Machine learning entrance exam 2025 — worked solutions

Generated with Lemma

November 10, 2026

Contents

Problem 1: Analyze properties of a convergent sequence of reals. **2**
Question . . . 2
Background . . . 2
Sub-Question 1: Part (a): Must the sequence be monotonic? . . . 2
Sub-Question 2: Part (b): Must the sequence have a monotonic subsequence? . . . 3
Sub-Question 3: Part (c): Must the sequence have a lower bound? . . . 3

Problem 2: Analyze the convergence of the series Σ (from n=1 to ∞) (-1)^n / √n. **5**
Question . . . 5
Background . . . 5
Sub-Question 1: Part (a): Absolute Convergence . . . 5
Sub-Question 2: Part (b): Monotonicity of Partial Sums . . . 6
Sub-Question 3: Part (c): Convergence of Partial Sums . . . 7

Problem 3: Analyze properties of a full-rank square matrix. **9**
Question . . . 9
Background . . . 9
Sub-Question 1: Part (a): Image of A . . . 9
Sub-Question 2: Part (b): Image of A^T . . . 10
Sub-Question 3: Part (c): A^T as a Monomorphism . . . 10

Problem 4: Analyze a diagonal matrix with roots of unity. **12**
Question . . . 12
Background . . . 12
Sub-Question 1: Part (a): Nonsingularity of A . . . 12
Sub-Question 2: Part (b): Eigenvalue 1 . . . 13
Sub-Question 3: Part (c): Eigenvalue -1 . . . 14

Problem 5: Calculate dimensions of spaces of linear maps. **15**
Question . . . 15
Background . . . 15
Sub-Question 1: **Part (a):** Analysis of dim $L(X, Y)$ . . . 15
Sub-Question 2: **Part (b):** Analysis of dim $L(Y, X)$ . . . 16
Sub-Question 3: **Part (c):** Analysis of a Nested Space of Maps . . . 16

Problem 6: Analyze properties of complex numbers. **18**
Question . . . 18
Background . . . 18
Sub-Question 1: **Part (a):** if $|a| = |b|$, then $a = b$ or $a = -b$ . . . 18
Sub-Question 2: **Part (b):** if $|a + b| = 0$, then $a = b = 0$ . . . 19
Sub-Question 3: **Part (c):** if $a^2 + b^2 = 0$, then $a = b = 0$ . . . 20

Problem 7: Analyze the cardinality of an inverse image. **21**
Question . . . 21
Background . . . 21
Sub-Question 1: Part (a): Is $|f^{-1}(A)| = |A|$ always true? . . . 21
Sub-Question 2: Part (b): Is $|f^{-1}(A)| \leq |A|$ always true? . . . 22
Sub-Question 3: Part (c): Is $|f^{-1}(A)| \geq |A|$ always true? . . . 22

Problem 9: Evaluate a nested summation. **24**
Question . . . 24
Background . . . 24
Sub-Question 1: Variant (a): Comparison with $\binom{n}{3}$ . . . 24
Sub-Question 2: Variant (b): Comparison with $\binom{n+2}{3}$ . . . 26
Sub-Question 3: Variant (c): Comparison with $\binom{n+3}{3}$ . . . 26

Problem 12: Analyze divisibility properties of binomial coefficients. **27**
Question . . . 27
Background . . . 27
Sub-Question 1: Part (a): n is divisible by 2 or 3 . . . 27
Sub-Question 2: Part (b): $C(n, 5)$ is divisible by 5 . . . 28
Sub-Question 3: Part (c): $C(n, 6)$ is divisible by 6 . . . 28

Problem 13: Calculate conditional and joint probabilities. **30**
Question . . . 30
Background . . . 30
Sub-Question 1: Part (a): Joint Probability from Conditional Probability 30
Sub-Question 2: Part (b): Union of Partitioning Events . . . 31
Sub-Question 3: Part (c): Bayesian Inference and the Law of Total Probability . . . 31

Problem 14: Analyze a condition for independence of random variables. **33**

Question . . . 33
Background . . . 33
Sub-Question 1: **Variant (a):** Equality of Probability Products . . . 34
Sub-Question 2: **Variant (b):** Linearity of Expectation . . . 34
Sub-Question 3: **Variant (c):** Independence of Random Variables . 35

Problem 15: Analyze a three-state time-homogeneous Markov chain. 36
Question . . . 36
Background . . . 36
Sub-Question 1: Part (a): Is the mean recurrence time for state 3 less than or equal to 6? . . . 36
Sub-Question 2: Part (b): Is the mean recurrence time for state 3 greater than or equal to 6? . . . 37
Sub-Question 3: Part (c): Is this Markov chain aperiodic? . . . 38

Problem 16: Analyze the distribution of a transformed random variable. 39
Question . . . 39
Background . . . 39
Sub-Question 1: Part (a): Is $f_Y(t) = f_X(-t)$ almost everywhere? . . . 39
Sub-Question 2: Part (b): Is $F_Y(t) = 1 - F_X(-t)$? . . . 40
Sub-Question 3: Part (c): Is $E(Y) = -E(X)$? . . . 41

Problem 17: Interpret machine learning test metrics. 43
Question . . . 43
Background . . . 43
Sub-Question 1: Variant (a): Identification of Healthy Patients . . . 43
Sub-Question 2: Variant (b): Health Status of Patients Classified as Healthy . . . 44
Sub-Question 3: Variant (c): Overall Classification Accuracy . . . 45

Problem 18: Analyze the standard k-means algorithm. 47
Question . . . 47
Background . . . 47
Sub-Question 1: Variant (a): a partition of observations into k groups with the lowest possible variances of distances within groups . . . 47
Sub-Question 2: Variant (b): a set of groups such that their middle points (centroids) are observations from the input data . . . 48
Sub-Question 3: Variant (c): the same results in different runs on the same input data . . . 49

Problem 19: Characterize the Kolmogorov-Smirnov test. 50
Question . . . 50
Background . . . 50

Sub-Question 1: Part (a): Testing against a fully specified normal distribution . . . 50
Sub-Question 2: Part (b): Testing against a normal distribution with estimated parameters . . . 51
Sub-Question 3: Part (c): Assumption of finite variance . . . 52

Problem 20: Define the False Discovery Rate (FDR). 53
Question . . . 53
Background . . . 53
Sub-Question 1: Variant (a): Probability of at least one Type I error . 53
Sub-Question 2: Variant (b): Expected proportion of false rejections . 54
Sub-Question 3: Variant (c): Product of p-values and number of tests 55

Problem 10: Count triangles in a specific graph. 56
Question . . . 56
Background . . . 56
Sub-Question 1: Calculate the total number of triangles and evaluate the given options. . . . 56

Problem 11: Solve a combinatorial distribution problem. 59
Question . . . 59
Background . . . 59
Sub-Question 1: Variant (a): The number of ways is 20. . . . 59
Sub-Question 2: Variant (b): The number of ways is $\binom{13}{3}$. . . . 60
Sub-Question 3: Variant (c): The number of ways is 35. . . . 60

Problem 21: Analyze properties of n-node red-black trees. 62
Question . . . 62
Background . . . 62
Sub-Question 1: Part (a): The number of red nodes is greater by 1000 than the number of non-null black nodes. . . . 62
Sub-Question 2: Part (b): The number of non-null black nodes is 3 times greater than the number of red nodes. . . . 63
Sub-Question 3: Part (c): The number of nodes on the longest path from the root to a leaf is 1.5 times greater than the number of nodes on the shortest path. . . . 64

Problem 22: Analyze disjoint-set (union-find) forest complexity. 66
Question . . . 66
Background . . . 66
Sub-Question 1: Variant (a): Tree Structure . . . 66
Sub-Question 2: Variant (b): Single Operation Worst-Case Time . . . 67
Sub-Question 3: Variant (c): Total Time for a Sequence of Operations 68

Problem 23: Analyze an SQL GROUP BY query. 69
Question . . . 69
Background . . . 69

Sub-Question 1: a) The number of rows returned is greater than or equal to the number of rows in T . . . 69
Sub-Question 2: b) The number of rows returned is less than or equal to the number of rows in T . . . 70
Sub-Question 3: c) The sum of all values in the second column of the result is equal to the sum of all values in column B of table T . . 70

Problem 24: Analyze a mutual exclusion algorithm. 72
Question . . . 72
Background . . . 72
Sub-Question 1: (a) Does the solution ensure mutual exclusion? . . . 73
Sub-Question 2: (b) May the solution lead to a deadlock? . . . 74
Sub-Question 3: (c) Does the solution ensure alternation? . . . 74

Problem 25: Analyze message reception in communication models. 76
Question . . . 76
Background . . . 76
Sub-Question 1: Variant (a): Direct Asynchronous Communication . . 76
Sub-Question 2: Variant (b): Direct Synchronous Communication . . 77
Sub-Question 3: Variant (c): Indirect Communication via a Tuple Space 78

Problem 26: Analyze Python code with list aliasing. 80
Question . . . 80
Background . . . 80
Sub-Question 1: Variant a . . . 80
Sub-Question 2: Variant b . . . 81
Sub-Question 3: Variant c . . . 82

Problem 27: Analyze Python code with variable scope. 83
Question . . . 83
Background . . . 83
Sub-Question 1: Variant (a): Calling `f1()` . . . 84
Sub-Question 2: Variant (b): Calling `f2()` . . . 84
Sub-Question 3: Variant (c): Calling `f3()` . . . 85

Problem 28: Analyze Python code with OOP and name mangling. 87
Question . . . 87
Background . . . 87
Sub-Question 1: Variant (a): `print(a.x)` . . . 87
Sub-Question 2: Variant (b): `print(a.__x)` . . . 88

Problem 29: Classify IP addresses. 89
Question . . . 89
Background . . . 89
Sub-Question 1: Variant (a): 10.10.0.1 is globally routable . . . 89
Sub-Question 2: Variant (b): 127.0.0.1 is a loopback address . . . 90
Sub-Question 3: Variant (c): 255.255.255.255 is a broadcast address . 91

Problem 30: Interpret memory values based on endianness. **92**
Question 92
Background 92
Sub-Question 1: Part (a): Big-Endian at Address A 92
Sub-Question 2: Part (b): Little-Endian at Address A+2 93
Sub-Question 3: Part (c): x86 Architecture at Address A+4 93

Problem 1: Analyze properties of a convergent sequence of reals.

Question

Let (a_n) be a convergent sequence of real numbers. Is it true that this sequence must:

(a) be monotonic?

(b) have a monotonic subsequence?

(c) have a lower bound?

Background

This problem tests fundamental properties of convergent sequences in real analysis. A sequence of real numbers (a_n) is said to converge to a limit $L \in \mathbb{R}$ if, for every $\epsilon > 0$, there exists a natural number N such that for all $n > N$, we have $|a_n - L| < \epsilon$. The questions probe the relationship between convergence and two other key properties: monotonicity and boundedness.

A sequence is monotonic if it is either non-decreasing ($a_n \leq a_{n+1}$ for all n) or non-increasing ($a_n \geq a_{n+1}$ for all n). A sequence is **bounded** if there exists a number $M > 0$ such that $|a_n| \leq M$ for all n. Being bounded implies having both an upper and a lower bound.

Sub-Question 1: Part (a): Must the sequence be monotonic?

Does convergence imply that the sequence (a_n) is monotonic?

Step 1

The answer is **NO**.

Step 2

Justification through counterexample. The core idea is to find a sequence that approaches a limit but does not do so in a consistently increasing or decreasing manner. An oscillating sequence whose oscillations decay is a perfect candidate.

Consider the sequence $a_n = \frac{(-1)^n}{n}$ for $n \geq 1$.

- **Convergence**: The sequence converges to 0. This is because $\lim_{n\to\infty} |a_n| = \lim_{n\to\infty} \frac{1}{n} = 0$, which implies $\lim_{n\to\infty} a_n = 0$.
- **Monotonicity**: The sequence is not monotonic. Let's examine the first few terms: $a_1 = -1$, $a_2 = \frac{1}{2}$, $a_3 = -\frac{1}{3}$. We have $a_1 < a_2$ but $a_2 > a_3$.

Since the direction of the inequality changes, the sequence is neither non-decreasing nor non-increasing.

Since we have found a sequence that is convergent but not monotonic, the statement that convergence implies monotonicity is false.

Sub-Question 2: Part (b): Must the sequence have a monotonic subsequence?

Does convergence imply that the sequence (a_n) must have a monotonic subsequence?

Step 1

The answer is **YES**.

Step 2

This is a direct consequence of the **Monotone Subsequence Theorem**.

The theorem states that every sequence of real numbers has a monotonic subsequence.

The non-trivial insight here is recognizing the scope of this theorem. It is a universal property of real sequences. It applies to *all* sequences, whether they are convergent, divergent, bounded, or unbounded. Since the given sequence (a_n) is a sequence of real numbers, the theorem applies directly. The property of convergence is, in this case, extra information that is not needed to answer the question.

Therefore, any convergent sequence, by virtue of being a real sequence, must contain a monotonic subsequence.

Sub-Question 3: Part (c): Must the sequence have a lower bound?

Does convergence imply that the sequence (a_n) must have a lower bound?

Step 1

The answer is **YES**.

Step 2

This follows from the theorem that **every convergent sequence is bounded**. A bounded sequence, by definition, has both an upper and a lower bound.

The non-trivial logical step is to understand *why* convergence implies boundedness. This can be shown by construction from the definition of a limit.

1. Let the sequence (a_n) converge to a limit L. By the definition of convergence, for a chosen ϵ, there is an index N after which all terms of the sequence are close to L. Let's fix $\epsilon = 1$. Then there exists some $N \in \mathbb{N}$ such that for all $n > N$, we have $|a_n - L| < 1$.

2. This inequality, $|a_n - L| < 1$, is equivalent to $L - 1 < a_n < L + 1$. This establishes that all terms of the sequence's 'tail' (i.e., for $n > N$) are bounded below by $L - 1$.

3. Now, consider the 'head' of the sequence: the finite set of terms $\{a_1, a_2, \ldots, a_N\}$. Since this is a finite set of real numbers, it must have a minimum value. Let $m = \min\{a_1, a_2, \ldots, a_N\}$.

4. To find a lower bound for the *entire* sequence, we take the minimum of the lower bound for the head and the lower bound for the tail. Thus, a lower bound for the entire sequence (a_n) is given by $\min\{m, L-1\}$. Since we have successfully constructed a lower bound, one must exist.

Problem 2: Analyze the convergence of the series Σ (from n=1 to ∞) (-1)^n / √n.

Question

Consider the series $S = \sum_{n=1}^{\infty} \frac{(-1)^n}{\sqrt{n}}$. Determine whether the following statements are true:

(a) The series is absolutely convergent.

(b) The series has an increasing sequence of partial sums.

(c) The series has a convergent sequence of partial sums.

Background

This problem tests the understanding of infinite series convergence, a fundamental concept in real analysis. The key is to distinguish between different types of convergence and the tests used to establish them. The series in question is an alternating series, which requires a specific set of tools for analysis.

1. Absolute vs. Conditional Convergence: A series $\sum a_n$ is **absolutely convergent** if the series of its absolute values, $\sum |a_n|$, converges. A crucial theorem states that absolute convergence implies convergence. However, the converse is not true. A series that converges but does not converge absolutely is called **conditionally convergent**. This distinction is the non-trivial core of the problem.

2. The p-Series Test: This is a standard test for series of the form $\sum_{n=1}^{\infty} \frac{1}{n^p}$. The series converges if and only if $p > 1$. It diverges for $p \leq 1$. This test is essential for analyzing the absolute convergence of the given series.

3. The Alternating Series Test (Leibniz's Test): For an alternating series $\sum(-1)^n b_n$ (or $\sum(-1)^{n+1} b_n$), if the sequence of magnitudes b_n satisfies three conditions: (i) $b_n > 0$ for all n, (ii) $b_{n+1} \leq b_n$ (it is non-increasing), and (iii) $\lim_{n\to\infty} b_n = 0$, then the series converges. The non-trivial intuition is that the partial sums oscillate, but the amplitude of oscillation shrinks to zero, forcing convergence.

4. Sequence of Partial Sums: The convergence of a series $\sum a_n$ is **defined** as the convergence of its sequence of partial sums, $S_k = \sum_{n=1}^{k} a_n$. Questions about the properties of this sequence are direct questions about the series itself.

Sub-Question 1: Part (a): Absolute Convergence

Is the series absolutely convergent?

Step 1

The answer is **NO**.

Step 2

Justification: To test for absolute convergence, we must examine the convergence of the series of the absolute values of the terms. The terms of our series are $a_n = \frac{(-1)^n}{\sqrt{n}}$.

Step 3

The series of absolute values is:

$$\sum_{n=1}^{\infty} \left| \frac{(-1)^n}{\sqrt{n}} \right| = \sum_{n=1}^{\infty} \frac{1}{\sqrt{n}} = \sum_{n=1}^{\infty} \frac{1}{n^{1/2}}$$

Step 4

This is a p-series of the form $\sum \frac{1}{n^p}$ with $p = 1/2$. According to the p-series test, the series diverges if $p \leq 1$. Since $1/2 \leq 1$, the series of absolute values diverges.

Step 5

The non-trivial logical step is recognizing that the failure of the absolute series to converge directly answers the question. Since $\sum |a_n|$ diverges, the original series is not absolutely convergent.

Sub-Question 2: Part (b): Monotonicity of Partial Sums

Does the series have an increasing sequence of partial sums?

Step 1

The answer is **NO**.

Step 2

Justification: Let $S_k = \sum_{n=1}^{k} a_n$ be the sequence of partial sums. For this sequence to be increasing, we must have $S_{k+1} > S_k$ for all $k \geq 1$. The difference between consecutive partial sums is simply the next term in the series: $S_{k+1} - S_k = a_{k+1}$. Therefore, for S_k to be increasing, all terms a_{k+1} must be positive.

Step 3

The terms of our series, $a_n = (-1)^n/\sqrt{n}$, are not all positive. They alternate in sign. For example, $a_1 = -1$, $a_2 = 1/\sqrt{2}$, $a_3 = -1/\sqrt{3}$, and so on.

