

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

Lecture - 10
Frequent Elements: Space Saving and Count Min

Hello everybody. Welcome to the Scalable Data Science course. Today's lecture is also about Frequent Elements and will do 2 algorithms today; one on space saving and the other on count min sketch. My name is Anirban, and I am faculty at computer science in engineering in IIT Gandhinagar. So, before we start just a brief review of the streaming model that we have seen that we are dealing with the only the arrival only streams. And the elements of the stream could be element ids. We have only really dealt with I mean examples at I have shown you really about element ids.

(Refer Slide Time: 00:53)

Page 7/15

Streaming model revisited

- Data is seen as incoming sequence
 - can be just element-ids, or (id, frequency update) tuple
- Arrival only streams
- Arrival + departure
 - Negative updates to frequencies possible
 - Can represent fluctuating quantities, e.g.

IIT Gandhinagar
Indian Institute of
Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

NPTEL

But it could also totally work with the; I mean when you have the id and positive frequency updates. Because he could always imagine the positive frequency update as multiple copies of an item, let us coming in a stream, ok. Today will sort of talk a little bit on, but arrival only streams, but then also talk about I mean some data structures that work with arrival plus departure models, ok.

(Refer Slide Time: 01:21)

Review: Frequency Estimation in one pass

- Given input stream, length m , want a sketch that can answer frequency queries at the end
 - For give item x , return an estimate of the frequency
- Deterministic algorithm by Misra and Gries $\epsilon \sim 1\%$
 - f_x = original frequency of item x . Return \hat{f}_x such that
$$f_x - \epsilon m \leq \hat{f}_x \leq f_x$$
 - Space = $O(\frac{1}{\epsilon} \log n)$ $\frac{2}{\epsilon} \log n$

IIT Gandhinagar Indian Institute of Technology Gandhinagar NPTEL ONLINE CERTIFICATION COURSES

And in last class with it this deterministic we tackle the question of frequency estimation on pass, right. That given a particular stream that is of length m , we want, I mean I should use this words sketch a little carefully. Because today we are going to distinguish between what we call as sketch, and what is an little more precisely may be right.

So, we want a data structure, let see and very loosely we are calling it a sketch that can answer frequency queries at the end. Which means that if the frequency of a particular item is f_x , I want to written some estimate of it, ok. And this is a kind of guarantee that we got, that if f_x is original frequency of an item x . And you have user defined error parameter epsilon, right which presumably something likes fairly small let us say 1 percent as an example. And then you can written an estimate \hat{f}_x of f_x for in a item x such that \hat{f}_x lies in this interval. And the amount of space that you need is not even order it is actually exactly $\frac{1}{\epsilon} \log n$. I mean rather $\frac{2}{\epsilon} \log n$.

So, let us talk about very similar algorithm to what we have, ok.

(Refer Slide Time: 02:52)

Space Saving Algorithm

- Keep k counters and items in hand

Initialize:

- Set all counters to 0

Process(x)

- if x is same as any item in hand, increment its counter
- else if number of items $< k$, store x with counter = 1
- else replace item with smallest counter by x , increment counter

Query(q)

- If q is in hand return its counter, else 0

Handwritten note: $2k \log n$

IIT Gandhinagar
Indian Institute of Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

So, you may, I mean if you remember what are Misra Gries algorithm was it look very much like this, that we keep k counters, and items in hand you initialize by setting all counters to 0. And then when we try to process an item, right we say that the first 2 steps of the same, if x is same as any item in hand of course in incremented counter. If you do not know enough items in hand yet, if the number of items in hand is less than k of course, it is store x with counter 1.

But what we do if you have already have k items in hand, and next does not match any of them. So, what remember what the Misra Gries doing, but Misra Gries were do is that it could drop x this copy of x , right it would drop this item. And then it would decrease the counts of the existing items that you have, right. And this let to this charging argument that we have we had for giving the guarantee. The space saving algorithm right developed by (Refer Time: 03:56) others. They are something slightly different. What it does is that it does not through it does not really through away x , on the contrary, right it looks at the item that you have an hand, that has the smallest value of the counter, ok. It throws away that item replaces that item id with x and it increments a counter id, since it is strange and will through come soon example right, but remember what is doing.

