**Zuha Aqib** 26106

## Question 1

Show that this gives

$$\mathbb{E}[X] = 2\sum_{j=1}^{n}(H_j - 1),$$

where $H_j = \sum_{k=1}^{j} \frac{1}{k}$ is the $j^{\text{th}}$ harmonic number.

(Some reference and understanding has been taken from this lecture: `https://www.cs.cmu.edu/~avrim/451f11/lectures/lect0906.pdf`)

To begin this question, we first need to understand what $X_{ij}$ is. It is the binary random variable to be 1 if the algorithm does compare the ith smallest and jth smallest elements in the course of sorting, and 0 if it does not. Let X denote the total number of comparisons made by the algorithm,

$$X = \sum_{1\le i<j\le n} X_{ij}$$

which can be re-written in the form of two summations as

$$X = \sum_{j=1}^{n}\sum_{i=1}^{j-1} X_{ij}$$

in which $j$ is fixed and $i$ iterates over $j$. We then apply Expectation ($\mathbb{E}$) on both sides,

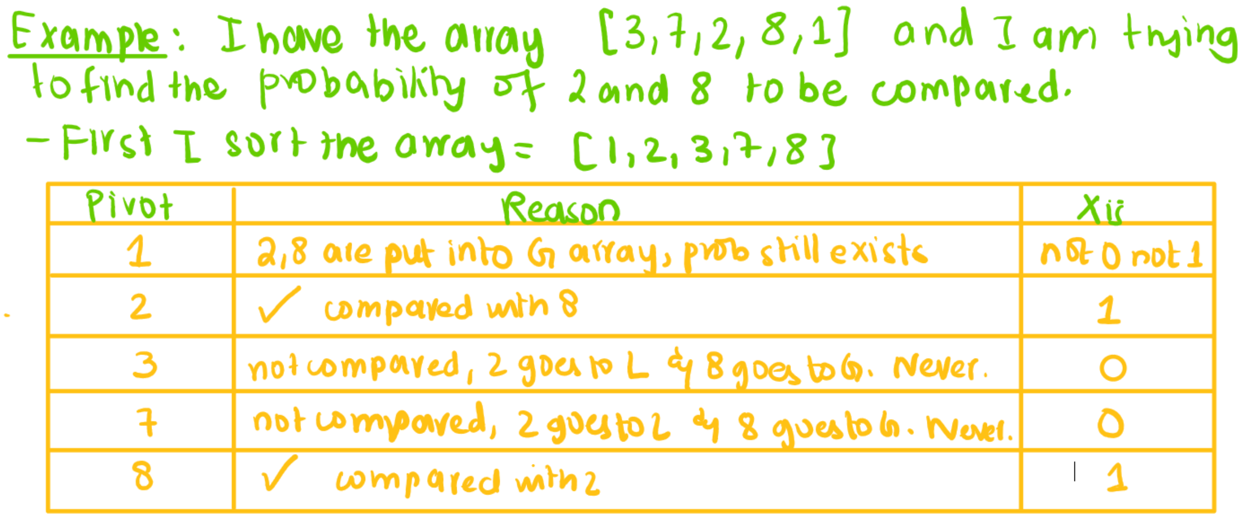
$$\mathbb{E}[X] = \sum_{j=1}^{n}\sum_{i=1}^{j-1} \mathbb{E}[X_{ij}]$$

So what we need to understand is when the $i^{th}$ smallest element and $j^{th}$ smallest element will be compared. For better understanding, lets denote the $i^{th}$ smallest element as $n^i$ and the $j^{th}$ smallest element as $n^j$.

So if we pick a pivot that is

- between $n^i$ and $n^j$, then both would be split into two different arrays $L$ and $G$ and therefore never compared ($X_{ij}$ is 0)
- exactly $n^i$ or $n^j$, then they would be compared and $X_{ij}$ is 1
- not $n^i$ or $n^j$ and is lesser/greater than both, they wouldn't be compared however they would fall in the same array ($L$ or $G$) and would still have the chance (probability) to be compared

So right now, we are focused on the chance of either $n^i$ or $n^j$ being the pivot.



Example: I have the array [3,7,2,8,1] and I am trying to find the probability of 2 and 8 to be compared.

- First I sort the array = [1,2,3,7,8]

| Pivot | Reason | $X_{ij}$ |
|---|---|---|
| 1 | 2,8 are put into G array, prob still exists | not 0 not 1 |
| 2 | ✓ compared with 8 | 1 |
| 3 | not compared, 2 goes to L & 8 goes to G. Never. | 0 |
| 7 | not compared, 2 goes to L & 8 goes to G. Never. | 0 |
| 8 | ✓ compared with 2 | 1 |

Figure 1: Understanding of pivots and when $X_{ij}$ will be 1

From this example we can see that we are only focused on the elements between $n^i$ and $n^j$. We do not care about the elements before $n^i$ and after $n^j$ as it only delays $X_{ij}$. Thus the number of elements between $n^i$ and $n^j$ is $j - i$ and then $+1$ for $n^i$ itself.
There are $j - i + 1$ elements between $n^i$ and $n^j$. The probability of $n^i$ being chosen the pivot out of all the elements between $n^i$ and $n^j$ is

$$Pr(n^i \text{ selected as pivot}) = \frac{1}{j - i + 1}$$

and the same for the probability of $n^j$ being selected as pivot,

$$Pr(n^j \text{ selected as pivot}) = \frac{1}{j - i + 1}$$

Thus

$$Pr(n^i \text{ or } n^j \text{ selected as pivot}) = \frac{1}{j - i + 1} + \frac{1}{j - i + 1} = \frac{2}{j - i + 1}$$

When $n^i$ or $n^j$ is the pivot, this is the only time $X_{ij}$ is 1. Thus we can denote

$$\mathbb{E}[X_{ij}] = \frac{2}{j - i + 1}$$

which makes

$$\mathbb{E}[X] = \sum_{j=1}^{n} \sum_{i=1}^{j-1} \frac{2}{j - i + 1}$$

we can bring the 2 out as it is a constant in the summation

$$\mathbb{E}[X] = 2 \sum_{j=1}^{n} \sum_{i=1}^{j-1} \frac{1}{j - i + 1}$$

here we can introduce $k$ which is equal to $j-i+1$. Thus we can re-evaluate the start stop points of summation as

$$k = j - i + 1$$

when $i = 1$, $k = j - 1 + 1 = j$ and when $i = j-1$, $k = j - (j-1) + 1 = j - j + 1 + 1 = 2$ thus the inner summation is now from $k = 2$ to $k = j$ which makes

$$\mathbb{E}[X] = 2 \sum_{j=1}^{n} \sum_{k=2}^{j} \frac{1}{k}$$

we can rewrite this as

$$\mathbb{E}[X] = 2 \sum_{j=1}^{n} \left( \frac{1}{2} + \ldots + \frac{1}{j} \right)$$

now we can see that this is a sequence of Harmonic Numbers [1]. However this is from $\frac{1}{1}$ to $\frac{1}{k}$ for $H_k$. In our case we have it from $\frac{1}{2}$ to $\frac{1}{k}$ for $H_k$. Thus we can subtract 1 from $H_j$ and thus re-write the equation as

$$\mathbb{E}[X] = 2 \sum_{j=1}^{n} (H_j - 1)$$

(we can remove the $k$ summation as $H_j$ is substituted to loop till $j$). Now we can see that we have proved what we needed to prove. And thus,

$$\mathbb{E}[X] = 2 \sum_{j=1}^{n} (H_j - 1)$$

[1]The harmonic number $H_k$ is a number in the sequence of harmonic numbers, which is defined as

$$H_k = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k}$$

This is cited from: `https://en.wikipedia.org/wiki/Harmonic_number`

**Zuha Aqib** 26106

Collaborators: Zehra Ahmed, Farah Inayat, Hamna Inam, Zara Masood

## Question 2

Suppose we have a tree where each node has three children, so a tree of depth $k$ has $N = 3^k$ leaves. Place a $MAJORITY$ gate at each node which outputs the majority of its children's truth values. Show that by evaluating the subtrees in random order, we can evaluate the entire tree while only looking at

$$\left(\frac{8}{3}\right)^k = N^{\log_3(8/3)}$$

leaves on average.