Step 4

Let's demonstrate with the first few partial sums:

- $S_1 = a_1 = -1$
- $S_2 = S_1 + a_2 = -1 + \frac{1}{\sqrt{2}} \approx -0.293$. Here, $S_2 > S_1$.
- $S_3 = S_2 + a_3 = -1 + \frac{1}{\sqrt{2}} - \frac{1}{\sqrt{3}} \approx -0.870$. Here, $S_3 < S_2$.

Step 5

Since the sequence of partial sums both increases and decreases, it is not an increasing sequence. The non-trivial insight is that this is a general property of any alternating series whose terms are non-zero: its sequence of partial sums can never be monotonic.

Sub-Question 3: Part (c): Convergence of Partial Sums

Does the series have a convergent sequence of partial sums?

Step 1

The answer is **YES**.

Step 2

Justification: This question is a direct rephrasing of "Does the series converge?". The series is $\sum_{n=1}^{\infty}(-1)^n \frac{1}{\sqrt{n}}$. This is an alternating series. We can apply the Alternating Series Test with $b_n = \frac{1}{\sqrt{n}}$. We must check the three conditions:

1. **Positivity:** $b_n = \frac{1}{\sqrt{n}} > 0$ for all $n \geq 1$. This is clearly true.
2. **Non-increasing:** We need to show that $b_{n+1} \leq b_n$. This means $\frac{1}{\sqrt{n+1}} \leq \frac{1}{\sqrt{n}}$. Since $n + 1 > n$ and the square root function is increasing, we have $\sqrt{n+1} > \sqrt{n}$. Taking the reciprocal reverses the inequality, so the condition holds.
3. **Limit is zero:** We must check if $\lim_{n\to\infty} b_n = 0$. Indeed, $\lim_{n\to\infty} \frac{1}{\sqrt{n}} = 0$.

Step 3

Since all three conditions of the Alternating Series Test are satisfied, the series converges. By definition, this means its sequence of partial sums converges.

Step 4

The final non-trivial insight comes from combining the results of (a) and (c). The series converges (from part c), but it does not converge absolutely (from part a). Therefore, the series is a classic example of a ***conditionally convergent*** series. This demonstrates a complete understanding of the hierarchy of series convergence.

Problem 3: Analyze properties of a full-rank square matrix.

Question

Let A be a 2025x2025 matrix over $\mathbb{R}$ with rank 2025. Determine if the following statements are true or false:

Background

This problem tests the understanding of the fundamental properties of a square matrix having full rank. For an $n \times n$ matrix, having full rank (i.e., $\text{rank}(A) = n$) is a powerful condition that implies a host of other equivalent properties. These equivalences are collectively known as the Invertible Matrix Theorem. The given matrix A is 2025×2025 and its rank is 2025, so it satisfies these conditions.

Key concepts involved are:

1. Rank: The rank of a matrix is the dimension of its column space and is also equal to the dimension of its row space. It represents the number of linearly independent columns (or rows) of the matrix.

2. Image (Column Space): The image of a matrix A, denoted $\text{Im}(A)$, is the set of all possible outputs $A\mathbf{x}$. This is precisely the span of the columns of A, also known as the column space, $C(A)$.

3. Monomorphism: A linear transformation is a monomorphism if and only if it is injective (one-to-one). For a transformation represented by a matrix M, this is equivalent to its kernel (or null space) being the trivial space $\{\mathbf{0}\}$, which means the only solution to $M\mathbf{x} = \mathbf{0}$ is $\mathbf{x} = \mathbf{0}$.

Sub-Question 1: Part (a): Image of A

Is the image of A equal to $\mathbb{R}^{2025}$?

Step 1

Answer: YES

Step 2

The image of a matrix A, $\text{Im}(A)$, is defined as its column space, $C(A)$. The rank of a matrix is, by definition, the dimension of its column space. That is, $\text{rank}(A) = \dim(\text{Im}(A))$.

Step 3

We are given that A is a 2025×2025 matrix and $\text{rank}(A) = 2025$. Therefore, $\dim(\text{Im}(A)) = 2025$.

Step 4

The image of A is a subspace of $\mathbb{R}^{2025}$. A fundamental result in linear algebra states that if a subspace S of a finite-dimensional vector space V has the same dimension as V, then the subspace must be the entire space, i.e., $S = V$. In this case, $\text{Im}(A)$ is a 2025-dimensional subspace of $\mathbb{R}^{2025}$, so it must be that $\text{Im}(A) = \mathbb{R}^{2025}$.

Sub-Question 2: Part (b): Image of A ^{T}

Is the image of A^T equal to $\mathbb{R}^{2025}$?

Step 1

Answer: YES

Step 2

The non-trivial jump in logic here relies on a cornerstone theorem of linear algebra: for any matrix A, its rank is equal to the rank of its transpose.

$$\text{rank}(A) = \text{rank}(A^T)$$

This holds because the dimension of the column space of a matrix (column rank) is always equal to the dimension of its row space (row rank). The row space of A is the column space of A^T.

Step 3

Given $\text{rank}(A) = 2025$, it immediately follows that $\text{rank}(A^T) = 2025$.

Step 4

The matrix A^T is also a 2025×2025 matrix with full rank. We can now apply the exact same logic from Part (a) to the matrix A^T. The image of A^T is a subspace of $\mathbb{R}^{2025}$, and its dimension is $\dim(\text{Im}(A^T)) = \text{rank}(A^T) = 2025$. Therefore, $\text{Im}(A^T) = \mathbb{R}^{2025}$.

Sub-Question 3: Part (c): A^T as a Monomorphism

Is A^T the matrix of a monomorphism from $\mathbb{R}^{2025}$ to $\mathbb{R}^{2025}$?

Step 1

Answer: YES

Step 2

A linear transformation is a monomorphism if it is injective. For a linear map $T : V \to W$, injectivity is equivalent to the condition that its kernel is trivial, i.e., $\text{Ker}(T) = \{\mathbf{0}\}$. The dimension of the kernel is called the nullity.

Step 3

The key logical step is to connect the rank of the matrix to its nullity using the Rank-Nullity Theorem. For any linear map from an n -dimensional space represented by a matrix M, the theorem states:

$$\text{rank}(M) + \text{nullity}(M) = n$$

Step 4

In our case, the matrix is $M = A^T$, and it maps from $\mathbb{R}^{2025}$ to $\mathbb{R}^{2025}$, so $n = 2025$. From Part (b), we know that $\text{rank}(A^T) = 2025$. Substituting these values into the theorem:

$$2025 + \text{nullity}(A^T) = 2025$$

Step 5

This implies that $\text{nullity}(A^T) = 0$. A nullity of 0 means the kernel of A^T contains only the zero vector. This is the condition for injectivity. Therefore, the linear transformation represented by A^T is a monomorphism.

Problem 4: Analyze a diagonal matrix with roots of unity.

Question

Let A be a 2025×2025 diagonal matrix over $\mathbb{C}$ whose diagonal elements are all the 2025 -th roots of unity. Determine if:

(a) A is nonsingular.

(b) 1 is an eigenvalue of A.

(c) -1 is an eigenvalue of A.

Background

This problem tests fundamental properties of diagonal matrices, eigenvalues, and complex numbers, specifically the roots of unity. The key concepts are:

1. n-th Roots of Unity: For a positive integer n, the n -th roots of unity are the complex numbers z that satisfy the equation $z^n = 1$. There are exactly n distinct roots, given by $\omega_k = e^{2\pi ik/n}$ for $k = 0, 1, \ldots, n-1$.

2. Properties of Diagonal Matrices: For a diagonal matrix, the eigenvalues are precisely the entries on its main diagonal. The determinant of the matrix is the product of these diagonal entries.

3. Nonsingular Matrices: A square matrix is nonsingular (or invertible) if and only if its determinant is non-zero. This is equivalent to stating that 0 is not an eigenvalue of the matrix.

Sub-Question 1: Part (a): Nonsingularity of A

Is the matrix A nonsingular?

Step 1

A matrix is nonsingular if its determinant is non-zero. For the given diagonal matrix A, the determinant is the product of its diagonal elements. The diagonal elements are the $n = 2025$ distinct n -th roots of unity.

The non-trivial step is to compute this product. The n -th roots of unity are the roots of the polynomial $P(z) = z^n - 1$. By Vieta's formulas, the product of the roots of a polynomial $a_n z^n + \cdots + a_1 z + a_0$ is $(-1)^n a_0/a_n$. For $P(z)$, we have $a_n = 1$ and $a_0 = -1$. Thus, the product of the roots is $(-1)^n(-1)/1 = -(-1)^n = (-1)^{n+1} = (-1)^{n-1}$.

Alternatively, using the exponential form $\omega_k = e^{2\pi ik/n}$:

$$\det(A) = \prod_{k=0}^{n-1} \omega_k = \prod_{k=0}^{n-1} e^{2\pi ik/n} =$$
$$\exp\left(\sum_{k=0}^{n-1} \frac{2\pi ik}{n}\right) = \exp\left(\frac{2\pi i}{n} \sum_{k=0}^{n-1} k\right)$$

The sum of the first $n-1$ non-negative integers is $\frac{(n-1)n}{2}$. Substituting this into the exponent:

$$\det(A) = \exp\left(\frac{2\pi i}{n} \frac{n(n-1)}{2}\right) = \exp(i\pi(n-1)) = (e^{i\pi})^{n-1} = (-1)^{n-1}$$

For our specific case, $n = 2025$, which is an odd number. Therefore, $n-1 = 2024$ is even.

$$\det(A) = (-1)^{2024} = 1$$

Since $\det(A) = 1 \neq 0$, the matrix A is nonsingular.

Answer: YES

Sub-Question 2: Part (b): Eigenvalue 1

Is 1 an eigenvalue of A?

Step 1

The eigenvalues of a diagonal matrix are its diagonal entries. The diagonal entries of A are the 2025 -th roots of unity. Therefore, the question is equivalent to asking whether 1 is a 2025 -th root of unity.

A complex number z is a 2025 -th root of unity if it satisfies $z^{2025} = 1$. We check this for $z = 1$:

$$1^{2025} = 1$$

The condition is satisfied. In fact, for any integer $n \geq 1$, 1 is always an n -th root of unity (corresponding to $k = 0$ in the formula $e^{2\pi ik/n}$).

Since 1 is a 2025-th root of unity, it must be one of the diagonal entries of A, and therefore it is an eigenvalue of A.

Answer: YES

Sub-Question 3: Part (c): Eigenvalue -1

Is -1 an eigenvalue of A?

Step 1

Similar to part (b), this question is equivalent to asking whether -1 is a 2025 -th root of unity. We must check if $(-1)^{2025} = 1$.

The exponent, 2025, is an odd number. For any odd integer m, $(-1)^m = -1$. Therefore:

$$(-1)^{2025} = -1$$

Since $-1 \neq 1$, -1 is not a 2025 -th root of unity.

The non-trivial logical jump is understanding the general condition: -1 is an n -th root of unity if and only if n is an even integer. This is because $-1 = e^{i\pi}$, so we need $(-1)^n = (e^{i\pi})^n = e^{in\pi} = 1$. This holds if and only if $n\pi$ is an even multiple of π, which means n must be even.

Since -1 is not a 2025-th root of unity, it is not a diagonal element of A and therefore not an eigenvalue.

Answer: NO

Problem 5: Calculate dimensions of spaces of linear maps.

Question

Let $L(U, V)$ be the space of linear maps from a vector space U to a vector space V. Let X and Y be linear spaces over $\mathbb{R}$ with $\dim X = 4$ and $\dim Y = 7$. Determine if the following statements are true:

(a) $\dim L(X, Y) = 28$

(b) $\dim L(Y, X) = 11$

(c) $\dim L(L(Y, Y), L(X, \mathbb{R})) = 196$

Background

This problem tests the understanding of the dimension of a vector space of linear transformations. The key insight is that for finite-dimensional vector spaces U and V over a field F, the set of all linear maps from U to V, denoted $L(U, V)$ (or $\mathrm{Hom}(U, V)$), itself forms a vector space over F. The dimension of this space is given by a fundamental formula:

$$\dim(L(U, V)) = (\dim U) \cdot (\dim V)$$

The non-trivial justification for this formula lies in the connection between linear maps and matrices. A linear map $T : U \to V$ is completely determined by its action on a basis of U. Let $\{u_1, \ldots, u_n\}$ be a basis for U (so $\dim U = n$) and $\{v_1, \ldots, v_m\}$ be a basis for V (so $\dim V = m$). For each basis vector u_j, its image $T(u_j)$ is a vector in V and can be uniquely expressed as a linear combination of the basis vectors of V. This requires m coefficients. Since there are n basis vectors in U, a total of $n \cdot m$ coefficients are needed to define the map T. These coefficients form an $m \times n$ matrix, establishing an isomorphism between $L(U, V)$ and the space of $m \times n$ matrices, $M_{m,n}(\mathbb{R})$, whose dimension is mn.

Part (c) involves the vector space $\mathbb{R}$. When considered as a vector space over itself, $\dim(\mathbb{R}) = 1$. The space $L(X, \mathbb{R})$ is the dual space of X, denoted X^*, and its dimension is equal to the dimension of X.

Sub-Question 1: Part (a): Analysis of dim $L(X, Y)$

Is the statement $\dim L(X, Y) = 28$ true?

Step 1

We directly apply the dimension formula for spaces of linear maps: $\dim(L(U, V)) = (\dim U) \cdot (\dim V)$.

Step 2

Substitute the given dimensions for X and Y: $\dim X = 4$ and $\dim Y = 7$.

$$\dim L(X, Y) = (\dim X) \cdot (\dim Y) = 4 \cdot 7 = 28$$

Step 3

The calculated dimension is 28, which matches the value given in the statement.

Step 4

Answer: YES

Sub-Question 2: Part (b): Analysis of dim $L(Y, X)$

Is the statement $\dim L(Y, X) = 11$ true?

Step 1

Again, we apply the dimension formula, but with the roles of the domain and codomain spaces reversed.

$$\dim L(Y, X) = (\dim Y) \cdot (\dim X) = 7 \cdot 4 = 28$$

Step 2

The calculated dimension is 28. The statement claims the dimension is 11. This is incorrect.

Step 3

The non-trivial insight here is recognizing a common pitfall. The value 11 comes from adding the dimensions ($7 + 4 = 11$). The dimension of a direct sum of vector spaces is the sum of their dimensions (i.e., $\dim(X \oplus Y) = \dim X + \dim Y$), but this rule does not apply to the space of linear maps, which involves multiplication.

Step 4

Answer: NO

Sub-Question 3: Part (c): Analysis of a Nested Space of Maps

Is the statement $\dim L(L(Y, Y), L(X, \mathbb{R})) = 196$ true?

Step 1

This problem involves a nested application of the dimension formula. Let's define $U = L(Y, Y)$ and $V = L(X, \mathbb{R})$. The problem asks for dim $L(U, V)$. We must first compute the dimensions of the domain space U and the codomain space V.

Step 2

Step 1: Calculate the dimension of the domain space, $U = L(Y, Y)$. This is the space of linear operators (or endomorphisms) on Y.

$$\dim(U) = \dim(L(Y, Y)) = (\dim\ Y) \cdot (\dim\ Y) = 7 \cdot 7 = 49$$

Step 3

Step 2: Calculate the dimension of the codomain space, $V = L(X, \mathbb{R})$. This is the dual space of X. We use the fact that $\mathbb{R}$ as a vector space over itself has dimension 1.

$$\dim(V) = \dim(L(X, \mathbb{R})) = (\dim\ X) \cdot (\dim\ \mathbb{R}) = 4 \cdot 1 = 4$$

Step 4

Step 3: Calculate the final dimension using the dimensions of U and V.

$$\dim(L(U, V)) = (\dim\ U) \cdot (\dim\ V) = 49 \cdot 4 = 196$$

Step 5

The calculated dimension is 196, which matches the statement.

Step 6

Answer: YES

Problem 6: Analyze properties of complex numbers.

Question

For all complex numbers a and b, determine if the following implications hold:

(a) if $|a| = |b|$, then $a = b$ or $a = -b$

(b) if $|a + b| = 0$, then $a = b = 0$

(c) if $a^2 + b^2 = 0$, then $a = b = 0$

Background

This problem tests the understanding of fundamental properties of complex numbers. The key to solving these variants is to avoid the common pitfall of applying properties from real numbers directly to the complex domain. The behavior of the modulus, addition, and squaring differs in crucial ways.

1. Modulus ($|z|$): For a complex number $z = x + iy$, the modulus $|z| = \sqrt{x^2 + y^2}$ represents its distance from the origin in the complex plane. This is a crucial geometric interpretation. While for real numbers, only two points (x and $-x$) share the same positive modulus, in the complex plane, all points on a circle of radius r centered at the origin have modulus r.

2. Zero Modulus Property: The property $|z| = 0 \iff z = 0$ holds for all complex numbers. This is a foundational property used in analysis.

3. Squares of Complex Numbers: For any non-zero real number x, $x^2 > 0$. This is not true for complex numbers. The existence of the imaginary unit i, defined by $i^2 = -1$, means that the sum of squares $a^2 + b^2$ can be zero even if a and b are non-zero.

Sub-Question 1: Part (a): if $|a| = |b|$, then $a = b$ or $a = -b$

Does having an equal modulus imply that two complex numbers are either identical or additive inverses?

Step 1

Answer: NO.

Step 2

The condition $|a| = |b|$ has a geometric interpretation: the complex numbers a and b are equidistant from the origin in the complex plane. If this distance is $r > 0$, then both a and b lie on the circle of radius r centered at the origin.

Step 3

The non-trivial logical step is recognizing that this circle contains infinitely many points. The conclusion $a = b$ or $a = -b$ would only be true if the circle contained only two points (or one if $r = 0$). This is true for real numbers (which lie on a 1D line), but not for complex numbers (which lie on a 2D plane).

Step 4

Counterexample: Let $a = 1$ and $b = i$. We calculate their moduli:

$$|a| = |1| = \sqrt{1^2 + 0^2} = 1$$
$$|b| = |i| = |0 + 1i| = \sqrt{0^2 + 1^2} = 1$$

So, $|a| = |b| = 1$. However, it is clear that $1 \neq i$ and $1 \neq -i$. The premise is true, but the conclusion is false. Therefore, the implication does not hold.