So, it is not throwing away I mean it is not decreasing any I mean any counter all, right. It replaces the smallest counter id, with the id of x and it increments the counter, well incrementing the counter is initiative because your accounting for the arrival of x right,

but it does a mysterious, why are we assigning some other items count to that of x . And what happens when you query?

Well, that is fairly simple, if the if the q is in hand right if q is if your if the algorithm is tracking q of the end, then you return it is counter else return 0, ok. So, why is this a useful algorithm? So, first of all, it should also be fairly easy to see that the space that you have is again that the space that you need is again too long $2k \log n$, right which is fairly I mean depending on the value of k this could be fairly nice.

So, that is so, there is no doubt about that, ok.

(Refer Slide Time: 05:33)

The slide titled "Example" shows a sequence of 10 colored dots: red, blue, red, blue, red, blue, red, blue, red, blue. To the right of the dots, there are handwritten red notes. At the top right, it says "k=2". Below that, there are several lines of text: "B", "R", "T", "Y", and circled numbers 1, 2, 3, 4, 5, 6. The slide also features logos for IIT Gandhinagar and NPTEL ONLINE CERTIFICATION COURSES at the bottom.

Let us draw a small example. Suppose we dealing with k equal 2, ok. So, first we track red count 1, blue count 1 then we get 2 copies of red. So, this is easy this becomes 3, then make it turquoise, ok. So now, what happens? Now we drop the blue we adding the turquoise, then we increment the count of this, this becomes 2. Now you get a blue, right and now again because turquoise is the one with the minimum count, you delete the id of turquoise you make it blue, and then increment these to 3. Then we get 2 reds these becomes 5. Then we get a blue, right. Well, I am already tracking blue so, this is becomes a 4, then we get a turquoise. So, this is still a smallest element smallest counter so, this becomes turquoise and this is becomes 5, these becomes yellow, ok.

So now there is a tie, he could do either it is just do this one. This becomes yellow and this becomes 6. This becomes blue, this becomes blue, this becomes 6. So, that simply weird, because of at the end of the stream we can a predicting the blue has 6, and that yellow has 6, while yellow does not really have 6. But this will be in some error, I mean we will see able to prove some kind of error bound for this, ok.

But what is the count of blue 1, 2, 3, 4. So, blue is not so far of, but yellow is really far of, right. And for red we should have our estimate is 0. So, does look like is this algorithm many good, right. It certainly more complicated looks for complicated then the Misra Gries algorithm. So, let us right analyze it.

(Refer Slide Time: 07:33)

Page 13/20

Analysis

$f_x > \epsilon m$ $k = \frac{1}{\epsilon}$

Claim 1: All items with true count $> \epsilon m$ are present in hand at the end

Claim 2: For every element x , the estimate \hat{f}_x satisfies

$$\underline{f_x} \leq \hat{f}_x \leq \underline{f_x + \epsilon m}$$

We will prove only Claim 1.

IIT Gandhinagar
Indian Institute of Technology Gandhinagar

NPTEL ONLINE
CERTIFICATION COURSES

It turns out that you can get similar guarantees for this algorithm, right. And let us break down the claim that we have in to 2 claims in the 2, although claim I mean claim 2 also will imply claim 1, but, but let us break it down, ok. Suppose we say in the first claim and the one that will actually prove in this particular lecture, says that if an item supposing we set, oh supposing we set k equal to 1 by ϵ ; which means that we are storing 1 by ϵ count I mean item ids as well as well as their counters.

If the true count of an item, right which means that if x is such that f_x is strictly bigger than ϵm right, then we can guarantee then we can guarantee that these item such an x is present in hand, right. This is what will guarantee ok; that means that will track all

these heavy heaters. Furthermore what will track what will say is that at consider the estimate \hat{f} of x .

So, remember what the estimate \hat{f} of x says because if an item is interact at the end the estimate is f the counter value, if you not tracking the item the estimate \hat{f} of x is 0, right. So, the estimate \hat{a} of x will satisfy, \hat{f} of x now is at least f of x . And \hat{f} of x is at most f of x plus ϵm . So now, instead of now we have a lower bound by f of x and then upper bound by f of x plus ϵm , right.