So here we understand that we have a tree of $k$ levels and each node has three children, thus there are $N$ leaves where $N$ and $k$ have the relationship

$$N = 3^k$$

So we have a choice that we can either evaluate every node - this is $N$ leaves we have to look at. But we want it to be faster - by looking at lesser number of leaves. We understand that every leaf has either the value 0 or 1. And out of 3, there will always be a majority, because 3 is an odd number (in even number children there could have been a tie possibility). So lets start evaluating them. First we go straight to the leaf node using DFS, *depth first search* algorithm. We go to the left-most node of $k-1$ level, not the $k^{th}$ level which is leaves. On the $(k-1)^{th}$ level, at each node, we have three leaves to evaluate. The values of the three leaves could be the following:

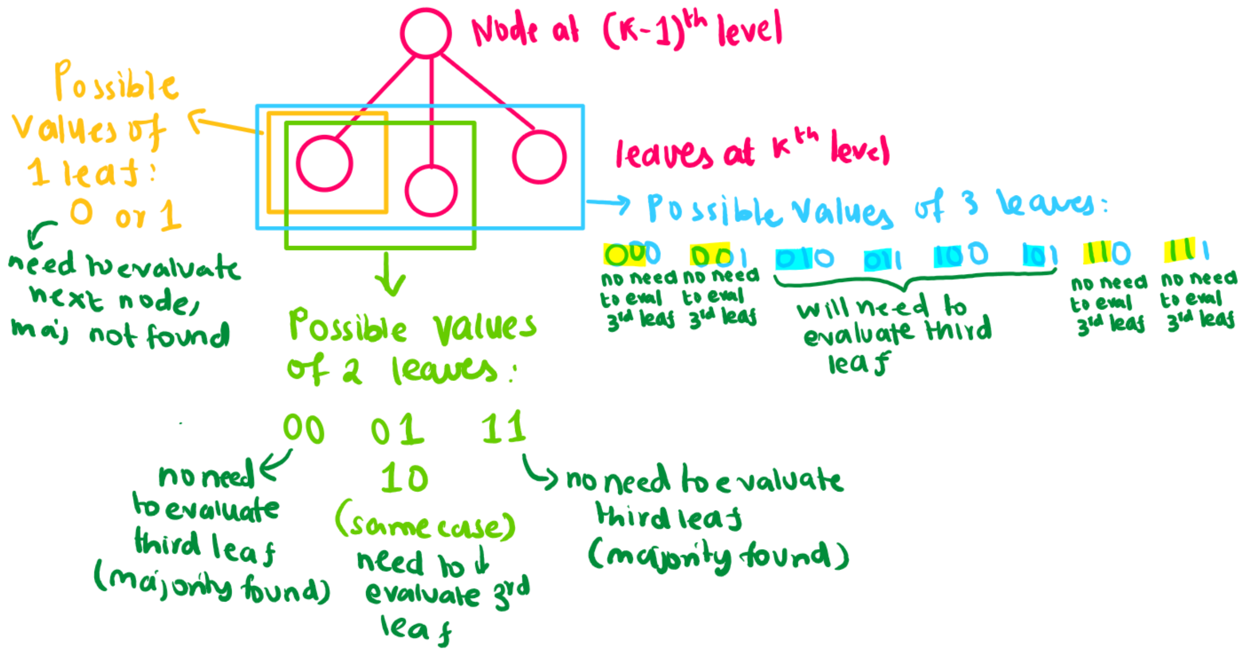

Node at (K-1)th level
Possible values of 1 leaf: 0 or 1
need to evaluate next node, maj not found
leaves at kth level
Possible values of 3 leaves:
000 001 010 011 100 101 110 111
no need to eval 3rd leaf
no need to eval 3rd leaf
will need to evaluate third leaf
no need to eval 3rd leaf
no need to eval 3rd leaf
Possible values of 2 leaves:
00 01 11
10
no need to evaluate third leaf (majority found)
(same case) need to evaluate 3rd leaf
no need to evaluate third leaf (majority found)

Figure 2: Understanding of when we need to evaluate the 1st, 2nd and 3rd leaf.

So here from the figure of annotations that has been attached, we see that the $1^{st}$ leaf always needs to be evaluated, and so does the $2^{nd}$ leaf, as we cannot interpret the $MAJORITY$ from just 1 leaf, we need to evaluate the $2^{nd}$ one as well to get an answer. The $3^{rd}$ however, is not always required to be evaluated. The $3^{rd}$ leaf evaluation is dependent on the $1^{st}$ and $2^{nd}$ leaf evaluation. How? We see this in the form of three cases:

The combination of $1^{st}$ and $2^{nd}$ leaf make:

- 00: $MAJORITY$ of 0 is found, $3^{rd}$ leaf doesn't need to be evaluated
- 11: $MAJORITY$ of 1 is found, $3^{rd}$ leaf doesn't need to be evaluated
- 01 or 10: cannot evaluate $MAJORITY$, will need to evaluate the $3^{rd}$ leaf

So we have three leafs. We can pick ANY pair, i.e.

$$\binom{3}{2} = 3 \text{ possible pairs}$$

So if we have the $1^{st}$, $2^{nd}$ and $3^{rd}$ leaf, we can make the pairs

$$\text{possible pairs: } 1^{st}\ 2^{nd},\ 2^{nd}\ 3^{rd} \text{ and } 1^{st}\ 3^{rd}$$

So we pick these pairs RANDOMLY. This is a RANDOMIZED DFS. Thus I can think of some probabilities, that out of the 3 leaves, there is always 1 leaf that will ALWAYS agree. Thus 2 leaves will disagree. That means there is a $\frac{1}{3}$ probability that I would just see the first 2 leaves and successfully evaluate the $MAJORITY$, while there is a $\frac{2}{3}$ probability that I would need to evaluate all 3 leaves to get the $MAJORITY$. We can formally say this as:

$$Pr(\text{first 2 leaves are same}) = \frac{1}{3}$$

$$Pr(\text{first 2 leaves are different}) = \frac{2}{3}$$

So to compute the average of how many leaves we are checking, we can say that

$$S_k = \text{the expected number of leaves I need to evaluate in depth } k$$
$$S_{k-1} = \text{the expected number of leaves I need to evaluate in depth } k-1$$

Thus we can formulate the equation:

$$S_k = (\text{no. of nodes}) * Pr(2\text{ nodes}) * S_{k-1} + (\text{no. of nodes}) * Pr(3\text{ nodes}) * S_{k-1}$$
$$= 2 * \frac{1}{3} * S_{k-1} + 3 * \frac{2}{3} * S_{k-1}$$
$$= \frac{2}{3} * S_{k-1} + 2 * S_{k-1}$$
$$= \frac{2 * S_{k-1} + 6 * S_{k-1}}{3}$$
$$= \frac{8 * S_{k-1}}{3}$$
$$= \frac{8}{3}S_{k-1}$$

If we call this recursively, then it is less than $\frac{8}{3}$ called $k$ times

$$\leq \left(\frac{8}{3}\right)^k$$

we know that $N = 3^k$ thus if we rearrange the formula, by taking $log_3$ on both sides

$log_3(N) = (log_3 3)^k$

$log_3(N) = k(log_3 3)$ (here we know $log_n n = 1$)

$log_3(N) = k(1)$

$log_3(N) = k$ (thus $k$ can be written as $log_3(N)$)

$$\leq \left(\frac{8}{3}\right)^{log_3(N)}$$

here we can apply a $log$ property that $a^{log_b(x)} = x^{log_b(a)}$

in this case: $\left(\frac{8}{3}\right)^{log_3(N)} = N^{log_3(frac83)}$

thus just like $a$ and $x$ can be flipped, $\frac{8}{3}$ and $N$ can be flipped

$$\leq N^{log_3(\frac{8}{3})}$$

now we can calculate the value of $log_3\left(\frac{8}{3}\right)$ however **we have proved** the required

$$\leq N^{\log_3\left(\frac{8}{3}\right)} \approx N^{0.893}$$

Thus we can see that instead of evaluating $N$ leaves, we are now evaluating $N^{0.893}$ leaves (which is smaller than $N^1$) on average. Thus we have proved that we will be evaluating

$$\left(\frac{8}{3}\right)^k = N^{\log_3(8/3)}$$

on average.

