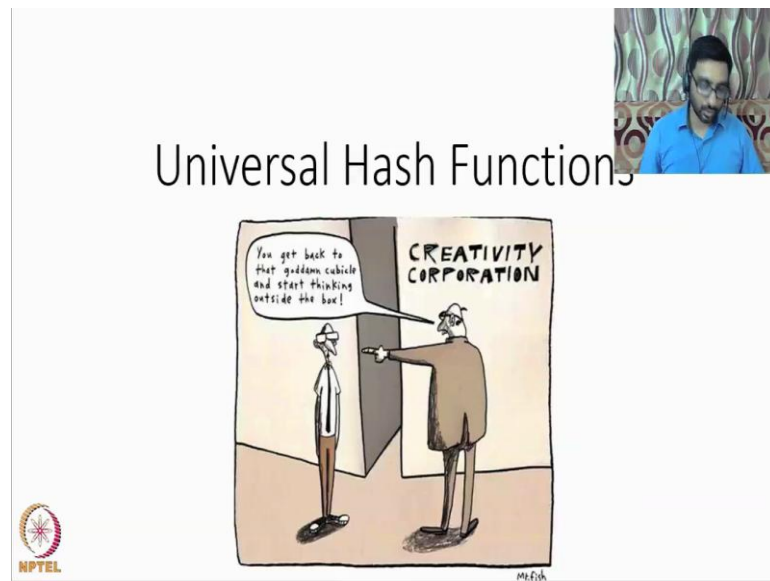


Algorithms for Big Data
Prof. John Ebenezer Augustine
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture - 29
Universal Hash Functions

(Refer Slide Time: 00:13)



Today, we are going to talk about Universal Hashing. Before we define universal hashing, let us just recollect, what we have looked at so far.

(Refer Slide Time: 00:23)

Recap

Balls in bins thrown independently and V.A.R.

Simple Uniform Hashing Assumption (SUHA)

Bloom Filters


Independence is expensive! k -wise independence
(i) Pairwise independence much cheaper (ii) Cheby cheb's words

Good Hash functions are particularly expensive

Uniform & independent

Deterministic behaviour

↑ expensive



We have looked at the balls in bins framework and we analyzed balls in bins using under the assumption that each ball is thrown into a bin that is chosen uniformly at random, and the choice of bins are mutually independent of each other and we used this balls in bins framework to analyze hashing, and we typically made a simple uniform hashing assumption. We assumed that when an item gets hashed its place in location that is chosen uniformly at random from all the possible hash locations and using this assumption we can invoke the ideas from the balls in bins framework.

For example, when we look at chain hashing, the worst case performance of chain hashing co-relates to the height of the most loaded bin in the balls in bins context. We also looked at independence in the last lecture, in particular we noticed that independence is somewhat expensive, but if we relax a notion of independence from mutual independence to some k -wise independence then it is cheaper. In particular, we looked at ways to generate pairwise independent random bits and pair wise independent random numbers.

We also noticed that while limiting ourselves to pairwise independence we did not lose out too much. We were still able to employ, for example, the Chebyshev's inequality. We also observed in the past that hash functions can be expensive and difficult to analyze

and hard to implement and so on and so forth and the problem comes from the fact that hash functions have to balance two requirements. On the one hand, hash functions should behave randomly they should have could randomness properties.

For example, when you hash an item it must be equally likely to be hash into any one of the locations and also we would like these hashed values to be mutually independent of each other. So, that is could randomness properties, but on the other hand when you hash an item it has to always hash in the exact same location, that is a very deterministic requirement because when you hash an item to store it and later on when you hash the item to search and receive it they must be in the same location, otherwise it is not going to work. So, these two requirements, the randomness requirement versus the deterministic requirement always brings about a bit of tension for us something that we need to resolve and here is an idea for how to resolve this tension.

(Refer Slide Time: 03:37)

How to Resolve the Tension ?

