# Project Member Application tactics

2026-27

For

Mathematics Club

Centre for Innovation
IIT Madras

# Instructions

## General Instructions

- Mention the following details in the start of your application.

| Name: | | Insti Name: |
|---|---|---|
| Roll: | Room: | Hostel: |
| Phone (WA): | CGPA: | Email: |

- Join the Aspiring Project TACTICS PM WhatsApp group for further updates.
- The recommended font is a standard font size 11-13.
- The applications have to be submitted in PDF format, named as:
  `<First_name>_<Roll_Number>_Mathematics_Club_TACTICS.pdf`

  For example, `Aditya_EE24B002_Mathematics_Club_TACTICS.pdf`.
- You can upload the finished applications in this Google Form.
- You may submit the completed application on or before **11:59 PM, 4th June 2026.**

## Note:

- Be concise and to the point. We prefer to see a detailed view of your thought process rather than just the final answer. Fully explained answers are far better than correct but poorly explained ones.
- It is fine even if you don't answer all the questions.
- Focus on the mandatory questions before attempting the bonus question.
- If you have any queries, you can reach out to the project leads anytime you want:
  - Pulkit Jindal (DA24B047): +91 81238 44770
  - Aditya Ramesh (EE24B002): +91 96544 76133
- We will follow a strict policy of NO LLM Usage. You may use the LLMs to help you learn the concepts but you may not use them in any way to aid your problem solving. If there is any doubt regarding the questions feel free to contact the Project leads, we will be more than happy to help.
- Adding on to the above point if someone is found we will be enforcing strict penalties on them
- The interview will be technically challenging and we urge you to prioritize depth of understanding rather than answering all the questions
- The References section will serve as your best friend when you are lost. It covers all the necessary theory for these questions. If you find any interesting resources you'd like us to add or feel something is missing from the section let us know!
- Have fun Apping and all the best!

# Contents

**1 Probability 1** **4**
1.1 Introduction . . . 4
1.2 Expected Time to Escape a Maze . . . 4
1.3 Random Distribution of Gummy Bear Bags . . . 4
1.4 Deleting Trees . . . 4
1.5 Estimating the Probability of Unseen Symbols . . . 4

**2 Probability 2** **5**
2.1 Introduction . . . 5
2.2 Buffon's Needle Problem . . . 6
2.3 Designing a Tournament . . . 6
2.4 Moment Generating Function of the Gamma Distribution . . . 7

**3 Analysis of Algorithms** **8**

**4 Hash Functions** **8**
4.1 Universal Hashing using Linear Functions . . . 8
4.2 Universal Hashing using Binary Matrices . . . 8
4.3 Hashing Distinctions . . . 9

**5 Permutations and Breaking Them** **11**
5.1 Introduction . . . 11
5.2 Inverting One-Way Functions and Hellman's Time-Space Tradeoff . . . 11

**6 Hash Collisions** **12**

**7 Bonus** **13**

**8 References** **14**

# HR Questions

We know its boring so let's finish it off first

1. Introduce yourself little bit, your hobbies, passions, or any quirks you would like us to know.
2. Why do you want to join this project? What skills or qualities do you possess that make you suitable for the work involved?
3. What other PoRs would you be taking part in in the upcoming semester? Include any applied to PoRs you're planning/have applied to as well as ones you already have.

# §1 Probability 1

## §1.1 Introduction

This Section is designed to test discrete probability, mixed with questions related to expectation and its properties. We expect you to familiarize yourself with definitions of expectation and conditional expectation, discrete distribution etc. Feel free to briefly describe your learning before beginning the section.

## §1.2 Expected Time to Escape a Maze

You find yourself in a room at the center of a maze with three exits.

- Exit 1 leads outside the maze after 2 minutes.
- Exit 2 leads back to the same room after 4 minutes.
- Exit 3 leads back to the same room after 6 minutes.

Each time you are in the room, you independently choose one of the three exits uniformly at random.
Find the expected amount of time required to leave the maze.

## §1.3 Random Distribution of Gummy Bear Bags

A total of 20 bags of Haribo gummy bears are independently distributed among 20 students in a certain Stat 110 section. Each bag is assigned uniformly at random to one of the 20 students.