**Zuha Aqib** 26106

## Question 3

(a) Let $\tilde{p}$ denote the estimate returned by $MeanEstimate(\epsilon)$. Prove that $\mathbb{E}(\tilde{p}) = p$.

In this problem we understand that we are flipping a coin with unknown probability of getting a *head* or a *tail*.

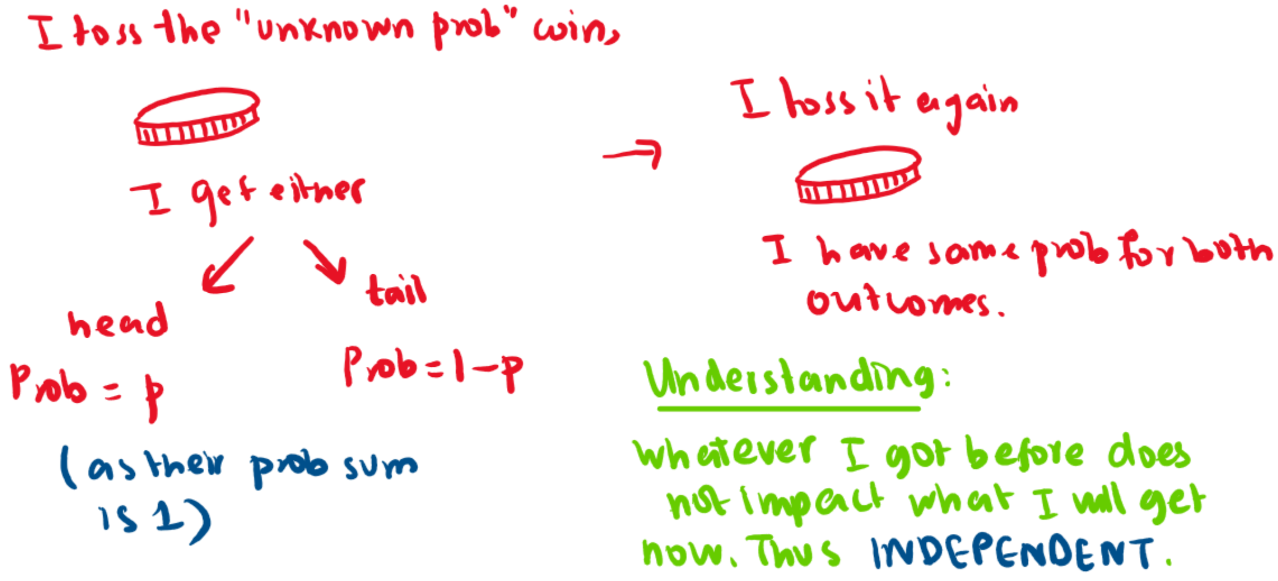

I toss the "unknown prob" coin,
I get either
head
tail
Prob = p
Prob = 1 − p
(as their prob sum is 1)
I toss it again
I have same prob for both outcomes.
Understanding:
whatever I got before does not impact what I will get now. Thus INDEPENDENT.

Figure 3: Here we show the understanding of how each coin flip is independent to previous and after flips.

We define the probability of getting a *head* on the coin as $p$.

We flip the coin $N$ times, and we count the number of times we get a *head*. $\tilde{p}$ is the overall probability of *heads* in $N$ times. We also understand that every flip is an **independent** flip and thus no probability is dependent on each other.

The **formula for** $\tilde{p}$ is

$$\tilde{p} = \frac{no.\ of\ heads}{N}$$

Here we begin by setting a value for each coin flip. In the algorithm, we are counting number of *head* flips, thus we can say that upon introducing a indicator random variable $X_i$ for every coin flip then $X_i$ is 1 when we get a *head* and for a *tail* we can assign the opposite, 0. We assign 1 to *head* because in the algorithm we can see that for every *head* we add 1 to the *count* variable, thus we can assign 1 to *head*, and because a *tail* has no effect (as in the algorithm), we can assign it 0.

$$X_i = \begin{cases} 1 & \text{if the } i\text{-th flip results in } heads \\ 0 & \text{otherwise (the } i\text{-th flip is not a head and is a } tail) \end{cases}$$

Since every coin flip is independent, thus $X_i$ is also independent and has a probability $p$. Thus, we can say this is a *Bernoulli trial* [2] which is a random experiment with two possible outcomes, each independent of other outcomes.

Thus,

$$\mathbb{E}[X_i] = p$$

Each coin flip has an expected value of $p$. The expanded version would be:

$$\begin{aligned}\mathbb{E}[X_i] &= 1(p) + 0(p-1)\\ &= p + 0\\ &= p\end{aligned}$$

As discussed before, $\tilde{p}$ is the probability of getting *head*s on $N$ flips. Thus if I get a *head*, I add 1, and if I don't, I add 0. Thus for every flip, I add $X_i$. Thus I can formulate,

$$\tilde{p} = \frac{X_1 + ... + X_N}{N}$$

which is the same as saying,

$$\tilde{p} = \frac{1}{N}\sum_{i=1}^{N} X_i$$

This can simply be defined as the *sample* mean of $N$ independent *Bernoulli trial*s.

[2]Bernoulli Trials and Binomial Distribution are the fundamental topics in the study of probability and probability distributions. Bernoulli's Trials are those trials in probability where only two possible outcomes are Success and Failure or True and False. Due to this fact of two possible outcomes, it is also called the Binomial Trial. cited from: `https://www.geeksforgeeks.org/bernoulli-trials-binomial-distribution/`

So now let's calculate $\mathbb{E}[\tilde{p}]$, which is

$$\begin{aligned}
\mathbb{E}[\tilde{p}] &= \mathbb{E}\left[\text{The formula we defined for } \tilde{p} \text{ above}\right] \\
&= \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N} X_i\right] \\
&= \text{here we can apply "Property of Expectation with constants"}^{3} \text{ which says that} \\
&\quad \text{constants can be taken out of random variables} \\
&= \frac{1}{N}\,\mathbb{E}\left[\sum_{i=1}^{N} X_i\right] \\
&= \text{lets open the summation} \\
&= \frac{1}{N}\,\mathbb{E}\left[X_1 + ... + X_N\right] \\
&= \text{now we can apply "Linearity of Expectation"}^{4} \text{ which states that if two} \\
&\quad \text{things are being summed inside } \mathbb{E}, \text{ we can take them out and apply } \mathbb{E} \text{ separately} \\
&\quad \text{thus we can take out the summation} \\
&= \frac{1}{N}\,\mathbb{E}\left[X_1\right] + ... + \mathbb{E}[X_N] \\
&= \frac{1}{N}\sum_{i=1}^{N}\,\mathbb{E}\left[X_i\right] \\
&= \text{now we can apply that } \mathbb{E}[X_i] = p \text{ as we discussed above} \\
&= \frac{1}{N}\sum_{i=1}^{N} p \\
&= \text{this summation is just equal to } p \text{ being added } N \text{ times thus we can write it as } Np \\
&= \frac{1}{N}Np \\
&= p
\end{aligned}$$

Thus we have shown that $\mathbb{E}[\tilde{p}] = p$.