Sub-Question 2: Part (b): if $|a + b| = 0$, then $a = b = 0$

Does the sum of two complex numbers having a modulus of zero imply that both numbers must be zero?

Step 1

Answer: NO.

Step 2

The core property to use here is that for any complex number z, $|z| = 0$ if and only if $z = 0$.

Applying this to the premise, $|a + b| = 0$ is logically equivalent to $a + b = 0$.

Step 3

The non-trivial jump is to realize the statement's logical flaw. It claims that $a + b = 0$ implies $a = 0$ and $b = 0$. However, $a + b = 0$ simply means that $a = -b$. This is true for any pair of additive inverses, not just the trivial pair (0, 0).

Step 4

Counterexample: Let $a = 1$ and $b = -1$. Then $a + b = 1 + (-1) = 0$. The modulus is $|a+b| = |0| = 0$. The premise is satisfied. However, $a \neq 0$ and $b \neq 0$, so the conclusion is false.

Sub-Question 3: Part (c): if $a^2 + b^2 = 0$, then $a = b = 0$

If the sum of the squares of two complex numbers is zero, must both numbers be zero?

Step 1

Answer: NO.

Step 2

This statement is true for real numbers. If $a, b \in \mathbb{R}$, then $a^2 \geq 0$ and $b^2 \geq 0$. The only way their sum can be zero is if both terms are zero, i.e., $a = 0$ and $b = 0$.

Step 3

The non-trivial jump for complex numbers is understanding that the square of a complex number is not necessarily non-negative. The existence of i with the property $i^2 = -1$ breaks this property. This allows for non-zero numbers whose squares sum to zero.

The equation $a^2 + b^2 = 0$ can be factored in the complex numbers as a difference of squares, since $+b^2 = -(-b^2) = -(ib)^2$:

$$a^2 + b^2 = a^2 - (ib)^2 = (a - ib)(a + ib) = 0$$

This implies that $a - ib = 0$ or $a + ib = 0$, which means $a = ib$ or $a = -ib$. This has many non-zero solutions.

Step 4

Counterexample: Let $a = 1$ and $b = i$. Then we calculate the sum of their squares:

$$a^2 + b^2 = 1^2 + i^2 = 1 + (-1) = 0$$

The premise is satisfied, but neither $a = 1$ nor $b = i$ is zero. Therefore, the implication does not hold.

Problem 7: Analyze the cardinality of an inverse image.

Question

Let $f : \mathbb{N} \to \mathbb{N}$ be a function and $A \subset \mathbb{N}$ be a finite subset of natural numbers. The inverse image is defined as $f^{-1}(A) = \{n \in \mathbb{N} | f(n) \in A\}$. Is it always true that the cardinality $|f^{-1}(A)|$ satisfies the following conditions?

Background

This problem tests the fundamental understanding of functions, sets, and cardinality. Specifically, it probes the properties of the inverse image (or preimage) of a set under a function. The core of the problem lies in the fact that the function f is completely arbitrary. No assumptions are made about its properties, such as being injective (one-to-one), surjective (onto), or bijective. This lack of constraints is the critical point.

The cardinality of the inverse image $|f^{-1}(A)|$ counts how many elements from the domain $\mathbb{N}$ are mapped into the target set A. Since f can be many-to-one, a single element in A can be the image of zero, one, a finite number, or even an infinite number of elements from $\mathbb{N}$. The non-trivial insight is to exploit this freedom to construct counterexamples for all three proposed relationships.

Sub-Question 1: Part (a): Is $|f^{-1}(A)| = |A|$ always true?

Determine if the cardinality of the inverse image of A is always equal to the cardinality of A.

Step 1

Answer: NO.

Step 2

Justification: The statement $|f^{-1}(A)| = |A|$ implies a one-to-one correspondence between the elements of A and the elements of its preimage. However, the definition of a function allows for a "many-to-one" mapping. We can construct a function where many (or even infinitely many) domain elements map to a single element in A, which would violate this equality.

Step 3

Counterexample: Consider the finite set $A = \{1\}$. Its cardinality is $|A| = 1$. Let's define the function $f : \mathbb{N} \to \mathbb{N}$ as the constant function $f(n) = 1$ for all $n \in \mathbb{N}$.

The inverse image of A is the set of all natural numbers that map to 1:

$$f^{-1}(A) = f^{-1}(\{1\}) = \{n \in \mathbb{N} | f(n) = 1\}$$

Since $f(n) = 1$ for all n, this set is the entire set of natural numbers, $f^{-1}(A) = \mathbb{N}$.

The cardinality of the inverse image is $|f^{-1}(A)| = |\mathbb{N}| = \infty$. Since $\infty \neq 1$, we have $|f^{-1}(A)| \neq |A|$, and the statement is false.

Sub-Question 2: Part (b): Is $|f^{-1}(A)| \leq |A|$ always true?

Determine if the cardinality of the inverse image of A is always less than or equal to the cardinality of A.

Step 1

Answer: NO.

Step 2

Justification: This statement is even more restrictive than part (a) and fails for the same reason. If a function maps an infinite number of domain elements to a single element $a \in A$, then the cardinality of the inverse image $|f^{-1}(A)|$ will be infinite. An infinite quantity cannot be less than or equal to $|A|$, which is specified to be finite.

Step 3

Counterexample: We can use the exact same counterexample from part (a).

Let $A = \{1\}$, so $|A| = 1$. Define $f(n) = 1$ for all $n \in \mathbb{N}$. As established before, $f^{-1}(A) = \mathbb{N}$ and $|f^{-1}(A)| = \infty$.

The inequality $\infty \leq 1$ is false. Thus, the statement is disproven.

Sub-Question 3: Part (c): Is $|f^{-1}(A)| \geq |A|$ always true?

Determine if the cardinality of the inverse image of A is always greater than or equal to the cardinality of A.

Step 1

Answer: NO.

Step 2

Justification: The non-trivial logical jump here is realizing that this statement implies that the function f must map at least one domain element to each element of A. That is, A must be a subset of the image of f ($A \subseteq \text{Im}(f)$). However, since f is arbitrary, we can easily define a function whose image does

not intersect with A at all. In such a case, no element from the domain maps to any element in A, making the inverse image the empty set.

Step 3

Counterexample: Consider the finite set $A = \{1, 2\}$. Its cardinality is $|A| = 2$. Let's define the function $f : \mathbb{N} \to \mathbb{N}$ as $f(n) = n+10$. The image of this function is $\text{Im}(f) = \{11, 12, 13, \dots\}$.

The inverse image of A is the set of all natural numbers n such that $f(n)$ is in A:

$$f^{-1}(A) = \{n \in \mathbb{N} | n + 10 \in \{1, 2\}\}$$

There is no natural number n for which $n + 10 = 1$ or $n + 10 = 2$. Therefore, the inverse image is the empty set: $f^{-1}(A) = \emptyset$.

The cardinality of the inverse image is $|f^{-1}(A)| = |\emptyset| = 0$. The inequality $0 \geq 2$ is false. Thus, the statement is disproven.

Problem 9: Evaluate a nested summation.

Question

For $n \geq 0$, let S_n be the sum defined as:

$$S_n = \sum_{i=1}^{n} \sum_{j=1}^{i} \sum_{k=1}^{j} 1$$

Determine if this sum is equal to the following expressions involving binomial coefficients.

Background

This problem requires evaluating a triply nested summation. This can be approached through two distinct methods: direct algebraic evaluation, which is mechanical but computationally intensive, and a more elegant combinatorial argument that reveals the underlying structure of the problem. The latter provides a deeper insight into why the result takes the form of a binomial coefficient.

The non-trivial insight lies in reinterpreting the summation not as a sequence of arithmetic operations, but as a counting problem. The expression $\sum_{i=1}^{n} \sum_{j=1}^{i} \sum_{k=1}^{j} 1$ is precisely the number of integer triples (k, j, i) that satisfy the condition $1 \leq k \leq j \leq i \leq n$. This transforms the problem from algebra to combinatorics.

This counting problem is a classic example of 'combinations with repetition'. It is equivalent to choosing 3 numbers from the set $\{1, 2, \ldots, n\}$, with replacement allowed, where the order of selection does not matter (as we can always arrange the chosen numbers to satisfy $k \leq j \leq i$). A standard technique to solve this is to establish a bijection between these multisets and ordinary sets.

The key logical leap is the transformation of the non-strict inequality $1 \leq k \leq j \leq i \leq n$ into a strict inequality. Define a new set of integers (k', j', i') as follows: $k' = k$, $j' = j + 1$, and $i' = i + 2$. This mapping is a bijection. The original condition on (k, j, i) implies a new condition on (k', j', i'): $1 \leq k' < j' < i' \leq n + 2$. This new problem is equivalent to choosing 3 distinct numbers from the set $\{1, 2, \ldots, n + 2\}$, which is a standard combination problem. The number of ways to do this is $\binom{n+2}{3}$.

Sub-Question 1: Variant (a): Comparison with $\binom{n}{3}$

Is the sum S_n equal to $\binom{n}{3}$?

Step 1

To answer this, we first find a closed-form expression for S_n using direct summation. We evaluate the sums from the inside out.

1. Innermost sum: The sum is over a constant, so it evaluates to the upper limit.

$$\sum_{k=1}^{j} 1 = j$$

2. Middle sum: We substitute the result from the inner sum. This is the sum of the first i integers.

$$\sum_{j=1}^{i} j = \frac{i(i+1)}{2} = \frac{1}{2}(i^2 + i)$$

3. Outermost sum: We substitute the result from the middle sum into the final summation.

$$S_n = \sum_{i=1}^{n} \frac{1}{2}(i^2 + i) = \frac{1}{2}\left(\sum_{i=1}^{n} i^2 + \sum_{i=1}^{n} i\right)$$

4. Using the standard formulas for the sum of the first n integers and the sum of the first n squares:

$$S_n = \frac{1}{2}\left(\frac{n(n+1)(2n+1)}{6} + \frac{n(n+1)}{2}\right)$$

5. Algebraic simplification:

$$S_n = \frac{1}{2}\left(\frac{n(n+1)(2n+1) + 3n(n+1)}{6}\right) = \frac{n(n+1)}{12}((2n+1)+3) = \frac{n(n+1)(2n+4)}{12} = \frac{n(n+1)(n+2)}{6}$$

6. Comparison: The expression $\frac{n(n+1)(n+2)}{6}$ is the definition of the binomial coefficient $\binom{n+2}{3}$. We are asked to compare this to $\binom{n}{3} = \frac{n(n-1)(n-2)}{6}$.

Clearly, for $n \geq 3$, $\frac{n(n+1)(n+2)}{6} \neq \frac{n(n-1)(n-2)}{6}$. Therefore, the statement is false.

Answer: `NO`

Sub-Question 2: Variant (b): Comparison with $\binom{n+2}{3}$

Is the sum S_n equal to $\binom{n+2}{3}$?

Step 1

As demonstrated in the background section through a combinatorial argument, and verified by direct summation in the solution to variant (a), the value of the nested sum is:

$$S_n = \frac{n(n+1)(n+2)}{6}$$

By definition, the binomial coefficient $\binom{n+2}{3}$ is calculated as:

$$\binom{n+2}{3} = \frac{(n+2)!}{3!(n+2-3)!} = \frac{(n+2)!}{3!(n-1)!} = \frac{(n+2)(n+1)n(n-1)!}{6(n-1)!} = \frac{n(n+1)(n+2)}{6}$$

The closed-form expression for S_n is identical to the expression for $\binom{n+2}{3}$. Therefore, the statement is true.

Answer: `YES`

Sub-Question 3: Variant (c): Comparison with $\binom{n+3}{3}$

Is the sum S_n equal to $\binom{n+3}{3}$?

Step 1

From the previous analysis, we know that $S_n = \binom{n+2}{3} = \frac{n(n+1)(n+2)}{6}$.

We need to compare this with the binomial coefficient $\binom{n+3}{3}$.

$$\binom{n+3}{3} = \frac{(n+3)!}{3!(n+3-3)!} = \frac{(n+3)!}{3!n!} = \frac{(n+3)(n+2)(n+1)}{6}$$

Comparing the two expressions:

$$S_n = \frac{n(n+1)(n+2)}{6} \quad \text{vs.} \quad \binom{n+3}{3} = \frac{(n+1)(n+2)(n+3)}{6}$$

For any $n > 0$, we have $n \neq n+3$, so the expressions are not equal. Therefore, the statement is false.

Answer: `NO`

Problem 12: Analyze divisibility properties of binomial coefficients.

Question

Consider an integer $n > 100$. Given that $C(n, 2)$ is even and $C(n, 3)$ is divisible by 3, determine if it follows that:

(a) n is divisible by 2 or n is divisible by 3.

(b) $C(n, 5)$ is divisible by 5.

(c) $C(n, 6)$ is divisible by 6.

Background

The core of this problem is to translate the given divisibility properties of binomial coefficients into modular arithmetic constraints on n. The binomial coefficient is defined as $C(n, k) = \binom{n}{k} = \frac{n!}{k!(n-k)!}$.

Premise 1: $C(n, 2) = \frac{n(n-1)}{2}$ is even. This means $\frac{n(n-1)}{2} = 2k$ for some integer k, which implies $n(n-1)$ is divisible by 4. Since n and $n-1$ are consecutive integers, one is even and the other is odd. For their product to be divisible by 4, the even factor must be a multiple of 4. This leads to two cases:

$$n \equiv 0 \pmod 4 \quad \text{or} \quad n - 1 \equiv 0 \pmod 4 \implies n \equiv 1 \pmod 4$$

Premise 2: $C(n, 3) = \frac{n(n-1)(n-2)}{6}$ is divisible by 3. This implies $\frac{n(n-1)(n-2)}{6} = 3k$, so $n(n-1)(n-2)$ is divisible by 18. Divisibility by 18 requires divisibility by 2 and 9. The product of three consecutive integers is always divisible by $3! = 6$, so it is always divisible by 2. The non-trivial constraint is divisibility by 9. For the product to be divisible by 9, one of the factors $n, n-1$, or $n-2$ must be a multiple of 9.

$$n \equiv 0 \pmod 9 \quad \text{or} \quad n \equiv 1 \pmod 9 \quad \text{or} \quad n \equiv 2 \pmod 9$$

Sub-Question 1: Part (a): n is divisible by 2 or 3

Does it follow that n is divisible by 2 or n is divisible by 3?

Step 1

Answer: NO.

To disprove the claim, we seek a counterexample: an integer $n > 100$ that satisfies both premises but is not divisible by 2 or 3. This means n must be odd and not a multiple of 3.

We need to satisfy the combined premises:

- $(n \equiv 0 \text{ or } 1 \pmod 4)$ AND $(n \equiv 0, 1, \text{ or } 2 \pmod 9)$

To ensure n is not divisible by 2 or 3, let's select congruences that avoid these factors. We choose $n \equiv 1 \pmod 4$ (makes n odd) and $n \equiv 1 \pmod 9$ (makes n not a multiple of 3).

By the Chinese Remainder Theorem, these congruences imply $n \equiv 1 \pmod{36}$. We need $n > 100$. Let's test values of the form $n = 36k + 1$. For $k = 3$, we get $n = 36(3) + 1 = 109$.

The number $n = 109$ satisfies both premises by construction. However, 109 is not divisible by 2 (it is odd) and not divisible by 3 (the sum of its digits is 10). Thus, we have found a counterexample.

Sub-Question 2: Part (b): $C(n, 5)$ is divisible by 5

Does it follow that $C(n, 5)$ is divisible by 5?

Step 1

Answer: NO.

We can test the same counterexample, $n = 109$, which satisfies the premises. The divisibility of $C(n, k)$ by a prime p can be determined by the p -adic valuation, denoted v_p. The value is divisible by p if and only if $v_p(C(n, k)) \geq 1$.

Using Legendre's formula, $v_p(m!) = \sum_{i=1}^{\infty} \lfloor \frac{m}{p^i} \rfloor$, we can compute the 5-adic valuation of $C(109, 5)$:

$$v_5(C(109, 5)) = v_5(109!) - v_5(5!) - v_5(104!)$$

- $v_5(109!) = \lfloor \frac{109}{5} \rfloor + \lfloor \frac{109}{25} \rfloor = 21 + 4 = 25$
- $v_5(5!) = \lfloor \frac{5}{5} \rfloor = 1$
- $v_5(104!) = \lfloor \frac{104}{5} \rfloor + \lfloor \frac{104}{25} \rfloor = 20 + 4 = 24$

$$v_5(C(109, 5)) = 25 - 1 - 24 = 0$$

Since the 5-adic valuation is 0, $C(109, 5)$ is not divisible by 5. The counterexample holds.

Sub-Question 3: Part (c): $C(n, 6)$ is divisible by 6

Does it follow that $C(n, 6)$ is divisible by 6?

Step 1

Answer: YES.

We must prove that for any n satisfying the premises, $C(n,6)$ is divisible by 6. This is equivalent to proving it is divisible by both 2 and 3. We use the p-adic valuation method again.

Divisibility by 3: We need to show $v_3(C(n,6)) \geq 1$. We know $v_3(C(n,6)) = v_3(n(n-1)...(n-5)) - v_3(6!)$. Since $v_3(6!) = \lfloor 6/3 \rfloor = 2$, we must show $v_3(n(n-1)...(n-5)) \geq 3$. Premise 2 states that $v_3(n(n-1)(n-2)) \geq 2$. The product $n(n-1)...(n-5)$ contains the consecutive integers $n, n-1, n-2$ as well as $n-3, n-4, n-5$. The term $n-3$ (or $n-4$, or $n-5$) will contribute at least one more factor of 3. For instance, if $n \equiv 0 \pmod 9$, then $v_3(n) \geq 2$ and $n-3$ is a multiple of 3, so $v_3(n-3) \geq 1$. The total valuation is at least $2+1=3$. This logic holds for all three cases from Premise 2.

