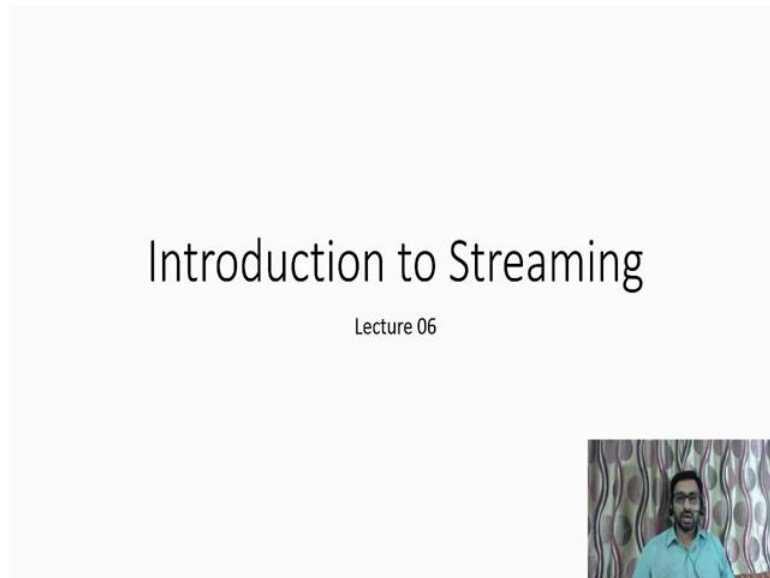


Algorithms for Big Data
Prof. John Ebenezer Augustine
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

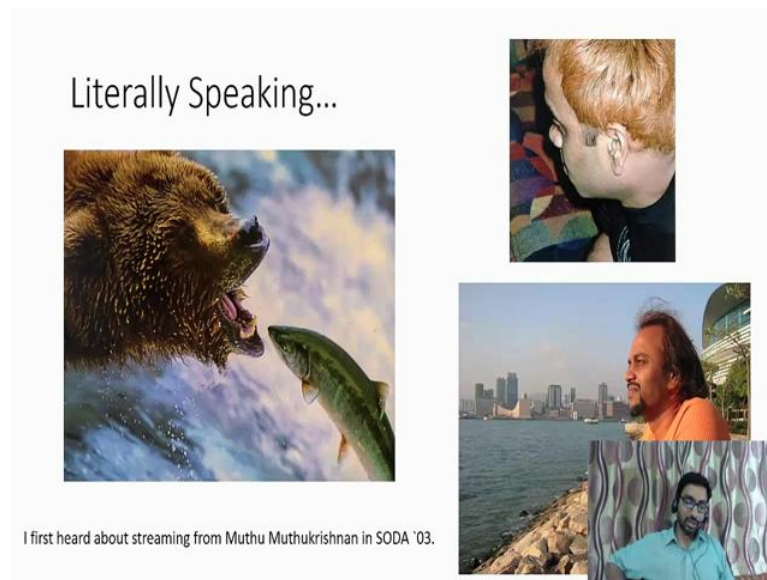
Lecture – 20
Introduction to Streaming

(Refer Slide Time: 00:20)



In today's lecture, we will talk about streaming. And many of you have already been introduced to streaming in our project discussions, some of you have presented problems based on streaming so as soon this is not very new to you. So, large (Refer Time: 00:29) jump right into the topic, but I thought talk little bit about the first time I came to know about streaming, this was several years ago to be precise.