Single hash function h : good deterministic behaviour
but suffers on randomness and independence

FAMILY OF HASH FUNCTIONS

Choose one uniformly at random

NPTEL

Now, think about a single hash function. This is going to have good deterministic properties because it is a single function. So, anytime you hash an item is always going to go to the same location, but when we choose a single item it suffers on the randomness point because it is a single hash function. So, how do we alleviate this situation? Instead of focusing on the single hash function, let us consider a family of hash functions. So, we

have this whole family of hash functions and when we actually need a hash function we choose one, but we choose one uniformly at random from this entire family. So, in this manner, it is easier for us to bring in the randomness requirements for hash functions, but we need to be bit careful when we do this.

(Refer Slide Time: 04:41)

Today: Universal Family of Hash Functions

Definition: Consider a family of hash functions \mathcal{H} .

$$h: U \rightarrow V \quad \forall h \in \mathcal{H}$$

Universe \leftarrow \leftarrow $[0, 1, 2, \dots, n-1]$

It is a k -universal family if, for an $h \in \mathcal{H}$ chosen UAR from \mathcal{H} and for subsets $\{x_1, x_2, \dots, x_k\} \subseteq U$ of cardinality k .

$$\Pr[h(x_1) = h(x_2) = \dots = h(x_k)] \leq \frac{1}{n^{k-1}}$$

Let us try to be a bit formal and let us try to define such a family with the correct properties that we required. So, we are going to define, what is called a universal family of hash functions and so it is going to be such family \mathcal{h} and each member of the family is a hash function that will map universe to a set of locations, and such a family is going to be called a k universal family of hash functions.

If the following condition applies, now choose let say you choose a hash function h uniformly at random from this family and let us pick x_1, x_2 up to x_k which is a subset of the universe and this subset has to be of cardinality at most k , then we will require the probability that each of the items hash into the same locations should be at most 1 over n to the k minus 1 . If this condition holds then this family is called a k universal family of hash functions and remember this condition must hold for hash function h chosen uniformly at random. So, the randomness is based on this choice of hash function and it must hold for any subset of at most k items drawn from the universe U .

(Refer Slide Time: 06:23)

EXAMPLE:

$$U = \{0, 1, 2, \dots, m-1\}$$
$$V = \{0, 1, 2, \dots, n-1\}$$

$m \geq n$

$p \geq m$ is a prime number and 1

$$h_{a,b}(x) = ((ax + b) \bmod p) \bmod m$$
$$\mathcal{H} = \{h_{a,b} \mid 1 \leq a \leq p-1, 0 \leq b \leq p-1\}$$

2-Universal \rightarrow a cannot equal 0

I might think this is a hard requirement, but is actually quite easy to obtain. So, let us for example, focus on pair wise or rather 2-universal family of hash functions. So, here is a one way to produce such a family. Now, our universe is ranging from 0 to m minus 1 and the hash locations are ranging from 0 to n minus 1, and let us pick a prime number p that is larger than m and let us define a hash function h with parameters a and b . This hash function will take an x , x is in the set b will take an item x and hash into the location x plus b , first you mod a with p and then you mod with m . Well, that is just a single hash function, but remember we have parameters a and b .

So, with by extending our choice of values for a and b ranging from 1 to p minus 1 and b ranging from 0 to p minus 1. We will get not just one hash function, but a family of hash functions and this family of hash functions is 2-universal of family of hash functions and we are going to skip to prove, but its suffices for us to know that this 2-universal family of hash functions and as you can obviously, see the construction is quite straight forward. Let me just point out one minor technical detail, b ranges from 0 to p minus, but a only ranges from 1 to p minus 1. So, if we want to generate or pick hash function uniformly random, we pick at a uniformly random from 1 to p minus 1, b uniformly random from to 0 to p minus 1 and that immediately implies choice of our hash function h a comma b and that is our choice of hash function.