1. Find the expected total number of bags received by the first three students.
2. Find the expected number of students who receive at least one bag.

## §1.4 Deleting Trees

Let $T_h$ be a perfect binary tree of height $h$, where the root is at depth 0 and every internal node has exactly two children. Thus,

$$|V(T_h)| = 2^{h+1} - 1.$$

Consider the following random deletion process:

At each discrete time step, one of the currently remaining vertices is chosen uniformly at random. Once a vertex $v$ is selected, the entire subtree rooted at $v$ (including $v$ itself and all of its descendants) is removed from the tree.

The process continues until no vertices remain.

Let $X_h$ denote the number of deletion steps required to completely erase the tree.
What is $\mathbb{E}[X_h]$

## §1.5 Estimating the Probability of Unseen Symbols

Suppose $X_1, X_2, \ldots, X_n$ are i.i.d. samples drawn from an unknown discrete distribution over a countable alphabet $\mathcal{A}$, where symbol $i \in \mathcal{A}$ occurs with probability $p_i$.

For each symbol $i$, define

$$N_i = \sum_{j=1}^{n} \mathbf{1}\{X_j = i\},$$

the number of times symbol $i$ appears in the sample.

Define the *missing mass*

$$M_n = \sum_{i \in \mathcal{A}} p_i \mathbf{1}\{N_i = 0\},$$

which represents the total probability of symbols not observed in the sample.

Also define

$$K_1 = \sum_{i \in \mathcal{A}} \mathbf{1}\{N_i = 1\},$$

the number of symbols that appear exactly once in the sample.

1. Show that
$$\mathbb{E}[M_n] = \sum_{i \in \mathcal{A}} p_i (1 - p_i)^n.$$

2. Show that
$$\mathbb{E}[K_1] = \sum_{i \in \mathcal{A}} n p_i (1 - p_i)^{n-1}.$$

3. Compare the two expressions above and explain why
$$\frac{K_1}{n}$$
may be viewed as a natural estimator for the missing mass $M_n$.

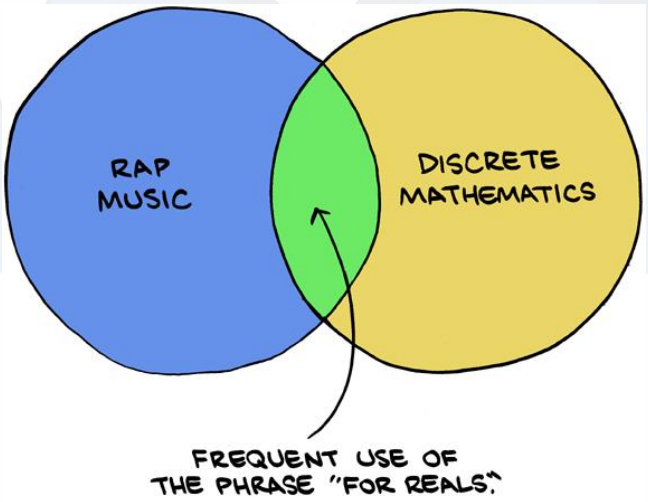


Figure 1: Trying to explain why Rap is like Math

# §2 Probability 2

## §2.1 Introduction

This section dives deeper into probability. The section is designed to test you on the concepts of continuous probability. Naturally we expect you to learn and understand the definitions of probability with continuous random variables. Further, the section will test you on concentration inequalities and Moment Generating

Functions, which are quite important to this project. Again feel free to explain your learning in these fundamental topics before attempting the questions!

## §2.2 Buffon's Needle Problem

Suppose a floor is marked with infinitely many parallel lines spaced distance $d$ apart. A needle of length $0 < \ell \leq d$ is dropped uniformly at random onto the floor.

Let:

- $X$ denote the distance from the center of the needle to the nearest line,
- $\Theta$ denote the acute angle between the needle and the parallel lines.

