IAE ACADEMY INSTITUTE FOR ENGINEERS

GATE CS · 2022

IAE Academy free study material · Updated 27 October 2026

GATE CS 2022 Solved Paper

INSIDE

- All 65 questions, in the order of the paper
- The official answer to every question
- Step-by-step worked solutions
- The answer key on one page

Question paper and official answer key: IIT Kharagpur

Online, with the questions linked to their topic notes: **iaeacademy.in/study/papers/cs-2022**

FREE COUNSELLING

Preparing for GATE CS? Talk to IAE Academy.

Or call **+91 84204 36524**

Every day, 10:30 am to 7:30 pm

Chat on WhatsApp

Free. Not for sale. Pass it on.

About this paper

65 questions, 100 marks. Question types: 32 MCQ · 15 MSQ · 18 NAT.

Official answer key: IIT Kharagpur

We placed each question under a subject and topic ourselves; the exam paper does not label them.

This paper online, with the questions linked to their topic notes:
iaeacademy.in/study/papers/cs-2022

Answer key

Q	TYPE	MARKS	ANSWER
Q.1	MCQ	1	**D**
Q.2	MCQ	1	**C**
Q.3	MCQ	1	**D**
Q.4	MCQ	1	**A**
Q.5	MCQ	1	**B**
Q.6	MCQ	2	**B**
Q.7	MCQ	2	**D**
Q.8	MCQ	2	**D**
Q.9	MCQ	2	**B**
Q.10	MCQ	2	**D**
Q.11	MCQ	1	**A**
Q.12	MCQ	1	**D**
Q.13	MCQ	1	**D**
Q.14	MCQ	1	**A**
Q.15	MCQ	1	**A**
Q.16	MCQ	1	**B**
Q.17	MCQ	1	**A**
Q.18	MCQ	1	**B**
Q.19	MCQ	1	**C**
Q.20	MCQ	1	**C**
Q.21	MCQ	1	**D**
Q.22	MCQ	1	**C**
Q.23	MSQ	1	**B, C, D**
Q.24	MSQ	1	**A, B, D**
Q.25	MSQ	1	**A, B**
Q.26	MSQ	1	**A, D**
Q.27	MSQ	1	**A, B, C**
Q.28	NAT	1	**509**
Q.29	NAT	1	**5**
Q.30	NAT	1	**36**
Q.31	NAT	1	**8**
Q.32	NAT	1	**7**
Q.33	NAT	1	**0.85**
Q.34	NAT	1	**-0.5**
Q.35	NAT	1	**4**
Q.36	MCQ	2	**A**
Q.37	MCQ	2	**D**
Q.38	MCQ	2	**C**
Q.39	MCQ	2	**A**
Q.40	MCQ	2	**C**
Q.41	MCQ	2	**B**
Q.42	MCQ	2	**D**
Q.43	MCQ	2	**A**
Q.44	MCQ	2	**A**
Q.45	MCQ	2	**D**
Q.46	MSQ	2	**A, B, C**
Q.47	MSQ	2	**A, B, C**
Q.48	MSQ	2	**B, C, D**
Q.49	MSQ	2	**A, B, C**
Q.50	MSQ	2	**A, B, C**
Q.51	MSQ	2	**A, B, C**
Q.52	MSQ	2	**A**
Q.53	MSQ	2	**A, C, D**
Q.54	MSQ	2	**A, B, D**
Q.55	MSQ	2	**B, D**
Q.56	NAT	2	**2**
Q.57	NAT	2	**0.5**
Q.58	NAT	2	**24**
Q.59	NAT	2	**7.07 to 7.09**
Q.60	NAT	2	**33**
Q.61	NAT	2	**1.42 to 1.45**
Q.62	NAT	2	**0**
Q.63	NAT	2	**153**
Q.64	NAT	2	**0.6**
Q.65	NAT	2	**80**

General Aptitude Q.1 to Q.10

Q.1 MCQ · 1 MARK

General Aptitude › Grammar & vocabulary

The _____ is too high for it to be considered _____.

(A) fair / fare

(B) faer / fair

(C) fare / fare

(D) fare / fair

Official answer: **D**

SOLUTION

1. The first blank follows "The" and is the subject of "is too high", so it must be a noun for something that can be high, such as a price or charge.
2. "fare" (noun) means the price charged for a journey, so "The fare is too high" fits. "fair" used as a noun means an exhibition or fun event, which is not something described as "too high" in this sense.
3. The second blank follows "considered" and needs an adjective meaning reasonable or just: "fair". "fare" is not an adjective.
4. The sentence reads "The fare is too high for it to be considered fair."
5. Option (A) reverses the two words. Option (B) uses "faer", which is not a word. Option (C) puts "fare" in the second blank, where an adjective is needed. Only option (D), fare / fair, works.

KEY IDEA

Fare and fair are homophones. Decide the part of speech each blank needs. The first blank is a noun (fare, the price of travel). The second is an adjective (fair, reasonable).

WATCH OUT FOR

Picking the options by sound alone, since both words are pronounced the same. This leads to (A), which reverses the words, or to (C), which puts the noun "fare" where an adjective is needed.

Q.2 MCQ · 1 MARK

General Aptitude › Numerical

A function y(x) is defined in the interval [0, 1] on the x-axis as

y(x) = 2 if 0 ≤ x < 1/3; 3 if 1/3 ≤ x < 3/4; 1 if 3/4 ≤ x ≤ 1.

Which one of the following is the area under the curve for the interval [0, 1] on the x-axis?

(A) $\frac{5}{6}$

(B) $\frac{6}{5}$

(C) $\frac{13}{6}$

(D) $\frac{6}{13}$

Official answer: **C**

SOLUTION

1. The function is piecewise constant, so the area under the curve is the sum of the areas of three rectangles (height times width).
2. Piece 1: height 2 on $\left[0, \frac{1}{3}\right)$, width $\frac{1}{3}$, area $= \frac{2}{3} = \frac{8}{12}$.
3. Piece 2: height 3 on $\left[\frac{1}{3}, \frac{3}{4}\right)$, width $\frac{3}{4} - \frac{1}{3} = \frac{9-4}{12} = \frac{5}{12}$, area $= \frac{15}{12}$.
4. Piece 3: height 1 on $\left[\frac{3}{4}, 1\right]$, width $1 - \frac{3}{4} = \frac{1}{4}$, area $= \frac{1}{4} = \frac{3}{12}$.
5. Total area:

$$\frac{8}{12}+\frac{15}{12}+\frac{3}{12}=\frac{26}{12}=\frac{13}{6}$$

6. Check: the widths $\frac{1}{3}+\frac{5}{12}+\frac{1}{4}=\frac{4+5+3}{12}=1$, so the pieces cover $[0,1]$ exactly. The area is $\frac{13}{6}$, option (C).

KEY IDEA

For a piecewise-constant function, the area is the sum of height times width over each piece. Use a common denominator, and check that the widths add up to the full interval.

WATCH OUT FOR

Taking the middle width as $\frac{3}{4}-\frac{1}{3}$ and getting it wrong, for example $\frac{1}{2}$ or $\frac{2}{3}$. Another slip is averaging the heights, $\frac{2+3+1}{3}=2$, instead of weighting each by its width.

Q.3 MCQ · 1 MARK

General Aptitude › Numerical

Let r be a root of the equation $x^2+2x+6=0$.

Then the value of the expression $(r+2)(r+3)(r+4)(r+5)$ is

(A) 51

(B) -51

(C) 126

(D) -126

Official answer: **D**

SOLUTION

1. Since r is a root, $r^2+2r+6=0$, so $r^2=-2r-6$.
2. Pair the factors whose constants add to 7: $(r+2)(r+5)=r^2+7r+10$ and $(r+3)(r+4)=r^2+7r+12$.
3. Substitute $r^2=-2r-6$: $(r+2)(r+5)=(-2r-6)+7r+10=5r+4$ and $(r+3)(r+4)=(-2r-6)+7r+12=5r+6$.
4. Multiply: $(5r+4)(5r+6)=25r^2+50r+24$.
5. Substitute again: $25(-2r-6)+50r+24=-50r-150+50r+24$.
6. The r terms cancel, leaving $-150+24=-126$. This is the same for either root, so the value does not depend on which root is chosen.
7. Hence the value is -126, option (D).

KEY IDEA

Pair the factors whose constants add to the same total, here $(r+2)(r+5)$ and $(r+3)(r+4)$. Both pairs then contain r^2+7r, which reduces to a linear expression in r using $r^2=-2r-6$. The r terms cancel in the final product.

WATCH OUT FOR

Multiplying the four factors in the order given gives a degree-4 polynomial that is slow to reduce and easy to slip on. Another slip is solving for the complex roots $-1\pm i\sqrt{5}$ and substituting them directly, which invites arithmetic errors with i. A third is a sign slip when substituting $r^2=-2r-6$.

Q.4 MCQ · 1 MARK

General Aptitude › Reasoning

Given below are four statements.

Statement 1: All students are inquisitive.

Statement 2: Some students are inquisitive.

Statement 3: No student is inquisitive.

Statement 4: Some students are not inquisitive.

From the given four statements, find the two statements that CANNOT BE TRUE simultaneously, assuming that there is at least one student in the class.

(A) Statement 1 and Statement 3

(B) Statement 1 and Statement 2

(C) Statement 2 and Statement 4

(D) Statement 3 and Statement 4

Official answer: **A**

SOLUTION

1. The class has at least one student, so it is not empty.
2. **Statement 1 and Statement 3:** Statement 1 says every student is inquisitive. Statement 3 says no student is inquisitive. Take any one student in the class. Statement 1 says that student is inquisitive and Statement 3 says that student is not, so both cannot hold. They cannot be true together.
3. **Statement 1 and Statement 2:** If all students are inquisitive, then at least one student is inquisitive, because the class is non-empty. Statement 1 implies Statement 2, so both can be true.
4. **Statement 2 and Statement 4:** A class with one inquisitive student and one who is not makes both true.
5. **Statement 3 and Statement 4:** If no student is inquisitive, then any student is not inquisitive, so some student is not inquisitive. Statement 3 implies Statement 4, so both can be true.
6. Only Statement 1 and Statement 3 cannot both be true, so the answer is (A).

KEY IDEA

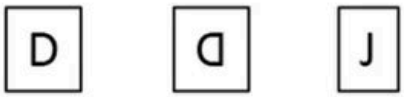
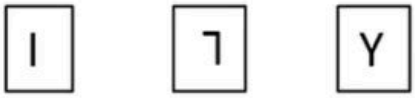
"All" and "No" about the same non-empty group are contradictory. Each of the other three pairs can be true together. Without a student in the class, "All" and "No" would both be true trivially, which is why the question states that there is at least one student.

WATCH OUT FOR

Treating "Some are" and "Some are not" as contradictory. Both can be true of the same group. Also forgetting that the non-empty assumption is what makes "All" and "No" clash.

Q.5 MCQ · 1 MARK

General Aptitude › Spatial

Q.5	A palindrome is a word that reads the same forwards and backwards. In a game of words, a player has the following two plates painted with letters.  From the additional plates given in the options, which one of the combinations of additional plates would allow the player to construct a five-letter palindrome. The player should use all the five plates exactly once. The plates can be rotated in their plane.
(A)	
(B)	
(C)	
(D)	

Official answer: **B**

Q.6 MCQ · 2 MARKS

General Aptitude › Reading

Some people believe that “what gets measured, improves”. Some others believe that “what gets measured, gets gamed”. One possible reason for the difference in the beliefs is the work culture in organizations. In organizations with good work culture, metrics help improve outcomes. However, the same metrics are counterproductive in organizations with poor work culture.

Which one of the following is the CORRECT logical inference based on the information in the above passage?

(A) Metrics are useful in organizations with poor work culture

(B) Metrics are useful in organizations with good work culture

(C) Metrics are always counterproductive in organizations with good work culture

(D) Metrics are never useful in organizations with good work culture

Official answer: **B**

SOLUTION

1. The passage says that in organizations with good work culture, metrics help improve outcomes. In organizations with poor work culture, the same metrics are counterproductive.
2. A correct inference must follow from these two statements alone, with no extra assumption.
3. (A) says metrics are useful in organizations with poor work culture. The passage says they are counterproductive there, so (A) contradicts it.
4. (B) says metrics are useful in organizations with good work culture. The passage says metrics help improve outcomes there, so (B) follows directly.
5. (C) says metrics are always counterproductive in organizations with good work culture. The passage says they help improve outcomes there, so (C) contradicts it.
6. (D) says metrics are never useful in organizations with good work culture. This also contradicts the statement that they help improve outcomes there.
7. Only (B) follows from the passage.

KEY IDEA
In an inference question, choose the option that restates what the passage says. Reject any option that contradicts a stated fact or adds absolutes such as "always" or "never" that the passage does not support.

WATCH OUT FOR
Choosing (A) by mixing up the two cultures, or choosing (C) or (D) because the passage mentions that metrics can be gamed or counterproductive. That applies only to poor work culture.

Q.7 MCQ · 2 MARKS

General Aptitude › Numerical

In a recently conducted national entrance test, boys constituted 65% of those who appeared for the test. Girls constituted the remaining candidates and they accounted for 60% of the qualified candidates.

Which one of the following is the correct logical inference based on the information provided in the above passage?

(A) Equal number of boys and girls qualified

(B) Equal number of boys and girls appeared for the test

(C) The number of boys who appeared for the test is less than the number of girls who appeared

(D) The number of boys who qualified the test is less than the number of girls who qualified

Official answer: **D**

SOLUTION

1. Let N be the number of candidates who appeared. Boys appeared: $0.65N$. Girls are the remaining candidates, so girls appeared: $0.35N$.
2. Since $0.65N > 0.35N$, more boys than girls appeared. This rules out (B) (equal numbers appeared) and (C) (boys fewer than girls).
3. Let Q be the number of candidates who qualified. Girls are 60% of the qualified candidates, so girls qualified $= 0.6Q$.
4. Boys are the remaining 40% of the qualified candidates, so boys qualified $= 0.4Q$.
5. Since $0.4Q < 0.6Q$ for any $Q > 0$, fewer boys than girls qualified. This rules out (A) and confirms (D).

6. The 65% share of boys among those who appeared does not change this split, because the 60% girls figure is given directly for the qualified candidates.

KEY IDEA
Keep two groups apart: those who appeared (65% boys) and those who qualified (60% girls). Each percentage applies only to its own group. The qualified split alone settles the comparison.

WATCH OUT FOR
Carrying the 65% boys share from the appeared group over to the qualified group, and wrongly concluding that more boys qualified. The two percentages describe different groups.

Q.8 MCQ · 2 MARKS

General Aptitude › Statistics & probability

A box contains five balls of same size and shape. Three of them are green coloured balls and two of them are orange coloured balls. Balls are drawn from the box one at a time. If a green ball is drawn, it is not replaced. If an orange ball is drawn, it is replaced with another orange ball.

First ball is drawn. What is the probability of getting an orange ball in the next draw?

(A) $\frac{1}{2}$
(B) $\frac{8}{25}$
(C) $\frac{19}{50}$
(D) $\frac{23}{50}$

Official answer: **D**

SOLUTION

1. The colour of the first ball is not given, so split on it (total probability): $P(O_2) = P(G_1)\,P(O_2 \mid G_1) + P(O_1)\,P(O_2 \mid O_1)$.
2. First green: $P(G_1) = \frac{3}{5}$. A green ball is not replaced, so the box now holds 2 green and 2 orange balls (4 balls). Hence $P(O_2 \mid G_1) = \frac{2}{4} = \frac{1}{2}$.
3. First orange: $P(O_1) = \frac{2}{5}$. An orange ball is replaced by another orange ball, so the box is unchanged at 3 green and 2 orange balls. Hence $P(O_2 \mid O_1) = \frac{2}{5}$.
4. Combine: $P(O_2) = \frac{3}{5} \cdot \frac{1}{2} + \frac{2}{5} \cdot \frac{2}{5} = \frac{3}{10} + \frac{4}{25}$.
5. Over a common denominator of 50: $\frac{15}{50} + \frac{8}{50} = \frac{23}{50}$, which is option (D).

KEY IDEA
The first ball's colour is unknown, so condition on it. Use the law of total probability, and update the box after each case: green is removed, orange is swapped for another orange so the box does not change.

WATCH OUT FOR
Treating both cases as "without replacement" and using $\frac{2}{4}$ throughout. Another slip is forgetting that the replaced orange ball leaves the box at 3 green and 2 orange. A further slip is answering $\frac{2}{5}$, which ignores the first draw's effect.

Q.9 MCQ · 2 MARKS

General Aptitude › Geometry

The corners and mid-points of the sides of a triangle are named using the distinct letters P, Q, R, S, T and U, but not necessarily in the same order. Consider the following statements:

- The line joining P and R is parallel to the line joining Q and S.
- P is placed on the side opposite to the corner T.
- S and U cannot be placed on the same side.

Which one of the following statements is correct based on the above information?

(A) P cannot be placed at a corner
(B) S cannot be placed at a corner

(C) U cannot be placed at a mid-point

(D) R cannot be placed at a corner

Official answer: **B**

SOLUTION

1. Name the triangle ABC with mid-points a (of BC), b (of CA), c (of AB). T is a corner, and by symmetry let $T = A$. Then P lies on the side opposite A, so $P \in \{B, a, C\}$.
2. Which lines through two of the six points are parallel? Take $A = (0,0)$, $B = (4,0)$, $C = (0,4)$, so $a = (2,2)$, $b = (0,2)$, $c = (2,0)$. The only parallel pairs are a mid-line and the side it is parallel to: $bc \parallel BC$ (points B, a, C), $ac \parallel AC$ (points A, b, C), $ab \parallel AB$ (points A, c, B). The medians are not parallel to any other joining line. So one of the lines PR, QS is a mid-line (two mid-points) and the other lies along the parallel side (two of the three points on it).
3. Case 1: PR is the mid-line. Then P and R are mid-points, and P is on BC, so $P = a$. If $R = b$, the mid-line $ab \parallel AB$, so Q, S are the two points of AB other than A, namely $\{c, B\}$, and $U = C$. B and C share the side BC, so S cannot be B. Hence $S = c$, a mid-point. If $R = c$, the mid-line $ac \parallel AC$, so $\{Q, S\} = \{b, C\}$ and $U = B$. Again C shares BC with B, so $S = b$, a mid-point.
4. Case 2: QS is the mid-line, so S is a mid-point. Examples that satisfy all three statements: $S = b$, $Q = c$, $P = C$, $R = B$, $T = A$, $U = a$. Here $PR = CB \parallel bc = QS$, P is on BC opposite A, and b (on AC) and a (on BC) share no side. The mid-lines ac and ab do not work in this case. For ac, $\{S, Q\} = \{a, c\}$ and $U = B$, but a lies on BC and c lies on AB, both sides through B. For ab, $\{S, Q\} = \{a, b\}$ and $U = C$, but a lies on BC and b lies on AC, both sides through C.
5. In every valid arrangement S is a mid-point, so S cannot be at a corner. Option (B) is correct.
6. The other options fail, using the arrangement $S = b, Q = c, P = C, R = B, T = A, U = a$ of step 4: P is the corner C, so (A) is false. R is the corner B, so (D) is false. U is the mid-point a, so (C) is false.

KEY IDEA

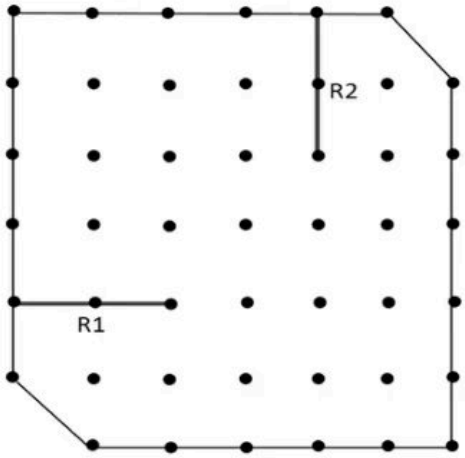
In a triangle with its three mid-points, the only parallel lines are a mid-line and the side it is parallel to. So PR and QS must be a mid-line and its parallel side, and the mid-line's end points are both mid-points.

WATCH OUT FOR

Assuming a parallel pair can come from any two lines, for example a median and a side. Or checking only one arrangement and calling an option false or true from it, without going through every placement that satisfies the three statements.