Divisibility by 2: We need to show $v_2(C(n,6)) \geq 1$. We know $v_2(6!) = \lfloor 6/2 \rfloor + \lfloor 6/4 \rfloor = 3+1=4$. We must show $v_2(n(n-1)...(n-5)) \geq 5$. Consider the two cases from Premise 1.

- *Case* $n \equiv 0 \pmod 4$: $v_2(n) \geq 2$. The other even factors in the product are $n-2$ and $n-4$. $v_2(n-2) = v_2(4k-2) = 1$. $v_2(n-4) = v_2(4k-4) \geq 2$. Total valuation: $v_2(\textit{product}) \geq 2+1+2=5$.
- *Case* $n \equiv 1 \pmod 4$: $v_2(n-1) \geq 2$. The other even factors are $n-3$ and $n-5$. $v_2(n-3) = v_2(4k+1-3) = v_2(4k-2) = 1$. $v_2(n-5) = v_2(4k+1-5) = v_2(4k-4) \geq 2$. Total valuation: $v_2(\textit{product}) \geq 2+1+2=5$.

In both cases, $v_2(C(n,6)) = v_2(\textit{product}) - 4 \geq 5-4 = 1$. Since $C(n,6)$ is divisible by both 2 and 3, it must be divisible by 6.

Problem 13: Calculate conditional and joint probabilities.

Question

Given the probabilities $P(B_1) = 1/4$, $P(B_2) = 3/4$, $P(A|B_1) = 1/2$, and $P(A|B_2) = 1/3$. Determine if the following statements are true:

(a) $P(A \cap B_1) = 1/8$

(b) $P(B_1 \cup B_2) = 1$

(c) $P(B_2|A) \geq 1/4$

Background

The critical, unstated assumption in this problem stems from the observation that $P(B_1) + P(B_2) = 1/4 + 3/4 = 1$. This is not a coincidence; it implies that the events B_1 and B_2 **form a partition** of the sample space Ω. A partition is a set of events that are both *mutually exclusive* ($B_1 \cap B_2 = \emptyset$) and *collectively exhaustive* ($B_1 \cup B_2 = \Omega$). This recognition is the foundational insight required to apply the Law of Total Probability and Bayes' Theorem correctly, which are essential for solving parts (b) and (c). Without this understanding, the relationship between the events remains ambiguous.

Sub-Question 1: Part (a): Joint Probability from Conditional Probability

Determine if $P(A \cap B_1) = 1/8$.

Step 1

This question is a direct application of the definition of conditional probability. The core relationship is:

$$P(A|B_1) = \frac{P(A \cap B_1)}{P(B_1)}$$

To find the joint probability $P(A \cap B_1)$, we rearrange this formula into the multiplicative rule:

$$P(A \cap B_1) = P(A|B_1)P(B_1)$$

Substituting the given values:

$$P(A \cap B_1) = \left(\frac{1}{2}\right) \times \left(\frac{1}{4}\right) = \frac{1}{8}$$

The calculated value matches the statement.

Answer: YES

Sub-Question 2: Part (b): Union of Partitioning Events

Determine if $P(B_1 \cup B_2) = 1$.

Step 1

This question tests the understanding of a partition, as established in the background. We can solve this computationally using the general formula for the union of two events:

$$P(B_1 \cup B_2) = P(B_1) + P(B_2) - P(B_1 \cap B_2)$$

Since B_1 and B_2 form a partition, they are mutually exclusive. By definition, this means their intersection is the empty set, $B_1 \cap B_2 = \emptyset$, and therefore $P(B_1 \cap B_2) = 0$.

The formula simplifies to:

$$P(B_1 \cup B_2) = P(B_1) + P(B_2) = \frac{1}{4} + \frac{3}{4} = 1$$

Conceptually, because B_1 and B_2 are also collectively exhaustive, their union constitutes the entire sample space Ω. By the axioms of probability, $P(\Omega) = 1$.

Answer: YES

Sub-Question 3: Part (c): Bayesian Inference and the Law of Total Probability

Determine if $P(B_2|A) \geq 1/4$.

Step 1

This question requires calculating an 'inverted' conditional probability. We are given $P(A|B_i)$ and asked to find $P(B_2|A)$. This is the classic scenario for applying Bayes' Theorem:

$$P(B_2|A) = \frac{P(A|B_2)P(B_2)}{P(A)}$$

The denominator, $P(A)$, is not directly given. This is the non-trivial step: we must first calculate the total probability of event A. Since $\{B_1, B_2\}$ forms a partition of the sample space, we can use the Law of Total Probability:

$$P(A) = P(A|B_1)P(B_1) + P(A|B_2)P(B_2)$$

Substituting the known values to find $P(A)$:

$$P(A) = \left(\frac{1}{2}\right)\left(\frac{1}{4}\right) + \left(\frac{1}{3}\right)\left(\frac{3}{4}\right) = \frac{1}{8} + \frac{3}{12} = \frac{1}{8} + \frac{1}{4} = \frac{1+2}{8} = \frac{3}{8}$$

Now we can substitute this result back into Bayes' Theorem:

$$P(B_2|A) = \frac{P(A|B_2)P(B_2)}{P(A)} = \frac{(1/3)(3/4)}{3/8} = \frac{1/4}{3/8} = \frac{1}{4} \times \frac{8}{3} = \frac{8}{12} = \frac{2}{3}$$

Finally, we check if the inequality holds: Is $2/3 \geq 1/4$? To compare, we can use a common denominator of 12: $8/12 \geq 3/12$. The inequality is true.

Answer: YES

Problem 14: Analyze a condition for independence of random variables.

Question

Let X and Y be random variables with values from the set $\{0, 1, 2\}$. Suppose that for any $x, y \in \{0, 1, 2\}$ we have:

$$P(X = x \wedge Y = y) = E[X] \cdot P(Y = y)$$

Does this imply that:

Background

This problem tests the understanding of fundamental concepts in probability theory: independence of random variables and properties of expectation. A key logical step is to analyze the consequences of the given, rather unusual, condition.

The standard definition for the independence of two discrete random variables X and Y is:

$$P(X = x \wedge Y = y) = P(X = x) \cdot P(Y = y) \quad \forall x, y$$

Comparing this to the given condition, $P(X = x \wedge Y = y) = E[X] \cdot P(Y = y)$, suggests a relationship between $P(X = x)$ and $E[X]$.

Let's make this relationship explicit. By the law of total probability, we can find the marginal probability $P(X = x)$ by summing the joint probabilities over all possible values of Y:

$$P(X = x) = \sum_{y \in \{0,1,2\}} P(X = x \wedge Y = y)$$

Substituting the given condition:

$$P(X = x) = \sum_{y \in \{0,1,2\}} (E[X] \cdot P(Y = y)) = E[X] \cdot \sum_{y \in \{0,1,2\}} P(Y = y)$$

Since the sum of probabilities over the entire sample space of Y must be 1 ($\sum_y P(Y = y) = 1$), we arrive at a critical, non-trivial implication:

$$P(X = x) = E[X] \quad \text{for all } x \in \{0, 1, 2\}$$

This result is central to answering all parts of the question. It asserts that the probability mass function of X is a constant, equal to the single numerical value of its expectation. We will explore the validity of this assertion in the solution.

Sub-Question 1: Variant (a): Equality of Probability Products

Does the condition imply $P(X = 1) \cdot P(Y = 2) = P(X = 2) \cdot P(Y = 1)$?

Step 1

Answer: NO.

Step 2

From the background analysis, the given condition implies that $P(X = x) = E[X]$ for all x. Therefore, the probability mass function of X must be constant for $x = 0, 1, 2$. This directly means:

$$P(X = 1) = E[X] \quad \text{and} \quad P(X = 2) = E[X]$$

This leads to the conclusion that $P(X = 1) = P(X = 2)$.

Step 3

Substituting $P(X = 1) = P(X = 2)$ into the equation from the question:

$$P(X = 1) \cdot P(Y = 2) = P(X = 1) \cdot P(Y = 1)$$

Step 4

If we assume $P(X = 1) \neq 0$ (which is equivalent to $E[X] \neq 0$), we can divide both sides by $P(X = 1)$, which would require $P(Y = 2) = P(Y = 1)$.

However, the initial condition places no constraints on the distribution of Y beyond it being a valid probability distribution. We can construct a simple counterexample. Let Y have the distribution $P(Y = 0) = 0.5, P(Y = 1) = 0.3, P(Y = 2) = 0.2$. In this case, $P(Y = 2) \neq P(Y = 1)$. Thus, the implication does not hold in general.

Sub-Question 2: Variant (b): Linearity of Expectation

Does the condition imply $E[X + Y] = E[X] + E[Y]$?

Step 1

Answer: YES.

Step 2

The property $E[X + Y] = E[X] + E[Y]$ **is known as the linearity of expectation**. This is a fundamental theorem of probability theory.

Step 3

The non-trivial point here is recognizing that this property holds for any two random variables X and Y defined on the same probability space, regardless of whether they are independent or dependent. The condition given in the problem, $P(X = x \wedge Y = y) = E[X] \cdot P(Y = y)$, is entirely irrelevant to this property. The statement is true by definition, so the implication holds.

Sub-Question 3: Variant (c): Independence of Random Variables

Does the condition imply X and Y are independent?

Step 1

Answer: NO.

Step 2

For X and Y to be independent, the condition $P(X = x \wedge Y = y) = P(X = x) \cdot P(Y = y)$ must hold. The problem states that $P(X = x \wedge Y = y) = E[X] \cdot P(Y = y)$. For these two conditions to be equivalent, it must be that $P(X = x) = E[X]$ for all $x \in \{0, 1, 2\}$.

Step 3

This is the non-trivial leap: we must rigorously check if the condition $P(X = x) = E[X]$ is possible for a random variable with values in $\{0, 1, 2\}$. Let $c = E[X]$. Then we must have $P(X = 0) = c$, $P(X = 1) = c$, and $P(X = 2) = c$.

- From the axiom that probabilities sum to 1: $P(X = 0) + P(X = 1) + P(X = 2) = 1 \implies c + c + c = 3c = 1 \implies c = 1/3$. This implies that $E[X]$ must be $1/3$.
- From the definition of expected value: $E[X] = \sum_x x \cdot P(X = x) = 0 \cdot P(X = 0) + 1 \cdot P(X = 1) + 2 \cdot P(X = 2)$. Substituting $P(X = x) = c = 1/3$, we get $E[X] = 0(1/3) + 1(1/3) + 2(1/3) = 3/3 = 1$.

Step 4

We have reached a contradiction: $E[X]$ must be both $1/3$ and 1. This is impossible. Therefore, the condition $P(X = x) = E[X]$ cannot hold for any non-trivial random variable X defined on $\{0, 1, 2\}$. Consequently, the given condition cannot be equivalent to the definition of independence.

Problem 15: Analyze a three-state time-homogeneous Markov chain.

Question

A time-homogeneous Markov chain has three states {1, 2, 3}, all of which are recurrent. The mean recurrence time for state 1 is $m_1 = 2$, and for state 2 is $m_2 = 4$. The questions are to determine if (a) the mean recurrence time for state 3 is less than or equal to 6, (b) the mean recurrence time for state 3 is greater than or equal to 6, and (c) this Markov chain is aperiodic.

Background

This problem hinges on a fundamental theorem of finite Markov chains. A key, non-trivial insight is that for a finite-state Markov chain, if all states are recurrent, the chain forms a single communicating class and is therefore **irreducible**. An irreducible, recurrent Markov chain is guaranteed to have a unique stationary distribution, denoted by the vector $\pi = (\pi_1, \pi_2, \pi_3)$.

The components of this distribution, π_i, represent the long-run proportion of time the chain spends in state i. By definition, the probabilities must sum to unity: $\sum_{i=1}^{3} \pi_i = 1$.

The crucial link between the stationary distribution and the mean recurrence time m_i (the expected time to return to state i having started in i) is given by the Kac's formula for finite irreducible chains:

$$\pi_i = \frac{1}{m_i}$$

This relationship is intuitive: if the chain spends a large fraction of its time in state i (high π_i), it must return to that state frequently, implying a short mean recurrence time (low m_i).

Finally, aperiodicity is a structural property of the state transition graph. A state is aperiodic if the greatest common divisor (GCD) of the lengths of all possible paths from the state back to itself is 1. The stationary distribution, and thus the mean recurrence times, can be identical for both periodic and aperiodic chains. Therefore, knowledge of m_i alone is insufficient to determine aperiodicity.

Sub-Question 1: Part (a): Is the mean recurrence time for state 3 less than or equal to 6?

Determine if $m_3 \leq 6$.

Step 1

The core argument combines the two fundamental properties discussed in the background. First, the sum of the probabilities in the stationary distribution is 1. Second, each probability is the reciprocal of the corresponding mean recurrence time.

$$\pi_1 + \pi_2 + \pi_3 = 1 \quad \Longrightarrow \quad \frac{1}{m_1} + \frac{1}{m_2} + \frac{1}{m_3} = 1$$

Step 2

We are given $m_1 = 2$ and $m_2 = 4$. Substituting these values into the equation allows us to solve for m_3.

$$\frac{1}{2} + \frac{1}{4} + \frac{1}{m_3} = 1$$

Step 3

Solving for $1/m_3$:

$$\frac{3}{4} + \frac{1}{m_3} = 1$$
$$\frac{1}{m_3} = 1 - \frac{3}{4} = \frac{1}{4}$$

This directly implies that $m_3 = 4$.

Step 4

We now check if this result satisfies the condition $m_3 \leq 6$. Since $4 \leq 6$ is true, the answer is YES.

Sub-Question 2: Part (b): Is the mean recurrence time for state 3 greater than or equal to 6?

Determine if $m_3 \geq 6$.

Step 1

Using the result derived in Part (a), we have already uniquely determined the mean recurrence time for state 3.

$$m_3 = 4$$

Step 2

We check this value against the condition $m_3 \geq 6$. The statement $4 \geq 6$ is false.

Step 3

Therefore, the answer is NO.

Sub-Question 3: Part (c): Is this Markov chain aperiodic?

Determine if the property of aperiodicity can be concluded from the given information.

Step 1

The non-trivial logical step here is to recognize the distinction between properties derived from the stationary distribution (like mean recurrence time) and properties derived from the transition graph's structure (like aperiodicity).

Step 2

The mean recurrence times $m_1 = 2, m_2 = 4, m_3 = 4$ uniquely determine the stationary distribution $\pi = (1/2, 1/4, 1/4)$. However, multiple different transition matrices P can result in the same stationary distribution π. Some of these matrices may correspond to periodic chains, while others correspond to aperiodic chains.

Step 3

For a conceptual counterexample, consider two chains. Chain A is a simple cycle $1 \to 2 \to 3 \to 1$. It is periodic with period 3. Its stationary distribution is $\pi_A = (1/3, 1/3, 1/3)$. Chain B has a transition matrix where all entries are positive ($P_{ij} > 0$). This chain is aperiodic. It is possible to construct such a P that has the stationary distribution $\pi = (1/2, 1/4, 1/4)$. It is also possible to construct a periodic chain with the same stationary distribution (though it would be more complex than a simple cycle).

Step 4

Since the given information only constrains the stationary distribution and not the specific transition structure, we cannot determine whether the chain is aperiodic or periodic. The statement that it ***is*** aperiodic is not necessarily true.

Thus, the answer is NO.

Problem 16: Analyze the distribution of a transformed random variable.

Question

Let X be a continuous random variable with probability density function (PDF) f_X and strictly increasing cumulative distribution function (CDF) F_X. Let $Y = -X$ be a new random variable with PDF f_Y and CDF F_Y. Determine if the following statements are true:

(a) $f_Y(t) = f_X(-t)$ almost everywhere.

(b) $F_Y(t) = 1 - F_X(-t)$.

(c) $E(Y) = -E(X)$, assuming the expectation exists.

Background

This problem tests fundamental principles of probability theory concerning the transformation of random variables. The key insight is that properties of a transformed variable (like $Y = g(X)$) can be derived from the properties of the original variable (X).

The most robust method for finding the distribution of Y is the CDF method. We first compute the CDF of Y, $F_Y(t) = P(Y \leq t)$, by expressing the event $Y \leq t$ in terms of an equivalent event involving X. Once the CDF is known, the PDF is found by differentiation: $f_Y(t) = \frac{d}{dt}F_Y(t)$. This approach avoids potential pitfalls of other methods, especially with non-monotonic transformations.

For expectations, the linearity of the expectation operator is a powerful tool. It states that $E[aX + b] = aE[X] + b$ for constants a and b. Recognizing that $Y = -X$ is a simple linear transformation allows for a direct solution for part (c).

Sub-Question 1: Part (a): Is $f_Y(t) = f_X(-t)$ almost everywhere?

Derive the probability density function (PDF) of Y and compare it to the PDF of X.

Step 1

YES. The statement is true.

The derivation relies on first finding the CDF of Y, $F_Y(t)$, and then differentiating it with respect to t to obtain the PDF, $f_Y(t)$. This two-step process is the standard, rigorous approach.

Step 2

Step 1: Find the CDF of Y. (This is the same derivation as in Part b).

$$F_Y(t) = P(Y \leq t) = P(-X \leq t) = P(X \geq -t)$$

The logical jump here is manipulating the inequality. Since X is a continuous random variable, we have $P(X \geq a) = 1 - P(X < a) = 1 - P(X \leq a) = 1 - F_X(a)$. Applying this with $a = -t$:

$$F_Y(t) = 1 - F_X(-t)$$

Step 3

Step 2: Differentiate the CDF of Y to find its PDF.

The PDF is the derivative of the CDF. We must apply the chain rule to differentiate $F_X(-t)$.

$$f_Y(t) = \frac{d}{dt}F_Y(t) = \frac{d}{dt}[1 - F_X(-t)] = -\frac{d}{dt}[F_X(-t)]$$

Using the chain rule, where the outer function is $F_X(\cdot)$ and the inner function is $-t$:

$$f_Y(t) = -\left(f_X(-t) \cdot \frac{d}{dt}(-t)\right) = -\left(f_X(-t) \cdot (-1)\right) = f_X(-t)$$

The qualifier "almost everywhere" is a standard technical condition in measure theory, acknowledging that PDFs are equivalent if they differ only on a set of points with zero probability, which does not affect any calculated probabilities.