Assume:

$$X \sim \text{Uniform}\left(0, \frac{d}{2}\right), \qquad \Theta \sim \text{Uniform}\left(0, \frac{\pi}{2}\right),$$

independently.

1. Show that the needle intersects a line if and only if

$$X \leq \frac{\ell}{2} \sin \Theta.$$

2. Using the joint density of $X$ and $\Theta$, show that the probability that the needle crosses a line is

$$\Pr(\text{crossing}) = \frac{2\ell}{\pi d}.$$

## §2.3 Designing a Tournament

This question will help you understand the applications of Chernoff bounds and union bounds more, as you'll deal with tuning parameters to optimize the efficiency of the system. You will also get an understanding of why NBA tournaments are designed the way they are, with more repeated matches being played as we get closer to the finals. For example, in the NBA, the early rounds are *best-of-five*, and the later rounds become *best-of-seven*. You will understand why, and also learn how to design tournaments with $n$ participants for large $n$.

Suppose there are $n$ teams, and they are totally ranked. That is, there is a well-defined best team, second ranked team and so on. It's just that we (the algorithm designer) don't know the ranking.

Moreover, assume that for any given match between two players, the better ranked team will win the match with probability

$$p = \frac{1}{2} + \delta,$$

independent of all other matches between these players and all other players also. Here $\delta$ is a small positive constant.

1. Let $n$ be a power of two, and fix an arbitrary tournament tree starting with $n/2$ matches, then $n/4$ matches and so on. What is the probability that the best team wins the tournament?

2. Use Chernoff bounds to bound the probability that the best team will not win the tournament, if each match-up occurs as a *best-of-k* series. How many games do you end up conducting in total to get a $1 - \epsilon$ probability of the best team winning?

3. Now design a tournament with a total of $O_\epsilon(n)$ games to get $1 - \epsilon$ probability of the best team winning eventually. $O_\epsilon(n)$ means $O(n)$ for all constant $\epsilon > 0$.

## §2.4 Moment Generating Function of the Gamma Distribution

Let $X_1, X_2, \ldots, X_n$ be independent exponential random variables with common rate parameter $\lambda > 0$, i.e.

$$f_{X_i}(x) = \lambda e^{-\lambda x}, \qquad x \geq 0.$$

Define

$$S_n = \sum_{i=1}^{n} X_i.$$

1. Compute the moment generating function (MGF) of an exponential random variable:

$$M_X(t) = \mathbb{E}[e^{tX}],$$

and show that for $t < \lambda$,

$$M_X(t) = \frac{\lambda}{\lambda - t}.$$

2. Using independence, show that the MGF of $S_n$ is

$$M_{S_n}(t) = \left(\frac{\lambda}{\lambda - t}\right)^n.$$

3. Recall that a Gamma random variable with shape parameter $n$ and rate parameter $\lambda$ has density

$$f_Y(y) = \frac{\lambda^n}{(n-1)!} y^{n-1} e^{-\lambda y}, \qquad y \geq 0.$$

Compute the MGF of $Y$ and show that

$$M_Y(t) = \left(\frac{\lambda}{\lambda - t}\right)^n.$$

4. Using uniqueness of MGFs, conclude that

$$S_n \sim \text{Gamma}(n, \lambda).$$


Figure 2: Me After Trying Section 2

# §3 Analysis of Algorithms

We'll keep this one simple. Read up on Quick Sort and Karatsuba Algorithm and understand their time complexities and write it in your own words. (Yes not gpt's words we prefer your broken english :D). You can find both of these by a simple google search.

# §4 Hash Functions

## §4.1 Universal Hashing using Linear Functions

Let $p$ be a prime number and let $t < p$. Consider the family of hash functions

$$\mathcal{H} = \{h_{a,b}(x) = ((ax + b) \bmod p) \bmod t \mid a \in \{1, 2, \ldots, p-1\},\ b \in \{0, 1, \ldots, p-1\}\}.$$

Suppose two distinct keys $x, y \in \{0, 1, \ldots, p-1\}$ are given.

1. Show that for any fixed pair $x \neq y$, the pair

$$(h_{a,b}(x), h_{a,b}(y))$$

is uniformly distributed over all possible values as $(a, b)$ are chosen uniformly at random.

