least one 0 at the end for bit stream is divisible by 4 it is at least 2 zeros at the end. If the bit stream is divisible by 2 to the k it is at least k zeros at end ok.

So, then see then; that means, that the position of the right most this say something position of the right most one right. We will get to the get to I mean making this precise in our next lecture and also if you if you take if you take this k to be something like log base 2 of n plus 1 right. The probability that a uniform chosen bit stream is divisible by is divisible by 2 to the k for this value of k, is really 1 by 2 to the power log base 2 of n plus 1 which is really 1 by n square right. Not base 1 by sorry it is 1 by 2 n it is 1 by 2 n not 1 by n square and; that means, that what you then say is that if we do not really expect any of them to divisible by 2 to the log n plus 1. Ok because that probability gets to be really small.

So, therefore, what it will mean is that that by tracking what is the maximum i such that 2 to the I divided it is v we can hope to get some estimate of how many distinct numbers we have, we have sort of input it to our in to our stream of how many distinct numbers, we have we have used in our stream.

So, let us sort of n now and in next class we will we will formulize this proof of the Flajolet Martin.

(Refer Slide Time: 36:24)



Page 10/20

## Summary

- Streaming model– useful abstraction
  - Estimating basic statistics also nontrivial
- Estimating number of distinct elements
  - Linear counting
  - Flajolet Martin

IIT Gandhinagar  
Indian Institute of  
Technology Gandhinagar

NPTEL ONLINE  
CERTIFICATION COURSES

NPTEL

So, just to summarize: in this class we introduce the streaming model to as useful extraction of when we cannot store the entire dataset into memory. We also got to the question of estimating basic statistics of our data. For instance, even the number of distinct counts. And you say that even that is really non-trivial for instance even sampling based algorithms really work simple sampling these algorithms. We saw at least one algorithm that that does the linear counting that uses order that.

However, uses order  $n$  space right. And now, we saw another very sort of magic like algorithm, by Flajolet Martin for which we have got in intuition we have not got into a actual proof of that, yet that is up will do in the next class.

Thank you.