Q.10 MCQ · 2 MARKS

General Aptitude › Spatial

Q.10	A plot of land must be divided between four families. They want their individual plots to be similar in shape, not necessarily equal in area. The land has equally spaced poles, marked as dots in the below figure. Two ropes, R1 and R2, are already present and cannot be moved. What is the least number of **additional** straight ropes needed to create the desired plots? A single rope can pass through three poles that are aligned in a straight line. 
(A)	2
(B)	4
(C)	5
(D)	3

Official answer: **D**

Computer Science and Information Technology Q.11 to Q.65

Q.11 MCQ · 1 MARK

Data Structures and Algorithms › Complexity

Which one of the following statements is TRUE for all positive functions $f(n)$?

(A) $f(n^2) = \Theta(f(n)^2)$, when $f(n)$ is a polynomial

(B) $f(n^2) = o(f(n)^2)$

(C) $f(n^2) = O(f(n)^2)$, when $f(n)$ is an exponential function

(D) $f(n^2) = \Omega(f(n)^2)$

Official answer: **A**

SOLUTION

1. Restrict attention to the class named in each option, then either prove the Θ/O/o/Ω relation holds for every function in that class, or find one positive function in that class for which it fails.
2. Option (A): let f(n) be any polynomial of degree k with positive leading coefficient c (forced positive since f(n)>0 for large n). Then f(n) = c·n^k + lower-order terms, so f(n) = Θ(n^k): the lower-degree terms are dominated by the leading term for large n, regardless of how many terms f has.
3. Substitute n → n²: f(n²) = Θ((n²)^k) = Θ(n^{2k}).
4. Square the relation f(n) = Θ(n^k): if $c_1 n^k \le f(n) \le c_2 n^k$ eventually, then $c_1^2 n^{2k} \le f(n)^2 \le c_2^2 n^{2k}$ eventually, so f(n)² = Θ(n^{2k}) as well.
5. Since f(n²) and f(n)² are both Θ(n^{2k}), and Θ is a transitive equivalence relation, f(n²) = Θ(f(n)²) holds for every polynomial f(n): option (A) is TRUE in general.
6. Option (B): counterexample f(n) = n. f(n²) = n², f(n)² = n². The two are equal (ratio → 1), so f(n²) is not o(f(n)²); (B) is false.
7. Option (C): counterexample f(n) = 2^n (exponential). f(n²) = 2^{n²}, f(n)² = 2^{2n}. Since n² − 2n → ∞, the ratio 2^{n²}/2^{2n} → ∞, so f(n²) is NOT O(f(n)²); (C) is false.
8. Option (D): counterexample f(n) = log n. f(n²) = 2 log n, f(n)² = (log n)². The ratio f(n²)/f(n)² = 2/log n → 0, so f(n²) = o(f(n)²), meaning it CANNOT also be Ω(f(n)²); (D) is false.
9. Only (A) survives as universally true for its stated class of functions.

KEY IDEA

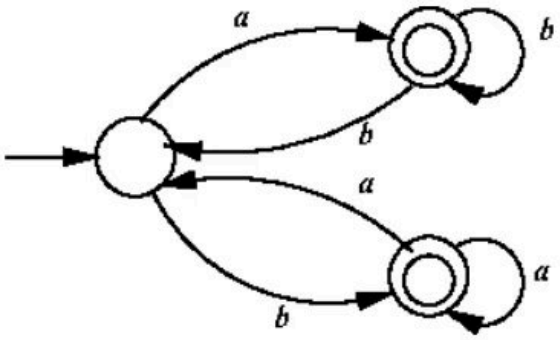
A polynomial of degree k satisfies $f(n) = \Theta(n^k)$ regardless of how many terms it has, because its growth rate is governed entirely by the leading (highest-degree) term; using this, $f(n^2)$ and $f(n)^2$ are both $\Theta(n^{2k})$, hence equal in Θ-order. Exponential and logarithmic functions break the other three options because squaring the argument ($n \to n^2$) affects an exponent multiplicatively/exponentially differently than squaring the function's value does.

WATCH OUT FOR

Proving option (A) only for a monomial like $f(n) = n^k$ and treating that as proof for "a polynomial" in general, without checking that a general polynomial (sum of several degree terms, arbitrary positive leading coefficient) still satisfies the same Θ-equivalence; the argument needs the general dominant-term (Θ-order) property of polynomials, not just the single-term case.

Q.12 MCQ · 1 MARK

Theory of Computation › Finite automata

Q.12	Which one of the following regular expressions correctly represents the language of the finite automaton given below?
	
(A)	$ab^*bab^* + ba^*aba^*$
(B)	$(ab^*b)^*ab^* + (ba^*a)^*ba^*$
(C)	$(ab^*b + ba^*a)^*(a^* + b^*)$
(D)	$(ba^*a + ab^*b)^*(ab^* + ba^*)$

Official answer: **D**

SOLUTION

1. Read the automaton from the figure. q_0 is the start state and is not accepting; q_1 and q_2 are accepting. Transitions: $q_0 \xrightarrow{a} q_1$, $q_1 \xrightarrow{b} q_1$ (loop), $q_1 \xrightarrow{b} q_0$ (return edge), $q_0 \xrightarrow{b} q_2$, $q_2 \xrightarrow{a} q_2$ (loop), $q_2 \xrightarrow{a} q_0$ (return edge). Since q_1 has two b-edges and q_2 has two a-edges, the machine is non-deterministic.
2. Since q_0 is not accepting, ε is rejected. Every accepted string ends in q_1 or q_2. Split an accepted string at the last time the walk is in q_0. Before that point the walk starts and ends at q_0. After it, the walk never returns to q_0 and ends in q_1 or q_2.
3. Closed walks from q_0 back to q_0 are of two kinds. Through q_1: read a, then any number of loop symbols b, then the return b, giving ab^*b. Through q_2: read b, then any number of loop symbols a, then the return a, giving ba^*a. Any closed walk is a sequence of these, so the prefix is $(ab^*b + ba^*a)^*$. This is the same as $(ba^*a + ab^*b)^*$, because the order inside a union does not matter.
4. The final part never returns to q_0. It is either $q_0 \xrightarrow{a} q_1$ followed by loops, giving ab^*, or $q_0 \xrightarrow{b} q_2$ followed by loops, giving ba^*. So the final part is $ab^* + ba^*$.
5. Therefore $L = (ba^*a + ab^*b)^*(ab^* + ba^*)$, which is option (D).
6. Check the other options on short strings. (A) needs at least aba, so it rejects a, which the automaton accepts ($q_0 \xrightarrow{a} q_1$). (C) accepts ε, because every factor can be empty, but q_0 is not accepting. (B) is a union of two expressions, and each one uses only one kind of closed walk. It rejects $abbaab = (ab)(ba)(ab)$, which the automaton accepts: $q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_0 \xrightarrow{b} q_2 \xrightarrow{a} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_1$. (D) accepts $abbaab$ through its prefix and final part. So only (D) is correct.

KEY IDEA
Split an accepted string at the last visit to the start state q_0. The part before it is a sequence of closed loops through q_1 (ab^*b) or q_2 (ba^*a). The part after it is a one-way trip to an accepting state (ab^* or ba^*).

WATCH OUT FOR
Treating the machine as deterministic and missing that q_1 and q_2 each have a loop and a return edge on the same symbol. Another mistake is choosing (B) or (C): (B) cannot mix visits to q_1 and q_2 before ending, and (C) accepts ε even though q_0 is not accepting.

Q.13 MCQ · 1 MARK

Compiler Design › Parsing

Which one of the following statements is TRUE?

(A) The $LALR(1)$ parser for a grammar G cannot have reduce-reduce conflict if the $LR(1)$ parser for G does not have reduce-reduce conflict.

(B) Symbol table is accessed only during the lexical analysis phase.

(C) Data flow analysis is necessary for run-time memory management.

(D) $LR(1)$ parsing is sufficient for deterministic context-free languages.

Official answer: **D**

SOLUTION

1. Check (A). LALR(1) merges LR(1) states that have the same core, and the merged state takes the union of their lookahead sets. Two reduce items that had disjoint lookaheads in separate LR(1) states can have overlapping lookaheads after the merge.
2. So a grammar can have no reduce-reduce conflict in LR(1) and still have one in LALR(1). (A) is false. The converse does hold: merging only adds lookaheads, so an LR(1) reduce-reduce conflict remains in LALR(1).
3. Check (B). The symbol table is used in lexical analysis, and also in syntax analysis, semantic analysis (type checking, scope), code generation and optimization. The word 'only' makes (B) false.
4. Check (C). Run-time memory management (stack frames, heap allocation, garbage collection) does not need data flow analysis. Data flow analysis is a compile-time technique used for optimization, so (C) is false.
5. Check (D). Every deterministic context-free language has an $LR(1)$ grammar, so an $LR(1)$ parser exists for every deterministic CFL. $LR(1)$ parsing is therefore sufficient for deterministic context-free languages, and (D) is true.
6. Answer: D.

KEY IDEA
The languages that LR(1) grammars generate are exactly the deterministic context-free languages. LALR(1) merges same-core LR(1) states and unions their lookaheads, so it can gain conflicts that LR(1) does not have.

WATCH OUT FOR
Choosing (A) because LALR(1) is built from LR(1). Merging states can create reduce-reduce conflicts that LR(1) does not have, though it never creates shift-reduce conflicts that LR(1) lacks.

Q.14 MCQ · 1 MARK

Database Management and Warehousing › Normal forms

In a relational data model, which one of the following statements is TRUE?

(A) A relation with only two attributes is always in BCNF.

(B) If all attributes of a relation are prime attributes, then the relation is in BCNF.

(C) Every relation has at least one non-prime attribute.

(D) BCNF decompositions preserve functional dependencies.

Official answer: **A**

SOLUTION

1. A relation is in BCNF when, for every non-trivial functional dependency $X \to Y$ that holds, X is a superkey.
2. Option A: let the relation be $R(A, B)$. A non-trivial dependency has exactly one attribute on its left side, so the possible sets of non-trivial dependencies are: none; $A \to B$ only; $B \to A$ only; or both.
3. If $A \to B$ holds, then $A^+ = \{A, B\}$, so A is a superkey. The same argument works for $B \to A$. With no non-trivial dependency, BCNF holds vacuously. So every two-attribute relation is in BCNF, and A is TRUE.
4. Option B: take $R(A, B, C)$ with $AB \to C$ and $C \to A$. Then $AB^+ = \{A, B, C\}$ and $BC^+ = \{A, B, C\}$, while A, B and C alone are not keys, so the candidate keys are AB and BC. Every attribute is prime. But in $C \to A$ the left side C has $C^+ = \{A, C\}$, which is not a superkey, so the relation violates BCNF. B is FALSE. (All attributes prime guarantees 3NF only.)
5. Option C: take $R(A, B)$ with $A \to B$ and $B \to A$. Both A and B are candidate keys, so every attribute is prime and the relation has no non-prime attribute. C is FALSE.
6. Option D: take $R(A, B, C)$ with $AB \to C$ and $C \to B$. The dependency $C \to B$ violates BCNF because $C^+ = \{B, C\}$ is not a superkey. Decomposing on it gives $R_1(C, B)$ and $R_2(C, A)$. The dependencies that hold on these parts are only $C \to B$ (and trivial ones), and $AB \to C$ cannot be derived from them, so it is lost. BCNF decomposition is always lossless but need not preserve dependencies. D is FALSE.
7. Hence the only true statement is A.

KEY IDEA

With two attributes, any non-trivial dependency has a single attribute on its left, and that attribute determines the other one, so it is a superkey. BCNF therefore always holds.

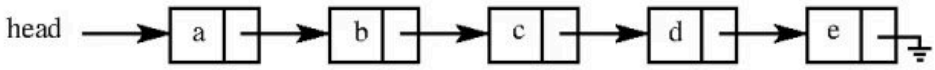
WATCH OUT FOR

Believing that "all attributes prime" implies BCNF. It only implies 3NF. Another slip is assuming BCNF decomposition preserves dependencies, which holds for 3NF synthesis but not for BCNF.

Q.15 MCQ · 1 MARK

Data Structures and Algorithms › Linear structures

Q.15 Consider the problem of reversing a singly linked list. To take an example, given the linked list below,



the reversed linked list should look like



Which one of the following statements is TRUE about the time complexity of algorithms that solve the above problem in $O(1)$ space?

(A) The best algorithm for the problem takes $\theta(n)$ time in the worst case.

(B) The best algorithm for the problem takes $\theta(n \log n)$ time in the worst case.

(C) The best algorithm for the problem takes $\theta(n^2)$ time in the worst case.

(D) It is not possible to reverse a singly linked list in $O(1)$ space.

Official answer: **A**

SOLUTION

1. Restate the problem: reverse a singly linked list a→b→c→d→e into e→d→c→b→a using only O(1) extra space (no auxiliary array/list, no recursion stack).
2. Standard iterative in-place algorithm: prev = NULL, curr = head; while curr != NULL: { next = curr→link; curr→link = prev; prev = curr; curr = next; }. This uses exactly three pointer variables, independent of list size, so extra space is O(1).
3. Trace the loop: at each iteration exactly one node's link field is redirected from its old successor to the previous node, and the three pointers each advance by one position; each iteration does a constant number of operations.
4. The loop executes exactly n times, once per node, since curr steps through the whole list from head to NULL, so the total running time is Θ(n) (both O(n) and Ω(n)).
5. Lower bound argument: to correctly reverse the list, every node's link field must be inspected and rewritten at least once (any node left untouched would still point the old way), so no algorithm, regardless of space usage, can run in better than Θ(n) time. This confirms Θ(n) is the best (tight) achievable bound, not just an upper bound.
6. Check against the options: (A) Θ(n): matches exactly. (B) Θ(n log n) and (C) Θ(n²) both claim slower tight bounds than what is achievable, so both are false. (D) is false since the O(1)-space, Θ(n)-time algorithm above exists.
7. Conclusion: option (A) is TRUE.

KEY IDEA

A singly linked list can be reversed in-place using three pointers (prev, curr, next), touching each node exactly once. This gives $\Theta(n)$ time using only $O(1)$ extra space, and no algorithm can do better than $\Theta(n)$ since every node's link must be inspected and changed at least once.

WATCH OUT FOR

Assuming an $O(1)$-space constraint forces a slower algorithm (choosing B, C, or D) because students associate space-restricted algorithms with recursion (which needs $O(n)$ stack space) rather than the standard iterative three-pointer technique, which needs only constant extra space and still runs in linear time.

Q.16 MCQ · 1 MARK

Data Structures and Algorithms › Sorting & hashing

Suppose we are given n keys, m hash table slots, and two simple uniform hash functions h_1 and h_2. Further suppose our hashing scheme uses h_1 for the odd keys and h_2 for the even keys. What is the expected number of keys in a slot?

(A) $\frac{m}{n}$

(B) $\frac{n}{m}$

(C) $\frac{2n}{m}$

(D) $\frac{n}{2m}$

Official answer: **B**

SOLUTION

1. Let the n keys be split into two disjoint groups by parity: n/2 keys hashed via h1 (a simple uniform hash function into m slots) and n/2 keys hashed via h2 (also simple uniform into the same m slots).
2. By definition of a simple uniform hash function, each key it processes lands in any fixed slot with probability 1/m, independently of the other keys in its own group.
3. Fix any one slot. For the h1-group: the expected number of its keys landing in this slot is $\frac{n}{2} \times \frac{1}{m} = \frac{n}{2m}$, since each of the n/2 keys contributes probability 1/m via the indicator-expectation argument (linearity of expectation over indicator variables).
4. Similarly, for the h2-group: the expected number of its keys landing in this same slot is also $\frac{n}{2} \times \frac{1}{m} = \frac{n}{2m}$.
5. By linearity of expectation, the total expected number of keys (from both groups combined) landing in the fixed slot is the sum of the two group expectations: $\frac{n}{2m} + \frac{n}{2m} = \frac{n}{m}$.
6. This is exactly the same result as using a single simple uniform hash function on all n keys directly ($n \times \frac{1}{m} = \frac{n}{m}$). Splitting the keys across two independent uniform hash functions by any fixed partition does not change the expected per-slot load, because linearity of expectation only requires that each individual key be uniformly distributed across slots, not that all keys share one hash function.
7. Hence the expected number of keys in a slot is $\frac{n}{m}$, which is option (B).

KEY IDEA
Linearity of expectation applies regardless of which hash function each key uses: as long as every key independently lands in a uniformly random slot (among the m slots), the expected number of keys per slot is (total keys)/(total slots) = n/m, whether one hash function or several partition the keys.

WATCH OUT FOR
Assuming that splitting the keys across two independent hash functions by parity somehow doubles or halves the expected load (picking (C) $2n/m$ or (D) $n/2m$), instead of applying linearity of expectation separately to each half and adding.

Q.17 MCQ · 1 MARK

Computer Organization and Architecture › I/O, interrupts & DMA

Which one of the following facilitates transfer of bulk data from hard disk to main memory with the highest throughput?

(A) DMA based I/O transfer

(B) Interrupt driven I/O transfer

(C) Polling based I/O transfer

(D) Programmed I/O transfer

Official answer: **A**

SOLUTION

1. Programmed I/O: the CPU actively polls the device status register and copies every single data word itself. The CPU is fully occupied for the entire transfer, so it is the bottleneck and throughput is low for bulk data.
2. Polling-based I/O: functionally the same overhead pattern as programmed I/O here. The CPU repeatedly checks device readiness and moves data word by word, wasting CPU cycles and limiting throughput.
3. Interrupt-driven I/O: the CPU is freed from busy-waiting between transfers, but it is still interrupted for each unit of data (or small chunk) and must run an interrupt-service routine to move it. This per-unit CPU intervention still caps throughput for large, bulk transfers.
4. DMA-based I/O: a DMA controller is set up once with the source address, destination address and word count, then transfers the entire block directly between disk and memory, only stealing individual bus cycles from the CPU. There is no per-word CPU intervention, so data moves essentially at bus/device speed.
5. Since bulk transfer throughput is limited precisely by how much CPU intervention is needed per word, and DMA removes that intervention almost entirely, DMA gives the highest throughput among the four options.
6. Hence the correct option is (A) DMA based I/O transfer.

KEY IDEA
Throughput for bulk transfers is limited by how much CPU intervention is required per word/chunk moved; ranking from most to least CPU intervention: programmed/polling I/O (every word) > interrupt-driven I/O (every unit/chunk) > DMA (none; only bus cycle-stealing), so DMA gives the highest throughput.

WATCH OUT FOR
Confusing "interrupt-driven" with "no CPU involvement" and picking (B). Interrupt-driven I/O still requires the CPU to run an ISR for each data unit/chunk, so it does not scale to bulk transfers the way DMA does, which offloads the entire block move to a dedicated controller.

Q.18 MCQ · 1 MARK

Computer Organization and Architecture › ALU & control unit

Let R1 and R2 be two 4-bit registers that store numbers in 2's complement form. For the operation R1+R2, which one of the following values of R1 and R2 gives an arithmetic overflow?

(A) R1 = 1011 and R2 = 1110

(B) R1 = 1100 and R2 = 1010

(C) R1 = 0011 and R2 = 0100

(D) R1 = 1001 and R2 = 1111

Official answer: **B**

SOLUTION

1. Recall the overflow rule for 2's complement addition on n bits: overflow happens iff both operands have the same sign and the result's sign differs from that common sign. A 4-bit register represents integers in $[-8, 7]$.
2. Option A: $R1 = 1011 = -5$, $R2 = 1110 = -2$ (both negative). Binary sum $1011 + 1110 = 1\,1001$; discarding the carry-out, the 4-bit result is $1001 = -7$, matching the true sum $-5 + (-2) = -7$ and staying negative like the operands, so no overflow.
3. Option B: $R1 = 1100 = -4$, $R2 = 1010 = -6$ (both negative). Binary sum $1100 + 1010 = 1\,0110$; the 4-bit result is $0110 = +6$, a positive value, while the true sum is $-4 + (-6) = -10$, which is outside $[-8, 7]$. Same-sign operands giving an opposite-sign result means arithmetic overflow.
4. Option C: $R1 = 0011 = 3$, $R2 = 0100 = 4$ (both positive). Sum $0111 = 7$, matching $3 + 4 = 7$, within range and same sign, so no overflow.
5. Option D: $R1 = 1001 = -7$, $R2 = 1111 = -1$ (both negative). Binary sum $1001 + 1111 = 1\,1000$; the 4-bit result is $1000 = -8$, matching the true sum $-7 + (-1) = -8$, the boundary value, still representable and negative, so no overflow.
6. Only option B produces a result whose sign contradicts the common sign of its same-signed operands, so B is the unique case of arithmetic overflow.