2. Prove that

$$\Pr[h_{a,b}(x) = h_{a,b}(y)] \leq \frac{1}{t}.$$

## §4.2 Universal Hashing using Binary Matrices

Let $m, n \in \mathbb{N}$ with $m < n$. Let $M$ be a uniformly random binary matrix of dimension $m \times n$ over the field $\mathbb{F}_2$. Define the hash function

$$h_M(x) = Mx,$$

where $x \in \{0, 1\}^n$ is viewed as a column vector over $\mathbb{F}_2$, and all operations are performed modulo 2.

1. Let $x \neq y$. Show that
$$h_M(x) = h_M(y)$$
if and only if
$$M(x - y) = 0.$$

2. Prove that
$$\Pr[h_M(x) = h_M(y)] = \frac{1}{2^m}.$$

## §4.3 Hashing Distinctions

1. Let
$$\mathcal{H}_k = \{h_{ab} : x \mapsto (ax + b) \bmod k \mid a, b \in \mathbb{F}_p,\ a \neq 0\}$$
be a 2-universal hash family.

Prove the following lemma.

**Lemma 1.** For every set $S \subseteq \mathbb{F}_p$,

a) if $|S| < k$, then
$$\Pr_{h \leftarrow \mathcal{H}_k}[\exists x \in S : h(x) = 0] < \frac{1}{4};$$

b) while, if $|S| \geq 2k$, then
$$\Pr_{h \leftarrow \mathcal{H}_k}[\exists x \in S : h(x) = 0] \geq \frac{3}{8}.$$

2. Consider a stream of integers
$$x_1, x_2, \ldots$$
where each $x_i \in [n]$. For a parameter $k$, where $1 \leq k \leq n$, design a streaming algorithm with the following guarantees:

a) if the stream contains strictly less than $k$ distinct elements, the output is No with probability at least
$$1 - \frac{1}{n^2},$$

b) while, if the stream contains at least $2k$ distinct elements, then the output is Yes with probability at least
$$1 - \frac{1}{n^2}.$$

The algorithm should proceed in three phases:

- Initialization: (pre-processing done before making the pass)
- Processing: ($x \in [n]$ is the element to process)
- Output: (called after the input is processed)

Fill in the details of the algorithm. You need not provide implementation details but should be precise in your description.

Prove the guarantees of the algorithm designed above and calculate the space used by the algorithm (in $O(\cdot)$ notation).

3. Using the above, design a streaming algorithm that outputs an estimate of the number of distinct elements, up to a factor 2 with probability at least

$$1 - \frac{1}{n},$$

where $n$ is a parameter passed to the algorithm.

As before, the algorithm should proceed in three phases:

- Initialization
- Processing
- Output

Prove the guarantees of your algorithm and calculate the total space used by the algorithm.


Figure 3: Hash Ception

# §5 Permutations and Breaking Them

## §5.1 Introduction

This section is the first introduction to our project and what it deals with.

## §5.2 Inverting One-Way Functions and Hellman's Time-Space Tradeoff

Modern cryptography relies heavily on the existence of *one-way functions*. Informally, a one-way function is a function that is easy to compute in the forward direction, but computationally difficult to invert.

For example, suppose

$$f : [N] \to [N]$$

is a function that can be evaluated very efficiently. Given an input $x$, computing $f(x)$ is easy. However, given only a value

$$y = f(x),$$

the task of recovering *any* $x$ such that $f(x) = y$ may be computationally difficult.

A common heuristic in cryptography is to model such a function as behaving like a completely random permutation over $[N]$. In this problem, we will study the complexity of inverting such functions, and investigate the tradeoff between:

- the amount of memory available to the attacker, and
- the amount of time required to invert the function.

Suppose $f : [N] \to [N]$ is a uniformly random permutation, and suppose you are given a challenge value

$$y = f(x),$$

where $x \in [N]$ is unknown.

Your goal is to recover $x$.

A trivial approach is exhaustive search: for every $z \in [N]$, compute $f(z)$ and check whether

$$f(z) = y.$$

This requires $T = N$ time and essentially no memory.