Sub-Question 2: Part (b): Is $F_Y(t) = 1 - F_X(-t)$?

Derive the cumulative distribution function (CDF) of Y from first principles.

Step 1

YES. The statement is true.

The derivation proceeds directly from the definition of the CDF.

Step 2

Step 1: Start with the definition of $F_Y(t)$.

$$F_Y(t) = P(Y \leq t)$$

Step 3

Step 2: Substitute $Y = -X$ and manipulate the inequality.

The crucial step is to rewrite the event $\{Y \leq t\}$ in terms of X. Multiplying the inequality by -1 reverses its direction.

$$P(Y \leq t) = P(-X \leq t) = P(X \geq -t)$$

Step 4

Step 3: Express the probability in terms of the CDF of X.

For any continuous random variable, the probability of being greater than or equal to a value is 1 minus the probability of being less than that value. Since it's continuous, $P(X < a) = P(X \leq a) = F_X(a)$.

$$P(X \geq -t) = 1 - P(X < -t) = 1 - P(X \leq -t) = 1 - F_X(-t)$$

Step 5

Combining these steps gives the final result:

$$F_Y(t) = 1 - F_X(-t)$$

Sub-Question 3: Part (c): Is $E(Y) = -E(X)$?

Assuming $E(X)$ exists, determine the relationship between $E(Y)$ and $E(X)$.

Step 1

YES. The statement is true.

This follows directly from the linearity of the expectation operator, which is one of its most fundamental properties.

Step 2

Step 1: Apply the linearity of expectation.

The expectation operator $E[\cdot]$ is a linear operator. This means that for any random variable Z and constants a, b, we have $E[aZ + b] = aE[Z] + b$. The non-trivial jump is simply recognizing that $Y = -X$ is a linear transformation where $a = -1$ and $b = 0$.

$$E(Y) = E(-X) = E(-1 \cdot X) = -1 \cdot E(X) = -E(X)$$

This holds true as long as $E(X)$ is well-defined (i.e., the integral $\int |x| f_X(x)dx$ is finite), which is given in the problem statement.

Step 3

Alternative Method: Using the Definition of Expectation

We can also prove this from the integral definition of expectation and the result from part (a).

$$E[Y] = \int_{-\infty}^{\infty} t f_Y(t)dt = \int_{-\infty}^{\infty} t f_X(-t)dt$$

Perform a change of variables with $u = -t$, so $t = -u$ and $dt = -du$. The limits of integration are reversed.

$$E[Y] = \int_{\infty}^{-\infty} (-u) f_X(u)(-du) = \int_{\infty}^{-\infty} u f_X(u)du$$

Flipping the limits of integration introduces a negative sign:

$$E[Y] = -\int_{-\infty}^{\infty} u f_X(u)du = -E[X]$$

This confirms the result obtained from the linearity property.

Problem 17: Interpret machine learning test metrics.

Question

An ML test for a disease has a sensitivity of 0.5 and a specificity of 0.7. We need to determine the truth of the following statements:

(a) The test identifies 70% of healthy patients as healthy.

(b) 70% of the patients classified as healthy are indeed healthy.

(c) The test correctly classifies more than 60% of the patients.

Background

This problem tests the understanding of fundamental binary classification metrics. Let D be the event that a patient is diseased, and H be the event that a patient is healthy. Let T_P be the event of a positive test result and T_N be the event of a negative test result.

Sensitivity (or True Positive Rate) is the probability of a positive test result given that the patient is diseased. It measures how well the test identifies actual positives.

$$\text{Sensitivity} = P(T_P|D) = 0.5$$

Specificity (or True Negative Rate) is the probability of a negative test result given that the patient is healthy. It measures how well the test identifies actual negatives.

$$\text{Specificity} = P(T_N|H) = 0.7$$

A critical non-trivial distinction in this problem is between probabilities conditioned on the true state of the patient (D or H) versus probabilities conditioned on the test outcome (T_P or T_N). The latter, known as predictive values, depend on the prevalence of the disease in the population, which is an unknown parameter here. Let $p = P(D)$ be the prevalence.

Sub-Question 1: Variant (a): Identification of Healthy Patients

Is the statement 'The test identifies 70% of healthy patients as healthy' true?

Step 1

This statement translates directly into a conditional probability. The condition is that a patient is 'healthy' (event H), and the outcome is that the test 'identifies them as healthy', which means a negative test result (event T_N). Therefore, the statement is asserting that $P(T_N|H) = 0.7$.

Step 2

By definition, $P(T_N|H)$ is the specificity of the test. The problem statement explicitly gives the specificity as 0.7.

Step 3

Thus, the statement is a direct restatement of the given information.

Step 4

Answer: YES

Sub-Question 2: Variant (b): Health Status of Patients Classified as Healthy

Is the statement '70% of the patients classified as healthy are indeed healthy' true?

Step 1

This statement describes a different conditional probability. The condition is that a patient is 'classified as healthy', meaning they received a negative test result (event T_N). The outcome is that they 'are indeed healthy' (event H). The statement asserts that $P(H|T_N) = 0.7$.

Step 2

This probability, $P(H|T_N)$, is known as the Negative Predictive Value (NPV). The non-trivial jump here is recognizing that this is **not** the same as specificity, $P(T_N|H)$. To find the NPV, we must apply Bayes' theorem, which incorporates the base rate or prevalence of the disease, $p = P(D)$.

Step 3

$$\text{NPV} = P(H|T_N) = \frac{P(T_N|H)P(H)}{P(T_N)}$$

Using the law of total probability, the denominator is $P(T_N) = P(T_N|H)P(H) + P(T_N|D)P(D)$. We are given $P(T_N|H) = 0.7$ and $P(T_P|D) = 0.5$, which implies $P(T_N|D) = 1 - P(T_P|D) = 1 - 0.5 = 0.5$. Let $P(D) = p$, so $P(H) = 1 - p$.

$$\text{NPV}(p) = \frac{0.7 \cdot (1-p)}{0.7 \cdot (1-p) + 0.5 \cdot p} = \frac{0.7 - 0.7p}{0.7 - 0.2p}$$

Step 4

The value of the NPV is a function of the prevalence p. Since p is not given, the NPV cannot be determined. For instance, if $p = 0.1$, NPV ≈ 0.926. If $p = 0.5$, NPV ≈ 0.583. The value is not constant and not necessarily 0.7.

Step 5

Answer: NO

Sub-Question 3: Variant (c): Overall Classification Accuracy

Is the statement 'The test correctly classifies more than 60% of the patients' true?

Step 1

This statement refers to the overall accuracy of the test. Accuracy is the probability of a correct classification, regardless of whether the patient is diseased or healthy. We can calculate this using the law of total probability, conditioning on the true state of the patient.

$$\text{Accuracy} = P(\text{Correct}) = P(\text{Correct}|D)P(D) + P(\text{Correct}|H)P(H)$$

Step 2

A correct classification for a diseased patient is a positive test, so $P(\text{Correct}|D) = P(T_P|D) = \text{Sensitivity} = 0.5$. A correct classification for a healthy patient is a negative test, so $P(\text{Correct}|H) = P(T_N|H) = \text{Specificity} = 0.7$.

Step 3

Again, let the prevalence be $p = P(D)$. The accuracy is then a function of p:

$$\text{Accuracy}(p) = (0.5)p + (0.7)(1-p) = 0.5p + 0.7 - 0.7p = 0.7 - 0.2p$$

Step 4

The non-trivial step is to analyze the range of this function for the valid domain of prevalence, $p \in [0, 1]$. The function Accuracy(p) is linear and decreasing.

- Maximum accuracy occurs at $p = 0$: Accuracy$(0) = 0.7$ (or 70%).

- Minimum accuracy occurs at $p = 1$: Accuracy$(1) = 0.7 - 0.2 = 0.5$ (or 50%).

Since the accuracy can range from 50% to 70%, we cannot guarantee that it is always greater than 60%. For the statement to be false, we need to find a prevalence p where $0.7 - 0.2p \leq 0.6$. This simplifies to $0.1 \leq 0.2p$, or $p \geq 0.5$. If the disease prevalence is 50% or more, the accuracy will be 60% or less. As we don't know the prevalence, we cannot make the claim.

Step 5

Answer: NO

Problem 18: Analyze the standard k-means algorithm.

Question

The standard k-means algorithm with parameter k for n observations returns:

(a) a partition of observations into k groups with the lowest possible variances of distances within groups,

(b) a set of groups such that their middle points (centroids) are observations from the input data,

(c) the same results in different runs on the same input data.

Background

The k-means algorithm is an unsupervised learning method used for clustering. Its objective is to partition a set of n observations into k distinct, non-overlapping clusters. The algorithm aims to minimize the within-cluster sum of squares (WCSS), also known as inertia. This is the sum of squared distances between each observation and the centroid of its assigned cluster.

$$J = \sum_{j=1}^{k} \sum_{x_i \in C_j} \|x_i - \mu_j\|^2$$

Here, C_j is the j-th cluster, and μ_j is the centroid (mean) of the points in C_j. The algorithm proceeds iteratively:

1. Initialization: Randomly select k data points as the initial centroids.

2. Assignment Step: Assign each observation to the cluster whose centroid is nearest (e.g., using Euclidean distance).

3. Update Step: Recalculate the centroid for each cluster by taking the mean of all data points assigned to it.

The algorithm repeats steps 2 and 3 until the cluster assignments stabilize. This process is a heuristic and is an example of an Expectation-Maximization (EM) algorithm. It is guaranteed to converge, but its final state depends critically on the initial choice of centroids.

Sub-Question 1: Variant (a): a partition of observations into k groups with the lowest possible variances of distances within groups

Step 1

Answer: NO

Step 2

The core of this question lies in the distinction between a local and a global optimum. The k-means algorithm is a greedy, iterative heuristic. It is guaranteed to converge to a ***local*** *minimum of the within-cluster sum of squares (WCSS), but not necessarily the* ***global*** *minimum.*

Step 3

The phrase "lowest possible variances" implies finding the global minimum of the WCSS objective function. However, the final clustering solution produced by k-means is highly dependent on the initial, randomly chosen centroids. Different initializations can lead the algorithm to converge to different local minima. Some of these minima may be substantially worse (i.e., have higher WCSS) than the global minimum.

Step 4

Finding the globally optimal solution for the k-means objective is an NP-hard problem. The iterative refinement strategy of k-means is a computationally feasible approximation, but it provides no guarantee of global optimality. Therefore, it does not guarantee the "lowest possible" variance.

Sub-Question 2: Variant (b): a set of groups such that their middle points (centroids) are observations from the input data

Step 1

Answer: NO

Step 2

The non-trivial insight here is understanding how a centroid is defined and calculated in the algorithm's update step. A centroid is the arithmetic mean (the geometric center) of all the data points assigned to its cluster.

Step 3

Mathematically, for a cluster C_j, the centroid μ_j is computed as:

$$\mu_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

Step 4

The result of this averaging operation is a point in the feature space. There is no constraint in the algorithm that forces this calculated mean to coincide

with one of the original data points, x_i. While it might happen by coincidence (e.g., for data points {1, 2, 3}, the mean is 2), it is not a general property. The centroid can, and usually does, exist at a location where no actual data point lies.

Step 5

This property distinguishes k-means from other clustering algorithms like k-medoids, where the cluster center (the medoid) is explicitly constrained to be one of the actual observations from the input dataset.

Sub-Question 3: Variant (c): the same results in different runs on the same input data

Step 1

Answer: NO

Step 2

The critical non-trivial point is the role of stochasticity in the standard algorithm. The k-means algorithm is not deterministic. Its outcome is contingent on the ***initialization step****, which is typically random.*

Step 3

The algorithm begins by selecting k initial centroids. In the standard implementation, this selection is made by randomly choosing k points from the dataset. Since this starting configuration is random, running the algorithm multiple times on the same data will almost certainly lead to different starting centroids.

Step 4

As established in the analysis of variant (a), different initializations can lead the algorithm to converge to different local minima. Consequently, the final cluster assignments, the final centroid positions, and the final WCSS value can vary from run to run.

Step 5

This is why, in practice, it is standard procedure to run k-means multiple times (e.g., 10 or more times) with different random seeds and select the run that yields the best result (i.e., the lowest WCSS). This non-determinism is a fundamental characteristic of the algorithm that must be managed in its application.

Problem 19: Characterize the Kolmogorov-Smirnov test.

Question

The Kolmogorov-Smirnov (K-S) test:

(a) tests whether an input sample comes from a normal distribution of given parameters.

(b) tests whether an input sample comes from a normal distribution of unknown parameters.

(c) assumes that the input sample comes from a distribution of finite variance.

Background

The one-sample Kolmogorov-Smirnov (K-S) test is a non-parametric goodness-of-fit test. It is used to determine if a sample of data is drawn from a specific, continuous theoretical distribution. The core of the test is a comparison between the Empirical Distribution Function (EDF) of the sample and the Cumulative Distribution Function (CDF) of the hypothesized theoretical distribution.

The EDF, denoted $F_n(x)$, for a sample of size n is the proportion of sample points less than or equal to x. The test statistic, D_n, is the maximum absolute difference between the EDF and the theoretical CDF, $F(x)$:

$$D_n = \sup_x |F_n(x) - F(x)|$$

A critical, non-trivial requirement for the validity of the standard K-S test is that the theoretical distribution $F(x)$ **must be fully specified** in advance. This means all of its parameters (e.g., mean and variance for a normal distribution) must be known and not estimated from the data being tested.

Sub-Question 1: Part (a): Testing against a fully specified normal distribution

Does the Kolmogorov-Smirnov test test whether an input sample comes from a normal distribution of given parameters?

Step 1

Answer: YES

Step 2

This scenario describes the canonical use case for the one-sample K-S test. The phrase "given parameters" is key. It signifies that the theoretical distribution is fully specified.

Step 3

For instance, if we want to test whether a sample comes from a normal distribution with a specific mean μ_0 and variance σ_0^2, the null hypothesis is:

$$H_0 : \text{The data are drawn from } \mathcal{N}(\mu_0, \sigma_0^2).$$

Since the CDF of $\mathcal{N}(\mu_0, \sigma_0^2)$ is completely determined by the given parameters μ_0 and σ_0^2, the K-S test is directly applicable. The test statistic D_n can be computed and compared against the critical values of the Kolmogorov distribution.

Sub-Question 2: Part (b): Testing against a normal distribution with estimated parameters

Does the Kolmogorov-Smirnov test test whether an input sample comes from a normal distribution of unknown parameters?

Step 1

Answer: NO

Step 2

This is a critical subtlety in statistical testing. If the parameters of the normal distribution (mean μ and variance σ^2) are unknown, they must be estimated from the sample data itself (e.g., using the sample mean $\bar{x}$ and sample variance s^2).

Step 3

Using the data twice—once to define the distribution and again to test against it—violates the assumption that $F(x)$ is pre-specified. The estimation process inherently makes the sample data fit the hypothesized distribution more closely than it would fit a truly independent theoretical distribution. This causes the K-S test statistic D_n to be systematically smaller.

Step 4

As a result, the distribution of the test statistic is no longer the standard Kolmogorov distribution. Using the standard critical values for the K-S test in this situation leads to an overly conservative test (i.e., the true Type I error rate is lower than the nominal level α), which reduces its power to detect non-normality.

Step 5

The correct procedure for this scenario is to use a modified version of the K-S test, known as the **Lilliefors test**, which provides the appropriate critical values for testing normality with estimated parameters.

Sub-Question 3: Part (c): Assumption of finite variance

Does the Kolmogorov-Smirnov test assume that the input sample comes from a distribution of finite variance?

Step 1

Answer: NO

Step 2

The K-S test is non-parametric, which means it makes very few assumptions about the underlying distribution of the data. Its mechanism is based entirely on the CDF, which is a fundamental property of any random variable, regardless of whether its moments (like mean or variance) exist or are finite.

Step 3

The test statistic $D_n = \sup_x |F_n(x) - F(x)|$ only involves probabilities and order statistics, not sums of powers of the observations. Therefore, the existence of moments is irrelevant to the test's validity.

Step 4

A classic counterexample is the Cauchy distribution. A random variable following a standard Cauchy distribution has a well-defined, continuous CDF, but its mean is undefined and its variance is infinite. The K-S test can still be validly applied to test whether a sample comes from a Cauchy distribution, as the procedure only requires its CDF.

Problem 20: Define the False Discovery Rate (FDR).

Question

When testing multiple statistical hypotheses, the false discovery rate (FDR) is:

(a) the probability of making a type I error in at least one of the tests,

(b) the expected proportion of falsely rejected null hypotheses among all rejected hypotheses,

(c) the product of all p-values and the number of tested hypotheses.

Background

The problem of multiple hypothesis testing arises when a researcher performs many statistical tests simultaneously. If each test is conducted at a significance level α (e.g., 0.05), the probability of making at least one Type I error (a false positive) across all tests inflates dramatically. To address this, various methods have been developed to control the overall error rate. The False Discovery Rate (FDR) is one such method, offering a different and often more powerful approach than controlling the Family-Wise Error Rate (FWER).

To formally define these concepts, consider a scenario where m null hypotheses are tested. The outcomes can be summarized in a contingency table:

	Not Rejected	Rejected	Total
Null is True	U	V	m_0
Null is False	T	S	m_1
Total	$m - R$	R	m

V is the number of Type I errors (false positives, or "false discoveries").

R is the total number of rejected hypotheses (total "discoveries").

Both V and R are random variables, as they depend on the observed data.

Sub-Question 1: Variant (a): Probability of at least one Type I error

Is FDR defined as the probability of making a type I error in at least one of the tests?

Step 1

Answer: NO.

Step 2

This definition describes the **Family-Wise Error Rate (FWER)**, not the FDR. The FWER is the probability of making one or more Type I errors among all the hypotheses tested.

$$\text{FWER} = P(V \geq 1)$$

Step 3

The non-trivial distinction lies in the type of control. FWER control is stringent; it aims to avoid even a single false positive across the entire family of tests. This is appropriate in fields like clinical trials where a single false positive can have severe consequences. However, this stringency comes at a cost: reduced statistical power. In exploratory analyses, such as genomics or neuroimaging, where thousands of tests are performed and some false positives are acceptable in exchange for finding more true effects, FWER control is often too conservative and can lead to missing important discoveries. The FDR was introduced as a less stringent, and therefore more powerful, alternative.