KEY IDEA

In 2's complement addition on an n-bit register, arithmetic overflow occurs only when both operands have the same sign but the result has the opposite sign. Adding operands of opposite sign can never overflow, since the true sum then always lies strictly between the two operands and hence within range.

WATCH OUT FOR

Confusing arithmetic overflow with a simple carry-out of the MSB. A carry-out (as in option D, 1001+1111) does not by itself indicate overflow; the correct test is whether the sign of the result contradicts the common sign of two like-signed operands.

Q.19 MCQ · 1 MARK

Operating Systems › Synchronisation

Q.19 Consider the following threads, T_1, T_2, and T_3 executing on a single processor, synchronized using three binary semaphore variables, S_1, S_2, and S_3, operated upon using standard `wait()` and `signal()`. The threads can be context switched in any order and at any time.

T_1	T_2	T_3

`while(true){`
`  wait(S3);`
`  print("C");`
`  signal(S2); }`
	`while(true){`
	`  wait(S1);`
	`  print("B");`
	`  signal(S3); }`
		`while(true){`
		`  wait(S2);`
		`  print("A");`
		`  signal(S1); }`

Which initialization of the semaphores would print the sequence BCABCABCA....?

(A) $S_1 = 1; S_2 = 1; S_3 = 1$

(B) $S_1 = 1; S_2 = 1; S_3 = 0$

(C) $S_1 = 1; S_2 = 0; S_3 = 0$

(D) $S_1 = 0; S_2 = 1; S_3 = 1$

Official answer: **C**

SOLUTION

1. Each thread waits on one semaphore and signals another. T_2 (prints B) signals S_3, which enables T_1 (prints C). T_1 signals S_2, which enables T_3 (prints A). T_3 signals S_1, which enables T_2 again. So the semaphores pass a single token round $T_2 \to T_1 \to T_3 \to T_2$, which gives the output BCABCA....
2. The output must begin with B, so at the start only T_2 may be able to pass its wait , and neither T_1 nor T_3 may be able to. That requires $S_1 = 1$, $S_2 = 0$ and $S_3 = 0$.
3. Trace from $(S_1, S_2, S_3) = (1, 0, 0)$. Only T_2 can pass wait(S_1) , because $S_2 = 0$ and $S_3 = 0$ block the others. T_2 prints B and signals S_3, giving $(0, 0, 1)$. Now only T_1 can proceed.
4. T_1 takes S_3 to 0, prints C and signals S_2, giving $(0, 1, 0)$. Now only T_3 can proceed. It takes S_2 to 0, prints A and signals S_1, giving $(1, 0, 0)$. The state is back at the start, so BCA repeats for ever, whatever the context-switch order.
5. The other options fail. (A) $(1, 1, 1)$ enables all three threads, so A or C can be printed first. (B) $(1, 1, 0)$ enables T_2 and T_3, so T_3 can print A before any B. (D) $(0, 1, 1)$ enables T_1 and T_3 but not T_2, so the first letter is C or A, never B.
6. Answer: (C).

KEY IDEA
Binary semaphores can pass a token around a cycle of threads. To force one fixed order, initialise exactly one semaphore to 1, the one the first thread in the order waits on, and set the rest to 0.

WATCH OUT FOR
Giving the starting semaphores of the other threads a value of 1 as well. Then several threads are enabled at once, and the context-switch order can print A or C before B.

Q.20 MCQ · 1 MARK

Engineering Mathematics › Linear algebra

Consider the following two statements with respect to the matrices $A_{m\times n}$, $B_{n\times m}$, $C_{n\times n}$ and $D_{n\times n}$.

Statement 1: $tr(AB) = tr(BA)$

Statement 2: $tr(CD) = tr(DC)$

where $tr()$ represents the trace of a matrix. Which one of the following holds?

(A) Statement 1 is correct and Statement 2 is wrong.

(B) Statement 1 is wrong and Statement 2 is correct.

(C) Both Statement 1 and Statement 2 are correct.

(D) Both Statement 1 and Statement 2 are wrong.

Official answer: **C**

SOLUTION

1. Sizes: A is $m \times n$ and B is $n \times m$, so AB is $m \times m$ and BA is $n \times n$. Both are square, so both traces are defined.
2. The diagonal entries of the product are $(AB)_{ii} = \sum_{j=1}^{n} a_{ij}b_{ji}$.
3. Take the trace: $tr(AB) = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}b_{ji}$.
4. Similarly $(BA)_{jj} = \sum_{i=1}^{m} b_{ji}a_{ij}$, so $tr(BA) = \sum_{j=1}^{n}\sum_{i=1}^{m} b_{ji}a_{ij}$. Entries are numbers, so $a_{ij}b_{ji} = b_{ji}a_{ij}$. This is the same finite double sum with the order of summation swapped, so $tr(AB) = tr(BA)$ and Statement 1 is correct.
5. Statement 2 is the special case $m = n$ with $A = C$ and $B = D$. Both are $n \times n$, so $tr(CD) = tr(DC)$ and Statement 2 is correct.
6. Both statements are correct, which is option (C).

KEY IDEA
The trace of a product is unchanged by swapping the factors: $tr(AB) = tr(BA)$ whenever both products exist, even if $AB \neq BA$ or the two products have different sizes.

WATCH OUT FOR
Concluding that Statement 2 is wrong because matrix multiplication is not commutative. $CD \neq DC$ in general, but their traces are always equal.

Q.21 MCQ · 1 MARK

Programming in C › Pointers & arrays

What is printed by the following ANSI C program?

```
#include<stdio.h>
int main(int argc, char *argv[])
{
   int x = 1, z[2] = {10, 11};
   int *p = NULL;
   p = &x;
   *p = 10;
   p = &z[1];
   *(&z[0] + 1) += 3;
   printf("%d, %d, %d\n", x, z[0], z[1]);
   return 0;
}
```

(A) 1, 10, 11

(B) 1, 10, 14

(C) 10, 14, 11

(D) 10, 10, 14

Official answer: **D**

SOLUTION

1. Initialization: x = 1; z[0] = 10, z[1] = 11 (array-initializer order).
2. p = &x; then *p = 10; writes through p into x, so x becomes 10. z[0] and z[1] are unaffected.
3. p = &z[1]; only changes what p points to. No dereference occurs, so no memory value changes at this line.
4. *(&z[0] + 1) += 3; &z[0] is int*, so &z[0] + 1 is the address of the next int in the array, which (by contiguity of array storage) is exactly &z[1]. This step is independent of the current value of p. It reads z[1] = 11, adds 3, and stores z[1] = 14.
5. Final values: x = 10, z[0] = 10 (never modified after initialization), z[1] = 14.
6. printf("%d, %d, %d\n", x, z[0], z[1]) prints: 10, 10, 14; option (D).

KEY IDEA

A pointer variable only affects memory when it is dereferenced; assigning a new address to `p` does nothing by itself. The expression `*(&z[0] + 1)` uses pointer arithmetic directly (array-to-pointer decay plus offset) and is completely independent of whatever `p` currently holds. It always denotes `z[1]` here regardless of the intervening `p = &z[1];` line.

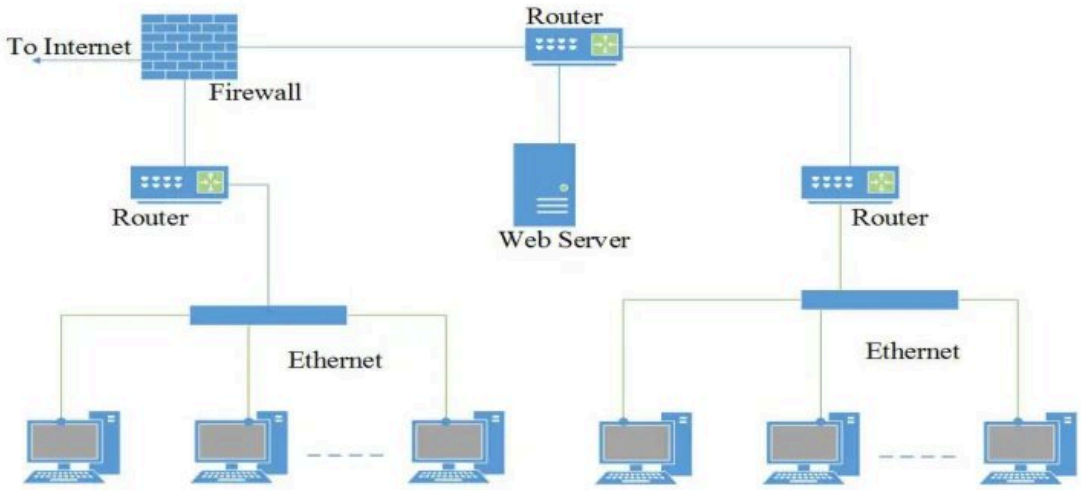
WATCH OUT FOR

Assuming that because `p` was just set to `&z[1]`, the very next statement `*(&z[0] + 1) += 3;` must be "using p" or must somehow interact with it, and then either double-adding through `p` or getting confused about which variable receives the `+= 3`. In fact `p` is a red herring for this line: `&z[0] + 1` is computed independently as the address of `z[1]`, so the update is simply `z[1] = 11 + 3 = 14`.

Q.22 MCQ · 1 MARK

Computer Networks › IPv4

Q.22 Consider an enterprise network with two Ethernet segments, a web server and a firewall, connected via three routers as shown below.



What is the number of subnets inside the enterprise network?

(A) 3

(B) 12

(C) 6

(D) 8

Official answer: **C**

SOLUTION

1. A subnet is a group of interfaces that can reach each other without passing through a router. In the drawing, every link between two devices that sit on opposite sides of a router is its own subnet, and each Ethernet segment is one subnet.
2. Leave out the firewall-to-Internet link. It lies outside the enterprise network.
3. Link from the firewall to the left router: 1 subnet.
4. Link from the firewall to the top router: 1 subnet.
5. Link from the top router to the web server: 1 subnet.
6. Link from the top router to the right router: 1 subnet.
7. Left Ethernet segment, with its hosts and the left router's interface: 1 subnet. Right Ethernet segment, with its hosts and the right router's interface: 1 subnet.
8. Total $= 1 + 1 + 1 + 1 + 1 + 1 = 6$, which is option (C).

KEY IDEA
Count one subnet for each point-to-point link between devices that routers separate, plus one for each Ethernet segment. Do not count the link out to the Internet.

WATCH OUT FOR
Counting the number of routers or devices instead of the networks between them. Another slip is counting the Internet link, which gives 7, or counting only the two Ethernet segments plus the routers, which gives 3 or 5.

Q.23 MSQ · 1 MARK

Theory of Computation › Turing machines

Which of the following statements is/are TRUE?

(A) Every subset of a recursively enumerable language is recursive.

(B) If a language L and its complement $\overline{L}$ are both recursively enumerable, then L must be recursive.

(C) Complement of a context-free language must be recursive.

(D) If L_1 and L_2 are regular, then $L_1 \cap L_2$ must be deterministic context-free.

Official answer: **B, C, D**

SOLUTION

1. (A) Take $L = \Sigma^*$, which is recursive and hence recursively enumerable. Every language over Σ is a subset of Σ^*. There are uncountably many languages over Σ but only countably many Turing machines, so some subset of Σ^* is not even recursively enumerable, let alone recursive. (A) is false.
2. (B) Let M_1 accept L and M_2 accept $\overline{L}$. On input w, run M_1 and M_2 alternately, one step at a time. Every string lies in exactly one of L and $\overline{L}$, so exactly one of the two machines eventually accepts. If M_1 accepts, accept w. If M_2 accepts, reject w. This machine halts on every input and decides L, so L is recursive. (B) is true.
3. (C) Every context-free language is decidable, for example by the CYK algorithm on a grammar in Chomsky normal form, which always halts. Swapping the accept and reject outcomes of this decider gives a decider for the complement. So the complement of a context-free language is recursive, even though it need not be context-free. (C) is true.
4. (D) Regular languages are closed under intersection, so $L_1 \cap L_2$ is regular. A DFA is a deterministic pushdown automaton that never uses its stack, so every regular language is deterministic context-free. (D) is true.
5. Correct options: B, C and D.

KEY IDEA
A language is recursive exactly when both it and its complement are recursively enumerable. Every context-free language is decidable, so its complement is recursive even when it is not context-free. Regular languages sit inside the deterministic context-free languages.

WATCH OUT FOR
Concluding that (C) is false because context-free languages are not closed under complement. They are not closed, but the complement is still decidable, and the option only asks for recursive, not context-free.

Q.24 MSQ · 1 MARK

Computer Organization and Architecture › Memory & cache

Let WB and WT be two set associative cache organizations that use LRU algorithm for cache block replacement. WB is a write back cache and WT is a write through cache. Which of the following statements is/are FALSE?

(A) Each cache block in WB and WT has a dirty bit.

(B) Every write hit in WB leads to a data transfer from cache to main memory.

(C) Eviction of a block from WT will not lead to data transfer from cache to main memory.

(D) A read miss in WB will never lead to eviction of a dirty block from WB.

Official answer: **A, B, D**

SOLUTION

1. A cache needs a dirty bit only to track whether a block's cached copy differs from main memory. In a write-through (WT) cache, every write is immediately propagated to memory, so cache and memory are always identical -- WT never needs a dirty bit. A write-back (WB) cache does need one, since writes are deferred. Statement (A) claims both WB and WT have a dirty bit, which is false because WT does not need one.
2. In a write-back (WB) cache, a write hit updates only the cache line and sets its dirty bit; the modified data is written to main memory later, typically only when that block is evicted. So a write hit does not, by itself, cause a transfer from cache to memory. Statement (B) is therefore false.
3. In a write-through (WT) cache, main memory is kept current after every write. Consequently, when a block is evicted from WT, memory already has the latest data and no write-back transfer is required. Statement (C) is therefore true (not false).
4. On a read miss in WB, the LRU-selected victim block may itself be dirty (modified since being cached); before it can be replaced, it must be written back to main memory. So a read miss can indeed cause eviction of a dirty block, contradicting the claim that it "will never" happen. Statement (D) is therefore false.
5. Collecting the false statements: A, B, and D are false, while C is true.

KEY IDEA

In a write-back cache, only a dirty (modified) block needs an eventual write to memory, and that write happens only on eviction, not on every write hit. A write-through cache never needs a dirty bit because memory is updated immediately on every write, so its blocks are never "ahead of" memory.

WATCH OUT FOR

Assuming every cache, regardless of write policy, needs a dirty bit, or assuming a write hit in a write-back cache must immediately propagate to main memory the way a write-through cache does.

Q.25 MSQ · 1 MARK

Database Management and Warehousing › Relational algebra

Consider the following three relations in a relational database.

$Employee(\underline{eId}, Name)$, $Brand(\underline{bId}, bName)$, $Own(\underline{eId}, \underline{bId})$

Which of the following relational algebra expressions return the set of eIds who own all the brands?

(A) $\Pi_{eId}(\Pi_{eId,bId}(Own) / \Pi_{bId}(Brand))$

(B) $\Pi_{eId}(Own) - \Pi_{eId}\big((\Pi_{eId}(Own) \times \Pi_{bId}(Brand)) - \Pi_{eId,bId}(Own)\big)$

(C) $\Pi_{eId}(\Pi_{eId,bId}(Own) / \Pi_{bId}(Own))$

(D) $\Pi_{eId}\big((\Pi_{eId}(Own) \times \Pi_{bId}(Own)) / \Pi_{bId}(Brand)\big)$

Official answer: **A, B**

SOLUTION

1. "Owns all brands" is a relational division: we want the employees e such that for every brand b in $Brand$, the pair (e, b) is in Own.
2. (A) $\Pi_{eId,bId}(Own) \div \Pi_{bId}(Brand)$ returns exactly the eId values that appear in Own paired with every bId of $Brand$. Projecting on eId keeps the same set. Correct.
3. (B) This is the textbook expansion of division. $\Pi_{eId}(Own) \times \Pi_{bId}(Brand)$ lists every (employee, brand) pair that would have to exist. Subtracting $\Pi_{eId,bId}(Own)$ leaves the missing pairs. Projecting on eId

gives the employees who lack at least one brand. Removing them from $\Pi_{eId}(Own)$ leaves the employees who own all brands. Correct.

4. (C) The divisor is $\Pi_{bId}(Own)$, the brands owned by someone, not all brands. If some brand in $Brand$ is owned by nobody, an employee who owns every owned brand is returned although he does not own all brands. Not correct in general.
5. (D) The dividend $\Pi_{eId}(Own) \times \Pi_{bId}(Own)$ pairs every owner with every owned brand, whatever that owner actually owns. Dividing by $Brand$ then only tests whether every brand is owned by somebody. If so, every owner is returned, including owners of a single brand. Not correct.
6. So the expressions that always return the employees owning all brands are (A) and (B).

KEY IDEA
Owning all brands is relational division by $Brand$. Division can be written directly (A) or expanded as "all owners minus owners with a missing (employee, brand) pair" (B).

WATCH OUT FOR
Dividing by $\Pi_{bId}(Own)$ (C) or building the dividend from owned brands only (D). Both compare against the brands that are owned, not against all brands in $Brand$.

Q.26 MSQ · 1 MARK

Operating Systems › Deadlock

Which of the following statements is/are TRUE with respect to deadlocks?

(A) Circular wait is a necessary condition for the formation of deadlock.

(B) In a system where each resource has more than one instance, a cycle in its wait-for graph indicates the presence of a deadlock.

(C) If the current allocation of resources to processes leads the system to unsafe state, then deadlock will necessarily occur.

(D) In the resource-allocation graph of a system, if every edge is an assignment edge, then the system is not in deadlock state.

Official answer: **A, D**

SOLUTION

1. (A) Deadlock needs all four Coffman conditions at once: mutual exclusion, hold and wait, no preemption and circular wait. So circular wait is a necessary condition. TRUE.
2. (B) When a resource has several instances, a cycle in the wait-for graph is necessary but not sufficient. A process in the cycle may be waiting for a resource that a process outside the cycle holds and will release. A cycle means deadlock only when every resource has a single instance. FALSE.
3. (C) An unsafe state means no safe sequence exists, so deadlock is possible but not certain. Processes may release resources before they make their maximum requests. FALSE.
4. (D) An assignment edge goes from a resource to a process. If every edge is an assignment edge, there are no request edges, so no process is waiting. A deadlock needs at least one blocked process, so there is no deadlock. TRUE.
5. Correct options: A and D.

KEY IDEA
Circular wait is necessary for deadlock. A cycle proves deadlock only with single-instance resources. An unsafe state means deadlock may happen, not that it must.

WATCH OUT FOR
Treating any cycle in a graph as a deadlock when resources have several instances. Treating an unsafe state as a deadlock that has already happened or must happen.

Q.27 MSQ · 1 MARK

Discrete Mathematics › Groups

Which of the following statements is/are TRUE for a group G?

(A) If for all $x, y \in G$, $(xy)^2 = x^2y^2$, then G is commutative.

(B) If for all $x \in G$, $x^2 = 1$, then G is commutative. Here, 1 is the identity element of G.

(C) If the order of G is 2, then G is commutative.

(D) If G is commutative, then a subgroup of G need not be commutative.

Official answer: **A, B, C**

SOLUTION

1. (A) Assume $(xy)^2 = x^2y^2$ for all $x, y \in G$. Then $xyxy = xxyy$. Multiply on the left by x^{-1} and on the right by y^{-1} to get $yx = xy$. So G is commutative. TRUE.
2. (B) If $x^2 = 1$ for all x, every element is its own inverse, so $x^{-1} = x$. Then $xy = (xy)^{-1} = y^{-1}x^{-1} = yx$ for all x, y. So G is commutative. TRUE.
3. (C) A group of order 2 is $\{1, a\}$. Every element is a power of a, so the group is cyclic, and a cyclic group is commutative. TRUE.
4. (D) Let H be a subgroup of a commutative group G. For $h_1, h_2 \in H \subseteq G$ we have $h_1h_2 = h_2h_1$, because the operation is the same as in G. So every subgroup of a commutative group is commutative. The statement says a subgroup need not be commutative, so it is FALSE.
5. Correct options: A, B and C.