(Refer Slide Time: 08:45)


Today: ^{Strongly} Universal Family of Hash Functions

Definition: Consider a family of hash functions \mathcal{H} .

$$h: U \rightarrow V \quad \forall h \in \mathcal{H}$$

It is a ^{strongly} ^{universal} k -universal family if, for an $h \in \mathcal{H}$ chosen UAR from \mathcal{H} and \forall subsets $\{x_1, x_2, \dots, x_k\} \subseteq U$ of cardinality k .

and for any choice of $y_i \in \{0, 1, \dots, n-1\}$, $1 \leq i \leq k$

$$\Pr(h(x_1) = y_1) \cap (h(x_2) = y_2) \cap \dots \cap (h(x_k) = y_k) = \frac{1}{n^k}$$


We can extend our theory of universal hash functions to something this likely stronger. So, here we are going to define, what is called a strongly universal family of hash functions? It is again a family in which each member is a hash function that maps the universe to the set v and for this family to be a strongly k -universal family of hash functions. Here is a condition that must apply, now again as before let us consider one of the hash function, hash functions h chosen uniformly at random and let assume x_1 to x_k is a subset of the universe and this subset is more of cardinality k and now for any choice of y_i ranging from 0 to n minus 1 basically y_i belongs to p .

Now, the probability that x_1 is hashed into y_1 and x_2 is hashed into y_2 and so on up to x_k hash into y_k should be at most 1 over n to the k , that my apologies, it is not at most one over n to the k , it should be exactly equal to 1 over n to the k . This is the requirement of for strongly k -universal family of hash functions and quite obviously, this is a stronger definition.

(Refer Slide Time: 10:27)

Example:

$$U = \{0, 1, 2, \dots, p-1\}$$
$$V = \{0, 1, 2, \dots, p-1\}$$

p is prime

$$h_{a,b}(x) = (ax + b) \bmod p$$
$$\mathcal{H} = \{ h_{a,b} \mid 0 \leq a, b \leq p-1 \}$$

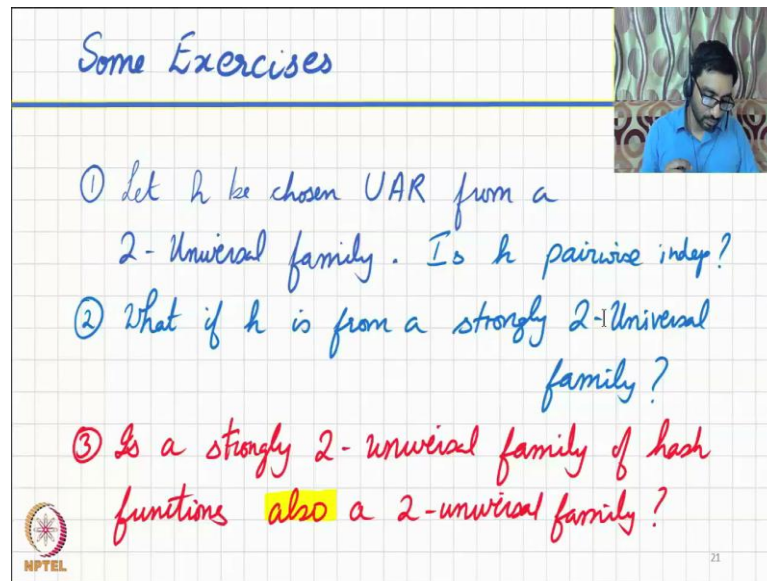
↳ Strongly 2-Universal

NPTEL 18

But the nice thing is such a family can still be constructed quite easily, especially if we all require is a strongly 2-universal family of hash functions. So, again how do we do that? Well, the universal it says ranging from 0 to p minus 1 and here p is a prime number and also let assume that v ranges from 0 to p minus 1. Again we define each individual hash functions based on parameters a , and b . So, $h_{a,b}(x)$ will take the item x and hash into the location $ax + b \bmod p$ and this family induced by the various choices of a and b .