[4] The "Property of Expectation with constants" states that constants can be factored out of expectation, which means,

$$\mathbb{E}[cX] = c\,\mathbb{E}[X]$$

thus constants can be taken out from expectations. cited from: `https://www.geeksforgeeks.org/properties-of-expected-value/`

[4] The "Linearity of Expectation" states that everything be summed inside an expectation can have expectation applied seperately, which means

$$\mathbb{E}[X+Y] = \mathbb{E}[X] + \mathbb{E}[Y]$$

thus summations can be seperated in expectations. cited from: `https://www.geeksforgeeks.org/properties-of-expected-value/`

(b) Prove using Chebyshev's inequality that if we set $N = \lceil \frac{\alpha}{\epsilon^2} \rceil$ for some appropriate constant $\alpha$, then we have

$$\Pr(|\tilde{p} - p| > \epsilon) < \frac{1}{4}.$$

Before we begin our proof, we need to first understand the *Chebyshev's Inequality* [5]. It is used to answer the question: what is the probability of some number to be $k$.standard deviations away from the mean? The equation is:

$$Pr(|X - \mathbb{E}[X]| \ \geq k\sigma) \leq \frac{1}{k^2}$$

$$\begin{aligned} where\ X &= \text{is the random variable} \\ \mathbb{E}[X] &= \text{the mean of } X \\ \sigma &= \text{the standard deviation of } X \\ k &= \text{how many standard deviations away it is, here } k \text{ is any number} \end{aligned}$$

The question here is that If I flip the coin $N$ times and calculate the probability of *head*s, how many flips are enough to guarantee that my estimate is close to the true probability most of the time? How many flips do I do to make it guaranteed that i will get a probability of *head* which is equal to $p$.

Lets first apply our coin-flipping question to the *Chebyshev Inequality.*

$$\begin{aligned} in\ this\ question,\ X &= \text{is the random variable which in this case is } \tilde{p} \\ \mathbb{E}[X] &= \text{the mean of } \tilde{p} \text{ which is } p \\ k &= \text{how many standard deviations away it is, here } k \text{ is any number} \\ \sigma &= \text{the standard deviation of } \tilde{p} \text{ which is we first calculate the variance:} \end{aligned}$$

[5] Chebyshev's inequality is a probabilistic inequality. It provides an upper bound to the probability that the absolute deviation of a random variable from its mean will exceed a given threshold. This is cited from `https://www.statlect.com/fundamentals-of-probability/Chebyshev-inequality`

$Var(X_i) = \mathbb{E}[(X_i - \mathbb{E}[X_i])^2]$ [6]

now if we replace $\mathbb{E}[X_i]$ with $p$,

$= \mathbb{E}[(X_i - p)^2]$

and now we expand the bracket $(X_i - p)^2$

$= \mathbb{E}[X_i^2 - X_i p - X_i p + p^2]$

$= \mathbb{E}[X_i^2 - 2X_i p + p^2]$

as discussed above, we can now apply *Linearity of Expectation*

$= \mathbb{E}[X_i^2] - \mathbb{E}[2X_i p] + \mathbb{E}[p^2]$

so here we know that $X_i$ is either 0 or 1. Thus $X_i^2 = 0^2 = 0 = X_i$ and $X_i^2 = 1^2 = 1 = X_i$. Thus we can say that $X_i^2 = X_i$ and $\mathbb{E}[X_i^2] = \mathbb{E}[X_i]$

$= \mathbb{E}[X_i] - \mathbb{E}[2X_i p] + \mathbb{E}[p^2]$

$= p - \mathbb{E}[2X_i p] + \mathbb{E}[p^2]$

now here we can apply *Property of Expectation with constants* and bring the $p$ and 2 out of $\mathbb{E}$

$= p - 2p\mathbb{E}[X_i] + \mathbb{E}[p^2]$

$= p - 2p.p + \mathbb{E}[p^2]$

$= p - 2p^2 + \mathbb{E}[p^2]$

again we can apply the *Property of Expectation with constants* because $p$ is a constant, and so is $p^2$

$= p - 2p^2 + p^2\mathbb{E}[1]$

$= p - 2p^2 + p^2.1$

$= p - 2p^2 + p^2$

$= p - p^2$

$= p(1-p)$

now here we know that the highest value of $p$ is 0.5 and the largest value of $(1-p)$ is also 0.5. Thus the largest value of $p(1-p)$ is 0.25 (this is also confirmed by the table below)

$= p(1-p) \leq \frac{1}{4}$

[6] This formula is cited from: https://www.ncl.ac.uk/webtemplate/ask-assets/external/maths-resources/statistics/descriptive-statistics/variance-and-standard-deviation.html

| $p$ | $1-p$ | $p(1-p)$ |
|---|---|---|
| 0 | 1 | 0 |
| 0.1 | 0.9 | 0.09 |
| 0.25 | 0.75 | 0.1875 |
| 0.5 | 0.5 | 0.25 |
| 0.75 | 0.25 | 0.1875 |
| 0.9 | 0.1 | 0.09 |
| 1 | 0 | 0 |

Table 1: Finding the largest value of $p$

because this is regarding $N$ flips, we define the *Variance of Average flips* as,

$$Var(X_i) = \frac{p(1-p)}{N}$$

and the relation between standard deviation and variance is

$$\sigma = \sqrt{Var(X_i)}$$

Thus defining the standard deviation as

$$\sigma = \sqrt{\frac{p(1-p)}{N}}$$

Lets compile our variables:

$$\begin{aligned} \textit{in this question, } X &= \tilde{p} \\ \mathbb{E}[X] &= p \\ k &= k \\ \sigma &= \sqrt{\frac{p(1-p)}{N}} \end{aligned}$$

Thus it makes our *Chebyshev Inequality* as

$$Pr\left(|\tilde{p}-p| \geq k\sqrt{\frac{p(1-p)}{N}}\right) \leq \frac{1}{k^2}$$

Here we are told that

$$\epsilon = k\sigma$$

i.e.

$$\epsilon = k\sqrt{\frac{p(1-p)}{N}}$$

Thus we can rearrange the formula to be

$$k = \frac{\epsilon}{\sqrt{\frac{p(1-p)}{N}}}$$

Thus our *Chebyshev Inequality* is now

$$Pr\left(|\tilde{p}-p| \;\geq \epsilon\;\right) \leq \frac{1}{\left(\frac{\epsilon}{\sqrt{\frac{p(1-p)}{N}}}\right)^2}$$

simplifying this further

$$Pr\left(|\tilde{p}-p| \;\geq \epsilon\;\right) \leq \frac{1}{\frac{\epsilon^2}{\frac{p(1-p)}{N}}}$$

$$Pr\left(|\tilde{p}-p| \;\geq \epsilon\;\right) \leq \frac{\frac{p(1-p)}{N}}{\epsilon^2}$$

$$Pr\left(|\tilde{p}-p| \;\geq \epsilon\;\right) \leq \frac{p(1-p)}{N\epsilon^2}$$

now we can apply the property we derived above that

$$p(1-p) \leq \frac{1}{4}$$

applying this in place of $p(1-p)$

$$Pr\left(|\tilde{p}-p| \;\geq \epsilon\;\right) \leq \frac{\frac{1}{4}}{N\epsilon^2}$$

$$Pr\left(|\tilde{p}-p| \;\geq \epsilon\;\right) \leq \frac{1}{4N\epsilon^2}$$

Now we want the inequality to be $\leq \frac{1}{4}$ thus we will replace the left hand side with this,

$$\frac{1}{4N\epsilon^2} \leq \frac{1}{4}$$

simplify it

$$\frac{1}{N\epsilon^2} \leq 1$$

rearrange it for what we are required to prove

$$1 \leq N\epsilon^2$$

flip it

$$N\epsilon^2 \geq 1$$

which makes

$$N \geq \frac{1}{\epsilon^2}$$

and here we introduce a constant $\alpha$ to always make it less than $\frac{1}{4}$

$$N \geq \frac{\alpha}{\epsilon^2}$$

Thus we have proved that using $N = \lceil \frac{\alpha}{\epsilon^2} \rceil$, we result in $< \frac{1}{4}$

**Zuha Aqib** 26106

Collaborators: Zehra Ahmed, Farah Inayat, Hamna Inam, Zara Masood

## Question 4

(a) Show a sequence of $n$ operations allowing both increments and decrements and starting from 0 that, without ever making the counter go negative, costs $\Omega(\log n)$ amortized per operation (i.e., $\Omega(n \log n)$ total cost).