KEY IDEA

Commutativity comes from cancelling in $xyxy = xxyy$ (A), from $x = x^{-1}$ giving $xy = (xy)^{-1} = yx$ (B), and from a group of order 2 being cyclic (C). A subgroup inherits commutativity from its parent group, so D fails.

WATCH OUT FOR

Calling D true by confusing it with the converse. A subgroup of a commutative group is always commutative. It is a non-commutative group that can have commutative subgroups.

Q.28 NAT · 1 MARK

Data Structures and Algorithms › Trees & heaps

Suppose a binary search tree with 1000 distinct elements is also a complete binary tree. The tree is stored using the array representation of binary heap trees. Assuming that the array indices start with 0, the 3rd largest element of the tree is stored at index ___.

Official answer: **509**

SOLUTION

1. The array representation of a complete binary tree (root at index 0, children of index i at $2i + 1$ and $2i + 2$) fixes only the tree's SHAPE. The node values still obey the binary-search-tree property, so ranking by value requires walking the shape with the in-order-predecessor rule, not reading heap order.
2. Locate the largest element: starting at the root (index 0), repeatedly move to the right child ($2i + 2$) as long as it exists. This traces $0 \to 2 \to 6 \to 14 \to 30 \to 62 \to 126 \to 254 \to 510$.
3. Check whether node 510 has any children. With $n = 1000$ nodes (valid indices 0 to 999): right child $= 2(510) + 2 = 1022 > 999$ (does not exist); left child $= 2(510) + 1 = 1021 > 999$ (does not exist). So index 510 is a leaf and holds the 1st largest (maximum) element.
4. In-order predecessor rule: if a node has no left subtree, its predecessor is its nearest ancestor for which it is a right descendant, here simply its parent. Parent of index i is $\left\lfloor \frac{i-1}{2} \right\rfloor$, so the parent of 510 is $\left\lfloor \frac{510-1}{2} \right\rfloor = 254$. So index 254 holds the 2nd largest element.

5. For the 3rd largest, find the in-order predecessor of node 254. Its left child is $2(254)+1=509$, and $509 \leq 999$, so this child exists: node 254 DOES have a left subtree.
6. When a node has a left subtree, its predecessor is the maximum (rightmost) element of that subtree, so it is enough to walk as far right as possible starting from node 509.
7. Check node 509's children: right child $= 2(509)+2 = 1020 > 999$ (does not exist); left child $= 2(509)+1 = 1019 > 999$ (does not exist). Node 509 is itself a leaf, so it is already the maximum of the subtree rooted at it.
8. Therefore the predecessor of 254, and hence the 3rd largest element of the tree, is stored at index $\boxed{509}$.

KEY IDEA

A complete binary tree's array shape (root at index 0, children of index i at $2i+1$ and $2i+2$) determines only the SHAPE of the tree, not the BST order of values. So to find the k-th largest value, walk the shape using the in-order-predecessor rule starting from the rightmost node (the maximum), completely independent of any heap-order property.

WATCH OUT FOR

Assuming this is a max-heap and reading off the "3rd largest" as if array position reflected value magnitude directly (e.g., picking index 1 or 2, the heap's near-root children). The array only encodes tree shape here; the values still obey the BST property, so ranking requires an in-order-predecessor walk, not a heap-order read.

Q.29 NAT · 1 MARK

Compiler Design › Parsing

Consider the augmented grammar with {+, *, (,), id} as the set of terminals.

```
S' → S
S  → S + R | R
R  → R * P | P
P  → ( S ) | id
```

If I_0 is the set of two LR(0) items $\{[S' \rightarrow S\cdot],\ [S \rightarrow S \cdot +R]\}$, then $goto(closure(I_0), +)$ contains exactly _____ items.

Official answer: **5**

SOLUTION

1. Only a non-terminal immediately after the dot makes the closure add items. In $S' \rightarrow S\cdot$ the dot is at the end. In $S \rightarrow S \cdot +R$ the symbol after the dot is the terminal $+$. So $closure(I_0) = I_0$.
2. $goto(I_0, +)$ moves the dot over $+$ in every item that has $+$ right after the dot. Only $S \rightarrow S \cdot +R$ qualifies, so the kernel is the single item $S \rightarrow S + \cdot R$.
3. Take the closure of the kernel. The dot is before R, so add every R-production with the dot at the start: $R \rightarrow \cdot R * P$ and $R \rightarrow \cdot P$.
4. In $R \rightarrow \cdot P$ the dot is before P, so add every P-production with the dot at the start: $P \rightarrow \cdot(S)$ and $P \rightarrow \cdot id$.
5. In $R \rightarrow \cdot R * P$ the dot is before R, and both R-items are already present. In $P \rightarrow \cdot(S)$ and $P \rightarrow \cdot id$ the next symbols are terminals, so nothing more is added and the closure is complete.
6. The resulting set is $\{S \rightarrow S + \cdot R,\ R \rightarrow \cdot R * P,\ R \rightarrow \cdot P,\ P \rightarrow \cdot(S),\ P \rightarrow \cdot id\}$. It has exactly 5 items.

KEY IDEA

$goto(I, X)$ moves the dot over X in the items of I that have X after the dot. You then take the closure of that kernel, adding productions for each non-terminal that now follows a dot.

WATCH OUT FOR

Stopping at the kernel item $S \rightarrow S + \cdot R$ and forgetting the closure. Another slip is counting $S' \rightarrow S\cdot$ in the result, although it has no $+$ after its dot and so does not carry over.

Q.30 NAT · 1 MARK

Discrete Mathematics › Graph theory

Consider a simple undirected graph of 10 vertices. If the graph is disconnected, then the maximum number of edges it can have is _____.

Official answer: **36**

SOLUTION

1. A disconnected graph has at least two components. To maximise edges, first take exactly two components, with k and $10-k$ vertices, where $1 \le k \le 9$. Merging extra components into one of these only allows more edges, so two components is enough.
2. A simple graph on m vertices has at most $\binom{m}{2}$ edges, reached by K_m. So the number of edges is at most $f(k) = \binom{k}{2} + \binom{10-k}{2}$.
3. Expand: $f(k) = \dfrac{k(k-1) + (10-k)(9-k)}{2} = k^2 - 10k + 45$. This is a parabola opening upward with its minimum at $k = 5$, so on $1 \le k \le 9$ it is largest at the ends $k = 1$ and $k = 9$.
4. Check the values: $f(1) = 1 - 10 + 45 = 36$, $f(2) = 4 - 20 + 45 = 29$, $f(5) = 25 - 50 + 45 = 20$. These agree with $0 + \binom{9}{2} = 36$, $1 + \binom{8}{2} = 29$ and $10 + 10 = 20$.
5. The maximum is 36, reached by K_9 together with one isolated vertex (a graph with 10 vertices, disconnected).
6. General rule: a disconnected simple graph on n vertices has at most $\binom{n-1}{2}$ edges. Here $\binom{9}{2} = 36$.

KEY IDEA

A disconnected graph splits into components. The edge count is largest when one component is a complete graph on $n-1$ vertices and the other is a single isolated vertex, giving $\binom{n-1}{2}$ edges.

WATCH OUT FOR

Using $\binom{10}{2} = 45$ or $\binom{10}{2} - 1 = 44$. Removing one edge from K_{10} leaves the graph connected. To disconnect it you must remove all 9 edges at one vertex, which leaves $45 - 9 = 36$.

Q.31 NAT · 1 MARK

Database Management and Warehousing › Normal forms

Consider a relation $R(A, B, C, D, E)$ with the following three functional dependencies.

$AB \rightarrow C;\ BC \rightarrow D;\ C \rightarrow E;$

The number of superkeys in the relation R is _____.

Official answer: **8**

SOLUTION

1. Find the attributes that never appear on a right-hand side. A and B appear only on left sides, so no dependency can derive them. Every key must therefore contain both A and B.
2. Compute the closure of AB. Start with $\{A, B\}$. Using $AB \rightarrow C$ add C, giving $\{A, B, C\}$. Using $BC \rightarrow D$ add D. Using $C \rightarrow E$ add E.
3. So $(AB)^+ = \{A, B, C, D, E\}$, which is all of R. Hence AB is a superkey. Any key must contain $\{A, B\}$, so no proper subset of AB is a key, and AB is the only candidate key.
4. A superkey is any set of attributes that contains a candidate key. Here that means any set containing both A and B, with each of C, D, E free to be present or absent. A superset of AB has closure R because the closure of a superset contains $(AB)^+$.
5. Count: $2^3 = 8$ superkeys, namely AB, ABC, ABD, ABE, $ABCD$, $ABCE$, $ABDE$, $ABCDE$. Any set missing A or B cannot be a superkey, since its closure never gains the missing attribute.

KEY IDEA
Attributes that never appear on a right-hand side must be in every key. Here that is AB, and it already has closure R. Every superkey is a superset of the candidate key, so the count is $2^{(\text{number of other attributes})}$.

WATCH OUT FOR
Counting only the candidate key, or counting AB together with a few supersets. A superkey is any superset of a candidate key, so all $2^3 = 8$ supersets of AB count.

Q.32 NAT · 1 MARK

Discrete Mathematics › Counting

The number of arrangements of six identical balls in three identical bins is _____.

Official answer: **7**

SOLUTION

1. Balls are identical and bins are identical, so an arrangement is determined only by the multiset of bin contents. We therefore count the partitions of 6 into at most 3 parts. A bin may be empty, so fewer than 3 non-zero parts is allowed.
2. One non-empty bin: 6. This gives 1 arrangement.
3. Two non-empty bins: $5+1$, $4+2$, $3+3$. The order of the parts does not matter, so there are no others. This gives 3 arrangements.
4. Three non-empty bins: $4+1+1$, $3+2+1$, $2+2+2$. The largest part can be at most 4, since the other two parts are each at least 1. With largest part 4 the rest is $1+1$. With largest part 3 the rest is $2+1$. With largest part 2 the rest is $2+2$. This gives 3 arrangements.
5. Total $= 1+3+3 = 7$.

KEY IDEA
With identical balls and identical bins, only the unordered list of bin counts matters. The count is the number of partitions of 6 into at most 3 parts, which is 7.

WATCH OUT FOR
Counting ordered distributions (distinct bins) gives $\binom{8}{2} = 28$ instead of 7. Another slip is leaving out the case with empty bins, such as the single bin holding all 6 balls.

Q.33 NAT · 1 MARK

Computer Organization and Architecture › Memory & cache

A cache memory that has a hit rate of 0.8 has an access latency 10 ns and miss penalty 100 ns. An optimization is done on the cache to reduce the miss rate. However, the optimization results in an increase of cache access latency to 15 ns, whereas the miss penalty is not affected. The minimum hit rate (*rounded off to two decimal places*) needed after the optimization such that it should not increase the average memory access time is ____.

Official answer: **0.85**

SOLUTION

1. Recall AMAT (Average Memory Access Time) formula: $\text{AMAT} = \text{hit time} + (1 - \text{hit rate}) \times \text{miss penalty}$.
2. Compute AMAT before optimization: $\text{AMAT}_{old} = 10 + (1 - 0.8) \times 100 = 10 + 20 = 30$ ns.
3. Let h be the new (post-optimization) hit rate. Then $\text{AMAT}_{new} = 15 + (1 - h) \times 100$.
4. The optimization must not increase AMAT, so require $\text{AMAT}_{new} \leq \text{AMAT}_{old}$: $15 + (1 - h) \times 100 \leq 30$.
5. Simplify: $(1 - h) \times 100 \leq 15 \Rightarrow 1 - h \leq 0.15$.
6. Solve for h: $h \geq 0.85$.
7. Minimum hit rate needed = 0.85.

KEY IDEA
Average Memory Access Time (AMAT) = Hit time + Miss rate × Miss penalty. To keep AMAT unchanged after a latency increase, the miss rate (equivalently, hit rate) must improve enough to offset the added hit-time cost.

WATCH OUT FOR
Forgetting that AMAT uses (1 – hit rate) as the miss rate, or comparing the new access latency directly to the old AMAT instead of setting up the inequality AMAT_new ≤ AMAT_old and solving for h.

Q.34 NAT · 1 MARK

Engineering Mathematics › Calculus

The value of the following limit is _____.

$$\lim_{x \to 0^+} \frac{\sqrt{x}}{1 - e^{2\sqrt{x}}}$$

Official answer: **-0.5**

SOLUTION

1. Put $t = \sqrt{x}$. As $x \to 0^+$, $t \to 0^+$, so the limit becomes $\lim_{t \to 0^+} \frac{t}{1 - e^{2t}}$.
2. Direct substitution gives $\frac{0}{0}$, so we expand the exponential or use L'Hôpital's rule.
3. Series method: $e^{2t} = 1 + 2t + \frac{(2t)^2}{2!} + \cdots$, so $1 - e^{2t} = -2t - 2t^2 - \cdots = -2t\,(1 + t + \cdots)$.
4. Then $\frac{t}{1 - e^{2t}} = \frac{t}{-2t(1 + t + \cdots)} = -\frac{1}{2(1 + t + \cdots)}$, which tends to $-\frac{1}{2}$ as $t \to 0^+$.
5. Check with L'Hôpital's rule: differentiating the numerator and denominator gives $\lim_{t \to 0^+} \frac{1}{-2e^{2t}} = -\frac{1}{2}$.
6. So the limit equals $-\frac{1}{2} = -0.5$.

KEY IDEA
Substitute $t = \sqrt{x}$ to reduce the limit to the standard form $\frac{t}{1 - e^{2t}}$. Since $1 - e^{2t} \approx -2t$ near 0, the ratio tends to $-\frac{1}{2}$.

WATCH OUT FOR
Forgetting the minus sign in $1 - e^{2t}$ and answering $+0.5$. Another slip is forgetting the factor 2 from the exponent, which gives -1.

Q.35 NAT · 1 MARK

Computer Networks › DNS & HTTP

Consider the resolution of the domain name `www.gate.org.in` by a DNS resolver. Assume that no resource records are cached anywhere across the DNS servers and that iterative query mechanism is used in the resolution. The number of DNS query-response pairs involved in completely resolving the domain name is _____.

Official answer: **4**

SOLUTION

1. With an empty cache, the resolver knows only the addresses of the root name servers, so it starts there.
2. In iterative resolution the resolver itself contacts each server in turn. Each server either answers or returns a referral to the next server down.
3. Query 1 goes to a root server. It does not know www.gate.org.in , so it replies with a referral to the in server.
4. Query 2 goes to the in server. It replies with a referral to the server for org.in .
5. Query 3 goes to the org.in server. It replies with a referral to the server for gate.org.in .
6. Query 4 goes to the authoritative server for gate.org.in . It replies with the address record for www.gate.org.in , so resolution is complete.

7. Total: $1+1+1+1=4$ query-response pairs. The query from the client host to its own resolver is not counted.

KEY IDEA
With empty caches and iterative resolution, the resolver walks down the name one zone at a time: root, then in, then org.in, then gate.org.in. Each server is one query-response pair.

WATCH OUT FOR
Counting only three pairs by skipping the org.in server, or counting five by adding the client's own request to its resolver. The question asks only for the pairs the resolver exchanges with the DNS servers.

Q.36 MCQ · 2 MARKS

Discrete Mathematics › Recurrences

Which one of the following is the closed form for the generating function of the sequence $\{a_n\}_{n\geq 0}$ defined below?

$$a_n = \begin{cases} n+1, & n \text{ is odd} \\ 1, & \text{otherwise} \end{cases}$$

(A) $\dfrac{x(1+x^2)}{(1-x^2)^2} + \dfrac{1}{1-x}$

(B) $\dfrac{x(3-x^2)}{(1-x^2)^2} + \dfrac{1}{1-x}$

(C) $\dfrac{2x}{(1-x^2)^2} + \dfrac{1}{1-x}$

(D) $\dfrac{x}{(1-x^2)^2} + \dfrac{1}{1-x}$

Official answer: **A**

SOLUTION

1. Write $a_n = 1 + n\,[n \text{ odd}]$, so the sequence is $1, 2, 1, 4, 1, 6, \ldots$ The all-ones part is 1 for every $n \geq 0$, and the extra part is n at odd n and 0 at even n.
2. The all-ones part has generating function $\sum_{n\geq 0} x^n = \dfrac{1}{1-x}$.
3. The extra part is $\sum_{n \text{ odd}} nx^n$. First, $\sum_{n \text{ odd}} x^n = x + x^3 + x^5 + \cdots = \dfrac{x}{1-x^2}$.
4. Multiplying each coefficient by n corresponds to the operator $x\dfrac{d}{dx}$, so $\sum_{n \text{ odd}} nx^n = x\dfrac{d}{dx}\left(\dfrac{x}{1-x^2}\right)$.
5. Differentiate: $\dfrac{d}{dx}\dfrac{x}{1-x^2} = \dfrac{(1-x^2)+2x^2}{(1-x^2)^2} = \dfrac{1+x^2}{(1-x^2)^2}$.
6. Hence the extra part is $\dfrac{x(1+x^2)}{(1-x^2)^2}$.
7. Add the two parts: $G(x) = \dfrac{x(1+x^2)}{(1-x^2)^2} + \dfrac{1}{1-x}$, which is option (A).
8. Check: $\dfrac{x(1+x^2)}{(1-x^2)^2} = x + 3x^3 + \cdots$, so the coefficient of x^1 is $1+1=2=a_1$ and of x^3 is $3+1=4=a_3$. Option (B) would give $3+1=4$ at x^1, which is wrong, and (C) and (D) give $2+1=3$ and $1+1=2$ at x^1 but fail at x^3 (coefficients $4+1=5$ and $2+1=3$ against $a_3=4$).

KEY IDEA
Split the sequence into a constant part and an odd-index part. Get the odd-index part by applying $x\frac{d}{dx}$ to $\frac{x}{1-x^2}$, the generating function of the odd powers.

WATCH OUT FOR
Forgetting that n is a coefficient multiplier only on odd terms and using $\frac{x}{(1-x^2)^2}$ or $\frac{2x}{(1-x^2)^2}$. Also, differentiating $\frac{1}{1-x^2}$ and getting $\frac{2x}{(1-x^2)^2}$ is the series for $\sum 2k\,x^{2k}$, which is not the extra part here.

Q.37 MCQ · 2 MARKS

Discrete Mathematics › Graph theory

Consider a simple undirected unweighted graph with at least three vertices. If A is the adjacency matrix of the graph, then the number of 3-cycles in the graph is given by the trace of

(A) A^3

(B) A^3 divided by 2

(C) A^3 divided by 3

(D) A^3 divided by 6

Official answer: **D**

SOLUTION

1. The entry $(A^k)_{ii}$ counts the closed walks of length k that start and end at vertex i, because $(A^k)_{ij}$ counts walks of length k from i to j.
2. So $\text{tr}(A^3) = \sum_i (A^3)_{ii}$ is the total number of closed walks of length 3, summed over all starting vertices.
3. In a simple graph there are no loops and no multi-edges. A closed walk $i \to j \to k \to i$ then needs $i \neq j$, $j \neq k$ and $k \neq i$. So it is a triangle on three distinct vertices.
4. Each triangle gives 3 choices of starting vertex and 2 directions of travel. That is $3 \times 2 = 6$ closed walks.
5. Hence $\text{tr}(A^3) = 6 \times (\text{number of triangles})$.
6. Check on K_3: $A = J - I$ has eigenvalues $2, -1, -1$. So $\text{tr}(A^3) = 8 - 1 - 1 = 6$, which is 1 triangle times 6.
7. The number of 3-cycles is $\text{tr}(A^3)/6$, which is option (D).

KEY IDEA

$\text{tr}(A^3)$ counts closed walks of length 3. In a simple graph each triangle contributes exactly $3 \times 2 = 6$ of them, so divide the trace by 6.

WATCH OUT FOR

Dividing by 3 and forgetting the two directions of travel. Dividing by 2 and forgetting the three choices of starting vertex. Both give the wrong count.

Q.38 MCQ · 2 MARKS

Operating Systems › Memory management

Which one of the following statements is FALSE?

(A) The TLB performs an associative search in parallel on all its valid entries using page number of incoming virtual address.

(B) If the virtual address of a word given by CPU has a TLB hit, but the subsequent search for the word results in a cache miss, then the word will always be present in the main memory.

(C) The memory access time using a given inverted page table is always same for all incoming virtual addresses.

(D) In a system that uses hashed page tables, if two distinct virtual addresses V1 and V2 map to the same value while hashing, then the memory access time of these addresses will not be the same.