So, at least in example that you saw, you are getting both the item. So, satisfying this lower bound right, and it is fairly easy to see why this should be the case, why \hat{f} of x is bigger than equal to f of x . So, let us argue for the first claim at least because even that is not obvious.

(Refer Slide Time: 09:46)

Page 14/21

Analysis

Claim 1: All items with true count $> \epsilon m$ are present in hand at the end

- Smallest counter value, \underline{min} , is at most ϵm ✓
 - Counters sum to m , by induction
 - $1/\epsilon$ counters, so average is ϵm , hence smallest is less
- True count of an uncounted item is between 0 and \underline{min}
 - Proof by induction, true initially, \underline{min} increases monotonically
 - Consider last time the item was dropped

$f_y \leq \min_t \leq \min \leq \epsilon m$

So, that tricks in the trick in analyzing this is in understanding how the minimum counter behaves. Because in some sense, I mean if you have noticed in the example, as items move in an out of the set of k , right the minimum counter acts as the storage. It kind of stores the it can stores all the updates that have been made to items that were once in a storage and that have moved out, the minimum counter source at value, ok. So, analyzing that is the critical part. So, as the first statement and that is fairly easy, what will say is that at the let me call the minimum countered \min , right. And let me say that the value of the minimum counter at the end is \min .

So, what will say is that this value \min is at most ϵ , it is not that big, ok. Why is that the case? So, here is a critical point, that if you sum up all the counters, right you get m exactly. So, this is very unlike Misra Gries because here we not doing any deletions, right? So if a counter if an element moves out of the set, right it deposits its value with the minimum counter, right. So, therefore, the total sum of all the counters is still m . And because it is the minimum, the average because we have $1/\epsilon$ counters as the average is m/ϵ and therefore, the minimum of these counters is at most ϵ , right.

So, then that seems fairly obvious, this is done, ok. So, the next is the more tricky statement. What will say is that suppose consider an uncounted item at the end, right that is considered an item that is not present in this list of items that you have at the end, right look that item be y . So, the algorithm is tracking this list of items right and y is not present there, ok.

So, what we can say what we can claim is that the true count of this uncounted item is within 0 and \min . Remember \min is this smallest of this counter values of these of these items they have been tracked this is what will be claimed. So, y is this the case. So, let us imagine the last time y was dropped. So, let us say at step t y was dropped ok. So, what is it mean? That means that in the steps from t till m y never appeared away even, right that in this region in the interval from t to m right y 's frequency is 0 , ok.

So, because if it had appeared right, then this if it had appeared in this interval, then this would not have been the last dropped, ok. So now, consider the value of the min counter at this step, let me denote by \min of t right. So, it could be so, first of all it is not hard to see that \min of t is less than equal to \min . Because the value of the min counter always increases, right. It never decreases at least, right because everybody that is that is sort of everybody whose living deposits itself an min counter. And so, it always completely, it always increases.

So, and now because y was dropped at this step, the frequency of y at this step t is less than that of the min counter, at this step. So, f of y which is the frequency until step t , but which is also equal to the total frequency of y because y does any occurrence in the

interval t to m . So, therefore, f of y is less than equal to $\min t$ which is less than equal to \min .

And we have argued that \min is less than equal to ϵm , right. Which is something I mean, but this is a later on stronger statement, that the true count of an uncounted item is within 0 and \min , right. And therefore, the true count of any uncounted item is less than less than equal ϵm . Therefore, if an item does our true count bigger than ϵm , it must be present in hand at the end, ok. So, this so you need to think about this proof a little bit, ok.

So, just do that. Basically the same proof can be sort of modify the little bit to proof a stronger claim, but we will not do that. Because it is little sort of it little complicated, ok.

(Refer Slide Time: 14:27)

The slide is titled "Counter based vs 'sketch' based" and is part of an NPTEL presentation from IIT Gandhinagar. It compares two types of data structure methods for counting elements in streams.