Sub-Question 2: Variant (b): Expected proportion of false rejections

Is FDR defined as the expected proportion of falsely rejected null hypotheses among all rejected hypotheses?

Step 1

Answer: YES.

Step 2

This is the precise definition of the False Discovery Rate. The core idea is to first define the **False Discovery Proportion (FDP)**, which is the proportion of Type I errors among all rejected hypotheses for a specific experiment:

$$\text{FDP} = \begin{cases} V/R & \text{if } R > 0 \\ 0 & \text{if } R = 0 \end{cases}$$

Step 3

The FDP is a random variable because V and R depend on the data. We cannot control the FDP for a single realization of an experiment. The non-trivial logical leap is to control its **expectation** over repeated experiments. The FDR is this expected value:

$$\text{FDR} = E[\text{FDP}] = E\left[\frac{V}{R}\middle| R > 0\right] P(R > 0)$$

Step 4

Therefore, controlling the FDR at a level q (e.g., 0.05) provides a guarantee that, on average, no more than 5% of the rejected hypotheses (the "discoveries") will be false. This interpretation is intuitive and highly useful in exploratory research.

Sub-Question 3: Variant (c): Product of p-values and number of tests

Is FDR defined as the product of all p-values and the number of tested hypotheses?

Step 1

Answer: NO.

Step 2

This formula, $m \times \prod_{i=1}^{m} p_i$, does not correspond to any standard statistical measure used in multiple hypothesis testing. It is mathematically and conceptually unsound as a measure of error.

Step 3

The non-trivial reason this fails is that it misinterprets how p-values should be aggregated. Since each $p_i \in [0, 1]$, their product $\prod p_i$ will shrink towards zero very rapidly as m increases. Multiplying by m does not provide a stable or meaningful correction. Valid methods for combining p-values, such as Fisher's method (which considers $-2\sum \ln(p_i)$), operate on a completely different mathematical basis, typically transforming p-values before aggregation to create a test statistic with a known distribution.

Problem 10: Count triangles in a specific graph.

Question

Consider an undirected graph $G = (V, E)$ where the set of vertices is:

$$V = \{1, 2, \ldots, 100\}$$

An edge $\{m, n\}$ exists between two distinct vertices $m, n \in V$ if and only if their product $m \cdot n$ is an even number.

Determine which of the following statements is true regarding the number of triangles (cliques of size 3) in G.

Background

The non-trivial logical step in this problem is to translate the arithmetic condition for edge existence into a structural property of the graph based on vertex parity. The condition that a product $m \cdot n$ is even is not satisfied only when both m and n are odd. This insight allows us to reason about the graph's connectivity by partitioning the vertices.

Let's partition the vertex set V into two disjoint subsets: O for odd numbers and E for even numbers.

$$O = \{1, 3, 5, \ldots, 99\} \quad \Longrightarrow \quad |O| = 50$$

$$E = \{2, 4, 6, \ldots, 100\} \quad \Longrightarrow \quad |E| = 50$$

The edge condition now defines the graph's structure completely:

1. If $m, n \in O$, their product $m \cdot n$ is odd. Thus, **no edge exists between any two odd vertices**. The set O forms an independent set.

2. If at least one of m, n is in E, their product $m \cdot n$ is even. Thus, **an edge exists between any two even vertices, and between any even and any odd vertex**. The set E forms a clique, and the sets (E, O) are fully connected.

A triangle requires three mutually connected vertices. The fact that O is an independent set critically constrains the possible structure of any triangle: a triangle can contain at most one vertex from O.

Sub-Question 1: Calculate the total number of triangles and evaluate the given options.

Determine if the number of triangles in G is:

a) 19600

b) 61250

c) 80850

Step 1

Step 1: Identify Valid Triangle Structures

As established in the background, no two odd vertices are connected. Therefore, any set of three vertices containing two or more odd numbers (e.g., {O, O, E} or {O, O, O}) cannot form a triangle. This leaves only two possible structures for a triangle, based on the parity of its vertices:

- Case 1: All three vertices are from the set of even numbers, E. (Type {E, E, E})
- Case 2: Two vertices are from E and one vertex is from the set of odd numbers, O. (Type {E, E, O})

Step 2

Step 2: Case 1 - Counting {E, E, E} Triangles

The set E of 50 even vertices forms a clique (all vertices are interconnected). Therefore, any combination of 3 vertices chosen from E will form a triangle. The number of such triangles is given by the binomial coefficient "50 choose 3":

$$\text{Count}(\{E, E, E\}) = \binom{50}{3} = \frac{50!}{3!(50-3)!} = \frac{50 \times 49 \times 48}{3 \times 2 \times 1} = 50 \times 49 \times 8 = 19600$$

Step 3

Step 3: Case 2 - Counting {E, E, O} Triangles

For this type of triangle, we need to select 2 vertices from the 50 even numbers and 1 vertex from the 50 odd numbers. Since any two even vertices are connected, and every even vertex is connected to every odd vertex, any such selection will form a valid triangle. The number of ways to do this is the product of the number of ways to choose the even vertices and the number of ways to choose the odd vertex:

$$\text{Count}(\{E, E, O\}) = \binom{50}{2} \times \binom{50}{1} = \left(\frac{50 \times 49}{2}\right) \times 50 = 1225 \times 50 = 61250$$

Step 4

Step 4: Calculate the Total Number of Triangles

The total number of triangles is the sum of the counts from the two disjoint cases:

$$\text{Total Triangles} = \text{Count}(\{E, E, E\}) + \text{Count}(\{E, E, O\}) = 19600 + 61250 = 80850$$

Step 5

Step 5: Final Answer Justification

Based on our calculation, the total number of triangles in the graph is 80850. We can now evaluate the given options:

- **a) 19600:** NO. This value is incorrect as it only counts triangles of type {E, E, E} and completely omits the {E, E, O} case.
- **b) 61250:** NO. This value is incorrect as it only counts triangles of type {E, E, O} and completely omits the {E, E, E} case.
- **c) 80850:** YES. This value is the correct sum of the counts for all possible types of triangles in the graph.

Problem 11: Solve a combinatorial distribution problem.

Question

We want to give 10 indistinguishable cookies to 4 distinguishable people. In how many ways can we do this if each person is to receive an odd number of cookies?

Background

This problem is a classic exercise in combinatorics, often solved using the 'stars and bars' method, but with an additional constraint. The task is to find the number of integer solutions to an equation that models the distribution.

Let x_i be the number of cookies received by person i, for $i = 1, 2, 3, 4$. Since the cookies are indistinguishable and the total is 10, we can express the problem as finding the number of solutions to the equation:

$$x_1 + x_2 + x_3 + x_4 = 10$$

The crucial constraint is that each x_i must be a positive odd integer. This means $x_i \in \{1, 3, 5, ...\}$. To handle this, we perform a variable substitution. Any positive odd integer x_i can be written in the form $x_i = 2y_i + 1$, where y_i is a non-negative integer ($y_i \geq 0$). This transformation is the non-trivial logical step, as it converts the 'oddness' constraint into a standard 'non-negative integer' constraint, making the problem solvable with the standard stars and bars formula.

Sub-Question 1: Variant (a): The number of ways is 20.

Is it true that there are exactly 20 ways to distribute the cookies according to the given rules?

Step 1

We start with the equation and constraints established in the background: $x_1 + x_2 + x_3 + x_4 = 10$, where each x_i is a positive odd integer.

Apply the substitution $x_i = 2y_i + 1$, where $y_i \geq 0$ is an integer. This transforms the equation:

$$(2y_1 + 1) + (2y_2 + 1) + (2y_3 + 1) + (2y_4 + 1) = 10$$

Now, we simplify the equation:

$$2(y_1 + y_2 + y_3 + y_4) + 4 = 10$$
$$2(y_1 + y_2 + y_3 + y_4) = 6$$
$$y_1 + y_2 + y_3 + y_4 = 3$$

The problem is now reduced to finding the number of non-negative integer solutions to this new equation. This is a classic 'stars and bars' problem. We have $n = 3$ 'stars' (the sum) to distribute into $k = 4$ 'bins' (the variables). The formula is:

$$\binom{n+k-1}{k-1}$$

Plugging in our values:

$$\binom{3+4-1}{4-1} = \binom{6}{3} = \frac{6!}{3!(6-3)!} = \frac{6 \times 5 \times 4}{3 \times 2 \times 1} = 20$$

The calculation shows there are exactly 20 ways. Therefore, the statement is true.

Final Answer: YES

Sub-Question 2: Variant (b): The number of ways is $\binom{13}{3}$.

Is it true that the number of ways is given by the binomial coefficient $\binom{13}{3}$?

Step 1

From the analysis in part (a), the correct number of ways is 20.

Let's evaluate the given expression:

$$\binom{13}{3} = \frac{13!}{3!(13-3)!} = \frac{13 \times 12 \times 11}{3 imes 2 imes 1} = 13 \times 2 \times 11 = 286$$

The value $\binom{13}{3}$ corresponds to the number of non-negative integer solutions to $z_1 + z_2 + z_3 + z_4 = 10$, which is the version of the problem *without* the odd number constraint ($\binom{10+4-1}{4-1} = \binom{13}{3}$). This is a common mistake, ignoring the problem's central constraint.

Since $286 \neq 20$, the statement is false.

Final Answer: NO

Sub-Question 3: Variant (c): The number of ways is 35.

Is it true that there are 35 ways to distribute the cookies?

Step 1

The correct calculation from part (a) established the number of ways is 20.

Let's investigate where the number 35 might come from. An answer of 35 corresponds to the binomial coefficient $\binom{7}{3}$:

$$\binom{7}{3} = \frac{7!}{3!(7-3)!} = \frac{7 \times 6 \times 5}{3 imes 2 imes 1} = 35$$

This result would be obtained from the stars and bars formula $\binom{n+k-1}{k-1}$ if $k = 4$ and $n = 4$. This implies solving the equation $y_1 + y_2 + y_3 + y_4 = 4$. This is an incorrect simplification of the original transformed equation, which was $2(y_1 + y_2 + y_3 + y_4) = 6$. A simple arithmetic error (e.g., $10 - 2 = 8$ instead of $10 - 4 = 6$) could lead to this incorrect intermediate step.

Since the correct result is 20, the statement is false.

Final Answer: NO

Problem 21: Analyze properties of n-node red-black trees.

Question

For infinitely many n, there exists an n -node red-black tree with:

Background

A red-black tree is a self-balancing binary search tree that maintains its balance by satisfying a set of invariants. Let R be the number of red nodes and B be the number of non-null black nodes. The key invariants are:

1. Every node is either red or black.

2. The root is black.

3. Every leaf (NIL) is black.

4. If a node is red, then both its children are black. (The 'Red' property). This implies a red node cannot have a red parent.

5. For each node, all simple paths from the node to descendant leaves contain the same number of black nodes. (The 'Black-height' property). The number of black nodes on any root-to-leaf path is constant for a given tree, denoted $bh(T)$.

These properties impose strong structural constraints on the tree, particularly on the relative number of red and black nodes and on the lengths of paths.

Sub-Question 1: Part (a): The number of red nodes is greater by 1000 than the number of non-null black nodes.

Is it possible to have $R = B + 1000$?

Step 1

The answer is **NO**.

Step 2

The justification relies on a fundamental theorem of red-black trees that bounds the number of red nodes relative to the total number of nodes, which can be derived from the invariants.

Step 3

Consider Invariant 2 (root is black) and Invariant 4 (a red node's children must be black). A direct consequence of Invariant 4 is that a red node's parent must

be black. Since the root is also black, every red node in the tree has a black parent.

Step 4

This parent-child relationship between red and black nodes implies that red nodes can only appear as children of black nodes. On any path from the root, there cannot be two consecutive red nodes. This implies that at least half of the nodes on any path are black.

Step 5

A well-known property that follows from this is that the number of red nodes, R, is at most the number of black nodes, B. More formally, for a tree with $n = R + B > 1$ nodes, it can be proven that $R \leq B - 1$. This is because the total number of nodes n is at least $2R + 1$, so $R + B \geq 2R + 1$, which simplifies to $B \geq R + 1$.

Step 6

The condition $R = B + 1000$ directly violates the property $R < B$. Therefore, no such red-black tree can exist, for any n.

Sub-Question 2: Part (b): The number of non-null black nodes is 3 times greater than the number of red nodes.

Is it possible to have $B = 3R$ for infinitely many n?

Step 1

The answer is **YES**.

Step 2

To prove this, we need to construct an infinite family of red-black trees that satisfy this property. This can be done using a recursive construction based on a 'gadget' tree.

Step 3

Step 1: Construct a base case gadget, T_0. Consider the following 4-node tree: a black root with a black left child and a red right child. The red right child has one black child. This tree has $R = 1$ and $B = 3$. The ratio $B/R = 3$ holds. All red-black properties are satisfied (e.g., black-height is 2). This tree has 4 leaves (NIL nodes).

Step 4

Step 2: Define a recursive construction. Let T_k be a tree from our family. We define T_{k+1} by taking the tree T_k and replacing every NIL leaf with a copy of our base gadget T_0. Since the root of T_0 is black, it can be attached to any node in T_k without violating the 'Red' property.

Step 5

Step 3: Show that the property is maintained. Let R_k and B_k be the counts for T_k, and let L_k be the number of NIL leaves in T_k. Assume for T_k we have $B_k = 3R_k$. When constructing T_{k+1}, we add L_k copies of T_0.

$$R_{k+1} = R_k + L_k \cdot R_0 = R_k + L_k \cdot 1$$
$$B_{k+1} = B_k + L_k \cdot B_0 = B_k + L_k \cdot 3$$

Substituting the inductive hypothesis $B_k = 3R_k$:

$$B_{k+1} = 3R_k + 3L_k = 3(R_k + L_k) = 3R_{k+1}$$

The property holds for all $k \geq 0$. Since this construction can be repeated indefinitely, creating ever-larger trees, there exists such a tree for infinitely many n.

Sub-Question 3: Part (c): The number of nodes on the longest path from the root to a leaf is 1.5 times greater than the number of nodes on the shortest path.

Is it possible to have $l = 1.5s$ for infinitely many n? (where l is the longest and s is the shortest root-to-leaf path length)

Step 1

The answer is **YES**.

Step 2

The non-trivial insight is to analyze path lengths in terms of the tree's black-height, bh. The number of nodes on any root-to-leaf path is the sum of its black nodes (which is always bh) and its red nodes.

Step 3

A shortest path, s, will maximize black-node adjacencies. The shortest possible path has 0 red nodes, consisting of bh black nodes. So, $s \geq bh$.

A longest path, l, will maximize red nodes. Due to the 'Red' property, the path alternates colors as much as possible (e.g., B-R-B-R...). A path with bh black nodes can have at most $bh - 1$ red nodes. So, $l \leq bh + (bh - 1) = 2bh - 1$. This shows the ratio $l/s < 2$, so 1.5 is plausible.

Step 4

We need to find a construction where $l/s = 1.5$. Let's assume the shortest path is composed entirely of black nodes, so $s = bh$. Let the longest path have l_R red nodes, so $l = bh + l_R$. The condition becomes:

$$\frac{bh + l_R}{bh} = 1.5 \implies bh + l_R = 1.5 \cdot bh \implies l_R = 0.5 \cdot bh$$

This equation has integer solutions if bh is an even number. Let $bh = 2k$ for any integer $k \geq 1$. Then we require $l_R = k$. This is a valid number of red nodes, as $k \leq 2k - 1$ for $k \geq 1$.

Step 5

Therefore, for any $k \geq 1$, we can construct a tree with black-height $bh = 2k$ that has a shortest path of $s = 2k$ nodes (all black) and a longest path of $l = 2k + k = 3k$ nodes ($2k$ black, k red). The ratio is $l/s = 3k/2k = 1.5$. Since we can do this for any $k \geq 1$, we can generate an infinite sequence of such trees with increasing n.

Problem 22: Analyze disjoint-set (union-find) forest complexity.

Question

A disjoint-set (union-find) forest data structure contains n elements and uses union by size and path compression. It follows that:

(a) every tree in the forest is a binary tree.

(b) the worst-case time of a single `find` operation is $\Theta(\log n)$.

(c) the total time of any sequence of m `find` / `union` operations is $O(n + m)$.

Background

The disjoint-set union (DSU) or union-find data structure is designed to efficiently manage a partition of a set into disjoint subsets. It is represented as a forest, where each tree corresponds to a subset, and the root of the tree serves as the unique representative of that subset. The structure is typically implemented using a simple parent-pointer array, where each node stores a reference to its parent. The root of a tree points to itself or a null value.

To achieve high efficiency, two heuristics are crucial: **union by size** (or rank) and **path compression**. Union by size dictates that during a union operation, the root of the smaller tree is attached as a child to the root of the larger tree. This heuristic alone guarantees that the height of any tree with k nodes is at most $O(\log k)$. This is the key to analyzing the worst-case complexity of a single operation.

Path compression is applied during a `find` operation. After the root of the tree is found, all nodes on the traversal path from the starting node to the root are made direct children of the root. This flattens the tree structure over time, drastically speeding up future `find` operations.

The combination of these two heuristics results in an amortized time complexity that is nearly constant. The total time for m operations on n elements is tightly bounded by $O((n + m)\alpha(n))$, where $\alpha(n)$ is the extremely slow-growing inverse Ackermann function. It is critical to distinguish this *amortized* complexity from the *worst-case* complexity of a single operation.

Sub-Question 1: Variant (a): Tree Structure

Is it true that every tree in the forest is a binary tree?

Step 1

Answer: NO

Step 2

The fundamental logical step is to understand how the forest structure evolves. In the parent-pointer representation of a disjoint-set forest, a node only stores a pointer to its parent. There is no mechanism that restricts a parent node to having at most two children. The trees are general trees, not necessarily binary trees.

A `union` operation makes the root of one tree a new child of the root of another. The second root can acquire an arbitrary number of children through successive `union` operations.