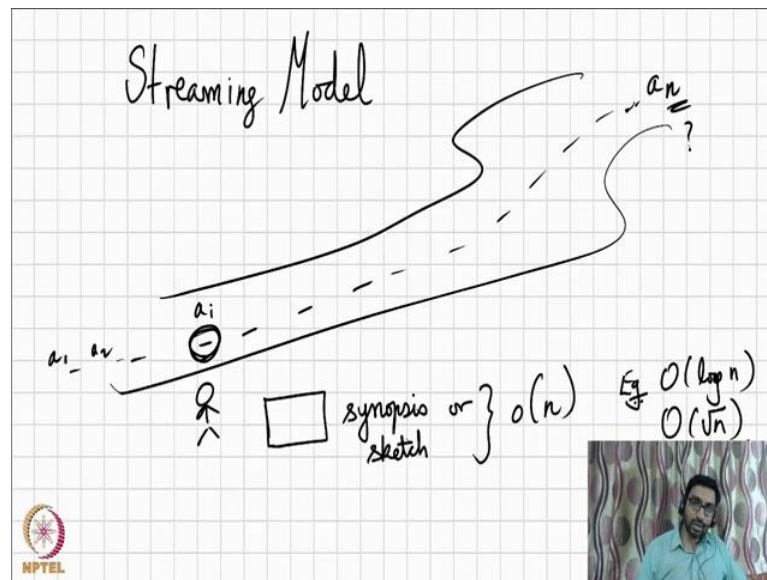
(Refer Slide Time: 00:45)



It was actually SODA 2003, and this notion of streaming the topic of streaming was introduced by the plenary speaker of SODA 2003 was Muthu Muthukrishnan pictured here. A very interesting personality, I believe I have not really met him face to face, but I have seen him several times, you know excellent speaker, and one characteristic about him is a every time I see him, he has such a different hair style, different hair color, so anyway.

This is how he introduced the streaming you are standing; you know on a stream or by a stream and you see the different types of fish go by. And he introduced the notion of streaming as you know you see all this fish going by and you have some questions in your mind about the fish. For a (Refer Time: 01:49), you cannot catch and store all the fish. So, you have to the let the fish go by, and then you at the end of the day after long day of you know fishing, you have some questions about the type of fish that you saw the biggest fish, the smallest fish, the average size and you know that is just give the started then many, many such questions. And so this is the kind of colorful perspective of which I was introduce to streaming.

(Refer Slide Time: 02:28)



So, without further do, let us get started let me introducing the streaming model. So, the data flows as a stream and you get to see data one, one item at a time and here you are looking at the data. And each time you get to see one data item and let say I give (Refer Time: 02:50), so let say the data item start from a 1, a 2 you currently looking at a i and so on, and the data string keeps coming in at every time step. And you do not get to stored the entire data at each point in time; you can do some temporary computation which we do not really care about how much computation we do.

But the essential thing we do is that we have some sort of a synopsis or a sketch that we maintain. And typically this synopsis or sketch is going to be much smaller than the size of the streams. So, we do not assume that we know the size of the stream, so this quantity is an unknown. And however, the synopsis itself as assume to be very, very small and typically little o of n of n small as o of $\log n$ o of square of n and in fact, we are going to see one example were we only use of o of $\log n$ (Refer Time: 04:10). So, with this let us jump into like a problem.

(Refer Slide Time: 04:32)

We don't know the number of items n
Need to count the number of items
counter of $O(\log n)$ bits
 $\Pr(\hat{n} \in [n - \epsilon n, n + \epsilon n]) \geq 1 - \delta$
 (ϵ, δ) - approximation

Let us say that we are getting a steady stream of items, then we do not even know how many items are there in the stream, but we want to count the number of items, and this is of course, quite easy because all we need is a counter of $O(\log n)$ bits. And in fact, with counter of $O(\log n)$ bits, we can count n accurately. The challenge is what happens in the situation where the number n is very large and you want to count using very few numbers of bits, and this could happen in cases like embedded systems.

Let us see a lot of data, but there design to be working with such resource constraint context that memory is at a premium. And so in situations like this, it is interesting that we can actually get a rough estimate of the number of items with far fewer numbers of bits. In fact, what we are going to see now is a way to estimate n within an epsilon fraction.

So, in other words, we are going to get an estimate \hat{n} that is going to belong to n minus epsilon n and n plus epsilon n . We are going to be able to give this guaranty with probability at least one minus delta, and such an approximation is called an epsilon delta approximation. And believe it or not this problem was actually studied way back in the seventies; and in that in some senses it not very surprising because memory was a lot more expensive in back time, so it was important to design algorithms of this nature even back time.