- Counter based methods
 - Misra-Gries, Space-Saving,
 - Work for arrival only streams
 - In practice somewhat more efficient: space, and especially update time
- Sketch based methods
 - "Sketch" is informally defined as a "compact" data structure that allows both inserts and deletes
 - Use hash functions to compute a linear transform of the input
 - Work naturally for arrivals + departure (CM needs modification)

The slide footer includes the IIT Gandhinagar logo, the text "IIT Gandhinagar Indian Institute of Technology Gandhinagar", the NPTEL logo, and "NPTEL ONLINE CERTIFICATION COURSES". A small video inset shows a man in a blue shirt speaking.

So, instead, but will do is move to other kinds of data structures. And here is where the definition of sketch comes in, right. See, the data structures that we have looked at until now, this Misra Gries the 2 algorithms for frequency saving Misra Gries and space saving these are known as counter based algorithms, right. Because see what is really happening is that we are sort of adding and dropping count I mean adding and deleting element ids from this set and we are just keeping counters.

So, that I mean this is more a matter of I mean nomenclature, but these are not known as sketches. People your find these refer to as counter based algorithms. In practice actually

for I mean they only really work for arrival only streams they proofs work only for arrival only streams. But in practice all arrival only streams there substantially more efficient in terms for space as well as the update time than the other set of more general algorithms that we look at in a little bit. So, if these are not sketches then what is the sketch? I mean again informally; a sketch is defined as a compact data structure that allows both inserts and deletes, ok. So, you should think of it as kind of more having some linearity kind of properties, right that you can both do updates you can do different, I mean, both positive and negative updates to the frequency of the of an item, right then we will call it as sketch, ok.

So, what will do in this sketches is that will somehow compute a linear transformation of the input. And we will show you over the; I mean we will caste even the sum in this the algorithms at we have that we seen in this form later. And because you can do both positive and negative updates, it naturally works for most of it naturally works for arrival and departure, right. Some of it will need a little bit of modification, but not major modification.

(Refer Slide Time: 16:30)

Page 16/23

Count-min sketch [Cormode Muthukr]

- Model input stream as a vector over U
 - f_x is the entry for dimension x
- Creates a small summary $w \times d$
- Use w hash functions, each maps $U \rightarrow [1, d]$

$(00 f_x \dots)$
 $\leftarrow u \text{ (id. 1)}$
 $\leftarrow \text{(id. + c)}$

w

IIT Gandhinagar
 Indian Institute of Technology Gandhinagar
 NPTEL ONLINE
 CERTIFICATION COURSES

So, the first sketch algorithm that will considered is known as the count min sketch. And this was developed by Cormode and Muthukrishnan. So, in order to so think about this sketch, you I mean again the right we have to think about this is to say that we have we have frequency. So, think of the inputs stream as a frequency vector, right.

So, think of it is a frequency vector for the dimension is the is u right, I mean the elements that have not appeared in the stream have frequency is 0, and the and the elements that have appeared have the frequency f_x , right. And then what we are trying to do is that, we get this frequency vector not in terms of I mean in terms of only the updates, in terms of only the updates to the frequency vector in; because we see the element ids with the frequency implicit sort of update 1 or we will see the element id, with the update plus sum plus c, right. Now the question is that given the stream of updates, can you answer queries about this the resulting frequency vector, ok. So, what count min sketched us is that, it sort of it creates a small summery and the summary has 2 parameters. It has a w parameter which has a d parameter, ok.

So, the summary will be outside w times d; there will be w hash functions. Each of them will map each element of the universe to one of d, right. So, the hash function one will map each element of the each element of the universe from 1 to d the hash function 2 will map from 1 to d and so on, right. So, how do we use this hash functions, ok.

(Refer Slide Time: 18:36)

Page 17/24

Count Min Sketch

Initialize

- Choose $h_1, \dots, h_w, A[w, d] \leftarrow 0$

Process(x, c):

- For each $i \in [w], A[i, h_i(x)] += c$

Query(q):

- Return $\min_{i \in [w]} A[i, h_i(x)]$

So, here is the here is the algorithms. So, in the same sort of frame work that we have been writing others streaming algorithms. That first we do the initialization. In initialization we choose h_1 to h_w . And at this point let us again assume that h_1 to h_w are chosen uniformly at random, right that they are there it is a completely random hash function.