So our aim here is to show that the total cost of performing $n$ operations (a mix of increments and decrements) on a binary counter is $\Omega(nlog(n))$. Here we want to maximise our cost fully so that we can find a tight lower bound on the increment, however there is a catch - we have to start from 0.

So to begin this question, lets consider a counter with $k$ bits where

$$k = log_2(n)$$

And we define a sequence of operations where:

- we first increment to the value $2^{k-1} - 1$ which means we do $2^{k-1} - 1$ increments where each increment is of $O(1)$ thus the total cost is $2^{k-1} - 1$
- then we oscillate between the $(2^{k-1} - 1)^{th}$ value and the $(2^{k-1})^{th}$ value. If we call the $(2^{k-1} - 1)^{th}$ value $A$ and the $(2^{k-1})^{th}$ value $B$. So we are at value $A$ and we increment to $B$ and then decrement to $A$ then again decrement to $B$, and so on, we do this $2^{k-1}$ times. In each oscillation, we flip all $k$-bits, thus each increment/decrement is $O(k)$ where $k = log_2(n)$ thus the cost is $O(log_2(n))$ for each operation

So why do we do this sequence? Because to find the lower bound, we want to maximise our cost. Our cost depends on the number of bits flipped, and upon incrementing in practice in class, we see that the highest number of flips are when we flip all the bits - that is when we increment from the $(2^{k-1} - 1)^{th}$ value (in which the $MSB$-bit is 0 and rest all bits are 1, and we go to the $(2^{k-1})^{th}$ value in which the $MSB$-bit is 1 and rest bits are 0 - this is when ALL the bits are flipped. Similarly, when we decrement from this value, all bits are flipped again, and when we increment, again all bits are flipped - thus we are in the most EXPENSIVE increment/decrements. So the worst case would be that for all $n$ operations we just do this increment/decrement. However, we first need to GET TO THAT position. To get to that position, we need to get to the $(2^{k-1} - 1)^{th}$ value for which we will NEED to do $2^{k-1} - 1$ increments. Then, in the remaining operations left, we just oscillate between the highest increment decrement to maximise our cost.

So what is the total cost of this sequence? How many times do we do the first and second part? First, from the formula of $k$ we did

$$k = log_2(n)$$

lets derive $n$, here we know that the logarithmic function of

$$log_b(x) = y \text{ can be rewritten as } b^y = x$$

so thus we can rewrite the $k$ equation as

$$n = 2^k$$

So we are doing $2^k$ operations. Out of those $2^k$ operations, we first do the first phase (increment to $2^{k-1} - 1$) which is $2^{k-1} - 1$ operations, thus

$$\begin{aligned}
\text{operations left } &= n - 1^{st} \text{ phase} \\
&= 2^k - (2^{k-1} - 1) \\
&= 2^k - 2^{k-1} + 1 \\
&= 2^k - \frac{2^k}{2} + 1 \\
&= \frac{2.2^k - 2^k}{2} + 1 \\
&= \frac{2^k}{2} + 1 \\
&= 2^{k-1} + 1
\end{aligned}$$

so out of the $2^{k-1}+1$ operations we are left with, we oscillate between $A$ and $B$ by incrementing and decrementing.

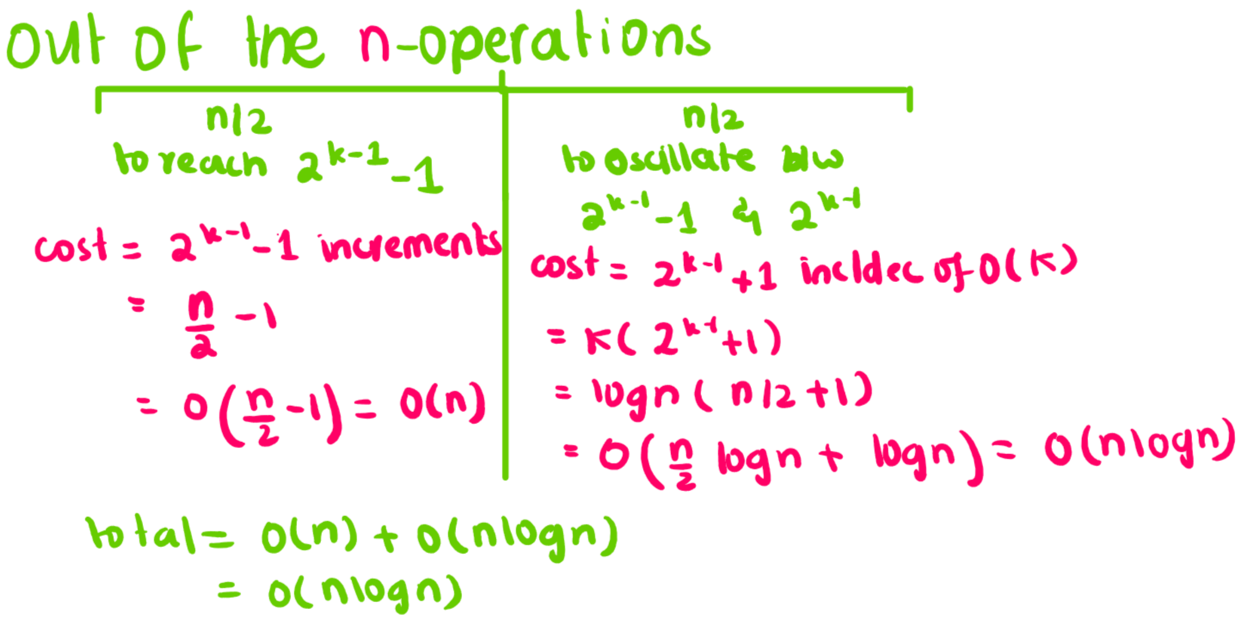


Figure 4: Understanding of the cost division in both phases and thus the totaling of the cost

## Algorithm:

```
        def BinaryCounter(k):
            max_val = 2^(k-1) - 1
            counter = 0
            t_cost = 0

            # Phase 1: Increment to max_val
            while counter < max_val:
                counter = counter + 1
                t_cost += cost(counter-1, counter)

            # Phase 2: Oscillations
            for i = 1 to 2^(k-1):
                if i % 2 = 0:
                    counter = counter + 1
                    t_cost += cost(counter-1, counter)
                else:
                    counter = counter - 1
                    t_cost += cost(counter-1, counter)

        def cost(start, end):
            return number_of_flipped_bits(start, end)

```

Listing 1: Algorithm for $k$-bit highest cost sequence

So what is the cost?