We can prove is a strongly 2-universal family of hash functions. Again, we going to skip the proof, but it is obvious to see that this family can be easily constructed and well that mainly the family you need not be constructed the, we can draw hash function uniformly at random from this family very easily. So, all we need to do is draw 2 numbers, a and b from 0 to p minus 1 and that implies the choice of the hash function.

(Refer Slide Time: 11:45)



The slide is titled "Some Exercises" and contains three handwritten questions in blue and red ink. A small video inset in the top right shows a man with glasses and a blue shirt. The NPTEL logo is in the bottom left, and the number 21 is in the bottom right.

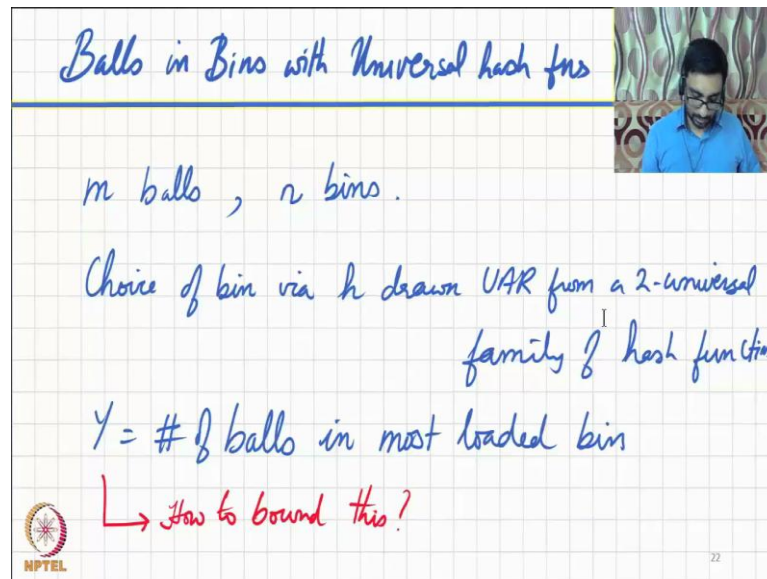
Some Exercises

- ① Let h be chosen UAR from a 2-Universal family. Is h pairwise indep?
- ② What if h is from a strongly 2-Universal family?
- ③ Is a strongly 2-universal family of hash functions also a 2-universal family?

So, before we proceed, let us ask ourselves some questions, I would like you to think about these questions and answer them for yourself, so that you have a sense of confidence about your understanding of this universal family of hash functions. Now, let us suppose, h is drawn uniformly at random from a 2-universal family of hash functions, my question is h pairwise independent, think about that and answer the same question.

Now, under the context of drawing h uniformly at random from a strongly 2-universal family of hash functions, is h now pairwise independent? What is going on here finally, is a strongly 2-universal family of hash function always also a 2-universal family of hash functions, think about these three questions and make sure your understanding of hash universal family of hash function is clear that way. Now, we defined this universal family of hash functions, how useful are they?

(Refer Slide Time: 13:13)




Balls in Bins with Universal hash fns

m balls, n bins.

Choice of bin via h drawn UAR from a 2-universal family of hash functions

$Y = \#$ of balls in most loaded bin

→ How to bound this?

 22

For example, our first concern would be their application in a very fundamental context, for example, the balls in bins context what happens when we replace the notion of balls choosing bins uniformly a random with the notion of balls choosing their bins based on some hash functions that is strong uniformly at random from, say 2-universal family of hash functions.

Let us think about that. So, we have m balls n bins, we use a hash function h to take each ball and hash into a particular location at particular bin and this h is drawn uniformly at random from 2-universal family of hash functions. So, what is our (Refer Time: 14:14) and now to understand that let us define random variable y and is defined as the number of balls in the max loaded bin and obviously, our question now is how, what is the reason of bound on the y ?