Again I will point out to you that you will not really need completely random hash functions, I mean, something like $\log 1$ over Δ independences fine or something like a some constant some key independences is fine some value of k , k universality is fine. And you also initialized this A $w \times d$ matrix to be all 0's, ok. Now what do I do? When I get a process x , right; so, I could get the item id itself or the item id with some of did. You look at each other hash functions, right and then you try to see where this particular hash functions would put this element.

So, you look at the corresponding so, for 1 you look at 1, this is where hash function 1 say is that x should go so, you update this entry, right. Hash function 2 says that hash function 2 puts elements in this column so, update this entry. Hash function 3 puts elements in this column so, you update this entry, and you update each of them exactly by c right.

So, when you query so, what you do when a query? So, this is interesting, right because you have you have w estimates now. One for each of the hash functions. So, you can look at $A_i h_i$ of x , $A_1 h_1$ of x , $A_2 h_2$ of x and $A_w h_w$ of x . And all these A I mean here w is 3 so, all these 3 buckets. What do you do? Well, you could have considered many you could have calculated the average, you could have taken the median. The first thing will when analyze is what happens if it take them minimum, right. And why is that are interesting thing to do, ok. And that is why the name count min sketch comes from, ok.

(Refer Slide Time: 20:53)

Example

	h1	h2
Blue	2	1
Yellow	1	2
Red	1	3
Blue	3	2

So, let us again run a small example, that suppose we have this stream, and suppose I have written the hash functions out like this. So, each column represents hash function. So, h 1 maps blue to position 2's, let say let say this is position 1, this is position 2, this is position 3. H 1 maps blue to position 2, yellow to position 1, red to position 1, and blue inter caste position 3, and h 2 maps.

Similarly, so now, first we see a red right. So, if you first see a red, then we get so, h 1 maps red to position 1. So, I get one here, right then I see a blue sorry, and h to maps red is position 3 So, I may so, I get 1 count of there, right. Then I get a blue right so blue h 1 is mapped to position 2, and h 2 is map to position 1. Can I get another red, here, here, another red, here, here. Then turquoise; turquoise is position 3, position 2, position 3 and position 2. Then blue; blue is position 2, position 1, position 2, position 1, red again 3 1 3 red 1 3 and then blue is 2 1, 2 1 turquoise is 3 2, 3 2. Yellow is 1 2, 1 2 and then blue is 2 1, 2 1.

So now the count here is the resulting count here is equal to 6, the resulting count here is equal to 4 and so on. That is earlier counter is 4, that is earlier counter is 5 and so on it is clear, ok. So, so this is the data structure that you will have with the end this matrix A, A w d, right? And suppose we are given the query of the red right. So, I look at the position 1, I mean I get a count of 6 here, and I look at position 3 I get a count of 5 here. So, it

return the minimum of 5 and 6 which means that I returned 5 on query red. So, that is how the algorithm works.

(Refer Slide Time: 23:24)

Guarantees

Space = $O(wd) \log n$
 Update time = $O(w)$

Each item is mapped to one bucket per row

The slide features a 4x4 grid of blue squares. A red oval labeled $(x, +c)$ has three red arrows pointing to different buckets in the grid. The footer includes logos for IIT Gandhinagar and NPTEL ONLINE CERTIFICATION COURSES.

So, let us analyze this, why is this good? So, so first let us analyze the easy things, which is the run time the update time and the space. The space clearly is order $w d$, times when this $\log n$, right because you are keeping so many counters. What is the update time? Well, each hash function updates only one of the positions, ok. So, therefore, each hash I mean each element does the blue updates, and therefore, the update time is order w .

(Refer Slide Time: 24:03)

Guarantees

- $w = \frac{2}{\epsilon}$ (with ϵ labeled as error) $d = \log\left(\frac{1}{\delta}\right)$ (with δ labeled as confidence)

$Y_1 \dots Y_w$ be the w estimates, i.e. $Y_i = A[i, h_i(x)]$, $\hat{f}_x = \min_i Y_i$ (with $Y_i \geq f_x$ written above)

Each estimate \hat{f}_x always satisfies $\hat{f}_x \geq f_x$ (with "Consider arrival only now" written to the right)

The footer includes logos for IIT Gandhinagar and NPTEL ONLINE CERTIFICATION COURSES.

So, what are the guarantees?