But I do not think the author realize how the important and fundamental these ideas are going to be, and how they are going to lead to very interesting work starting from the late nineties and two thousands and onward. So, this the algorithms we are going to study was introduced by Morris and later it was analyzed by (Refer Time: 07:02) in the mid 80s.

(Refer Slide Time: 07:09)

Morris Algorithm

$X \leftarrow 0$

When we see a new item

With probability 2^{-X}

$X \leftarrow X + 1$

Output $2^X - 1$

$\frac{1}{1} \quad \frac{1}{2} \quad \frac{1}{3} \quad \frac{1}{4} \quad \dots \quad \frac{1}{8} \quad \frac{1}{9} \quad \frac{1}{16}$

NPTEL

So, here is how this algorithm goes. We initialize a variable x to 0; and each time we see a data item, we update x with some probability. So, in particular, whenever we see a new item, we flip a biased point; and with bias 2 to the minus x , remember x is value that is going to be incremented over time; if this biased point comes up to the heads and that is going to happen the probability 2 to the minus x then we increment this quantity x . At the end of the stream, we come to the very end of the stream; now we need to know how many items we have seen we simply output 2 to the x minus 1. And why this is even correct, and why what is the inclusion behind the algorithm, you may wonder. So, what is going on is the following.

We start off with x equals to 0; first time, we see an item or x is going to be a 1. And then subsequently with probability have roughly speaking, you are going to increment in the next couple of occasions, so that would lead us to so this is items 1, 2, 3, 4, 8, now as you can see when we see the second and third item with probability half intuitively speaking, we would increment.

So, by time we come to 4, we would have probably reached x value could have reached 2 to 4 up to 7, you would be incrementing with probability 2^{-2} , so at one-fourth probability. So, between these ranges, you expect to increment by 1; and then between 8 and 15 you again hope to increment by 1; and from 16 to 31 you will increment by 1 on expectation and so on and so forth. So, every time the numbers double, you increment ones, this the rate of which we increment is algorithmic and number of bits we need to store the counter is only $\log \log n$, so that is the intuition, but let us look at the a little bit more carefully and more formally.

(Refer Slide Time: 09:57)

Claim: $E[2^x] = n+1$

Let X_n be the value of x after seeing item n

$$E[2^x] = \sum_{i=1}^{\infty} \Pr(X_{n-1}=i)$$

$$E[2^{X_n} | X_{n-1}=i]$$

$$= \sum_{i=1}^{\infty} \Pr(X_{n-1}=i) \cdot \left(\frac{1}{2^i} \cdot 2^{i+1} + \left(1 - \frac{1}{2^i}\right) 2^i \right)$$

MU, Pg 27

So, the claim, we would like to prove is that the expectation of the random variable 2^x is $n+1$, and this is why we output 2^x minus 1, so we output n . So, the question is why this claim is true. So, let X_n be the value of x after seeing item n .

So, now, the expectation of 2^x can be written in this way it is the summation of i going from one to infinity probability that X_{n-1} equal to i times now given that X_{n-1} equal to i , you want the expectation of 2^{X_n} . So, in case you are wondering where this comes from, you can refer to (Refer Time: 11:14) page number 27 that will be a good reference for you to verify this formula. So, now we have the summation or the overall i the probabilities of x and minus 1 times, and now we need to

expand out the expectations the conditional expectations that we have here. Of course, this 2 to the x_n , remember we were conditioning on $x_n - 1$ equal to i .

(Refer Slide Time: 12:02)

Morris Algorithm

$X \leftarrow 0$

When we see a new item

With probability 2^{-X}

$X \leftarrow X + 1$

Output $2^X - 1$

$\frac{1}{1} \frac{1}{2} \frac{1}{3} \frac{1}{4} \dots \frac{1}{8} \frac{1}{9} \frac{1}{16}$

NPTEL

And if you know that $x_n - 1$ equal to i , then from the algorithm we know that we are going to increment with probability 2 to the minus i , and stay the same with probability 1 minus 2 to the minus i which is exactly what is given here in this formula.