Official answer: **C**

SOLUTION

1. We need the FALSE statement, so check each one and find the one that cannot be guaranteed.
2. (A) A TLB is associative memory. The page number of the incoming virtual address is compared with all valid entries in parallel. True.

3. (B) A TLB hit means the page has a valid mapping to a frame, so the page is resident in main memory. If the word then misses in the cache, it is fetched from main memory, where it must be present. True.
4. (C) An inverted page table has one entry per physical frame. The entry for a virtual address is found by a hash lookup or a search, and the number of probes depends on the address. One address may be found at the first probe while another needs several probes along a collision chain. So the access time is not always the same, and this statement is FALSE.
5. (D) In a hashed page table, two addresses that hash to the same value share a bucket on a collision chain. One is found earlier in the chain and the other after traversing more entries, so their access times differ. True.
6. The false statement is (C).

KEY IDEA

Search an inverted or hashed page table through a hash bucket and collision chain, so lookup time varies with the address. A TLB hit guarantees the page is resident in main memory.

WATCH OUT FOR

Assuming an inverted page table gives constant-time lookup because it has one entry per frame. Another slip is doubting (B): a cache miss after a TLB hit still means the word is in main memory.

Q.39 MCQ · 2 MARKS

Database Management and Warehousing › Transactions

Let $R_i(z)$ and $W_i(z)$ denote read and write operations on a data element z by a transaction T_i, respectively. Consider the schedule S with four transactions.

$S : R_4(x)\, R_2(x)\, R_3(x)\, R_1(y)\, W_1(y)\, W_2(x)\, W_3(y)\, R_4(y)$

Which one of the following serial schedules is conflict equivalent to S?

(A) $T_1 \rightarrow T_3 \rightarrow T_4 \rightarrow T_2$

(B) $T_1 \rightarrow T_4 \rightarrow T_3 \rightarrow T_2$

(C) $T_4 \rightarrow T_1 \rightarrow T_3 \rightarrow T_2$

(D) $T_3 \rightarrow T_1 \rightarrow T_4 \rightarrow T_2$

Official answer: **A**

SOLUTION

1. Two operations conflict if they belong to different transactions, access the same data item and at least one of them is a write. For each conflicting pair, draw an edge $T_i \rightarrow T_j$ when T_i's operation comes first in S. A serial schedule is conflict equivalent to S exactly when it is a topological order of this precedence graph.
2. Item x: the operations in order are $R_4(x),\ R_2(x),\ R_3(x),\ W_2(x)$. The only write is $W_2(x)$. The reads $R_4(x)$ and $R_3(x)$ come before it and belong to other transactions, so $T_4 \rightarrow T_2$ and $T_3 \rightarrow T_2$. $R_2(x)$ is T_2 's own operation and gives no edge.
3. Item y: the operations in order are $R_1(y),\ W_1(y),\ W_3(y),\ R_4(y)$.
4. $R_1(y)$ and $W_1(y)$ both come before $W_3(y)$, and $R_1(y)$ against $W_3(y)$ is a read-write conflict. This gives $T_1 \rightarrow T_3$ ($W_1(y)$ against $W_3(y)$ is a write-write conflict and gives the same edge).
5. $W_1(y)$ comes before $R_4(y)$, so $T_1 \rightarrow T_4$. $W_3(y)$ comes before $R_4(y)$, so $T_3 \rightarrow T_4$.
6. The precedence graph has edges $T_1 \rightarrow T_3,\ T_1 \rightarrow T_4,\ T_3 \rightarrow T_4,\ T_3 \rightarrow T_2,\ T_4 \rightarrow T_2$. Every edge goes from a transaction in the list T_1, T_3, T_4, T_2 to a later one, so there is no cycle and S is conflict serializable.
7. Topological order: T_1 is the only transaction with no incoming edge, so it is first. After removing it, only T_3 has no incoming edge (it still needs nothing but T_1), so T_3 is second. Then T_4 (its predecessors T_1, T_3 are done), and T_2 is last. The order is forced, hence unique: $T_1 \rightarrow T_3 \rightarrow T_4 \rightarrow T_2$.

8. Check the other options against the edges. Option B puts T_4 before T_3, violating $T_3 \rightarrow T_4$. Option C puts T_4 before T_1, violating $T_1 \rightarrow T_4$. Option D puts T_3 before T_1, violating $T_1 \rightarrow T_3$. Only option (A) respects every edge.

KEY IDEA

Build the precedence graph from conflicting operations (different transactions, same item, at least one write). A serial schedule is conflict equivalent to S exactly when it is a topological order of that graph, and here the order is unique.

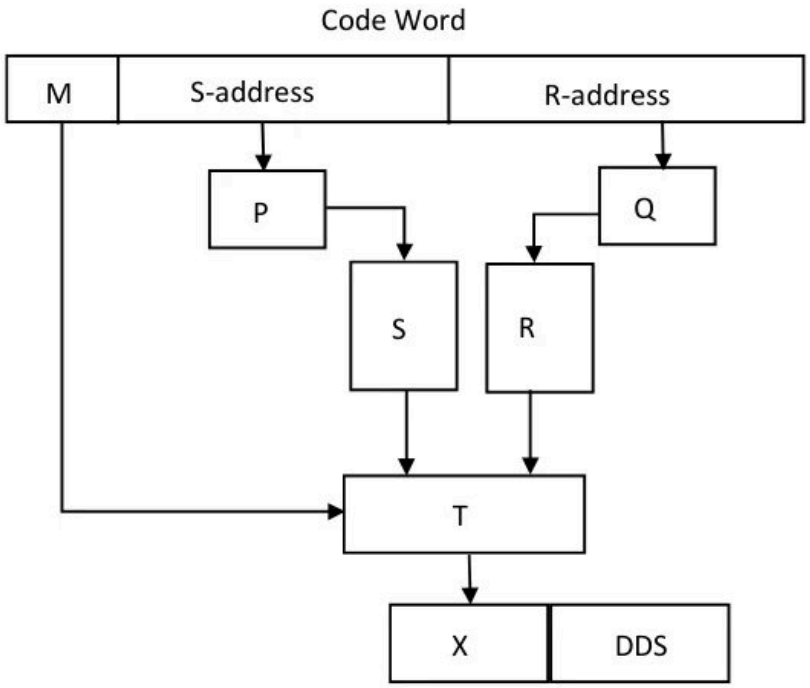
WATCH OUT FOR

Reading $R_2(x)$ as creating an edge with $W_2(x)$ (both are T_2's), or missing the edge $T_3 \rightarrow T_4$ from $W_3(y)$ before $R_4(y)$. That second slip makes T_4 and T_3 look interchangeable, so options B and D look plausible. Another slip is putting T_4 first because its read $R_4(x)$ is the first operation, forgetting that $R_4(y)$ comes after T_1's and T_3's writes.

Q.40 MCQ · 2 MARKS

Digital Electronics › Combinational

Q.40 Consider a digital display system (DDS) shown in the figure that displays the contents of register X. A 16-bit code word is used to load a word in X, either from S or from R. S is a 1024-word memory segment and R is a 32-word register file. Based on the value of mode bit M, T selects an input word to load in X. P and Q interface with the corresponding bits in the code word to choose the addressed word. Which one of the following represents the functionality of P, Q, and T?



(A)	P is 10:1 multiplexer;	Q is 5:1 multiplexer;	T is 2:1 multiplexer
(B)	P is 10:2^{10} decoder;	Q is 5:2^5 decoder;	T is 2:1 encoder
(C)	P is 10:2^{10} decoder;	Q is 5:2^5 decoder;	T is 2:1 multiplexer
(D)	P is 1:10 de-multiplexer;	Q is 1:5 de-multiplexer;	T is 2:1 multiplexer

Official answer: **C**

SOLUTION

1. S has $1024 = 2^{10}$ words, so addressing one word needs 10 bits: the S-address field is 10 bits wide.
2. R has $32 = 2^5$ words, so addressing one word needs 5 bits: the R-address field is 5 bits wide. With the mode bit M this gives $1 + 10 + 5 = 16$ bits, matching the 16-bit code word.
3. P takes the 10 address bits and must select exactly one of the 2^{10} words of S. A decoder turns an n-bit code into one of 2^n select lines, so P is a $10 : 2^{10}$ decoder.
4. Likewise Q takes 5 address bits and selects one of the 2^5 words of R, so Q is a $5 : 2^5$ decoder.

5. T has two data inputs (the word from S and the word from R), one output going to X, and the single control bit M. Choosing one of two inputs under one select line is a $2:1$ multiplexer.
6. P and Q only turn address bits into word-select lines; no data word passes through them. A $10:1$ or $5:1$ multiplexer would output a single bit and could not select a word, and a $1:10$ or $1:5$ de-multiplexer has one data input and takes only a one-bit input, not a 10-bit or 5-bit address. So options A and D are wrong.
7. An encoder does the reverse of a decoder (many lines to a binary code), so it cannot choose between two data words: option B is wrong. The answer is C.

KEY IDEA

Address bits to word-select lines is a decoder ($n:2^n$). Choosing one of two data words under a control bit is a multiplexer. The widths follow from $\log_2$ of the word counts: $\log_2 1024 = 10$ and $\log_2 32 = 5$.

WATCH OUT FOR

Calling P and Q multiplexers because they "select" a word. A multiplexer selects among data inputs and produces one bit per output, while here the address bits must activate one of many word lines, which is a decoder's job. Another slip is picking an encoder for T, but an encoder converts one active line into a code and cannot choose between two data words.

Q.41 MCQ · 2 MARKS

Digital Electronics › Number systems

Consider three floating point numbers A, B and C stored in registers R_A, R_B and R_C, respectively as per IEEE-754 single precision floating point format. The 32-bit content stored in these registers (in hexadecimal form) are as follows.

R_A = 0xC1400000, R_B = 0x42100000, R_C = 0x41400000

Which one of the following is FALSE?

(A) $A + C = 0$

(B) $C = A + B$

(C) $B = 3C$

(D) $(B - C) > 0$

Official answer: **B**

SOLUTION

1. IEEE-754 single precision value: $(-1)^s \times 1.f \times 2^{e-127}$, with 1 sign bit, 8 exponent bits and 23 fraction bits.
2. R_A = 0xC1400000 = 1100 0001 0100 0000 ... : sign 1, exponent $10000010 = 130$, so $e - 127 = 3$; fraction 100...0 gives $1.1_2 = 1.5$. Hence $A = -1.5 \times 2^3 = -12$.
3. R_C = 0x41400000 = 0100 0001 0100 0000 ... : sign 0, exponent $10000010 = 130$ (power 2^3), fraction 100...0 gives 1.5. Hence $C = 1.5 \times 8 = 12$.
4. R_B = 0x42100000 = 0100 0010 0001 0000 ... : sign 0, exponent $10000100 = 132$ (power 2^5), fraction 0010 0...0 gives $1 + 2^{-3} = 1.125$. Hence $B = 1.125 \times 32 = 36$.
5. Check (A): $A + C = -12 + 12 = 0$, true.
6. Check (B): $A + B = -12 + 36 = 24 \neq 12 = C$, so $C = A + B$ is false.
7. Check (C): $3C = 36 = B$, true. Check (D): $B - C = 36 - 12 = 24 > 0$, true.
8. Only (B) is false.

KEY IDEA

Split each hexadecimal word into sign, 8-bit exponent (subtract the bias 127) and fraction, rebuild $(-1)^s \times 1.f \times 2^{e-127}$, then test each option with the decoded values $A = -12$, $B = 36$, $C = 12$.

WATCH OUT FOR

Misreading the exponent field or the fraction bits. For R_B the hex digits `2 1 0` give exponent `10000100` and fraction `0010...`, not `0100...`. Taking the exponent as 131 or the fraction as 1.25 gives the wrong value of B. Also forgetting the sign bit of R_A (first hex digit C = `1100`).

Q.42 MCQ · 2 MARKS

Operating Systems › CPU scheduling

Consider four processes P, Q, R, and S scheduled on a CPU as per round robin algorithm with a time quantum of 4 units. The processes arrive in the order P, Q, R, S, all at time t = 0. There is exactly one context switch from S to Q, exactly one context switch from R to Q, and exactly two context switches from Q to R. There is no context switch from S to P. Switching to a ready process after the termination of another process is also considered a context switch. Which one of the following NOT possible as CPU burst time (in time units) of these processes?

(A) P = 4, Q = 10, R = 6, S = 2

(B) P = 2, Q = 9, R = 5, S = 1

(C) P = 4, Q = 12, R = 5, S = 4

(D) P = 3, Q = 7, R = 7, S = 3

Official answer: **D**

SOLUTION

1. All four processes arrive at $t = 0$ in the order P, Q, R, S. In each round every unfinished process runs for $\min(4, \text{remaining})$ units. A switch is counted between consecutive CPU slices, including the switch to the next ready process after a process terminates.
2. The required switch counts are: $S \to Q$ exactly once, $R \to Q$ exactly once, $Q \to R$ exactly twice, and no $S \to P$.
3. **Option A** ($P = 4, Q = 10, R = 6, S = 2$): P finishes in round 1. Q runs 4 (6 left), R runs 4 (2 left), S runs 2 and finishes. Round 2: Q runs 4 (2 left), R runs 2 and finishes, Q runs 2 and finishes. Order: P, Q, R, S, Q, R, Q. Counts: $S \to Q$ once, $Q \to R$ twice, $R \to Q$ once, no $S \to P$. Possible.
4. **Option B** ($P = 2, Q = 9, R = 5, S = 1$): P finishes. Q runs 4 (5 left), R runs 4 (1 left), S runs 1 and finishes. Then Q runs 4 (1 left), R runs 1 and finishes, Q runs 1 and finishes. Order: P, Q, R, S, Q, R, Q, with the same counts as A. Possible.
5. **Option C** ($P = 4, Q = 12, R = 5, S = 4$): P finishes. Q runs 4 (8 left), R runs 4 (1 left), S runs 4 and finishes. Then Q runs 4 (4 left), R runs 1 and finishes, Q runs 4 and finishes. Order: P, Q, R, S, Q, R, Q, with the same counts as A. Possible.
6. **Option D** ($P = 3, Q = 7, R = 7, S = 3$): P finishes in round 1. Q runs 4 (3 left), R runs 4 (3 left), S runs 3 and finishes. Round 2: Q runs 3 and finishes, then R runs 3 and finishes. Order: P, Q, R, S, Q, R.
7. In D, $S \to Q$ occurs once and $Q \to R$ occurs twice, but Q finishes before R in round 2, so no $R \to Q$ switch ever occurs. The condition of exactly one $R \to Q$ switch fails, so D is not possible. The answer is D.

KEY IDEA

Simulate round robin with quantum 4, write the order of CPU slices, and count the adjacent pairs. The required switches ($S \to Q$ once, $R \to Q$ once, $Q \to R$ twice) need the order P, Q, R, S, Q, R, Q. That means P and S end in round 1, Q needs three slices and R needs two slices, and R finishes before Q.

WATCH OUT FOR

Forgetting that the move to the next ready process after a termination counts as a context switch. Another slip is checking only $S \to Q$ and $Q \to R$ in option D and missing that $R \to Q$ never occurs, because Q finishes in round 2 before R gets its second slice.

Q.43 MCQ · 2 MARKS

Programming in C › Pointers & arrays

What is printed by the following ANSI C program?

```
 #include<stdio.h>
int main(int argc, char *argv[])
{
      int a[3][3][3] =
      {{1, 2, 3, 4, 5, 6, 7, 8, 9},
       {10, 11, 12, 13, 14, 15, 16, 17, 18},
       {19, 20, 21, 22, 23, 24, 25, 26, 27}};
   int i = 0, j = 0, k = 0;
   for( i = 0; i < 3; i++ ){
      for(k = 0; k < 3; k++ )
         printf("%d ", a[i][j][k]);
      printf("\n");
   }
   return 0;
}
```

(A) 1 2 3
10 11 12
19 20 21

(B) 1 4 7
10 13 16
19 22 25

(C) 1 2 3
4 5 6
7 8 9

(D) 1 2 3
13 14 15
25 26 27

Official answer: **A**

SOLUTION

1. Array a[3][3][3] is declared with a flat initializer of 27 values, grouped only two levels deep into three lists of 9: {1..9}, {10..18}, {19..27}.
2. C fills a multidimensional array in row-major order regardless of how the initializer braces are nested: the rightmost (fastest-varying) index cycles first, so within each outer group of 9, the values fill a[i][0][0..2], then a[i][1][0..2], then a[i][2][0..2].
3. So a[0][0..2][0..2] = rows (1,2,3),(4,5,6),(7,8,9); a[1][0..2][0..2] = (10,11,12),(13,14,15),(16,17,18); a[2][0..2][0..2] = (19,20,21),(22,23,24),(25,26,27).
4. j is initialized to 0 and never modified anywhere in the program, so every access uses a[i][0][k] .

5. Outer loop: `i` runs 0 to 2. Inner loop: `k` runs 0 to 2, printing `a[i][0][k]` each time, followed by a newline after the inner loop finishes.
6. i=0: prints a[0][0][0], a[0][0][1], a[0][0][2] = `1 2 3`.
7. i=1: prints a[1][0][0], a[1][0][1], a[1][0][2] = `10 11 12`.
8. i=2: prints a[2][0][0], a[2][0][1], a[2][0][2] = `19 20 21`.
9. The full output is `1 2 3`, `10 11 12`, `19 20 21`. This is option (A).

KEY IDEA

A 3-D array `a[3][3][3]` initialized from a flat, doubly-nested brace list still fills strictly in row-major order (rightmost index varies fastest, regardless of how the initializer braces group the numbers visually). With `j` fixed at 0, the loops print `a[i][0][k]` for `i = 0..2, k = 0..2`, which are the first three elements of each of the three outer i-blocks.

WATCH OUT FOR

Assuming the flat initializer list fills the array in the visually-grouped chunks shown by the braces (giving option C), or confusing which index (i, j, or k) is held fixed and which varies fastest, leading to picking (B) or (D).

Q.44 MCQ · 2 MARKS

Programming in C › C basics

Q.44 What is printed by the following ANSI C program?

```
#include<stdio.h>
int main(int argc, char *argv[]){
    char a = 'P';
    char b = 'x';
    char c = (a & b) + '*';
    char d = (a | b) - '-';
    char e = (a ^ b) + '+';
    printf("%c %c %c\n", c, d, e);
    return 0;
}
```

ASCII encoding for relevant characters is given below

A	B	C	...	Z
65	66	67	...	90

a	b	c	...	z
97	98	99	...	122

*	+	–
42	43	45

(A) z K S

(B) 122 75 83

(C) * - +

(D) P x +

Official answer: **A**

SOLUTION

1. a = P = 80 = 01010000 in binary; b = x = 120 = 01111000 in binary.
2. c = (a & b) + asterisk: AND gives 01010000 = 80. 80 + 42 = 122 = z.
3. d = (a | b) - minus: OR gives 01111000 = 120. 120 - 45 = 75 = K.
4. e = (a ^ b) + plus: XOR gives 00101000 = 40. 40 + 43 = 83 = S.
5. printf prints c, d, e as characters: output is z K S, matching option A.

KEY IDEA
Bitwise AND/OR/XOR act on the binary ASCII codes of the char operands; the result is combined arithmetically with a symbol's ASCII code, and the resulting integer is printed as a character via percent-c.

WATCH OUT FOR
Mismapping the resulting decimal ASCII code to the wrong letter by miscounting the offset from A, e.g. taking 75 as L instead of K.

Q.45 MCQ · 2 MARKS

Engineering Mathematics › Linear algebra

Consider solving the following system of simultaneous equations using LU decomposition.

$$x_1 + x_2 - 2x_3 = 4$$
$$x_1 + 3x_2 - x_3 = 7$$
$$2x_1 + x_2 - 5x_3 = 7$$

where L and U are denoted as

$$L = \begin{bmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{bmatrix}, \quad U = \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{bmatrix}$$

Which one of the following is the correct combination of values for L_{32}, U_{33}, and x_1?