So, let us set w to be $2/\epsilon$, and let d to be $\log(1/\delta)$. So, remember what ϵ what we intuitively want ϵ and δ to be? ϵ we want the error parameter and δ we want the confidence of being correct, rather $1 - \delta$ to be the confidence of the incorrect; the probability of being correct, ok.

So, δ is a error probability. So now, as we mentioned, right let me define I mean we have at the end on we have querying we really have w estimates, let me call these Y_1 to Y_w . So, Y_i is the has the content of the cell $A[h_i(x)]$ of x , ok. So, in the final estimate that we written is really the minimum of these w of this y_i , and what happens is the minimum. So, consider the arrival only model for now because we need to change the algorithm a little bit to consider a deletions. So, in the A in the arrival only model, it is not hard to say that each Y_i is actually bigger than equal to f of x , ok.

Why is this? Well, it fairly easy because when x comes every instant of x your updating each of the cells $A[h_i(x)]$. So, therefore, I mean there could be other updates to the cell, because of other elements as we saw. So, therefore, each y_i is at least f of x and say and therefore, the minimum of the y_i is is at least f of x which means \hat{f}_x is at least f of x .

So, that is nice, but what about the upper bound right, ok.

(Refer Slide Time: 25:56)

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

- $w = \frac{2}{\epsilon}$ $d = \log\left(\frac{1}{\delta}\right)$ (The fraction $\frac{2}{\epsilon}$ is circled in red)
- $Y_1 \dots Y_w$ be the w estimates, i.e. $Y_i = A[h_i(x)]$, $\hat{f}_x = \min_i Y_i$
- Each estimate \hat{f}_x always satisfies $\hat{f}_x \geq f_x$ (with the handwritten note "Consider arrival only now" to the right)
- $E[Y_i] = \sum_{y: h_i(y)=h_i(x)} f_y = f_x + \frac{\epsilon(m - f_x)}{2} \leq f_x + \frac{\epsilon m}{2}$ (The last part of the equation is underlined in red)

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

Let us now calculate what is the expectation of Y_i ok. So, Y_i is clearly a random variable. And the randomness comes from the choice of the hash function. So, what is the expectation of Y_i ? The expectation of Y_i is a sum over all y such that map in to that map in to this particular bucket, that is mapped in to this particular bucket by this particular hash function, and the frequencies of these Y_i is, right? Summation number all Y_i such that $h_i(y)$ equal to $h_i(x)$ sum and the sum is and then you sum f_y for this y is, ok. And it is not hard to say that this expectation is really I mean x is there in the set, right.

So, this expectation is x plus for any other for any other Y_i for any other Y what is the chance that it will be a what is a chance at it will be a map to, what is a chance at it will be map to exactly this bucket, right? Because the size because it is uniform random hash functions the chance that any other x ; I mean any other y maps in to the same bucket of x is really 1 over w , right; which is ϵ by 2 , and the total frequency of all the other Y_s is really m minus the frequency of x .

So, therefore, the total frequency the expectation here of are all other y of are all y is not equal to x is the total frequency times the probability of any one of them mapping in to this bucket, right; which gives you ϵ by 2 times m minus f_x , right and this will right simply as f_x plus ϵ m by 2 , ok.

(Refer Slide Time: 28:08)

Page 18/38

Guarantees

- $\delta = \frac{2}{\epsilon}$ $W = \log\left(\frac{1}{\delta}\right)$

$Y_1 \dots Y_w$ be the w estimates, i.e. $Y_i = A[i, h_i(x)]$, $\hat{f}_x = \min_i Y_i$


Each estimate \hat{f}_x always satisfies $\hat{f}_x \geq f_x$

$$E[Y_i] = \sum_{y: h_i(y)=h_i(x)} f_y = f_x + \epsilon(m - f_x)/2$$


$P[h_i(y)=h_i(x)] = 1/d = \frac{2}{\epsilon}$

Applying Markov's inequality,


$$\Pr[Y_i - f_x > \epsilon m] \leq \frac{\epsilon(m - f_x)}{2\epsilon m} \leq \frac{1}{2}$$