Consider this counterexample: Start with $n = 4$ elements, initially in four separate sets: $\{1\}, \{2\}, \{3\}, \{4\}$. Perform the following sequence of operations:

1. `union(1, 4)`: Element 1 becomes a child of element 4.
2. `union(2, 4)`: Element 2 becomes a child of element 4.
3. `union(3, 4)`: Element 3 becomes a child of element 4.

After these operations, the root element 4 has three children: 1, 2, and 3. The resulting tree is not a binary tree.

Sub-Question 2: Variant (b): Single Operation Worst-Case Time

Is it true that the worst-case time of a single `find` operation is $\Theta(\log n)$?

Step 1

Answer: YES

Step 2

The non-trivial insight here is to distinguish the single-operation worst-case complexity from the more commonly cited amortized complexity. While path compression leads to the near-constant amortized time of $O(\alpha(n))$, it does not prevent a single, specific `find` operation from being slow.

The worst-case runtime of a single `find` operation is proportional to the depth of the element being found, which is bounded by the maximum height of any tree in the forest. The **union by size** heuristic is precisely what bounds the tree height. When we union two trees, the root of the smaller tree is attached to the root of the larger one. The depth of a node x increases only if the tree containing it is merged into a tree that is at least as large. This means that every time the depth of x increases by one, the size of the new tree containing x at least doubles. Since the total number of elements is n, this can happen at most $\log_2 n$ times. Thus, the maximum height of any tree is $O(\log n)$.

To establish the $\Omega(\log n)$ lower bound, one can construct a sequence of union operations that builds a tree of logarithmic height. This can be done by systematically merging pairs of trees of equal size. For example, union pairs of single-element sets, then pairs of 2-element sets, and so on. This creates a binomial-like tree structure with logarithmic height. A `find` on the deepest node in such a tree will take $\Omega(\log n)$ time. Since the complexity is both $O(\log n)$ and $\Omega(\log n)$, it is $\Theta(\log n)$.

Sub-Question 3: Variant (c): Total Time for a Sequence of Operations

Is it true that the total time of any sequence of m operations is $O(n + m)$?

Step 1

Answer: NO

Step 2

This question probes a very subtle point in theoretical computer science. The non-trivial jump is recognizing the formal definition of Big-O notation and the true nature of the inverse Ackermann function.

The tight upper bound on the total time for a sequence of m union-find operations on n elements, using both union by size and path compression, was proven by Robert Tarjan to be $\Theta((n + m)\alpha(n))$. The function $\alpha(n)$ is the inverse Ackermann function.

While $\alpha(n)$ grows incredibly slowly (e.g., $\alpha(n) < 5$ for any value of n that can be written down in the physical universe), it is **not** a constant. It is an unbounded, monotonically increasing function. In the formal definition of asymptotic notation, this means $(n+m)\alpha(n)$ grows asymptotically faster than $n+m$. Therefore, $O(n + m)$ is not a valid upper bound for the complexity. The statement is formally false.

Problem 23: Analyze an SQL GROUP BY query.

Question

A table T is defined as `CREATE TABLE T (A INT NOT NULL, B INT NOT NULL)`. Consider the following query:

```
SELECT A, SUM(B) FROM T GROUP BY A
```

It follows that:

Background

This problem tests the fundamental understanding of SQL aggregate functions and the `GROUP BY` clause. The core, non-trivial insight is to recognize that `GROUP BY` is a *partitioning* operation. It divides the entire set of rows from a table into disjoint subsets (groups) based on the distinct values of the specified column(s) — in this case, column A. After this partitioning, aggregate functions (like `SUM()`) operate independently on each partition. The final result set contains exactly one row per partition.

Sub-Question 1: a) The number of rows returned is greater than or equal to the number of rows in T

Is it true that the number of rows returned by the query is always greater than or equal to the number of rows in table T?

Step 1

The query `GROUP BY A` collapses all rows with an identical value in column A into a single output row. Therefore, the number of rows in the result set is equal to the number of *distinct* values in column A.

Let N be the total number of rows in table T, and let D be the number of distinct values in column A. By definition, the number of distinct values in a set cannot be greater than the total number of elements in that set. Thus, we always have $D \leq N$. The number of rows returned is D.

The statement claims $D \geq N$. This is only true in the specific case where $D = N$ (i.e., all values in column A are unique). However, the statement must hold for all possible table contents. Consider a simple counterexample where T contains two rows: $(1, 10)$ and $(1, 20)$. Here, $N = 2$, but there is only one distinct value in column A (which is 1), so $D = 1$. The query returns one row. The inequality $1 \geq 2$ is false.

Final Answer: **NO**. The number of returned rows is the number of distinct values in the grouping column, which cannot exceed the total number of rows.

Sub-Question 2: b) The number of rows returned is less than or equal to the number of rows in T

Is it true that the number of rows returned by the query is always less than or equal to the number of rows in table T?

Step 1

As established in the previous part, let N be the total number of rows in T and D be the number of distinct values in column A. The query returns D rows.

The statement claims $D \leq N$. This is the fundamental property of distinct counts. The number of distinct items in a collection can never exceed the total number of items in that collection.

- Limiting Case 1: Every row in T has a unique value for column A. In this scenario, each row forms its own group of size one. The number of distinct values D is equal to the total number of rows N. The inequality $D \leq N$ holds (as $N \leq N$).
- Limiting Case 2: At least two rows in T share the same value for column A. In this case, those rows are collapsed into a single group, and the number of distinct values D will be strictly less than the total number of rows N. The inequality $D \leq N$ still holds.

Since the inequality holds in all possible cases, the statement is true.

Final Answer: **YES**. The number of groups can at most be equal to the number of items being grouped.

Sub-Question 3: c) The sum of all values in the second column of the result is equal to the sum of all values in column B of table T

Is it true that if we sum up the values in the second column of the query's result set, we get the same value as summing up all values in column B of the original table T?

Step 1

The non-trivial logical step is to view the `GROUP BY` operation as creating a partition of the set of rows in table T. Let R be the set of all rows in T. The grouping operation partitions R into disjoint subsets $P_1, P_2, \ldots, P_D$, where D is the number of distinct values of A. By definition of a partition, $R = \bigcup_{i=1}^{D} P_i$ and $P_i \cap P_j = \emptyset$ for $i \neq j$.

The second column of the result set consists of the values $S_i = \sum_{r \in P_i} r.B$ for each partition P_i. That is, for each group, the query computes the sum of B values.

The question asks to evaluate the sum of all values in this second column, which is $\sum_{i=1}^{D} S_i$. Substituting the definition of S_i, we get:

$$\sum_{i=1}^{D} S_i = \sum_{i=1}^{D} \left(\sum_{r \in P_i} r.B \right)$$

Because the sets P_i form a partition of the total set of rows R, summing the sums of the parts is equivalent to summing over the whole. This is a fundamental property of summation.

$$\sum_{i=1}^{D} \left(\sum_{r \in P_i} r.B \right) = \sum_{r \in R} r.B$$

The right-hand side is precisely the definition of the sum of all values in column B of the original table T. Therefore, the statement is true.

Final Answer: **YES**. The sum of group-wise sums is always equal to the total sum because the grouping operation is a partition.

Problem 24: Analyze a mutual exclusion algorithm.

Question

Two processes, P1 and P2, attempt to ensure mutual exclusion using three shared integer variables: x, y, and z, all initialized to 0. Their respective code segments are:

Process P1:

```
L1: x=1; z=0; while(y!=z){}; critical_section(); x=0;
```

Process P2:

```
L2: y=1; z=1; while(x==z){}; critical_section(); y=0;
```

The problem asks whether this solution (a) ensures mutual exclusion, (b) may lead to a deadlock, and (c) ensures alternation between the processes.

Background

This problem tests understanding of fundamental concepts in concurrency control for shared-memory systems. The three key properties to evaluate are:

1. Mutual Exclusion: At most one process can be in its critical section at any given time.

2. Deadlock: A state where two or more processes are blocked forever, each waiting for an event that only another blocked process can cause.

3. Alternation: Processes are forced to strictly alternate their turns in entering the critical section.

The non-trivial logical jump in analyzing this specific algorithm lies in recognizing the implications of how modern computer architectures handle shared memory. The code uses simple variable assignments and checks (a software-only solution) without any atomic instructions or memory fences. In such cases, the behavior is highly dependent on the system's *memory consistency model*. Under a strict model like sequential consistency, all processes observe memory writes in the same order. However, most modern processors use weaker, more performant models where the order and visibility of writes across different processor cores are not guaranteed without explicit synchronization commands. This can lead to scenarios where a process reads a "stale" value of a shared variable, causing concurrency bugs like deadlock.

Sub-Question 1: (a) Does the solution ensure mutual exclusion?

Determine if it's possible for both processes P1 and P2 to be in their respective critical sections simultaneously.

Step 1

Answer: YES

First, let's establish the conditions required for each process to enter its critical section. A process exits its `while` loop and enters the critical section only when its loop condition becomes false.

- P1 enters when `y!=z` is false, i.e., when $y = z$.
- P2 enters when `x==z` is false, i.e., when $x \neq z$.

Step 2

Consider the state of the shared variables just before the processes check their loop conditions. Both processes signal their intent to enter by setting $x = 1$ (P1) and $y = 1$ (P2). The crucial variable is z, which is subject to a race condition. Its value will be either 0 (if P1's write is last) or 1 (if P2's write is last).

Step 3

Case 1: P1's write to z is last. The state before the loops is $(x = 1, y = 1, z = 0)$.

- P1's condition: $y = z \implies 1 = 0$. This is false. P1 waits.
- P2's condition: $x \neq z \implies 1 \neq 0$. This is true. P2 enters the critical section.

Step 4

Case 2: P2's write to z is last. The state before the loops is $(x = 1, y = 1, z = 1)$.

- P1's condition: $y = z \implies 1 = 1$. This is true. P1 enters the critical section.
- P2's condition: $x \neq z \implies 1 \neq 1$. This is false. P2 waits.

Step 5

In both possible non-deadlocking executions, the final value of z acts as a tie-breaker, allowing exactly one process to satisfy its entry condition. If a deadlock occurs (as shown in part b), neither process enters the critical section. Therefore, in all possible scenarios, mutual exclusion is preserved.

Sub-Question 2: (b) May the solution lead to a deadlock?

Analyze if there exists an execution trace where both processes become permanently blocked.

Step 1

Answer: YES

The potential for deadlock arises from the combination of a specific instruction interleaving and the effects of a weak memory consistency model, common in real-world systems. Without explicit synchronization primitives, a write to a shared variable by one process may not be immediately visible to another.

Step 2

Consider the following sequence of events:

1. P1 executes `x=1;`. State: $(x = 1, y = 0, z = 0)$.
2. A context switch occurs. P2 executes `y=1;`. State: $(x = 1, y = 1, z = 0)$.
3. A context switch occurs. P1 executes `z=0;` and proceeds to its loop. It checks `while(y!=z)`, which is `while(1!=0)`. The condition is true, and P1 begins to spin.
4. A context switch occurs. P2 executes `z=1;` and proceeds to its loop. It checks `while(x==z)`, which is `while(1==1)`. The condition is true, and P2 begins to spin.

Step 3

At this point, a deadlock has occurred. P1 is spinning, waiting for y to become equal to z. However, P2 has set $z = 1$, and P1 sees $y = 1$. P1's condition would become false if it re-read the new value of z. However, under a weak memory model, there's no guarantee P1 will see P2's write to z in a timely manner. More critically, P2 is also stuck. P2 is waiting for x to be unequal to z (i.e., for x to become 0), an action only P1 can perform after its critical section. Since P1 is blocked from entering its critical section, P2 is blocked forever. This constitutes a deadlock.

Sub-Question 3: (c) Does the solution ensure alternation?

Determine if the processes are forced to take turns entering the critical section.

Step 1

Answer: NO

The algorithm does not enforce strict alternation. A process that has just exited the critical section can, if scheduled appropriately, re-enter before the other process has a chance to run. We can show this with a counterexample.

Step 2

Consider the following execution trace, starting from the initial state ($x = 0, y = 0, z = 0$):

1. P1 is scheduled. It executes `x=1; z=0;`. The current state of shared variables is $(1, 0, 0)$.
2. P1 checks its condition `while(y!=z)`. With $y = 0$ and $z = 0$, the condition is false. P1 enters the critical section.
3. P1 completes its critical section and executes `x=0;`. The state returns to $(0, 0, 0)$.
4. Crucially, assume the scheduler immediately gives the CPU back to P1, before P2 has any chance to run.
5. P1 starts its loop again. It executes `x=1; z=0;` and once again checks the condition `while(y!=z)`. Since P2 has not run, y is still 0. The condition is $0! = 0$, which is false. P1 enters the critical section for a second consecutive time.

This sequence demonstrates that P1 can enter the critical section multiple times in a row, starving P2. Therefore, strict alternation is not guaranteed.

Problem 25: Analyze message reception in communication models.

Question

The problem asks whether it is possible for a process to receive a message after the process that sent the message has already terminated. This is evaluated under three different communication paradigms:

(a) Direct asynchronous communication

(b) Direct synchronous communication

(c) Indirect communication via a tuple space

Background

Understanding this problem requires distinguishing between the core properties of inter-process communication (IPC) models. The key axes of classification are synchronicity (synchronous vs. asynchronous) and directness (direct vs. indirect).

Synchronous vs. Asynchronous Communication: This distinction hinges on whether the `send` operation is blocking. In *synchronous* communication (also known as a rendezvous), the sender blocks until the receiver is ready to accept the message. The transfer is a tightly coupled event. In *asynchronous* communication, the `send` operation is non-blocking; the sender hands the message to a buffer or message queue and continues its execution immediately, decoupling the act of sending from the act of receiving.

Direct vs. Indirect Communication: This concerns how processes address each other. In *direct* communication, the sender explicitly names the recipient (e.g., `send(message, process_P)`). In *indirect* communication, messages are sent to an intermediary, such as a mailbox, port, or tuple space. This decouples processes in space (they don't need to know each other's identity) and potentially in time (they don't need to co-exist). A tuple space is a specific form of intermediary, an associative shared memory where processes can write, read, or take tuples (structured data) based on their content rather than an address.

Sub-Question 1: Variant (a): Direct Asynchronous Communication

Is it possible to receive a message after the process that sent it has terminated in the direct asynchronous communication model?

Step 1

Answer: YES

Step 2

The non-trivial logical step is to understand that 'asynchronous' implies the use of an intermediary buffer, even in a 'direct' communication model. The `send` operation is non-blocking. When a sender process executes `send(message, receiver_ID)`, the operating system or runtime environment copies the message into a message queue or buffer associated with the receiver. The sender's operation is now complete, and it is free to continue execution.

Step 3

This temporal decoupling allows for the following sequence of events:

1. Process A executes a `send` operation, targeting Process B. The message is placed in B's message queue.
2. Process A immediately continues its work, which might be minimal, and proceeds to terminate.
3. The message persists in the system buffer independently of Process A's existence.
4. At some later time, Process B is scheduled, executes a `receive` operation, and retrieves the message from its queue.

Step 4

Therefore, the receiver can indeed get the message long after the sender has terminated. The lifecycles of the two processes are decoupled with respect to the communication event.

Sub-Question 2: Variant (b): Direct Synchronous Communication

Is it possible to receive a message after the process that sent it has terminated in the direct synchronous communication model?

Step 1

Answer: NO

Step 2

The critical concept here is the 'rendezvous'. Synchronous communication mandates that the `send` operation is blocking. This isn't just a detail; it's the defining feature. The sender process's execution is halted at the `send` call and does not resume until the receiver has executed a corresponding `receive` operation and the message transfer is complete.

Step 3

The sequence of events is strictly ordered:

1. Process A calls `send(message, B)` and blocks.
2. Process B calls `receive()`.
3. The message is transferred from A to B.
4. Both processes are unblocked and resume their execution from the point after the communication calls.

Step 4

Since the sending process cannot proceed past the `send` call until the message has been received, it is logically impossible for it to terminate before the reception occurs. The lifecycles are tightly coupled at the point of communication.

Sub-Question 3: Variant (c): Indirect Communication via a Tuple Space

Is it possible to receive a message after the process that sent it has terminated in indirect communication via a tuple space?

Step 1

Answer: YES

Step 2

The non-trivial insight here is that indirect communication via an intermediary like a tuple space provides both *spatial decoupling* (processes don't need to know each other's identity) and *temporal decoupling* (processes don't need to exist at the same time). The tuple space acts as a persistent, shared data store.

Step 3

The operation of writing to a tuple space is analogous to an asynchronous send. It is typically non-blocking. The sender's interaction is with the tuple space itself, not with any specific receiver.

1. Process A executes a `write(tuple)` operation. The tuple is now stored in the tuple space.
2. Process A is now free to continue its work and terminate.
3. The tuple persists in the space, completely independent of Process A.

4. At any point in the future (seconds, hours, or days later, depending on the system's persistence guarantees), Process B can execute a `read(template)` or `take(template)` operation that matches the tuple and retrieve it.

Step 4

This model provides the strongest form of decoupling. The message's lifecycle is tied to the intermediary (the tuple space), not the sender or receiver, making it trivial for a message to be received after the sender has terminated.

Problem 26: Analyze Python code with list aliasing.

Question

Analyze the following Python code snippet and determine the truth value of the statements that follow. The initial code is:

```
a = b = [1, 2, 3]
a.append(4)
```

Each of the following statements assumes only the initial code has been executed, and each statement's operation is independent of the others.

Background

This problem tests the understanding of Python's data model, specifically how names (variables) relate to objects. In Python, names are references to objects stored in memory. The core concepts are:

1. Mutability: Objects can be mutable (their state can be changed after creation, e.g., lists, dictionaries) or immutable (their state cannot be changed, e.g., tuples, strings, integers). Lists are mutable.

2. Aliasing: The statement `a = b = [1, 2, 3]` creates a single list object and makes both names, `a` and `b`, refer to that same object. They are aliases for the same underlying data.

3. In-place Mutation vs. New Object Creation: Methods like `.append()` modify the object directly (in-place). Operations like concatenation (`+`) or slicing (`[:]`) create a new object. An assignment (`b = ...`) then rebinds the name on the left to the new object on the right.