$$\begin{aligned}
\text{total cost} &= \text{phase 1 (incrementing)} + \text{phase 2 (oscillating)} \\
&= \text{(one\_inc\_cost)(no\_of\_inc)} + \text{(one\_inc\_or\_dec\_cost)(no\_of\_inc\_or\_dec)} \\
&\quad \text{where inc} = \text{increment and dec} = \text{decrement} \\
&= (1)(2^{k-1} - 1) + (k)(2^{k-1} + 1)
\end{aligned}$$

phase 1: we discussed that each increment to $(2^{k-1} - 1)$ is $O(1)$ cost and there are $(2^{k-1} - 1)$ increments

phase 2: here each inc/dec consists of flipping $k$ bits, thus $O(k)$ cost and we perform this inc/dec $(2^{k-1} + 1)$ times

$$= 2^{k-1} - 1 + k.2^{k-1} + k$$

$$= \frac{2^k}{2} + \frac{k.2^k}{2} - 1 + k$$

we can apply the formula $n = 2^k$

$$= \frac{n}{2} + \frac{nk}{2} - 1 + k$$

$$= \frac{n + nk - 2 + 2k}{2}$$

we can apply the formula $k = log_2(n)$

$$= \frac{n + nlog_2(n) - 2 + 2log_2(n)}{2}$$

$$\approx nlog_2(n)$$

$$\approx \Omega(nlog_2(n))$$

THUS we have proved

$$\Omega(nlog(n)) \text{ as the total cost}$$

and

$$\frac{\Omega(nlog(n))}{n} = \Omega(log(n)) \text{ as the amortized cost}$$

---

Let us display this sequence using an example.
We use

$$k = 4$$

thus our counter is of 4 bits (highest value is 15). So,

FIRST PHASE (increment to $2^{k-1} - 1$)
So we increment to

$$2^{4-1} - 1 = 2^3 - 1 = 8 - 1 = 7$$

in which we perform 7 increments:

| **Step** | **Binary** | **No. of Flips** |
|---|---|---|
| $0 \to 1$ | $0000 \to 0001$ | 1 |
| $1 \to 2$ | $0001 \to 0010$ | 2 |
| $2 \to 3$ | $0010 \to 0011$ | 1 |
| $3 \to 4$ | $0011 \to 0100$ | 3 |
| $4 \to 5$ | $0100 \to 0101$ | 1 |
| $5 \to 6$ | $0101 \to 0110$ | 2 |
| $6 \to 7$ | $0110 \to 0111$ | 1 |

Table 2: Phase 1: incrementing to $2^{k-1} - 1$ from 0

SECOND PHASE (oscillating):
Now we oscillate between

$$2^{k-1} - 1 = 2^{4-1} - 1 = 7$$

and

$$2^{k-1} = 2^{4-1} = 8$$

and we do this $2^{k-1} + 1 = 9$ times

| **Step** | **Binary** | **No. of Flips** |
|---|---|---|
| $7 \to 8$ | $0111 \to 1000$ | 4 |
| $8 \to 7$ | $1000 \to 0111$ | 4 |
| $7 \to 8$ | $0111 \to 1000$ | 4 |
| $8 \to 7$ | $1000 \to 0111$ | 4 |
| $7 \to 8$ | $0111 \to 1000$ | 4 |
| $8 \to 7$ | $1000 \to 0111$ | 4 |
| $7 \to 8$ | $0111 \to 1000$ | 4 |
| $8 \to 7$ | $1000 \to 0111$ | 4 |
| $7 \to 8$ | $0111 \to 1000$ | 4 |

Table 3: Phase 2: oscillating between $2^{k-1} - 1$ and $2^{k-1}$

(b) Give a clear, coherent proof that with this representation, the amortized cost per operation is $O(1)$ (i.e., the total cost for the $n$ operations is $O(n)$). Hint: think about a "bank account" or "potential function" argument.

In this part, we are required to prove that the amortized cost of increment and decrement operations in the redundant *ternary number system* is $O(1)$. The redundant *ternary number system* represents numbers using *trits* (ternary digits), where each *trit* can take values from the set:

$$t_i \epsilon \{-1, 0, +1\}$$

To analyze the amortized cost, we introduce a potential function that helps us track the internal state of the system during each operation. The potential function $\Phi$ is defined as:

$$\Phi = \sum_{i=0}^{k-1} |t_i|$$

Here, $\Phi$ represents the **total number of non-zero trits** in the number at any point in time.

INCREMENTS OPERATIONS POLICY
To understand the amortized cost, let's break down how incrementing a ternary number works:

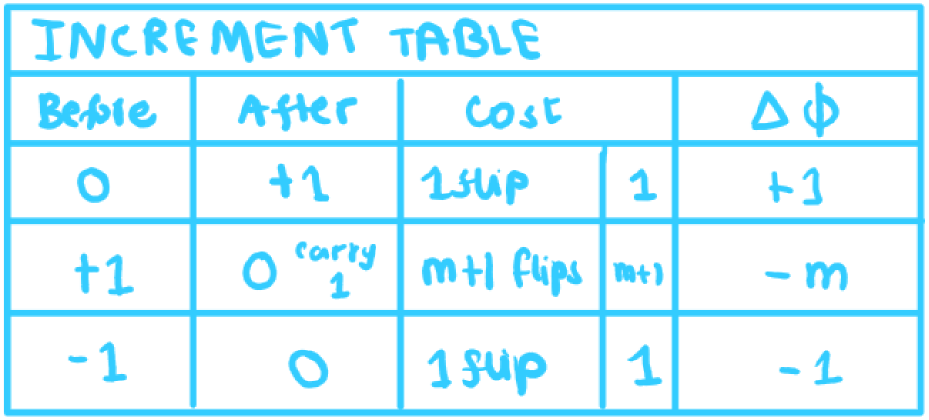
- If the current trit is 0, it becomes +1 (cost = 1).
- If the current trit is -1, it becomes 0 (cost = 1), and since a non-zero trit became zero, the potential function decreases by 1.
- If the current trit is +1, it becomes +2, which is not allowed. So, it is converted to 0, and the carry is propagated to the next trit. This propagation continues until no more carries are left.
  If the carry propagates over m positions, then the total number of trits that flip is m+1. However, each carry reduces the potential function by 1, making the net potential change:

  $$\Delta\Phi = -m$$

  The amortized cost is given by:

  $$a_i = c_i + \Delta\Phi = (m + 1) - m = 1 = O(1)$$

This shows that the amortized cost of the increment operation is constant.



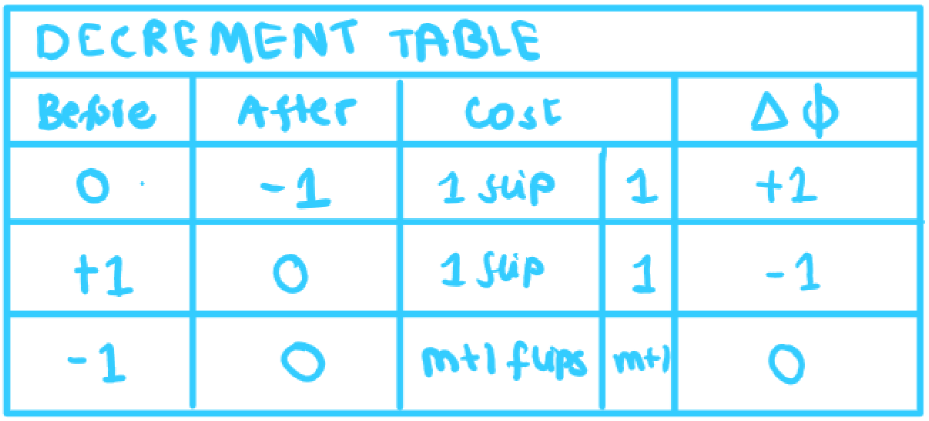
| INCREMENT TABLE | | | | |
|---|---|---|---|---|
| Before | After | Cost | | $\Delta\phi$ |
| 0 | +1 | 1 flip | 1 | +1 |
| +1 | 0 carry 1 | m+1 flips | m+1 | −m |
| −1 | 0 | 1 flip | 1 | −1 |

Figure 5: Increment Table for a *trit*

DECREMENTS OPERATIONS POLICY
Similarly, the decrement operation works as follows:

- If the current trit is 0, it becomes -1 (cost = 1).
- If the current trit is +1, it becomes 0 (cost = 1), and the potential function decreases by 1.
- If the current trit is -1, it becomes -2, which is invalid. So, it is converted to 0, and a borrow is propagated to the next trit.
  If the borrow propagates over m positions, the actual cost is $m+1$, but since each borrow cancels out its effect on the potential function, the net change in potential is:
  $$\Delta\Phi = 0$$
  Hence, the amortized cost is:
  $$a_i = c_i + \Delta\Phi = (m+1) + 0 = m+1 = O(1)$$

This shows that the amortized cost of the decrement operation is also constant.



| DECREMENT TABLE | | | | |
|---|---|---|---|---|
| Before | After | Cost | | $\Delta\phi$ |
| 0 | −1 | 1 flip | 1 | +2 |
| +1 | 0 | 1 flip | 1 | −1 |
| −1 | 0 | m+1 flips | m+1 | 0 |

Figure 6: Decrement table for a *trit*

Using the potential function method, we have shown that both increment and decrement operations have an amortized cost of O(1). Even though the actual cost might vary depending on the number of carry or borrow propagations, the potential function guarantees that the average cost per operation remains constant over multiple operations. Therefore, for n operations, the total cost is:

$$\sum_{i=1}^{n} a_i = O(n)$$

and the

amortized cost per operation is: $O(1)$

**Zuha Aqib** 26106

Collaborators: Zehra Ahmed, Farah Inayat, Hamna Inam, Zara Masood

# Question 5

(a) Prove that $\mathcal{H}$ is 2-uniform.