(A) $L_{32} = 2,\ U_{33} = -\frac{1}{2},\ x_1 = -1$

(B) $L_{32} = 2,\ U_{33} = 2,\ x_1 = -1$

(C) $L_{32} = -\frac{1}{2},\ U_{33} = 2,\ x_1 = 0$

(D) $L_{32} = -\frac{1}{2},\ U_{33} = -\frac{1}{2},\ x_1 = 0$

Official answer: **D**

SOLUTION

1. Write the system as $A\mathbf{x} = \mathbf{b}$ with $A = \begin{bmatrix} 1 & 1 & -2 \\ 1 & 3 & -1 \\ 2 & 1 & -5 \end{bmatrix}$ and $\mathbf{b} = (4, 7, 7)^T$. Use the Doolittle form $A = LU$ with unit diagonal in L, so the multipliers are $L_{21} = 1$ and $L_{31} = 2$.
2. Eliminate column 1. $R_2 \to R_2 - R_1$ gives $(0,\ 2,\ 1)$. $R_3 \to R_3 - 2R_1$ gives $(0,\ -1,\ -1)$.
3. Eliminate column 2. The multiplier is $L_{32} = \dfrac{-1}{2} = -\dfrac{1}{2}$. Then $R_3 \to R_3 + \frac{1}{2}R_2$ gives $U_{33} = -1 + \frac{1}{2} = -\dfrac{1}{2}$.
4. So $U = \begin{bmatrix} 1 & 1 & -2 \\ 0 & 2 & 1 \\ 0 & 0 & -\frac{1}{2} \end{bmatrix}$ and $L = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 2 & -\frac{1}{2} & 1 \end{bmatrix}$. Multiplying back, row 3 of LU is $2(1, 1, -2) - \frac{1}{2}(0, 2, 1) + (0, 0, -\frac{1}{2}) = (2, 1, -5)$, which matches A.
5. Forward substitution $L\mathbf{y} = \mathbf{b}$: $y_1 = 4$, $y_2 = 7 - 4 = 3$, $y_3 = 7 - 2(4) + \frac{1}{2}(3) = \frac{1}{2}$.
6. Back substitution $U\mathbf{x} = \mathbf{y}$: $-\frac{1}{2}x_3 = \frac{1}{2} \Rightarrow x_3 = -1$. Then $2x_2 + x_3 = 3 \Rightarrow x_2 = 2$.
7. From the first equation, $x_1 = 4 - x_2 + 2x_3 = 4 - 2 - 2 = 0$. Check in the other equations: $0 + 3(2) - (-1) = 7$ and $2(0) + 2 - 5(-1) = 7$, both hold.
8. So $L_{32} = -\frac{1}{2}$, $U_{33} = -\frac{1}{2}$ and $x_1 = 0$, which is option (D).

KEY IDEA
LU decomposition records the row-elimination multipliers in L and the reduced rows in U. The solution then comes from one forward and one back substitution. The product $U_{11}U_{22}U_{33} = \det A = -1$ gives a quick check on U_{33}.

WATCH OUT FOR
Writing L_{32} with the wrong sign or taking the pivot ratio upside down, which gives 2 instead of $-\frac{1}{2}$. Another slip is to forget that U_{33} is the third row after both eliminations, so the reduced entry -1 must be adjusted by $-L_{32} \cdot 1$.

Q.46 MSQ · 2 MARKS

Theory of Computation › Turing machines

Which of the following is/are undecidable?

(A) Given two Turing machines M_1 and M_2, decide if $L(M_1) = L(M_2)$.

(B) Given a Turing machine M, decide if $L(M)$ is regular.

(C) Given a Turing machine M, decide if M accepts all strings.

(D) Given a Turing machine M, decide if M takes more than 10^{73} steps on every string.

Official answer: **A, B, C**

SOLUTION

1. (A) Equivalence of two Turing machines is a property of a pair of machines, so Rice's theorem does not apply directly. Reduce from the totality problem instead: fix M_2 to be a machine that accepts every string, so $L(M_2) = \Sigma^*$. Then $L(M_1) = L(M_2)$ exactly when M_1 accepts all strings. Totality is undecidable (see C), so equivalence is undecidable.
2. (B) "$L(M)$ is regular" is a property of the language $L(M)$. It is non-trivial: a machine for $\emptyset$ has it, and a machine for $\{a^n b^n \mid n \geq 0\}$ does not. By Rice's theorem it is undecidable.
3. (C) "$L(M) = \Sigma^*$" is a property of the language $L(M)$. It is non-trivial: a machine accepting every string has it, and a machine accepting nothing does not. By Rice's theorem it is undecidable.
4. (D) This is a property of the running time, not only of the language, so Rice's theorem does not apply and we argue directly. Let $N = 10^{73}$, a fixed constant. "M takes more than N steps on every string" means that no string makes M halt within N steps (a string on which M loops forever counts as taking more than N steps).
5. In N steps the head starts at cell 0 and can reach at most cells $0, 1, \ldots, N$. So the first N steps depend only on the first $N + 1$ input symbols. Hence if some string w makes M halt within N steps, then the prefix of w of length at most $N + 1$ (or w itself if it is shorter) makes M halt within N steps in the same way.
6. So it is enough to simulate M for N steps on each of the finitely many strings over Σ of length at most $N + 1$. If any simulation halts within N steps the answer is NO, otherwise YES. This is a finite, if astronomically large, computation, so (D) is decidable.
7. The undecidable problems are A, B and C.

KEY IDEA
Rice's theorem makes every non-trivial property of L(M) undecidable (B, C), and equivalence reduces from totality (A). A bound on running time with a fixed constant is different: only finitely many input prefixes matter, so brute-force simulation decides it (D).

WATCH OUT FOR
Calling D undecidable because it is about Turing machines, or applying Rice's theorem to it. It is a property of running time, not of the language, and the fixed step bound makes it a finite check.

Q.47 MSQ · 2 MARKS

Theory of Computation › Context-free languages

Consider the following languages:

$L_1 = \{a^n w a^n \mid w \in \{a,b\}^*\}$

$L_2 = \{wxw^R \mid w, x \in \{a,b\}^*,\ |w|, |x| > 0\}$

Note that w^R is the reversal of the string w. Which of the following is/are TRUE?

(A) L_1 and L_2 are regular.

(B) L_1 and L_2 are context-free.

(C) L_1 is regular and L_2 is context-free.

(D) L_1 and L_2 are context-free but not regular.

Official answer: **A, B, C**

SOLUTION

1. L_1: here $w \in \{a,b\}^*$ is arbitrary (it may be empty). Taking $n = 1$ gives every string $a\,w\,a$. A larger n gives nothing new, because $a^n w a^n = a\,(a^{n-1} w a^{n-1})\,a$ already has the form $a\,w'\,a$. Hence $L_1 = a(a+b)^*a$, the strings of length at least 2 that start and end with a. If $n = 0$ is also allowed, $L_1 = \{a,b\}^*$. Either way L_1 is regular.
2. L_2: take $|w| = 1$, say $w = c \in \{a,b\}$. Then $wxw^R = c\,x\,c$ with x non-empty, which gives every string of length at least 3 that starts and ends with the same symbol.
3. A longer w gives nothing new. The string wxw^R starts with the first symbol of w and ends with that same symbol. Its length is at least 3 because $|w|, |x| > 0$. Writing c for the first symbol, the string is $c\,y\,c$ with y non-empty. So $L_2 = a(a+b)^+a + b(a+b)^+b$, which is regular.
4. So both L_1 and L_2 are regular. The reversal in L_2 looks as if it needs a stack. But the free middle part x means a finite automaton only has to compare the first and last symbols.
5. Every regular language is context-free, so L_1 and L_2 are both context-free.
6. Check the options. (A) true. (B) true, since regular implies context-free. (C) true, since L_1 is regular and L_2, being regular, is context-free. (D) false, because neither language is non-regular.

KEY IDEA

The free middle part (w in L_1, x in L_2) absorbs everything except the first and last symbols. So both languages reduce to a check on the end symbols, and both are regular. Every regular language is context-free.

WATCH OUT FOR

Reading wxw^R as a palindrome-like language that needs a stack and so is non-regular. The non-empty free x between w and w^R means only the first and last symbols matter, so this is not the language $\{ww^R\}$.

Q.48 MSQ · 2 MARKS

Theory of Computation › Context-free languages

Consider the following languages:

$L_1 = \{ww \mid w \in \{a,b\}^*\}$

$L_2 = \{a^n b^n c^m \mid m, n \geq 0\}$

$L_3 = \{a^m b^n c^n \mid m, n \geq 0\}$

Which of the following statements is/are FALSE?

(A) L_1 is not context-free but L_2 and L_3 are deterministic context-free.

(B) Neither L_1 nor L_2 is context-free.

(C) L_2, L_3 and $L_2 \cap L_3$ all are context-free.

(D) Neither L_1 nor its complement is context-free.

Official answer: **B, C, D**

SOLUTION

1. $L_1 = \{ww \mid w \in \{a, b\}^*\}$ is the copy language. It is not context-free (pumping lemma for CFLs, or the standard result). So the first half of (A) is correct.
2. $L_2 = \{a^n b^n c^m \mid m, n \geq 0\}$ is accepted by a DPDA that pushes a symbol for each a, pops one for each b, then reads any number of c's and accepts on empty stack. So L_2 is a deterministic context-free language. A grammar is $S \to XC$, $X \to aXb \mid \varepsilon$, $C \to cC \mid \varepsilon$.
3. $L_3 = \{a^m b^n c^n \mid m, n \geq 0\}$ is accepted by a DPDA that skips the a's, pushes for each b and pops for each c. So L_3 is a DCFL. Hence statement (A) is TRUE.
4. Statement (B) says L_2 is not context-free. But L_2 is context-free by step 2, so (B) is FALSE.
5. $L_2 \cap L_3 = \{a^n b^n c^n \mid n \geq 0\}$, which is not context-free (pumping lemma for CFLs). Statement (C) claims it is context-free, so (C) is FALSE.
6. The complement of L_1 is context-free. A string over $\{a, b\}$ is outside L_1 if its length is odd, or if its length is $2n$ and some position i and position $i + n$ hold different symbols. A PDA can guess the position, push a count while reading, and match it against the second half. This gives a CFL. Statement (D) says neither L_1 nor its complement is context-free, so (D) is FALSE.
7. The FALSE statements are B, C and D.

KEY IDEA

$L_1 = \{ww\}$ is not context-free but its complement is. $a^n b^n c^m$ and $a^m b^n c^n$ are DCFLs, and their intersection $a^n b^n c^n$ is not context-free, so CFLs are not closed under intersection.

WATCH OUT FOR

Assuming that because $\{ww\}$ is not context-free, its complement cannot be either. The complement is context-free because a PDA can guess one mismatching position. Another slip is treating $L_2 \cap L_3$ as context-free just because L_2 and L_3 are.

Q.49 MSQ · 2 MARKS

Data Structures and Algorithms › Graph algorithms

Consider a simple undirected weighted graph G, all of whose edge weights are distinct. Which of the following statements about the minimum spanning trees of G is/are TRUE?

(A) The edge with the second smallest weight is always part of any minimum spanning tree of G.

(B) One or both of the edges with the third smallest and the fourth smallest weights are part of any minimum spanning tree of G.

(C) Suppose $S \subseteq V$ be such that $S \neq \phi$ and $S \neq V$. Consider the edge with the minimum weight such that one of its vertices is in S and the other in $V \setminus S$. Such an edge will always be part of any minimum spanning tree of G.

(D) G can have multiple minimum spanning trees.

Official answer: **A, B, C**

SOLUTION

1. Since all edge weights are distinct, Kruskal's algorithm produces the (unique) MST: at every step it either accepts the next-cheapest edge (if it joins two different components) or rejects it (if it would close a cycle), and because weights are distinct there is never a tie to break arbitrarily.
2. (A): Let e_1 be the globally smallest edge, with endpoints u_1, v_1; Kruskal always accepts it, merging $\{u_1\}$ and $\{v_1\}$ into one component. When the second-smallest edge e_2 is considered, the only pair of vertices lying in the same component is $\{u_1, v_1\}$ itself. Since G is simple, no edge other than e_1 can join u_1 and v_1, so $e_2 \neq e_1$ cannot close a cycle at this stage. Hence Kruskal always accepts e_2: (A) is TRUE.

3. (B): By the same argument as above, e_1 and e_2 are always both accepted. Two cases arise for the components they create. Case 1: e_1, e_2 are vertex-disjoint: the only same-component vertex pairs are $\{u_1, v_1\}$ and $\{u_2, v_2\}$, each already realized by e_1 or e_2; so e_3 (being a different edge) cannot close a cycle and is always accepted, satisfying the statement. Case 2: e_1, e_2 share a vertex, say $e_1 = (A, B)$, $e_2 = (A, C)$, forming a 3-vertex component: among A, B, C the only edge not yet placed is BC, and it is the unique edge whose acceptance could ever be blocked by this component (its weight rank is necessarily ≥ 3, since ranks 1 and 2 are already e_1, e_2). If $BC = e_3$, it is rejected (it would close the triangle), but then all three possible edges among $\{A, B, C\}$ have been accounted for, so e_4 cannot lie inside this component and must be accepted. If $BC \neq e_3$ (its rank is ≥ 4 or the edge doesn't even exist in G), then e_3 itself cannot close this cycle and is accepted. Either way, at least one of e_3, e_4 is always accepted into the MST: (B) is TRUE.
4. (C): This is exactly the cut property of MSTs: for any nontrivial vertex partition $(S, V \setminus S)$, the minimum-weight edge crossing the cut lies in every MST. Since all weights are distinct, adding any other crossing edge before this one is impossible without first using it (any cycle created by a heavier crossing edge could be broken by removing a heavier edge on the cycle, and the minimum crossing edge is never that heavier edge), so it is always selected by Kruskal/Prim. (C) is TRUE.
5. (D): Distinct edge weights mean Kruskal's algorithm has no tie to break at any step, so the sequence of accept/reject decisions, and hence the MST itself, is completely determined. Therefore G has exactly one MST, not multiple: (D) is FALSE.

KEY IDEA

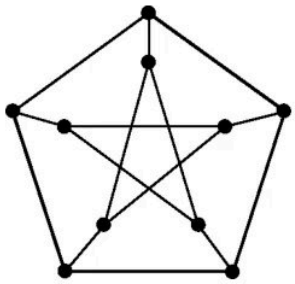
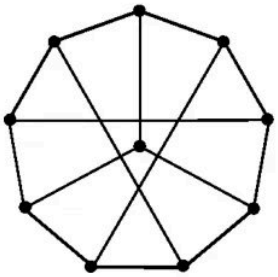
When all edge weights in a simple graph are distinct, Kruskal's algorithm has no ties to break, so the MST is unique; this forces the 1st and 2nd smallest edges to always be in the MST (they can't yet close any cycle), forces at least one of the 3rd/4th smallest edges in, and makes the cut-property minimum-crossing-edge argument airtight.

WATCH OUT FOR

Assuming "second smallest edge is always in the MST" generalizes to "the k-th smallest edge is always in the MST" for larger k; it does not (that's exactly why (B) is phrased as "one or both of the 3rd and 4th smallest", not "the 3rd smallest alone"); also assuming distinct weights only give a unique MST-*value* rather than a unique MST *itself*, which would wrongly make (D) look plausible.

Q.50 MSQ · 2 MARKS

Discrete Mathematics › Graph theory

Q.50	The following simple undirected graph is referred to as the Peterson graph.  Which of the following statements is/are TRUE?
(A)	The chromatic number of the graph is 3.
(B)	The graph has a Hamiltonian path.
(C)	The following graph is isomorphic to the Peterson graph. 
(D)	The size of the largest independent set of the given graph is 3. (A subset of vertices of a graph form an independent set if no two vertices of the subset are adjacent.)

Official answer: **A, B, C**

SOLUTION

1. Label the Peterson graph: outer 5-cycle $o_0, \ldots, o_4$, spokes $o_k - i_k$, inner edges $i_k - i_{k\pm2}$ (indices mod 5). It is 3-regular with 10 vertices and 15 edges, and it has girth 5.
2. (A) The outer 5-cycle is an odd cycle, so $\chi \geq 3$. A 3-colouring: $o_0, \ldots, o_4 = 1, 2, 1, 2, 3$ and $i_0, \ldots, i_4 = 2, 3, 3, 1, 1$. Outer edges: 1-2, 2-1, 1-2, 2-3, 3-1 all differ. Spokes: $(1,2), (2,3), (1,3), (2,1), (3,1)$ all differ. Inner edges $i_0i_2, i_1i_3, i_2i_4, i_3i_0, i_4i_1$ have colours $(2,3), (3,1), (3,1), (1,2), (1,3)$, all differ. So $\chi = 3$. TRUE.

3. (B) Hamiltonian path: $o_0, o_1, o_2, o_3, o_4, i_4, i_1, i_3, i_0, i_2$. The consecutive pairs are edges (o_4i_4 is a spoke; $i_4i_1, i_1i_3, i_3i_0, i_0i_2$ are inner edges) and all 10 vertices appear once. TRUE. (The Peterson graph has no Hamiltonian cycle, but a path exists.)
4. (C) A simple cubic graph on 10 vertices with girth 5 is unique: the Peterson graph. In the figure, number the 9-cycle $0, \ldots, 8$ clockwise from the top (1 = upper-right, 2 = right, 3 = lower-right, 4 = bottom-right, 5 = bottom-left, 6 = lower-left, 7 = left, 8 = upper-left) and call the centre c. The chords are $0c$, $3c$, $6c$, 2-7, 4-8, 1-5, so every vertex has degree 3 (15 edges).
5. Girth check: the centre's neighbours $0, 3, 6$ are pairwise non-adjacent and have no common neighbour other than c (neighbours of 0: $1, 8$; of 3: $2, 4$; of 6: $5, 7$), so no triangle or 4-cycle passes through c. Each chord spans 4 cycle edges, so a single chord gives only 5-cycles. A 4-cycle using two chords would need their ends joined by cycle edges: for 2-7 and 4-8, $7 \sim 8$ but $2 \not\sim 4$; for 2-7 and 1-5, $2 \sim 1$ but $7 \not\sim 5$; for 4-8 and 1-5, $4 \sim 5$ but $8 \not\sim 1$. So there is no 3- or 4-cycle, the girth is 5, and the graph is isomorphic to the Peterson graph. TRUE.
6. (D) Each of the two 5-cycles allows at most $\lfloor 5/2 \rfloor = 2$ independent vertices, so $\alpha \leq 4$. The set $\{o_0, o_2, i_3, i_4\}$ is independent: o_0 is adjacent only to o_1, o_4, i_0; o_2 only to o_1, o_3, i_2; i_3 only to i_0, i_1, o_3; i_4 only to i_1, i_2, o_4. So $\alpha = 4 \neq 3$. FALSE.
7. Hence the true statements are A, B and C.

KEY IDEA

The Peterson graph is the unique 3-regular graph on 10 vertices with girth 5, so another cubic 10-vertex graph with no triangle or 4-cycle is isomorphic to it. Its chromatic number is 3, it has a Hamiltonian path but no Hamiltonian cycle, and its largest independent set has size 4.

WATCH OUT FOR

Judging the isomorphism in (C) by how the drawing looks instead of checking degrees and girth. Another slip is assuming that because the Peterson graph has no Hamiltonian cycle it has no Hamiltonian path, or taking the independence number to be 3 instead of 4.

Q.51 MSQ · 2 MARKS

Data Structures and Algorithms › Complexity

Consider the following recurrence:

$$\begin{aligned} f(1) &= 1; \\ f(2n) &= 2f(n) - 1, \quad \text{for } n \geq 1; \\ f(2n+1) &= 2f(n) + 1, \quad \text{for } n \geq 1. \end{aligned}$$

Then, which of the following statements is/are TRUE?

(A) $f(2^n - 1) = 2^n - 1$

(B) $f(2^n) = 1$

(C) $f(5 \cdot 2^n) = 2^{n+1} + 1$

(D) $f(2^n + 1) = 2^n + 1$

Official answer: **A, B, C**

SOLUTION

1. Compute small values directly from the recurrence: $f(1) = 1$, and using $f(2n) = 2f(n) - 1$, $f(2n+1) = 2f(n) + 1$ repeatedly gives $f(2) = 1, f(3) = 3, f(4) = 1, f(5) = 3, f(6) = 5, f(7) = 7, f(8) = 1, \ldots, f(15) = 15, f(16) = 1$.
2. Within the block $n \in [2^k, 2^{k+1} - 1]$ the values traced above are exactly the consecutive odd numbers $1, 3, 5, \ldots, 2^k - 1$, so conjecture the closed form $f(n) = 2n - 2^{k+1} + 1$ for n in that block.
3. Verify by induction: if $n = 2m$ lies in block k, then m lies in block $k - 1$, so $f(n) = 2f(m) - 1 = 2\left(2m - 2^k + 1\right) - 1 = 2n - 2^{k+1} + 1$, matching the conjectured formula; the odd case $n = 2m + 1$

checks the same way using $f(n) = 2f(m) + 1$. So the closed form $f(n) = 2n - 2^{k+1} + 1$ for $n \in [2^k, 2^{k+1} - 1]$ holds for all $n \geq 1$.