4. Identity vs. Equality: The `is` operator checks for object identity (if two names refer to the same object in memory, i.e., `id(a) == id(b)`). The `==` operator checks for equality (if the contents of the objects are the same).

Sub-Question 1: Variant a

If we then execute `b = a + [5, 6]`, then `a` will be `[1, 2, 3, 4, 5, 6]`.

Step 1

First, let's establish the state after the initial setup. The line `a = b = [1, 2, 3]` creates one list object and assigns two names, `a` and `b`, to it. The line `a.append(4)` mutates this single object in-place. Therefore, after the setup, both `a` and `b` refer to the list `[1, 2, 3, 4]`.

Step 2

The expression `a + [5, 6]` involves the list concatenation operator. This operation does not modify `a`. Instead, it creates a new list object in memory, which is the result of concatenating `[1, 2, 3, 4]` and `[5, 6]`. The new list is `[1, 2, 3, 4, 5, 6]`.

Step 3

The assignment `b = ...` rebinds the name `b` to this newly created list. The name `a` is completely unaffected by this operation; it still refers to the original list object.

Step 4

Therefore, the final value of `a` remains `[1, 2, 3, 4]`. The statement claims it will be `[1, 2, 3, 4, 5, 6]`, which is incorrect.

Step 5

Answer: **NO**. The concatenation operator creates a new list and rebinds `b` to it, leaving the original list referenced by `a` unchanged.

Sub-Question 2: Variant b

If we then execute `b.append(5)`, then `a` will be `[1, 2, 3, 4, 5]`.

Step 1

As established, after the initial setup, `a` and `b` are aliases for the same list object: `[1, 2, 3, 4]`.

Step 2

The line `b.append(5)` calls the `.append()` method on the object referenced by `b`. This is an in-place mutation. It modifies the list object directly by adding an element to its end.

Step 3

Since `a` is an alias for the same object, the change made via `b` is visible through `a`. The object itself has been changed to `[1, 2, 3, 4, 5]`.

Step 4

The statement asserts that the final value of `a` will be `[1, 2, 3, 4, 5]`. This is correct.

Step 5

Answer: **YES**. Because `a` and `b` are aliases for the same mutable list object, an in-place mutation through one name is visible through the other.

Sub-Question 3: Variant c

If we then execute `b = a[:]`, the expression `a is b` evaluates to `True`.

Step 1

Again, the starting state is that `a` and `b` are aliases for the same list object, `[1, 2, 3, 4]`. At this point, `a is b` is `True`.

Step 2

The expression `a[:]` uses slicing to create a shallow copy of the list. This is a non-trivial leap: slicing does not return a view or a reference to the original, but an entirely new list object that has the same contents.

Step 3

The assignment `b = a[:]` rebinds the name `b` to this new copy. Now, `a` refers to the original list and `b` refers to the new, copied list. They are no longer aliases.

Step 4

The `is` operator checks for object identity. Since `a` and `b` now refer to two different objects in memory (even though their contents are equal), the expression `a is b` will evaluate to `False`. The statement claims it will be `True`, which is incorrect.

Step 5

Answer: **NO**. The slicing operation `a[:]` creates a new list object. Rebinding `b` to this new object breaks the alias, so `a` and `b` no longer refer to the same object.

Problem 27: Analyze Python code with variable scope.

Question

Consider the following Python code snippet:

```
x = 0

def f1():
    x += 2

def f2():
    global x
    x = 2

def f3():
    if False:
        x += 2

# Which of the following function calls, when uncommented,
# will result in the program printing the value 2?

# a) f1()
# b) f2()
# c) f3()

# (Assume only one function is called)
# print(x)
```

Which function, when called, results in the program printing the value 2 after the call?

Background

This problem dissects Python's variable scoping rules, specifically the distinction between local and global scopes and the mechanism for modifying global variables from within a function. Python's scope resolution follows the LEGB (Local, Enclosing, Global, Built-in) rule.

The non-trivial logical leap lies in how Python determines a variable's scope. When Python compiles the code for a function, it decides whether a variable is local or not. If there is *any* assignment operation to a variable within a function's body (e.g., $x = ...$, $x+ = ...$), that variable is implicitly considered local to that function for its entire scope. This decision is made at compile time, before the function is ever executed.

This pre-emptive classification can lead to an `UnboundLocalError`. If a variable is marked as local due to an assignment later in the function, attempting to read

from it before the assignment occurs will fail, as the interpreter will be looking for a local variable that has not yet been initialized. It will not fall back to the global scope.

To override this default behavior and explicitly modify a global variable, one must use the `global` keyword. This keyword informs the interpreter that all references and assignments to the specified variable within the function should target the global variable, not a newly created local one.

Sub-Question 1: Variant (a): Calling `f1()`

Does calling `f1()` result in the program printing 2?

Step 1

First, analyze the operation inside `f1()`: `x += 2`. This is an augmented assignment statement, which is functionally equivalent to `x = x + 2`.

Step 2

The presence of this assignment operation causes the Python compiler to treat `x` as a **local variable** throughout the entire scope of the function `f1()`. The global variable `x` is shadowed and inaccessible for reading within this function.

Step 3

At runtime, when `f1()` is called, the interpreter attempts to evaluate the right-hand side of the assignment, `x + 2`. To do this, it needs to fetch the current value of `x`. Since `x` has been marked as a local variable, it looks for it in the local scope. However, no value has been assigned to the local variable `x` yet.

Step 4

This results in an `UnboundLocalError: local variable 'x' referenced before assignment`. The program will crash, and the `print(x)` statement will not be executed. The value of the global `x` remains 0.

Step 5

Final Answer: NO. Calling `f1()` raises an error and does not result in printing 2.

Sub-Question 2: Variant (b): Calling `f2()`

Does calling `f2()` result in the program printing 2?

Step 1

The function `f2()` begins with the statement `global x`. This is an explicit directive to the Python interpreter.

Step 2

This directive overrides the default scoping behavior. It instructs the interpreter that any reference or assignment to the variable `x` within the scope of `f2()` should target the variable `x` that exists in the global scope.

Step 3

Consequently, the line `x = 2` does not create a new local variable. Instead, it modifies the existing global variable `x`, changing its value from 0 to 2.

Step 4

After the call to `f2()` completes, the program proceeds to the `print(x)` statement. This statement reads the value of the global variable `x`, which is now 2.

Step 5

Final Answer: YES. Calling `f2()` correctly modifies the global variable, and the program prints 2.

Sub-Question 3: Variant (c): Calling `f3()`

Does calling `f3()` result in the program printing 2?

Step 1

The code inside `f3()` is contained within an `if False:` block.

Step 2

A conditional statement with a constant boolean value of `False` creates a block of unreachable code. The statement `x += 2` will never be executed. Some static analysis tools or compilers might even warn about or remove this dead code.

Step 3

It is worth noting that even though the assignment `x += 2` is present in the function's source code, because it is unreachable, the Python interpreter does not treat `x` as a local variable. A call to `f3()` does not raise an error, unlike `f1()`.

Step 4

Calling `f3()` is effectively a no-operation (a 'no-op'). The function is called and immediately returns without altering the program state in any way. The global variable `x` remains at its initial value of 0.

Step 5

The subsequent `print(x)` statement will therefore print 0.

Step 6

Final Answer: NO. Calling `f3()` has no effect, and the program prints 0.

Problem 28: Analyze Python code with OOP and name mangling.

Question

Given the following Python code:

```
class A:
    def __init__(self):
        self.x = 1
        self.__x = 1

a = A()
```

Determine which of the following statements will execute without an error.

Background

Python's approach to attribute visibility differs from languages like Java or C++. It does not have strictly private attributes. Instead, it relies on conventions and a mechanism called name mangling.

1. Public Attributes: Attributes without a leading underscore (e.g., `self.x`) are considered public and are freely accessible from outside the class.

2. Single Underscore Prefix (`self._x`): This is a convention indicating that an attribute is intended for internal use. It does not, however, prevent external access. The Python interpreter does not treat it differently.

3. Double Underscore Prefix (`self.__x`): This triggers name mangling. The interpreter alters the name of the attribute to `_ClassName__attributeName`. This is not a security feature but a mechanism to prevent name clashes in subclasses. If a subclass defines its own `__x` attribute, it will be mangled with the subclass's name, thus avoiding an accidental override of the parent's attribute.

Sub-Question 1: Variant (a): `print(a.x)`

Will the statement `print(a.x)` execute without an error?

Step 1

The line `self.x = 1` in the `__init__` method creates a public instance attribute named `x`.

Step 2

Public attributes are, by definition, accessible from outside the class scope using dot notation on an instance of the class.

Step 3

Therefore, accessing `a.x` is a valid operation. The statement will execute correctly, printing the value `1`.

Step 4

Final Answer: YES

Sub-Question 2: Variant (b): `print(a.__x)`

Will the statement `print(a.__x)` execute without an error?

Step 1

The line `self.__x = 1` uses a double underscore prefix. This syntax instructs the Python interpreter to perform name mangling.

Step 2

The non-trivial step is understanding this transformation. The interpreter changes the name of the attribute from `__x` to `_A__x` (following the pattern `_ClassName__attributeName`). The attribute is stored on the instance under this new, mangled name.

Step 3

When the code attempts to access `a.__x`, it searches for an attribute with the literal name `__x` on the instance `a`. Since no attribute with this exact name exists (it was renamed to `_A__x`), Python raises an `AttributeError`.

Step 4

Note that access is still possible by using the mangled name directly: `print(a._A__x)` *would execute successfully.*

Step 5

Final Answer: NO

Problem 29: Classify IP addresses.

Question

For each of the following IP addresses, determine if the accompanying statement is true.

Background

This problem tests foundational knowledge of IPv4 address allocation, which is governed by standards set by the Internet Engineering Task Force (IETF) and managed by the Internet Assigned Numbers Authority (IANA). The IPv4 address space is not monolithic; it is partitioned into special-use ranges with distinct behaviors. Understanding these partitions is crucial for network configuration and security.

The non-trivial insight lies in understanding the *why* behind these special ranges:

1. Private (Non-Routable) Addresses (RFC 1918): The exhaustion of the 2^{32} IPv4 address space necessitated a mechanism for address reuse. RFC 1918 reserves specific blocks (10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16) for internal use within private networks. These are not unique globally. The key technology that makes this work is Network Address Translation (NAT), where a router maps multiple private IP addresses to a single public IP address for communication with the global internet. Internet backbone routers are configured to drop any traffic with a private source or destination address.

2. Loopback Addresses: The entire 127.0.0.0/8 block is reserved for loopback. A packet sent to any address in this range is immediately looped back by the host's own TCP/IP stack to its own input queue. It never reaches any physical network hardware. This creates a virtual network interface that is indispensable for client-server application testing on a single machine and for many forms of inter-process communication.

3. Broadcast Addresses: Broadcasting sends a packet to all hosts on a given network segment. The address 255.255.255.255 is the 'limited broadcast' address. The critical concept here is its scope: it is restricted to the local broadcast domain (e.g., the local Ethernet segment). Routers, which separate broadcast domains, do not forward these packets. This prevents a local broadcast from flooding the entire internet. This is distinct from a 'directed broadcast' (e.g., 192.168.1.255 for the 192.168.1.0/24 network), which is now largely deprecated due to its use in Smurf attacks.

Sub-Question 1: Variant (a): 10.10.0.1 is globally routable

Determine if the statement that the IP address 10.10.0.1 is globally routable is true.

Step 1

Final Answer: NO

Step 2

The IP address in question is 10.10.0.1.

Step 3

RFC 1918, titled "Address Allocation for Private Internets", reserves several blocks of the IPv4 address space for use in private networks. One of these reserved blocks is the 24-bit block 10.0.0.0/8, which includes all addresses from 10.0.0.0 to 10.255.255.255.

Step 4

The address 10.10.0.1 clearly falls within this range.

Step 5

By definition, addresses within this private range are not globally unique and must not be routed on the public internet. Internet routers are universally configured to drop packets that have a source or destination address within any of the RFC 1918 ranges. Therefore, 10.10.0.1 is not globally routable.

Sub-Question 2: Variant (b): 127.0.0.1 is a loopback address

Determine if the statement that the IP address 127.0.0.1 is a loopback address is true.

Step 1

Final Answer: YES

Step 2

The IP address in question is 127.0.0.1.

Step 3

The IANA has reserved the entire 127.0.0.0/8 address block (from 127.0.0.0 to 127.255.255.255) for loopback purposes. Any traffic sent to an address in this block is handled internally by the host's operating system and TCP/IP stack; it does not generate any traffic on the network.

Step 4

The specific address 127.0.0.1 is the standard, conventional address used for the loopback interface and is commonly associated with the hostname `localhost`. Since it belongs to the loopback block and is the primary address used for this function, the statement is correct.

Sub-Question 3: Variant (c): 255.255.255.255 is a broadcast address

Determine if the statement that the IP address 255.255.255.255 is a broadcast address is true.

Step 1

Final Answer: YES

Step 2

The IP address in question is 255.255.255.255. In IPv4 networking, this address has a specific, reserved meaning.

Step 3

As defined by IETF standards (specifically RFC 919), 255.255.255.255 is the 'limited broadcast' address. A packet with this destination address is intended for every host on the same local network segment as the sender.

Step 4

The key characteristic of this address is that routers will not forward packets destined for 255.255.255.255. This confines the broadcast to the local broadcast domain, preventing it from propagating across the internet. This mechanism is used by protocols like DHCP to discover servers on the local network. Therefore, the statement is correct.

Problem 30: Interpret memory values based on endianness.

Question

Six consecutive 8-bit bytes in memory, beginning at address A, hold the hexadecimal values: `06, 05, 04, 03, 02, 01`. This means the memory is laid out as follows: the byte at address A is `0x06`, the byte at address A+1 is `0x05`, and so on, until address A+5, which holds `0x01`. The problem asks for the decimal value of the 16-bit (two-byte) integer stored at different starting addresses under various byte-ordering conventions (endianness).

Background

This problem tests the fundamental concept of **endianness**, which refers to the order in which bytes of a multi-byte data type are stored in computer memory. There are two primary conventions:

1. Big-Endian: The most significant byte (MSB) of the data is placed at the lowest memory address. This is analogous to how numbers are written in most Western cultures (e.g., in the number 1541, the most significant digit '1' comes first). For a 16-bit hexadecimal number like $0xABCD$, the byte $0xAB$ (the MSB) would be stored at address N and the byte $0xCD$ (the LSB) at address $N + 1$.

2. Little-Endian: The least significant byte (LSB) of the data is placed at the lowest memory address. This may seem counter-intuitive but has hardware advantages. For the same number $0xABCD$, the byte $0xCD$ (the LSB) would be stored at address N and the byte $0xAB$ (the MSB) at address $N + 1$.

A critical piece of information for part (c) is that the **x86** processor architecture, which includes most modern desktop and laptop CPUs (Intel, AMD), is **little-endian**.

Sub-Question 1: Part (a): Big-Endian at Address A

Is the value of the 16-bit number stored at address A equal to 1541, assuming a big-endian architecture?

Step 1

First, we identify the bytes that constitute the 16-bit integer starting at address A. These are the bytes at addresses A and A+1. According to the problem statement, these bytes have the hexadecimal values `0x06` and `0x05`, respectively.

Next, we apply the big-endian rule. The byte at the lower memory address (A) is the most significant byte (MSB), and the byte at the higher address (A+1) is the least significant byte (LSB).

$$MSB = 0x06 \quad (from\ address\ A)$$
$$LSB = 0x05 \quad (from\ address\ A+1)$$

To find the value of the integer, we logically assemble the bytes in the order of significance (MSB followed by LSB) to form the 16-bit hexadecimal number `0x0605`. We then convert this value to decimal:

$$0x0605 = (6 \times 16^1 + 0) \times 16^1 + 5 \times 16^0 = 6 \times 16^2 + 5 = 6 \times 256 + 5 = 1536 + 5 = 1541$$

Final Answer: YES. The calculated value is 1541, which matches the value given in the question.

Sub-Question 2: Part (b): Little-Endian at Address A+2

Is the value of the 16-bit number stored at address A+2 equal to 772, assuming a little-endian architecture?

Step 1

The 16-bit integer starts at address A+2, so we need the bytes at addresses A+2 and A+3. These bytes are `0x04` and `0x03`.

Now, we apply the little-endian rule. This is the crucial non-trivial step: the byte at the lower memory address (A+2) is the least significant byte (LSB), while the byte at the higher address (A+3) is the most significant byte (MSB). The order in memory is the reverse of the order of significance.

$$LSB = 0x04 \quad (from\ address\ A+2)$$
$$MSB = 0x03 \quad (from\ address\ A+3)$$

To interpret the integer's value, we assemble the bytes according to their significance, not their memory order. We place the MSB first, which gives the hexadecimal number `0x0304`. Converting this to decimal:

$$0x0304 = 3 \times 16^2 + 0 \times 16^1 + 4 \times 16^0 = 3 \times 256 + 4 = 768 + 4 = 772$$

Final Answer: YES. The calculated value of 772 matches the given value.

Sub-Question 3: Part (c): x86 Architecture at Address A+4

Is the value of the 16-bit number stored at address A+4 equal to 258, assuming the x86 architecture?

Step 1

This part requires knowing that the x86 architecture is little-endian. With this information, the problem becomes identical in logic to part (b). The relevant bytes start at address A+4, so we read the bytes at A+4 and A+5. These are `0x02` and `0x01`.

Applying the little-endian rule, the byte at the lower address (A+4) is the LSB, and the byte at the higher address (A+5) is the MSB.

$$\begin{aligned} LSB &= 0x02 \quad (from\ address\ A+4) \\ MSB &= 0x01 \quad (from\ address\ A+5) \end{aligned}$$

We assemble the number by placing the MSB first, yielding `0x0102`. Its decimal equivalent is:

$$0x0102 = 1 \times 16^2 + 0 \times 16^1 + 2 \times 16^0 = 1 \times 256 + 2 = 258$$

Final Answer: YES. The calculated value of 258 is correct.