To prove that it is 2-uniform, we need to prove the property that

$$Pr(h_{A,B}(x_1, y_1) = h_{A,B}(x_2, y_2)) = \frac{1}{m^2}$$

for all distinct pairs $(x_1, y_1) \neq (x_2, y_2)$

To prove this, lets first define the random variables

$$w = A[x_1], x = A[x_2], y = B[y_1], z = B[y_2]$$

So we compute the variables

$$a = w \oplus y$$

and

$$b = x \oplus z$$

and we want to find the probability that

$$a = b \text{ has probability } \frac{1}{m^2}$$

So here we know that $w, x, y, z$ are all $l$-bit random strings and they are all independent. Thus each of them have

$$\text{number of combinations for each } = 2^l = m$$

which means there are

$$\text{total combinations} = m * m * m * m = m^4$$

So,

CASE 1: pairs are not equal
So in this case I get $x_1$ and $x_2$ and they are NOT EQUAL, out of the $m^4$ possibilites, and so to find $w$'s corresponding $y$ where $a$ is equal to $b$, we can rearrange the previous formula,

$$w = y \oplus a$$

and for $x$'s corresponding $z$, I can do the same:

$$x = z \oplus b$$

thus there is only one corresponding combination of $y$ and one corresponding combination of $z$, thus out of the $m^4$ combinations, only $m^2$ are possible, thus,

$$Pr(a\ AND\ b \text{ when } x_1 \neq x_2\ AND\ y_1 \neq y_2) = \frac{m^2}{m^4} = \frac{1}{m^2}$$

Thus we have show 2-uniform for this case.

CASE 2: $x_1, x_2$ are equal
So in this case I get $x_1$ and $x_2$, out of the $m^4$ possibilites, and they are EQUAL so now I am left with only $m^3$ possible combinations, and so to find $w$'s corresponding $y$ where $a$ is equal to $b$, we can rearrange the previous formula,

$$w = y \oplus a$$

and for $x$'s corresponding $z$, I can do the same:

$$x = z \oplus b$$

thus there is only one corresponding combination of $y$ and one corresponding combination of $z$, thus out of the $m^3$ combinations, only $m$ are possible, thus,

$$Pr(a = b \text{ when } x_1 = x_2) = \frac{m}{m^3} = \frac{1}{m^2}$$

Thus we have show 2-uniform for this case.

CASE 3: $y_1, y_2$ are equal
So in this case I get $y_1$ and $y_2$, out of the $m^4$ possibilites, and they are EQUAL so now I am left with only $m^3$ possible combinations, and so to find $y$'s corresponding $w$ where $a$ is equal to $b$, we can rearrange the previous formula,

$$y = w \oplus a$$

and for $z$'s corresponding $x$, I can do the same:

$$z = x \oplus b$$

thus there is only one corresponding combination of $w$ and one corresponding combination of $x$, thus out of the $m^3$ combinations, only $m$ are possible, thus,

$$Pr(a = b \text{ when } y_1 = y_2) = \frac{m}{m^3} = \frac{1}{m^2}$$

Thus we have show 2-uniform for this case.

Thus because all cases are proved to be 2-uniform, thus proved it is 2-uniform.

But lets see if I can generalize these cases more, lets try to first simplify the left hand side to something more meaningful and understandable.

$$\begin{aligned}
h_{A,B}(x_1,y_1) &= h_{A,B}(x_2,y_2) \\
&\text{where } h_{A,B}(x,y) = A[x] \oplus B[y] \\
A[x_1] \oplus B[y_1] &= A[x_2] \oplus B[y_2] \\
&\text{introduce XOR of } A[x_2] \oplus B[y_1] \text{ on both sides} \\
A[x_1] \oplus B[y_1] \oplus (A[x_2] \oplus B[y_1]) &= A[x_2] \oplus B[y_2] \oplus (A[x_2] \oplus B[y_1]) \\
&\text{because XOR is associative }^{7} \text{ and commutative }^{8}, \\
&\text{we can rearrange the equation on each side} \\
(A[x_1] \oplus A[x_2]) \oplus (B[y_1] \oplus B[y_1]) &= (A[x_2] \oplus A[x_2]) \oplus (B[y_2] \oplus B[y_1]) \\
&\text{the XOR of to self is equal to 0 (proved in table below)} \\
&\text{Thus } B[y_1] \oplus B[y_1] = 0 \\
(A[x_1] \oplus A[x_2]) \oplus 0 &= (A[x_2] \oplus A[x_2]) \oplus (B[y_2] \oplus B[y_1]) \\
&\text{Thus } A[x_2] \oplus A[x_2] = 0 \\
(A[x_1] \oplus A[x_2]) \oplus 0 &= 0 \oplus (B[y_2] \oplus B[y_1]) \\
&\text{the XOR of anything with 0 is always equal to what} \\
&\text{we started with. Thus we can remove the } \oplus 0 \\
(A[x_1] \oplus A[x_2]) &= (B[y_2] \oplus B[y_1])
\end{aligned}$$

| $x$ | $x \oplus x$ |
|---|---|
| 0 | 0 |
| 1 | 0 |

Table 4: XOR of a value with itself. The property of XOR is that if two values are same, its 0, if they are different, it is 1. Here when XOR is on self, values are same, and thus it always gives 0.

| $x$ | **0** | $x \oplus 0$ |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

Table 5: XOR of a value with 0 always gives back the value we started with. we can say that $a \oplus 0 = a$

[8] Associative means order does not matter and if three elements are being XOR-ed, any pair can be done first with the same result such as

$$a \oplus (b \oplus c) = (a \oplus b) \oplus c$$

It is cited from: `https://web.stanford.edu/class/archive/cs/cs103/cs103.1142/lectures/01/Small01.pdf` (slide 35 and 43)

[8] Commutative is the same as associative but it is for binary i.e. pairs, such as

$$a \oplus b = b \oplus a$$

It is cited from: `https://web.stanford.edu/class/archive/cs/cs103/cs103.1142/lectures/01/Small01.pdf` (slide 35 and 52)

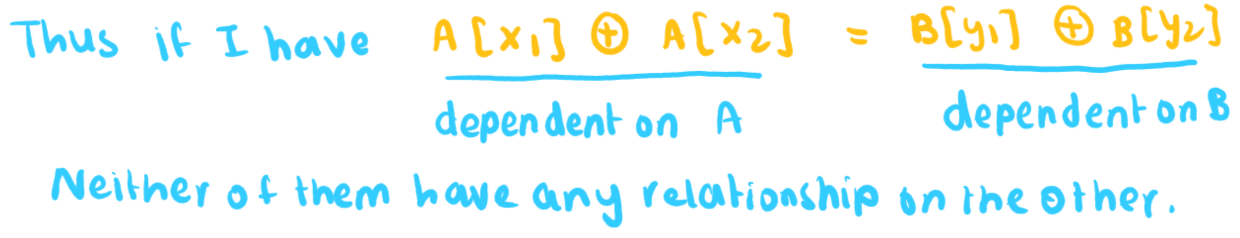
So we reach the simplification from

$$Pr(h_{A,B}(x_1, y_1) = h_{A,B}(x_2, y_2)) = \frac{1}{m^2}$$

to

$$Pr(A[x_1] \oplus A[x_2] = B[y_2] \oplus B[y_1]) = \frac{1}{m^2}$$

In this equation, we can see that inside the probability, the left hand side $A[x_1] \oplus A[x_2]$ is **dependent** on $A$ and the right hand side $B[y_2] \oplus B[y_1]$ is **dependent** on $B$.