4. Option A: $2^n - 1$ is the top of block $k = n - 1$ (range $[2^{n-1}, 2^n - 1]$). Then $f(2^n - 1) = 2(2^n - 1) - 2^n + 1 = 2^n - 1$, which matches the claim: TRUE.
5. Option B: 2^n is the bottom of block $k = n$ (range $[2^n, 2^{n+1} - 1]$). Then $f(2^n) = 2 \cdot 2^n - 2^{n+1} + 1 = 1$, which matches the claim: TRUE.
6. Option C: $5 \cdot 2^n = 2^{n+2} + 2^n$ lies strictly between 2^{n+2} and $2^{n+3} - 1$, so it is in block $k = n + 2$. Then $f(5 \cdot 2^n) = 2(5 \cdot 2^n) - 2^{n+3} + 1 = 10 \cdot 2^n - 8 \cdot 2^n + 1 = 2^{n+1} + 1$, which matches the claim: TRUE.
7. Option D: $2^n + 1$ lies in block $k = n$ (range $[2^n, 2^{n+1} - 1]$, for $n \geq 1$). Then $f(2^n + 1) = 2(2^n + 1) - 2^{n+1} + 1 = 3$, a constant, not equal to $2^n + 1$ in general. Testing $n = 2$: $f(2^2 + 1) = f(5) = 3 \neq 2^2 + 1 = 5$: FALSE.
8. So the TRUE statements are A, B and C.

KEY IDEA

The recurrence generates, block by block (n from 2^k to $2^{k+1} - 1$), the consecutive odd numbers $1, 3, 5, \ldots, 2^k - 1$, giving the closed form $f(n) = 2n - 2^{k+1} + 1$ whenever $2^k \leq n \leq 2^{k+1} - 1$. Once this closed form is established (and proved by induction using the two recurrence cases), every option reduces to identifying which block the given n falls in and substituting.

WATCH OUT FOR

Two common traps: (1) assuming the closed form applies uniformly without first identifying which power-of-two block the argument falls into: e.g. for option D, plugging $n = 2^n + 1$ into the wrong block's formula, or not testing a concrete value, can hide that $f(2^n + 1) = 3$ is constant rather than $2^n + 1$; (2) for option C, misjudging the block of $5 \cdot 2^n$ (it is easy to mistake it for block $k = n + 1$ instead of the correct $k = n + 2$, since $5 \cdot 2^n$ sits just above $4 \cdot 2^n = 2^{n+2}$).

Q.52 MSQ · 2 MARKS

Discrete Mathematics › Graph theory

Which of the properties hold for the adjacency matrix A of a simple undirected unweighted graph having n vertices?

(A) The diagonal entries of A^2 are the degrees of the vertices of the graph.

(B) If the graph is connected, then none of the entries of $A^{n-1} + I_n$ can be zero.

(C) If the sum of all the elements of A is at most $2(n - 1)$, then the graph must be acyclic.

(D) If there is at least a 1 in each of A's rows and columns, then the graph must be connected.

Official answer: **A**

SOLUTION

1. (A) For a simple undirected graph, A is symmetric with entries 0 or 1 and zero diagonal. So $(A^2)_{ii} = \sum_j A_{ij}A_{ji} = \sum_j A_{ij}^2 = \sum_j A_{ij}$, which is the number of neighbours of vertex i, i.e. its degree. True.
2. (B) $(A^{n-1})_{ij}$ counts walks of exactly $n - 1$ edges, not of at most $n - 1$ edges. In a bipartite graph the parity of a walk's length between two vertices is fixed by which sides they lie on.
3. Counterexample for (B): the path $1 - 2 - 3$ has $n = 3$ and is connected. Every walk from 1 to 2 has odd length, so $(A^2)_{12} = 0$. Also $I_{12} = 0$, so $(A^2 + I_3)_{12} = 0$. A zero entry exists, so (B) is False.
4. (C) The sum of all entries of A is $2|E|$, so the condition says $|E| \leq n - 1$. This does not force acyclicity: a triangle plus one isolated vertex has $n = 4$ and $|E| = 3 \leq 3$, yet it contains a cycle. False.
5. (D) A 1 in every row means no vertex is isolated. It does not imply connectivity: two disjoint edges on 4 vertices have no zero row or column, but the graph has 2 components. False.
6. Only (A) holds.

KEY IDEA

$(A^k)_{ij}$ counts walks of exactly k edges. $(A^2)_{ii}$ counts closed walks of length 2, which are the edges at i, so it equals the degree. The other three statements each fail on a small graph.

WATCH OUT FOR

Reading $(A^{n-1}+I)$ as "walks of at most $n-1$ edges" and so concluding that connected means no zero entries. A^{n-1} only counts walks of exactly $n-1$ edges, so parity can leave zeros (for example in a path). Another slip is treating "at most $n-1$ edges" or "no isolated vertex" as enough for acyclic or connected.

Q.53 MSQ · 2 MARKS

Engineering Mathematics › Linear algebra

Which of the following is/are the eigenvector(s) for the matrix given below?

$$\begin{bmatrix} -9 & -6 & -2 & -4 \\ -8 & -6 & -3 & -1 \\ 20 & 15 & 8 & 5 \\ 32 & 21 & 7 & 12 \end{bmatrix}$$

(A) $\begin{bmatrix} -1 \\ 1 \\ 0 \\ 1 \end{bmatrix}$

(B) $\begin{bmatrix} 1 \\ 0 \\ -1 \\ 0 \end{bmatrix}$

(C) $\begin{bmatrix} -1 \\ 0 \\ 2 \\ 2 \end{bmatrix}$

(D) $\begin{bmatrix} 0 \\ 1 \\ -3 \\ 0 \end{bmatrix}$

Official answer: **A, C, D**

SOLUTION

1. A nonzero vector v is an eigenvector of A exactly when $Av = \lambda v$ for some scalar λ. With only four candidates, multiply each by A and check whether the product is a scalar multiple of the candidate.
2. (A) $v = (-1, 1, 0, 1)^T$. Row 1: $9 - 6 + 0 - 4 = -1$. Row 2: $8 - 6 + 0 - 1 = 1$. Row 3: $-20 + 15 + 0 + 5 = 0$. Row 4: $-32 + 21 + 0 + 12 = 1$. So $Av = (-1, 1, 0, 1)^T = 1 \cdot v$, and v is an eigenvector with $\lambda = 1$.
3. (B) $v = (1, 0, -1, 0)^T$. Row 1: $-9 + 2 = -7$. Row 2: $-8 + 3 = -5$. The first two entries of Av are -7 and -5. Since v has second entry 0, a multiple λv would need second entry 0, but it is -5. So v is not an eigenvector.
4. (C) $v = (-1, 0, 2, 2)^T$. Row 1: $9 + 0 - 4 - 8 = -3$. Row 2: $8 + 0 - 6 - 2 = 0$. Row 3: $-20 + 0 + 16 + 10 = 6$. Row 4: $-32 + 0 + 14 + 24 = 6$. So $Av = (-3, 0, 6, 6)^T = 3v$, and v is an eigenvector with $\lambda = 3$.
5. (D) $v = (0, 1, -3, 0)^T$. Row 1: $-6 + 6 = 0$. Row 2: $-6 + 9 = 3$. Row 3: $15 - 24 = -9$. Row 4: $21 - 21 = 0$. So $Av = (0, 3, -9, 0)^T = 3v$, and v is an eigenvector with $\lambda = 3$.
6. Conclusion: options A, C and D are eigenvectors (for $\lambda = 1, 3, 3$) and option B is not, so the answer is A, C, D.

KEY IDEA

To test whether a given vector is an eigenvector, compute Av and check that it is a scalar multiple of v. There is no need to find the characteristic polynomial when the candidates are listed.

WATCH OUT FOR

Stopping after one correct option, although this is a select-all question. Another slip is to treat a vector as an eigenvector because some entries of Av are proportional to it, as happens in B (first entry -7 against 1). Every entry, including the zero entries of v, must scale by the same factor.

Q.54 MSQ · 2 MARKS

Computer Organization and Architecture › Memory & cache

Consider a system with 2 KB direct mapped data cache with a block size of 64 bytes. The system has a physical address space of 64 KB and a word length of 16 bits. During the execution of a program, four data words P, Q, R, and S are accessed in that order 10 times (i.e., PQRSPQRS...). Hence, there are 40 accesses to data cache altogether. Assume that the data cache is initially empty and no other data words are accessed by the program. The addresses of the first bytes of P, Q, R, and S are 0xA248, 0xC28A, 0xCA8A, and 0xA262, respectively. For the execution of the above program, which of the following statements is/are TRUE with respect to the data cache?

(A) Every access to S is a hit.

(B) Once P is brought to the cache it is never evicted.

(C) At the end of the execution only R and S reside in the cache.

(D) Every access to R evicts Q from the cache.

Official answer: **A, B, D**

SOLUTION

1. Cache size = 2 KB, block size = 64 bytes ⇒ number of blocks = 2048/64 = 32 ⇒ index needs $\log_2 32 = 5$ bits and offset needs $\log_2 64 = 6$ bits. Physical address space = 64 KB ⇒ address width = 16 bits ⇒ tag = 16 − 5 − 6 = 5 bits. So a 16-bit address splits as tag(5) | index(5) | offset(6).
2. Write each byte address as 16-bit binary and split into tag/index/offset:
$$P = \text{0xA248} = 1010\,0010\,0100\,1000 \Rightarrow \text{tag} = 20,\ \text{index} = 9,\ \text{offset} = 8$$
$$Q = \text{0xC28A} = 1100\,0010\,1000\,1010 \Rightarrow \text{tag} = 24,\ \text{index} = 10,\ \text{offset} = 10$$
$$R = \text{0xCA8A} = 1100\,1010\,1000\,1010 \Rightarrow \text{tag} = 25,\ \text{index} = 10,\ \text{offset} = 10$$
$$S = \text{0xA262} = 1010\,0010\,0110\,0010 \Rightarrow \text{tag} = 20,\ \text{index} = 9,\ \text{offset} = 34$$
3. P and S have the SAME (tag, index) = (20, 9). Since the block size is 64 bytes, offsets 8 and 34 both lie inside the same 64-byte block, so P and S are always brought into the cache together as one line: whichever of P, S is accessed first loads a block that already contains the other's data.
4. Q and R have the SAME index = 10 but DIFFERENT tags (24 vs 25), so they occupy the same cache set and permanently conflict with each other every time both are accessed in a period.
5. Trace period 1 (P, Q, R, S): P → compulsory miss, loads the tag-20 block into set 9 (this block covers offset 8 for P and offset 34 for S). Q → compulsory miss, loads tag-24 block into set 10. R → tag mismatch in set 10 (holds tag 24, needs tag 25) → miss, evicts Q, loads R's block. S → set 9 already holds tag 20 (loaded for P) → HIT, even on this very first access to S.
6. Trace every subsequent period (2nd through 10th), P, Q, R, S: P → set 9 was never touched by Q/R traffic (different index) → HIT. Q → set 10 currently holds R (tag 25) → miss, evicts R, loads Q (tag 24). R → set 10 currently holds Q (tag 24) → miss, evicts Q, loads R (tag 25). S → set 9 still holds the tag-20 block from P → HIT.
7. (A) Every one of the 10 accesses to S is a HIT (true even in period 1, since P's load already brought in S's block first) → TRUE.

8. (B) Set 9 (holding P's/S's block) is never disturbed by any Q or R access (they only ever touch set 10), so once P's block is loaded it is never evicted → TRUE.
9. (C) At the end of execution, set 9 holds the tag-20 block, which contains BOTH P's and S's data (same 64-byte line), and set 10 holds R (the last one loaded in the final period). So the resident data is P, R and S together, not 'only R and S': the claim that P is absent is false → FALSE.
10. (D) In every one of the 10 periods, Q is loaded into set 10 immediately before R is accessed, and R always misses (tag 25 ≠ tag 24) and evicts whatever is in set 10 at that moment, which is always Q → TRUE for all 10 accesses to R.

KEY IDEA

In a direct-mapped cache, only the (tag, index) pair of an address matters for placement; the offset just locates the byte within a fixed-size block. Here P and S land in the same (tag, index), so they always share one cache line, while Q and R share an index but differ in tag, so they permanently thrash each other's line every time both are accessed in a period.

WATCH OUT FOR

Treating P and S as independent cache entries because their offsets (8 vs 34) look far apart, instead of checking that both offsets fit inside the same 64-byte block once (tag, index) match. This leads students to wrongly mark S's first access as a miss and to wrongly believe P gets evicted, when in fact P and S occupy one shared, never-touched cache line while only Q and R's shared index thrashes.

Q.55 MSQ · 2 MARKS

Computer Networks › IPv4

Consider routing table of an organization's router shown below:

Subnet Number	Subnet Mask	Next Hop
12.20.164.0	255.255.252.0	$R1$
12.20.170.0	255.255.254.0	$R2$
12.20.168.0	255.255.254.0	Interface 0
12.20.166.0	255.255.254.0	Interface 1
default		$R3$

Which of the following prefixes in CIDR notation can be collectively used to correctly aggregate all of the subnets in the routing table?

(A) 12.20.164.0/20

(B) 12.20.164.0/22

(C) 12.20.164.0/21

(D) 12.20.168.0/22

Official answer: **B, D**

SOLUTION

1. Convert each mask to a prefix length. Mask 255.255.252.0 has third octet $252 = 11111100_2$, so it is /22. Mask 255.255.254.0 has third octet $254 = 11111110_2$, so it is /23.
2. Find the third-octet range of each subnet. 12.20.164.0/22 covers 164 to 167. 12.20.170.0/23 covers 170 to 171. 12.20.168.0/23 covers 168 to 169. 12.20.166.0/23 covers 166 to 167.
3. 12.20.166.0/23 lies inside 12.20.164.0/22, so it adds nothing new. The union of all the subnets is 12.20.164.0 to 12.20.171.255, that is third-octet values 164 to 171, eight values in all.
4. A /22 block holds 4 third-octet values and starts at a multiple of 4. So 12.20.164.0/22 covers 164 to 167, and 12.20.168.0/22 covers 168 to 171 (168 is a multiple of 4, so the block is aligned).
5. These two /22 prefixes together cover exactly 164 to 171, with nothing extra. So B and D, used together, aggregate all the subnets correctly.
6. Check the others. 12.20.164.0/21 is the aligned block 12.20.160.0/21, which covers 160 to 167. It misses 168 to 171, and it also covers 160 to 163, which are not in the table. It is not correct.

7. 12.20.164.0/20 is the aligned block 12.20.160.0/20, which covers 160 to 175. It includes every subnet, but it also covers 160 to 163 and 172 to 175, which are not in the table. Those addresses would be wrongly sent along this route instead of the default route R3. So it is not a correct aggregation.

KEY IDEA

Convert each mask to a prefix length, write each subnet as a range of third-octet values, and take the union (164 to 171). Choose the prefixes whose aligned blocks cover that union exactly, with no extra addresses.

WATCH OUT FOR

Picking the single prefix /20 because it contains all four subnets. It also covers 160 to 163 and 172 to 175, which are not in the table, so traffic for those addresses would be sent along this route instead of to the default route R3. Another slip is reading 12.20.164.0/21 as starting at 164, when the aligned /21 block starts at 160.

Q.56 NAT · 2 MARKS

Database Management and Warehousing › SQL

Q.56 Consider the relational database with the following four schemas and their respective instances.

Student(sNo, sName, dNo) Dept(dNo, dName)

Course(cNo, cName, dNo) Register(sNo, cNo)

Student		
sNo	sName	dNo
S01	James	D01
S02	Rocky	D01
S03	Jackson	D02
S04	Jane	D01
S05	Milli	D02

Dept	
dNo	dName
D01	CSE
D02	EEE

Course		
cNo	cName	dNo
C11	DS	D01
C12	OS	D01
C21	DE	D02
C22	PT	D02
C23	CV	D03

Register	
sNo	cNo
S01	C11
S01	C12
S02	C11
S03	C21
S03	C22
S03	C 23
S04	C11
S04	C12
S05	C11
S05	C21

SQL Query:

```
SELECT * FROM Student AS S WHERE NOT EXIST
 (SELECT cNo FROM Course WHERE dNo = "D01"
     EXCEPT
  SELECT cNo FROM Register WHERE sNo = S.sNo)
```

The number of rows returned by the above SQL query is_____.

Official answer: **2**

SOLUTION

1. The outer query keeps a student S only when the subquery returns no rows. The subquery is $\{\text{courses of D01}\} - \{\text{courses taken by } S\}$, and it must be empty. This is the division pattern.
2. Courses with dNo = "D01" are $\{C11, C12\}$. C23 belongs to D03, so it does not count.
3. So a student is kept exactly when both C11 and C12 appear in that student's registrations.

4.

sNo	Courses registered	$\{C11, C12\}$ − registered
$S01$	$C11, C12$	$\emptyset$
$S02$	$C11$	$\{C12\}$
$S03$	$C21, C22, C23$	$\{C11, C12\}$
$S04$	$C11, C12$	$\emptyset$
$S05$	$C11, C21$	$\{C12\}$

5. The difference is empty only for S01 and S04, so the query returns 2 rows.

KEY IDEA

NOT EXISTS (A EXCEPT B) means $A \subseteq B$. Here A is the set of D01 courses, $\{C11, C12\}$, and B is the set of courses the student registered for. So the query returns the students who registered for every D01 course (relational division).

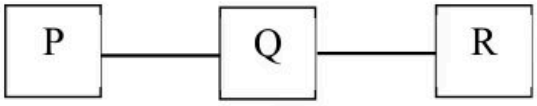
WATCH OUT FOR

Counting C23 (which belongs to D03) as a D01 course, or counting students by their own department. Another slip is reading the query as 'students who registered for any D01 course', which would give 4 rows.

Q.57 NAT · 2 MARKS

Computer Networks › Routing

Q.57 Consider a network with three routers P, Q, R shown in the figure below. All the links have cost of unity.



The routers exchange distance vector routing information and have converged on the routing tables, after which the link Q–R fails. Assume that P and Q send out routing updates at random times, each at the same average rate. The probability of a routing loop formation (*rounded off to one decimal place*) between P and Q, leading to count-to-infinity problem, is _____.

Official answer: **0.5**

SOLUTION

1. Before the failure the tables have converged. Q reaches R directly at cost 1. P reaches R through Q at cost 2.
2. The link Q-R fails. Q detects this and sets its distance to R to ∞. P has not heard yet and still holds the old entry: R at cost 2 via Q.
3. Only the order of the next two routing updates matters: whether Q's update reaches P first, or P's update reaches Q first.
4. Case 1: Q sends its update first. Q advertises R at ∞. P's route to R goes through Q, so P must follow Q and sets R to ∞. The tables agree and no loop forms.
5. Case 2: P sends its update first, advertising R at cost 2. Q has no route to R, so it accepts, because $2 + 1 = 3 < \infty$, and sets R to cost 3 via P. Q now sends R-bound packets to P, and P sends them back to Q. This is a routing loop. When Q next advertises cost 3, P changes its cost to 4, and so on, so the cost climbs without bound (count to infinity).
6. P and Q send updates at random times at the same average rate, so by symmetry each is equally likely to send first: $P(\text{P first}) = \frac{1}{2}$.
7. A loop forms exactly in Case 2, so $P(\text{loop}) = \frac{1}{2} = 0.5$.

KEY IDEA
After Q-R fails, a loop forms only if P advertises its stale route to R (cost 2 via Q) to Q before Q advertises that R is unreachable to P. With equal update rates each order is equally likely, so the probability is 1/2.

WATCH OUT FOR
Assuming a loop always forms (answer 1), or that it never forms because of split horizon. Here P's stale route goes through Q, so the order of the two updates decides the outcome.