IIT Gandhinagar
Indian Institute of
Technology Gandhinagar



NPTEL ONLINE
CERTIFICATION COURSES



So, this is nice. So now, we can apply Markov's inequality right, because, so, consider the random variable y minus f_x , right Y_i minus f_x . So, first of all, we say that this random variable is a non-negative random variable, right because of our previous argument that Y_i is always greater than equal to f_x . So, we have a non-negative random variable, what is expectation of this random variable expectation of this random variable is exactly this quantity, right is exactly this quantity, because it is equal to because it is equal to the expectation of Y_i minus f_x . So, so then we have a random variable non-negative random variable with this expectation therefore, the probability that this random variable will exceed the value ϵm , right is expectation divided by ϵm , which is at most half

So, **that is imply** algebra? So, therefore, the probability that each any one of the Y is crosses f_x plus ϵm is at most half and we are almost home.

(Refer Slide Time: 29:23)

Guarantee

- Since we are taking minimum of $\log\left(\frac{1}{\delta}\right)$ such random variables,

$$\Pr[\hat{f}_x > f_x + \epsilon m] \leq 2^{-\log\left(\frac{1}{\delta}\right)} \leq \delta$$

- Hence, with probability $1 - \delta$, for any query x

$$f_x \leq \hat{f}_x \leq f_x + \epsilon m$$

Page 27/30

IIT Gandhinagar Indian Institute of Technology Gandhinagar NPTEL ONLINE CERTIFICATION COURSES

Now, what are we doing? We are taking the minimum of $\log 1$ over δ of such random variables, right. Because we are taking the minimum of $\log 1$ over δ of such random variables; therefore, the chance that the minimum will cross if the minimum if the probability of any one of them being bigger than if f_x plus ϵm is half, if the minimum crosses if f_x plus ϵm ; that means, at all of them are bigger than f_x plus ϵm .

And the probability of that is $2^{-\log(1/\delta)}$ which is less than δ , right. So, therefore, the probability that the minimum of the Y is crosses $f(x) + \epsilon/m$ is less than δ . And hence with probability $1 - \delta$ for any query for any query x $f(x)$ lies in this interval $f(x) \leq \hat{f}(x) \leq f(x) + \epsilon/m$, ok. So, let me go back a little bit, because I might have made I think we made one small mistake, ok.

So, in here assume might already have noticed, the number of hash functions we are using w for the number of hash functions and w is equal to $1/\delta$, and d is equal to $2/\epsilon$. So, the rest of the argument will this follows, right the rest of the argument will this follows because the probability that $h_i(y) = h_i(x)$, now is $1/d$, right which is equal to $\epsilon/2$. So, we are just mixed up here, the notation between the w and the d , that now that is fixed, ok.

(Refer Slide Time: 31:7)

Page 19/37

Summary

- Two algorithms for frequency estimation
 - Counter based: Space Saving
 - Sketch based: Count-Min
 - CM sketch can also work for deletions if we change the minimum to median
- Guiding principle: use error bounds as design parameters of the data structure
- More to come...

IIT Gandhinagar Indian Institute of Technology Gandhinagar
 NPTEL ONLINE CERTIFICATION COURSES
 NPTEL

So, just to summarize in this lecture we looked it 2 algorithms of frequency estimation, one was counter base that was space saving. And the other one was sketch base that was count min.

So, while I did promise you, I mean sketch base algorithms at work with both updates and with both additions and deletions. As you mind imagine I mean as you might have noted the count min proof does not completely work for deletions, right. And that is because we are taking the minimum. And so, we were saying that each Y_i has to be I mean has to be bigger than equal to $f(x)$. But that will not be true if you are doing

deletions, because Y_i could be deleted and so, Y_i could be as small as $\frac{1}{n}$; however, you can easily fix this, you can easily fix this were taking the median instead of the min, right. And that you should try to work out yourself. If not you can look at the lecture notes that I pointed to at the end.

So, the entire algorithm for CM works, the only thing that you need to change is that you when you return the query in when you are answering the query you take the median of the estimate instead of the minimum. So, broad guiding principle that I would like to take is that you should that the way you design the sketches is that first you come up with error bounds, and confidence estimates and you use these as design parameters so the data structure. And this principle will come up over and over in the later parts of the course. So,

Thank you.