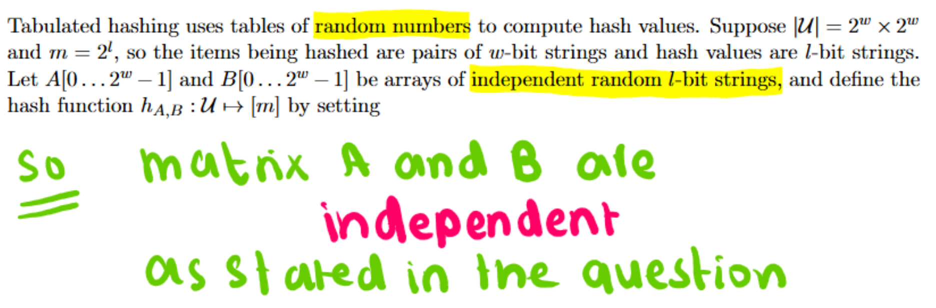


Thus if I have $\underbrace{A[x_1] \oplus A[x_2]}_{\text{dependent on A}} = \underbrace{B[y_1] \oplus B[y_2]}_{\text{dependent on B}}$

Neither of them have any relationship on the other.

Figure 7: Thus we can clearly see that the left hand side is fully matrix $A$ dependent and the right hand side is fully matrix $B$ dependent. They are not cross dependent, the left hand side does not have a relationship with matrix $B$ and the right hand side does not have a relationship with matrix $A$.

We also recogonize that $A$ and $B$ are independent random arrays of bit length $l$



Tabulated hashing uses tables of random numbers to compute hash values. Suppose $|\mathcal{U}| = 2^w \times 2^w$ and $m = 2^l$, so the items being hashed are pairs of $w$-bit strings and hash values are $l$-bit strings. Let $A[0 \ldots 2^w - 1]$ and $B[0 \ldots 2^w - 1]$ be arrays of independent random $l$-bit strings, and define the hash function $h_{A,B} : \mathcal{U} \mapsto [m]$ by setting

So matrix A and B are independent as stated in the question

Figure 8: (Extract from the question) It is stated that both matrixes are independent and random.

So now we know that the left hand side is completely independent from the right hand side. So each string is of $l$ bits. The number of combinations we can make of those $l$ bits with two values (0 or 1) for each can be $2^l$ combinations. We are given $m$ as

$$m = 2^l$$

So the probability of getting 1 combination out of $m$ combinations is

$$Pr(\text{one combination}) = \frac{1}{m}$$

So we

- select two strings out of matrix A, the probability for selecting one is $\frac{1}{m}$
- $\oplus$ them and thus get a single answer
- do the same for matrix $B$, probability is $\frac{1}{m}$
- $\oplus$ them to get a single answer

The probability that the answer from matrix $A$ is equal to matrix $B$ is the product of extracting the answer from $A$ and extracting of answer from $B$.

$$Pr(\text{matrix } A \text{ selects } x_1, x_2) = \frac{1}{m}$$

$$Pr(\text{matrix } B \text{ selects } y_1, y_2) = \frac{1}{m}$$

$$\begin{aligned} Pr(A \text{ AND } B) &= Pr(A) * Pr(B) \\ &= \frac{1}{m} * \frac{1}{m} \\ &= \frac{1}{m^2} \end{aligned}$$

Thus we can see that the probability of getting the same on the left hand side as on the right hand side is $\frac{1}{m^2}$. Thus

$$Pr(A[x_1] \oplus A[x_2] = B[y_2] \oplus B[y_1]) = \frac{1}{m^2}$$

Henceforth we have proved that

$$Pr(h_{A,B}(x_1, y_1) = h_{A,B}(x_2, y_2)) = \frac{1}{m^2}$$

and thus, $\mathcal{H}$ is 2-uniform.

(b) Prove that $\mathcal{H}$ is 3-uniform.

(Some reference has been used from this document: `https://www.cs.cmu.edu/~15750/notes/hashing.pdf`)

Now we aim to prove it is 3-unform. To prove that it is 2-uniform, we need to prove the property that

$$Pr(h_{A,B}(x_1, y_1) = h_{A,B}(x_2, y_2) = h_{A,B}(x_3, y_3)) = \frac{1}{m^3}$$

for all distinct pairs $(x_1, y_1) \neq (x_2, y_2) \neq (x_3, y_3)$

We do this by again defining random variables

$$w = A[x_1], x = A[x_2], y = B[y_1], z = B[y_2], s = A[x_3], t = B[y_3]$$

So we compute the variables

$$a = w \oplus y$$

and

$$b = x \oplus z$$

and

$$c = s \oplus t$$

and we want to find the probability that

$$a \ AND \ b \ AND \ c \text{ has probability } \frac{1}{m^3}$$

So here we know that $w, x, y, z, s, t$ are all $l$-bit random strings and they are all independent. Thus each of them have

$$\text{number of combinations for each } = 2^l = m$$

which means there are

$$\text{total combinations} = m * m * m * m * m * m = m^6$$

So,

CASE 1: none of them are equal to each other
So in this case I get $x_1$ and $x_2$ and $x_3$, AND THEY ARE ALL DIFFERENT, hence there are $m^6$ possibilites, and so to find $w$'s corresponding $y$ where $a$ is equal to $b$ is equal to $c$, we can rearrange the previous formula,

$$w = y \oplus a$$

and for $x$'s corresponding $z$, I can do the same:

$$x = z \oplus b$$

and for $s$'s corresponding $t$, I can do the same:

$$s = t \oplus c$$

thus there is only one corresponding combination of $y$ and one corresponding combination of $z$ and one corresponding combination of $t$, thus out of the $m^6$ combinations, only $m$ are possible, thus,

$$Pr(a\ AND\ b\ AND\ c \text{ when } x_1 \neq x_2 \neq x_3\ AND\ y_1 \neq y_2 \neq y_3) = \frac{m^3}{m^6} = \frac{1}{m^2}$$

Thus we have shown 3-uniform for this case.

CASE 2: $x_1, x_2, x_3$ are equal
So in this case I get $x_1, x_2, x_3$, out of the $m^6$ possibilites, and they are EQUAL so now I am left with only $m^3$ possible combinations, and so to find $w$'s corresponding $y$ where $a$ is equal to $b$, and $x$'s corresponding $z$ and $s$'s corresponding $t$, there is only ONE possible combination thus

$$Pr(a = b = c \text{ when } x_1 = x_2 = x_3) = \frac{1}{m^3}$$

Thus we have show 3-uniform for this case.

CASE 3: mixmatch
So in this case I get $x_1, x_2, x_3$, so here $x_1 = x_2 \neq x_3$ and $y_1 \neq y_2 = y_3$ so now I am left with only $m^4$ possible combinations for $w, y, z, s$ and so to find $y$'s corresponding $w$ where $a$ is equal to $b$ is equal to $c$, we can rearrange the previous formula,

$$y = w \oplus a$$

and for $z$'s corresponding $w$, I can do the same:

$$z = w \oplus b$$

and for $s$'s corresponding $y$, I can do the same:

$$s = y \oplus c$$

which is equal to

$$w \oplus b \oplus c$$

thus there is only one corresponding combination of $w$ and one corresponding combination of $x$, thus out of the $m^4$ combinations, only $m$ are possible, thus,

$$Pr(a = b = c \text{ when } x_1 = x_2 \neq x_3\ AND\ y_1 \neq y_2 = y_3) = \frac{m}{m^4} = \frac{1}{m^3}$$

Thus we have show 3-uniform for this case.

Thus because all cases are proved to be 3-uniform, thus proved it is 2-uniform.