At the other extreme, one could precompute and store the entire lookup table

$$(x, f(x))$$

for all $x \in [N]$. Then inversion can be done in constant time using $S = N$ memory.

The central question in this problem is:

*Can one simultaneously reduce both time and space?*

1. Suppose an algorithm uses memory $S$ and inversion time $T$.

   Explain why the two trivial strategies above achieve:

   $$(S, T) = (1, N) \qquad \text{and} \qquad (S, T) = (N, 1).$$

   Then propose your own strategy for inverting the permutation using intermediate values of $S$ and $T$. Try to make the product

   $$ST$$

as small as possible.

Give a precise description of your algorithm and analyze its running time and memory usage.

2. One remarkable idea due to Hellman is to precompute chains of iterates of the permutation.

   Starting from some initial point $x_0$, define a chain:

   $$x_1 = f(x_0), \qquad x_2 = f(x_1), \qquad \dots \qquad x_t = f(x_{t-1}).$$

   Instead of storing the entire chain, Hellman's idea is to store only:

   $$(x_0, x_t).$$

   During inversion, one repeatedly applies $f$ to the target value $y$, attempting to collide with one of the stored chain endpoints.

   Explain intuitively why this method can significantly reduce memory usage while still allowing inversion of many values.

3. Suppose we construct $S$ chains, each of length $T$, and assume for simplicity that the chains do not collide with one another.

   a) Explain why the total number of points covered is approximately

   $$ST.$$

   b) Show that if

   $$ST \approx N,$$

   then a constant fraction of the permutation can be inverted successfully.

   c) Conclude that Hellman's method achieves the tradeoff

   $$ST = N.$$

   Compare this with the trivial exhaustive-search and full-table methods. What exactly did I achieve by doing this tradeoff?

## §6 Hash Collisions

1. Let $[N] = \{1, 2, 3, \dots, N\}$. Let $f : [N] \to [N]$ be a random function.

   Consider the following algorithm:

   Pick some arbitrary $x \in [N]$

Let

$$y_0 = x$$

and for $j = 1$ to $N$,

$$y_j = f(y_{j-1}).$$

Let $E_i$ be the event that

$$y_i \in \{y_0, y_1, y_2, \ldots, y_{i-1}\}$$

and

$$|\{y_0, y_1, y_2, \ldots, y_{i-1}\}| = i.$$

a) What is $\Pr[E_i]$?

b) Let $K$ be the random variable denoting the minimum $i$ such that $E_i$ is 1. What is the expectation of $K$?

2. Let $[N] = \{1, 2, 3, \ldots, N\}$. Let $f : [N] \to [N]$ be a random function.

A three-collision in $f$ is a set of 3 values

$$x_1, x_2, x_3 \in [N]$$

such that

$$|\{x_1, x_2, x_3\}| = 3$$

and

$$f(x_1) = f(x_2) = f(x_3).$$

Suppose you can query $f$ $T$ times.

Design an algorithm that finds a 3-collision in $f$ with probability

$$\frac{T^3}{N^2}.$$

Give a very precise analysis of the success probability of your algorithm.

## §7 Bonus

A hard combinatorics counting puzzle for those interested

Consider a weighted complete bipartite graph between sets

$$\{A_1, A_2, \ldots, A_n\} \quad \text{and} \quad \{B_1, B_2, \ldots, B_n\},$$

where the edge between $A_i$ and $B_j$ has weight

$$|i - j|.$$

Count the number of Hamiltonian cycles that minimize the total length of the cycle. Two cycles are considered different if an edge from one cycle is not present in the other.

## §8 References

You may refer to the "reading materials" from MIT Probability for learning the relevant probability for Section 1. Additionally, their corresponding lecture series on YouTube is also quite helpful if you feel stuck in a topic.

For Section 2, for Markov and Chebyshev inequalities you may find it helpful to refer from here, and for Chernoff bounds (until Section 3) from here.

For the remaining sections, significant theory is not required. You may find it helpful to ask the project leads or do a Google search if you are confused about what any of the terms mean.