(Refer Slide Time: 14:32)

Bounding Y

What is the value of k for which:

$$\Pr[Y \geq k] \leq \frac{1}{2}$$

Conservatively we can bound k ensuring

$$\Pr[\# \text{ of colliding pairs} \geq \binom{k}{2}] \leq \frac{1}{2}$$

needs to be modeled

$\forall 1 \leq i < j \leq m$

$$X_{ij} = \begin{cases} 1 & \text{if ball } i \text{ and ball } j \text{ collide} \\ 0 & \text{otherwise} \end{cases}$$

$E[X_{ij}] \leq \frac{1}{m}$

Bin with k balls $\Rightarrow \binom{k}{2}$ pairs collide


NPTEL

In particular, we want to know what is the value of k for which the probability that y is at least k is bounded from above by a half and, in order to understand the probability of this event we are going to be conservative. Let us look at the bin with k balls, there must be at least k choose 2 pairs balls that collide just in that one bin. So, instead of bounding the probability that y will be at least k . We will be a bit more conservative and we will bound the probability that the number of colliding pairs is at least k choose 2, but this event, it is this number colliding pairs needs to be modeled a bit carefully, for that we are going to use an indicator random variable X_{ij} and that is X_{ij} is going to take the value one whenever ball i and ball j land at the same bin, otherwise it is going to take on the value 0.

So, X_{ij} is a very natural choice to model colliding balls. Clearly the expectation of X_{ij} is at most 1 over n because from our assumption that we are saying h drawn uniformly at random from 2-universal family of hash functions. We know that the probability with which ball i and ball j will collide is at most 1 over n . So, the expectation of the Bernoulli random variable X_{ij} is going to be at most 1 over m .

(Refer Slide Time: 16:16)


Bounding γ


$$X = \sum_{1 \leq i < j \leq m} X_{ij} \Rightarrow E[X] \leq \binom{m}{2} \frac{1}{n} \leq \frac{m^2}{2n}$$

Using Markov's inequality

$$\Pr(X \geq \frac{m^2}{n}) \leq \Pr(X \geq \frac{2m^2}{2n}) \leq \frac{1}{2}$$

Thus we must set $\binom{k}{2} \geq \frac{m^2}{n}$ or $\frac{k^2}{2} \geq \frac{m^2}{n}$ (conservative)

$$\Rightarrow k \geq m \sqrt{\frac{2}{n}} = \sqrt{2n} \text{ when } m=n$$


And of course, our interest is counting the total number of collisions. So, with that being the case we define upper case X to be the summation of all these X_{ij} 's and if we look at this way, the expectation of X is simply at most m choose two times 1 over n , which is at most m square over $2n$ and since we have the expectation, we can now apply Markov's inequality and using Markov's inequality, we get probability that X is at least m squared by n is at most for the probability that X is at least twice the expectation of X , this m square by $2, m$; remember is the expectation of X and that is of course, at most 1 by 2 . So, remember our goal is to find the k such that the most loaded bin k as k or more balls in bins.

We were in a bit conservative in bounding that probability, in order to get our value of k , we need to set k choose 2 to be at most m squared by n because k choose we do not want; we were bounding the probability that the number of collisions is at least k choose 2 . So, of course, k choose 2 we approximate that with k square by m by 2 and that leads us to the bound that k must be at least m times squared of 2 over m and this is 4 , you know arbitrary choice of m , but if we restrict m to be exactly equal to n , we get k to be at least square root of 2 times n . This result must be viewed under the back that if we use uniform and mutually independent choice of bins for each ball then the height of the max loaded bin is going to be at most O of \ln and over $\ln \ln n$.

This is significantly worst, but it is still very general in nature. So, it is worst, it is no longer logarithmic, it is proportional to the square of n , but we were able to get this result using hash functions drawn uniformly at random from 2-universal family of hash functions. So, it is in that sense very general in its application.