(Refer Slide Time: 12:18)

Claim: $E[2^X] = n + 1$

Let X_n be the value of X after seeing item n

$E[2^X] = \sum_{i=1}^{\infty} \Pr(X_{n-1}=i)$

$E[2^{X_n} | X_{n-1}=i]$ MU, Pg 27

$= \sum_{i=1}^{\infty} \Pr(X_{n-1}=i) \cdot \left(\frac{1}{2^i} \cdot 2^{i+1} + \left(1 - \frac{1}{2^i}\right) 2^i \right)$

$= \sum_{i=1}^{\infty} \Pr(X_{n-1}=i) \cdot \left(2 + \frac{2^i - 1}{2^i} \cdot 2^i \right)$

$= \sum_{i=1}^{\infty} \Pr(X_{n-1}=i) \cdot (1 + 2^i) = n + 1$

NPTEL



So, with one or getting my i 's and j 's mixed up, so lets us call this i 's. So with probability 1 over 2 to the i , we have we are going to increment. So, 2 to the x_n will become 2 to i

plus 1. And with remain probability to stay 2 to the i. So, let us expand this out; and with a few cancellations, we are going to get the submission of the probabilities times 1 plus 2 raise to the i, when we take the submission inside the (Refer Time: 13:02) and this is going to be two submissions.

(Refer Slide Time: 13:08)

$$\begin{aligned}
 &= \sum_{i=1}^{\infty} \Pr(X_{n-1}=i) + \sum_{i=1}^{\infty} \Pr(X_{n-1}=i) 2^i \\
 &= 1 + E[2^{X_{n-1}}] \\
 &= 1 + 1 + E[2^{X_{n-2}}] \\
 &= n-1 + E[2^{X_1}] = \underline{\underline{n+1}}
 \end{aligned}$$

Exercise: Show that $E[2^{2^x}] = E[2^{x^2}]$
 $\Rightarrow \text{Var}[2^x] = \theta(n^2)$ (using same tech)

The first submission is going to be 1, because it is submission over all possible probabilities or the probability space. And the other submission of this one let us look at this bit carefully so this is going to be nothing but the expectation of 2 to the x n minus 1, so this is nothing but 1 plus the expectation of 2 to the x n minus 1. And so now, what we have is the current set we can expand out so this is going to be 1 plus 1 plus E to the 2 and this can be expanded out and we finally, get n minus 1 plus the expectation of 2 the x 1, but x one is always 1. And therefore, the expectation of 2 to the 1 is going to be just 2 and that is going to evaluate to n plus 1 which is exactly what we wanted.

So, here is a quick exercise for you. This is just to give you some context of exercises design to move us towards on understanding of the variance of this random variable x. We want to show that the expectation 2 to the 2 x, which is essentially the expectation of 2 to the x square, goes up to theta of n square. And so you can show this by using the same technique to be used to show the expectation of 2 to the x. And the implication of this exercise is that the variance of 2 to the x is equal to theta of n square.

And when the variance theta of n square, the standard deviation is of the order of n; and if you work out the math, this is not going to be reasonable approach to get a good epsilon delta approximation. So, we need to be a bit more careful here. So, this a good starting point, but in order to get a good epsilon delta approximation, what we are going to do is not just have one estimate, but several estimates, and then take the average. And we will see that by doing that we can bring down the standard deviation of the estimate to a point where we get the appropriate epsilon delta approximation.

(Refer Slide Time: 16:23)

Algorithm Morris++
 $t = \Theta\left(\frac{1}{\epsilon^2 \delta}\right)$
 $X^1 = X^2 = \dots = X^t = 0$
 When each item arrives, execute independently $\forall 1 \leq i \leq t$
 With prob 2^{-X^i} , $X^i = X^i + 1$
 When stream ends, return $1 + \frac{1}{t} \sum_{i=1}^t X^i$

Someone call this algorithm Morris plus plus, because it is a one step improvement over the classic Morris algorithm. In this, we have a parameter t and that is parameter going to be theta of 1 over epsilon square delta. The exact constants can be worked out in the analysis. So, here is how we are going to this, so we are going to maintain t variables, we going to call them X 1, X 2 and so on up to X t, and we are going to initialize all of them to 0s. So, this is essentially implementing Morris algorithm in parallel. As each item arrives, we execute the following line in parallel for each value of i ranging from 1 to t.