Q.58 NAT · 2 MARKS

Discrete Mathematics › Graph theory

Let $G(V, E)$ be a directed graph, where $V = \{1, 2, 3, 4, 5\}$ is the set of vertices and E is the set of directed edges, as defined by the following adjacency matrix A.

$$A[i][j] = \begin{cases} 1, & 1 \le j \le i \le 5 \\ 0, & \text{otherwise} \end{cases}$$

$A[i][j] = 1$ indicates a directed edge from node i to node j. A directed spanning tree of G, rooted at $r \in V$, is defined as a subgraph T of G such that the undirected version of T is a tree, and T contains a directed path from r to every other vertex in V. The number of such directed spanning trees rooted at vertex 5 is _____.

Official answer: **24**

SOLUTION

1. Read the matrix: $A[i][j] = 1$ exactly when $j \le i$. So there is an edge $i \to j$ for every pair with $j < i$, plus a self-loop at every vertex. A self-loop can never lie in a tree, so ignore the loops.
2. Edges therefore go only from a larger-numbered vertex to a smaller one. Vertex v has incoming edges exactly from the vertices u with $u > v$. Vertex 5 has no incoming edge at all.
3. A spanning tree on 5 vertices has exactly 4 edges. Each vertex $v \neq 5$ must be reached by a directed path from 5, so the last edge of that path enters v. That gives each of the 4 non-root vertices at least one incoming tree edge. With only 4 edges in total, each non-root vertex has exactly one incoming tree edge and the root has none.
4. Conversely, pick any one parent $p(v) > v$ for each $v \in \{1, 2, 3, 4\}$. This gives 4 edges. Starting at any v and repeatedly moving to its parent, the label strictly increases, so no cycle can form and the walk must stop at a vertex with no parent, which is 5. Reversing the walk gives a directed path from 5 to v. The graph is therefore connected with 4 edges on 5 vertices, so its undirected version is a tree. Every such choice is a valid directed spanning tree rooted at 5.
5. Count the choices of parent. Vertex 4 may choose from $\{5\}$, so 1 way. Vertex 3 from $\{4, 5\}$, so 2 ways. Vertex 2 from $\{3, 4, 5\}$, so 3 ways. Vertex 1 from $\{2, 3, 4, 5\}$, so 4 ways.
6. The choices are independent, so the number of trees is

$$1 \times 2 \times 3 \times 4 = 4! = 24.$$

KEY IDEA
A directed spanning tree rooted at r is the same as choosing exactly one incoming edge (a parent) for every non-root vertex, provided the choices create no cycle. Here all edges point from larger to smaller labels, so no cycle is possible and the choices are free.

WATCH OUT FOR
Counting self-loops as available parents, which gives an extra factor for each vertex. Another slip is using the matrix-tree theorem on the undirected version, or forgetting that edges only go from larger to smaller labels, so vertex v has $5 - v$ possible parents and not 4.

Q.59 NAT · 2 MARKS

Computer Networks › Performance

Consider a 100 Mbps link between an earth station (sender) and a satellite (receiver) at an altitude of 2100 km. The signal propagates at a speed of 3×10^8 m/s. The time taken (in milliseconds, rounded off to two

decimal places) for the receiver to completely receive a packet of 1000 bytes transmitted by the sender is _____.

Official answer: **7.07 to 7.09**

SOLUTION

1. The receiver has the whole packet only when the last bit arrives. The last bit leaves the sender after one transmission time and then travels for one propagation time, so $T = t_{\text{trans}} + t_{\text{prop}}$.
2. Transmission time: the packet is $1000 \times 8 = 8000$ bits and the link rate is $100\ \text{Mbps} = 100 \times 10^6$ bits per second, so $t_{\text{trans}} = \dfrac{8000}{100 \times 10^6} = 8 \times 10^{-5}\ \text{s} = 0.08\ \text{ms}$.
3. Propagation time: the distance is the altitude, $2100\ \text{km} = 2.1 \times 10^6\ \text{m}$, and the speed is $3 \times 10^8\ \text{m/s}$, so $t_{\text{prop}} = \dfrac{2.1 \times 10^6}{3 \times 10^8} = 7 \times 10^{-3}\ \text{s} = 7\ \text{ms}$.
4. Total: $T = 0.08 + 7 = 7.08\ \text{ms}$.

KEY IDEA
The time to receive a whole packet is the transmission time (packet bits divided by link rate) plus the propagation time (distance divided by signal speed). The last bit arrives one propagation delay after it has been sent.

WATCH OUT FOR
Giving only the propagation time (7 ms) and forgetting the transmission time. Another slip is to use 1000 bits instead of 8000, or to count a round trip.

Q.60 NAT · 2 MARKS

Computer Networks › TCP

Consider the data transfer using TCP over a 1 Gbps link. Assuming that the maximum segment lifetime (MSL) is set to 60 seconds, the minimum number of bits required for the sequence number field of the TCP header, to prevent the sequence number space from wrapping around during the MSL is _____.

Official answer: **33**

SOLUTION

1. TCP numbers every byte, so the sequence number space must be at least as large as the number of bytes the sender can transmit within one MSL. Otherwise a sequence number could repeat while an old segment with that number is still alive in the network.
2. Link rate $= 1\ \text{Gbps} = 10^9$ bits/s $= \dfrac{10^9}{8} = 1.25 \times 10^8$ bytes/s.
3. Bytes sent in one MSL $= 1.25 \times 10^8 \times 60 = 7.5 \times 10^9$ bytes.
4. We need $2^n \geq 7.5 \times 10^9$.
5. $$2^{32} = 4.29 \times 10^9 < 7.5 \times 10^9 \leq 8.59 \times 10^9 = 2^{33}$$
6. So the minimum number of bits is $n = 33$.

KEY IDEA
TCP numbers bytes, not segments. The sequence number space must cover every byte the link can send in one MSL, so $2^n \geq \text{rate in bytes/s} \times \text{MSL}$.

WATCH OUT FOR
Using the rate in bits per second ($10^9 \times 60 = 6 \times 10^{10}$, which gives 36 bits). TCP sequence numbers count bytes, so the rate must first be divided by 8.

Q.61 NAT · 2 MARKS

Computer Organization and Architecture › Pipelining

A processor X_1 operating at 2 GHz has a standard 5-stage RISC instruction pipeline having a base CPI (cycles per instruction) of one without any pipeline hazards. For a given program P that has 30% branch instructions, control hazards incur 2 cycles stall for every branch. A new version of the processor X_2 operating at same clock frequency has an additional branch predictor unit (BPU) that completely eliminates

stalls for correctly predicted branches. There is neither any savings nor any additional stalls for wrong predictions. There are no structural hazards and data hazards for X_1 and X_2. If the BPU has a prediction accuracy of 80%, the speed up (rounded off to two decimal places) obtained by X_2 over X_1 in executing P is ___.

Official answer: **1.42 to 1.45**

SOLUTION

1. Compute the CPI of X_1. Without a BPU, every branch (30% of instructions) stalls for 2 cycles, so:
$$\text{CPI}(X_1) = 1 + 0.30 \times 2 = 1.6$$
2. Model the branch outcomes on X_2. The BPU gives 0 stall cycles for a correctly predicted branch, and the problem states there is "neither any savings nor any additional stalls for wrong predictions", i.e. a mispredicted branch stalls exactly as it did on X_1, for 2 cycles. With 80% prediction accuracy, the expected stall per branch is:
$$0.8 \times 0 + 0.2 \times 2 = 0.4 \text{ cycles}$$
3. Compute the CPI of X_2, applying that expected stall only to the 30% branch fraction:
$$\text{CPI}(X_2) = 1 + 0.30 \times 0.4 = 1.12$$
4. Both processors run at the same clock frequency and execute the same program P (same instruction count N), so execution time is proportional to CPI alone:
$$T = N \times \text{CPI} \times \text{Clock Period}$$
Hence:
$$\text{Speedup} = \frac{T(X_1)}{T(X_2)} = \frac{\text{CPI}(X_1)}{\text{CPI}(X_2)} = \frac{1.6}{1.12}$$
5. Evaluate and round to two decimal places:
$$\frac{1.6}{1.12} = \frac{160}{112} = \frac{10}{7} = 1.42857\ldots \approx 1.43$$

KEY IDEA

Speedup with the same clock frequency and same program reduces to a ratio of CPIs; build each CPI as (base CPI) + (branch fraction) × (expected stall cycles per branch), where the expected stall on the predictor-equipped processor is a probability-weighted average over the correct/incorrect prediction outcomes.

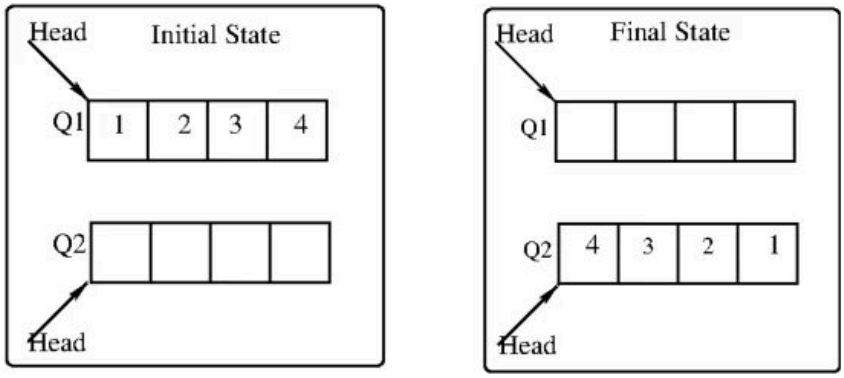
WATCH OUT FOR

Assuming the BPU also improves the misprediction penalty (e.g. treating it as 0 or as some reduced value), when the problem explicitly says a wrong prediction gets neither savings nor extra stall, i.e. it costs the same 2 cycles it always did. Another common slip is applying the 2-cycle/0.4-cycle stall to 100% of instructions instead of scaling by the 30% branch fraction, or forgetting that clock frequency being equal lets CPI ratios stand in directly for the speedup.

Q.62 NAT · 2 MARKS

Data Structures and Algorithms › Linear structures

Q.62 Consider the queues Q_1 containing four elements and Q_2 containing none (shown as the Initial State in the figure). The only operations allowed on these two queues are `Enqueue(Q,element)` and `Dequeue(Q)`. The minimum number of `Enqueue` operations on Q_1 required to place the elements of Q_1 in Q_2 in reverse order (shown as the Final State in the figure) without using any additional storage is_____.



Official answer: **0**

SOLUTION

1. The question counts only Enqueue operations performed on Q_1. Enqueue operations on Q_2 and all Dequeue operations are free as far as the count is concerned. So we look for a method that never enqueues anything back into Q_1.
2. Required final state: Q_2 read from head to tail is $4, 3, 2, 1$. Q_1 gives up its elements only in the order $1, 2, 3, 4$ (head first), so each element has to be moved into place inside Q_2 as it arrives. To do this we rotate Q_2: Dequeue its head and at once Enqueue that element back onto Q_2. The element is held for only one step, just as in any transfer between the two queues, so no additional storage is used.
3. Move 1: Dequeue (Q_1) gives 1, then Enqueue (Q_2, 1). Now $Q_2 = [1]$. No rotation is needed.
4. Move 2: Dequeue (Q_1) gives 2, then Enqueue (Q_2, 2), so $Q_2 = [1, 2]$. Rotate Q_2 once (take 1 off the head and put it at the tail): $Q_2 = [2, 1]$.
5. Move 3: Dequeue (Q_1) gives 3, then Enqueue (Q_2, 3), so $Q_2 = [2, 1, 3]$. Rotate Q_2 twice (move 2, then 1, to the tail): $Q_2 = [3, 2, 1]$.
6. Move 4: Dequeue (Q_1) gives 4, then Enqueue (Q_2, 4), so $Q_2 = [3, 2, 1, 4]$. Rotate Q_2 three times (move 3, 2, 1 to the tail): $Q_2 = [4, 3, 2, 1]$, which is the Final State. Q_1 is empty.
7. Count: every Enqueue above was on Q_2: 4 transfers plus $0 + 1 + 2 + 3 = 6$ rotations, 10 in total. The number of Enqueue operations on Q_1 is 0.
8. A count cannot be negative, so 0 is the minimum. The minimum number of Enqueue operations on Q_1 is $\mathbf{0}$.

KEY IDEA

Only Enqueue operations on Q_1 are counted. Move each element of Q_1 straight into Q_2, then rotate Q_2 (Dequeue from Q_2, Enqueue back onto Q_2) until the new element reaches the head. All of the reordering happens inside Q_2, so Q_1 is never enqueued into and the minimum is 0.

WATCH OUT FOR

Doing the reordering by rotating Q_1 to bring 4, then 3, then 2 to its head. That takes $3 + 2 + 1 = 6$ Enqueues on Q_1 and gives 6. The rotations can be done on Q_2 instead, where they do not count. A second mistake is counting every Enqueue on both queues (10) when the question asks only about Q_1.

Q.63 NAT · 2 MARKS

Operating Systems › File systems & disks

Consider two files systems A and B, that use contiguous allocation and linked allocation, respectively. A file of size 100 blocks is already stored in A and also in B. Now, consider inserting a new block in the middle of the file (between 50th and 51st block), whose data is already available in the memory. Assume that there are enough free blocks at the end of the file and that the file control blocks are already in memory. Let the number of disk accesses required to insert a block in the middle of the file in A and B are n_A and n_B, respectively, then the value of $n_A + n_B$ is _____.

Official answer: **153**

SOLUTION

1. **File A (contiguous).** To open a gap after block 50, blocks 51 to 100 must each move one position towards the end: that is $100 - 50 = 50$ blocks. There are enough free blocks at the end, so block 101 is available.
2. Each moved block costs one read and one write, so $50 \times 2 = 100$ accesses. Move from the last block backwards, so nothing is overwritten before it has been copied.
3. Writing the new block (already in memory) into the gap costs 1 access. So $n_A = 100 + 1 = 101$.
4. **File B (linked).** There is no random access, so block 50 is reached by following pointers from the start. Reading blocks 1 to 50 costs 50 accesses (the file control block is in memory, so the first block's address is free).
5. Write the new block, with its pointer set to the old block 51: 1 access.
6. Block 50 is already in memory, so rewrite it with its pointer changed to the new block: 1 access. So $n_B = 50 + 1 + 1 = 52$.
7. $n_A + n_B = 101 + 52 = 153$.

KEY IDEA

Contiguous allocation must physically shift every block after the insertion point (a read and a write each), while linked allocation must walk to the insertion point and then change only two pointers.

WATCH OUT FOR

Forgetting that a shifted block costs two accesses (read and write), or forgetting to rewrite block 50 with the new pointer in the linked file.

Q.64 NAT · 2 MARKS

Operating Systems › Virtual memory

Consider a demand paging system with four page frames (initially empty) and LRU page replacement policy. For the following page reference string

7, 2, 7, 3, 2, 5, 3, 4, 6, 7, 7, 1, 5, 6, 1

the page fault rate, defined as the ratio of number of page faults to the number of memory accesses (rounded off to one decimal place) is _____.

Official answer: **0.6**

SOLUTION

1. There are 4 frames and 15 references. Simulate LRU, evicting the page whose last use is oldest.
2.

Ref	Result	Evicted	Frames after
7	fault	–	7
2	fault	–	7, 2
7	hit	–	7, 2
3	fault	–	7, 2, 3
2	hit	–	7, 2, 3
5	fault	–	7, 2, 3, 5
3	hit	–	7, 2, 3, 5

3. Reference 4 (position 8): the frames are full. Last uses are 7 at position 3, 2 at 5, 5 at 6 and 3 at 7, so 7 is least recently used. Fault, evict 7. Frames: 2,3,5,4.
4. Reference 6 (position 9): last uses are 2 at 5, 3 at 7, 5 at 6 and 4 at 8, so evict 2. Fault. Frames: 6,3,5,4.
5. Reference 7 (position 10): last uses are 3 at 7, 5 at 6, 4 at 8 and 6 at 9, so evict 5. Fault. Frames: 6,3,7,4. The next 7 (position 11) is a hit.
6. Reference 1 (position 12): last uses are 3 at 7, 4 at 8, 6 at 9 and 7 at 11, so evict 3. Fault. Frames: 6,1,7,4.
7. Reference 5 (position 13): last uses are 4 at 8, 6 at 9, 7 at 11 and 1 at 12, so evict 4. Fault. Frames: 6,1,7,5. The last two references, 6 and 1, are hits.
8. Faults occur on 7, 2, 3, 5, 4, 6, 7, 1, 5, which is 9 faults in 15 accesses.
9. $$\text{Page fault rate} = \frac{9}{15} = 0.6$$

KEY IDEA
Under LRU, on a fault with full frames evict the page whose most recent use is oldest. Count faults over all 15 accesses and divide by 15.

WATCH OUT FOR
Counting only faults after the frames fill, or evicting by first arrival (FIFO) instead of by last use. Another slip is dividing by the number of distinct pages instead of the 15 accesses.

Q.65 NAT · 2 MARKS

Compiler Design › Syntax-directed translation

Consider the following grammar along with translation rules.

$S \rightarrow S_1 \# T \quad \{S.\text{val} = S_1.\text{val} * T.\text{val}\}$
$S \rightarrow T \quad \{S.\text{val} = T.\text{val}\}$
$T \rightarrow T_1 \,\%\, R \quad \{T.\text{val} = T_1.\text{val} \div R.\text{val}\}$
$T \rightarrow R \quad \{T.\text{val} = R.\text{val}\}$
$R \rightarrow id \quad \{R.\text{val} = id.\text{val}\}$

Here # and % are operators and id is a token that represents an integer and $id.\text{val}$ represents the corresponding integer value. The set of non-terminals is $\{S, T, R, P\}$ and a subscripted non-terminal indicates an instance of the non-terminal.

Using this translation scheme, the computed value of $S.\text{val}$ for root of the parse tree for the expression 20#10%5#8%2%2 is _____.

Official answer: **80**

SOLUTION

1. The grammar is left-recursive in both $S \to S_1 \# T$ and $T \to T_1 \% R$, so both # and % are left-associative. % is generated inside T, which sits below S, so % binds tighter than #.
2. Group the input by #: $20 \# (10\%5) \# (8\%2\%2)$.
3. First T is $T \to R$ with $R \to id$, so $T.\text{val} = 20$.
4. Second T uses $T \to T_1 \% R$: $T.\text{val} = 10 \div 5 = 2$.
5. Third T is left-associative, so it is $(8 \div 2) \div 2 = 4 \div 2 = 2$.
6. S is left-recursive, so $S.\text{val} = (20 * 2) * 2$.
7. Hence $S.\text{val} = 40 * 2 = 80$.

KEY IDEA

The grammar fixes the parse. A left-recursive rule makes its operator left-associative, and an operator generated lower in the grammar (% in T) binds tighter than one generated higher up (# in S). Parse the string first, then evaluate each node bottom-up.

WATCH OUT FOR

Using the usual meaning of `#` and `%`, or evaluating strictly left to right. That computes $((((20 \cdot 10) \div 5) \cdot 8) \div 2) \div 2 = 80$ here by coincidence, but the operations are not in that order. Another slip is grouping $8\%2\%2$ as $8 \div (2 \div 2) = 8$ instead of $(8 \div 2) \div 2 = 2$.

FROM IAE ACADEMY

Preparing for GATE CS?

IAE Academy teaches GATE CS in Ballygunge, Kolkata, and online. Counselling is free: ask us about batches, fees or your preparation.

FREE COUNSELLING

Chat on WhatsApp

Call +91 84204 36524

Open every day, 10:30 am to 7:30 pm.

Scan to chat on WhatsApp

THE GATE CS COURSE

Classroom batches in Ballygunge and online batches. Fees and details are on the course page.

Scan: course page

See the course

FREE ON IAEACADEMY.IN

More GATE CS papers, question by question with the official answers, and free notes by subject.

Scan: more papers

More solved papers

Found this useful? Send it to a friend preparing for GATE.

Send to a friend on WhatsApp

IAE Academy free study material. The questions are those of the GATE CS 2022 question paper, organised by IIT Kharagpur; the answers are from its official answer key. Prepared and published by IAE Academy, Kolkata. Free to read and free to share, unaltered and with attribution. Not affiliated with, endorsed by or produced by IIT Kharagpur or the GATE committee.

IAE ACADEMY INSTITUTE FOR ENGINEERS

GATE coaching in Kolkata since 2012 · EC, EE, IN, CS and IT, DA, BT and CE · in the classroom and online

227A Rash Behari Ave, Ballygunge, Kolkata 700019

+91 84204 36524 · iaeacademy.in · Updated 27 October 2026

Get directions