And we do this independently. So, when for any given value of i, with probability 2 raise to the minus X i, so it only uses the random the variable X i. With probability 2 to the minus X i, we increment X i, so these are the variables X 1, X 2 up to X t are manipulated completely independent of each other. But at the end when the stream ends

we simply output the average of the estimates. So, when the stream ends, we simply return 1 plus t average of 2 to the X i.

(Refer Slide Time: 18:39)

$$Z = \frac{1}{t} \sum_{i=1}^t 2^{X_i}$$

$$E[Z] = n+1$$

$$\text{Var}[Z] = \text{Var}\left(\frac{1}{t} \sum_{i=1}^t 2^{X_i}\right) = \frac{1}{t^2} \text{Var}\left(\sum_{i=1}^t 2^{X_i}\right)$$

$$= \frac{1}{t} \text{Var}(2^X) = O\left(\frac{n^2}{t}\right)$$

So, let us analyze this Morris plus plus algorithm. Let us use z to denote the average value of 2 to the x i which is essentially 1 over t submission 1 to t 2 to the X i. So, first question is what is the expectation of this z. Well, the expectation of each of the 2 to the X i, we know is n plus one and you sum them up and divide by t so that is essentially going to be n plus 1, so there is no change there. So going to get a good estimate with a correct expectation, but hopefully the standard deviation is improved. So let us see how we can show that. So for that, we will have to look at the variance of z. So, we first simply expand out z, so this is what shows up here.

Now of course, we can bring the t outside the variants and that gives this square when we bring out at a constant term outside of the variants. And more over the variance of each of the 2 to the x i is equal to each other. And so this is the summation is really summation over equal variance terms so that and it summed from i equal to 1 to t, so that t can be brought out as well so it becomes 1 by t variance of 2 to the x. And of course, we know the variant of 2 to the x is sum theta n square, you should have worked it out as an exercise. So, if that is a case then this variant becomes theta of n square divided by t, so as we increase t we can get better variance.

(Refer Slide Time: 20:58)

$$E[Z] = n+1$$
$$\Pr(|Z - n - 1| > \epsilon n) \leq \frac{\text{Var}(Z)}{\epsilon^2 n^2} \quad (\text{By Chebyshev})$$
$$= \Theta\left(\frac{n^2}{t}\right) \cdot \frac{1}{\epsilon^2 n^2}$$
$$\hookrightarrow \Theta\left(\frac{1}{\epsilon^2 \delta}\right)$$
$$= \delta$$

Memory Requirement is $\Theta\left(\frac{\log \log n}{\epsilon^2 \delta}\right)$

Exercise: $\Theta\left(\frac{\log 1/\delta \log \log n}{\epsilon^2}\right)?$

So, now assuming it is sufficiently large look at the probability that is random variable z is far away from n . So, in particular z minus n minus 1, remember at the expectation z is equal to n plus 1, so the probability that z minus the expectation of z is greater than ϵn is at most the variance of z over ϵ square n square. And of course, this is come from by Chebyshev's inequality. So, applying our knowledge of the variance of z , we get θ of n square over t and times 1 over ϵ square n square.

So, now the question is well we have we know the t is θ of 1 over ϵ square δ for suitable constants. So, if we substitute that some other terms are going to get canceled out and this probability can be brought down to δ for a suitable choice of constants. So, just to conclude this, each execution of the Morris algorithm will require $\log \log n$ bits, but we repeated t times where t is state of 1 over ϵ square δ , so our total memory is θ of $\log \log n$ divided by ϵ square δ . So, now, here is an exercise for you.

Extend this our Morris plus plus algorithm in order to achieve memory requirement of only $\log 1$ over δ times $\log \log n$ divided by ϵ square. So, instead of essentially instead of 1 over δ term, we are going to have $\log 1$ over δ , so that is the pre significant improvement. As I hint, please note that you will have to use the (Refer Time: 23:19) of bound to achieve this bound.