GATE

Subject : CS 2006 - SOLUTIONS

(Q. NO. 1 – 20) 1 MARKS

1. According to Homer's rule

$$\begin{aligned} P(x) &= a_0 + a_1x + a_2x^2 + a_3x^3 \\ &= a_0 + x\left(a_1 + a_2x + a_3x^2\right) \\ &= a_0 + x\left(a_1 + x\left(a_2 + a_3x\right)\right) \end{aligned}$$

So we need 3 multiplications as follows :

(1) $p = a_3 \times x$

(2) $q = (a_2 + p) \times x$

(3) $r = (a_1 + q) \times x$

(A) is the answer.

2. $|X| = x$

$|Y| = y$

$|Z| = z$

$W = X \times Y$

$\therefore |W| = xy$

E is the set of all subsets of W

$\therefore |E| = 2^{xy}$

Number of functions from a set of m elements to a set of n elements are n^m

$\therefore$ Number of functions from Z to E = $(2^{xy})^z$

$= 2^{xyz}$

(D) is the answer.

3.

X_{10}	1	2	3	5	7	8	9
1	1	2	3	5	7	8	9
2	2	4	6	0	4	6	8
3	3	6	9	5	1	4	7
5	5	0	5	5	5	0	5
7	7	4	1	5	9	6	3
8	8	6	4	0	6	4	2
9	9	8	7	5	3	2	1

(A) is true as it is not closed because we are getting the elements 0,4,6 which do not belong to the given set.

(B) is true. 2 does not have an inverse because there is no such element x such that

$(x \times 2) \bmod 10 = e$

For multiplication, e = 1

(C) is false. Inverse of 3 is 7

$(3\times7) \bmod 10 = 1$

i.e. $(3\times(3)^{-1}) \bmod 10 = e$

so, 3 has an inverse which is 7.

(D) is true because there is no such element x such that

$(x \times 8) \bmod 10 = 1$

(C) is the answer.

4. R is defined as $(x, y)R(u, v)$ if $x < u$ and $y > v$

R is not reflexive as we cannot have $(x, y)R(x, y)$ because $x < x$ and $y < y$ both are false.

- So, R is not an equivalence relation as equivalence relation should be reflexive, symmetric and transitive.
- Also R is not a partial order set (POSET) as POSET should be reflexive, anti-symmetric and transitive.
- R is also not a TOSET (Total Order Set) because for a relation to be TOSET it should satisfy the properties of POSET as well as comparable property.

So, (A) is the answer.

5. TTL (Time to live) field is used in the IP datagram header to prevent the packets from looping indefinitely in the network.

The TTL value is decremented by 1 for every hop of the packet. When the value of TTL reaches 0, the packet cannot hop further. Thus, looping is avoided in the network.

(C) is the answer

6.

PROCESS	A.T.	B. T.
A	0	10
B	2	20
C	6	30

Using SRTF,

A	A	B	C

0 2 10 30 60

Context switches are nothing but switching between two processes.

Here, we have only two context swithches – from A to B and from B to C.

(B) is the answer.

7. For making LR(0) item sets, we need to first augment the given grammar with the production rule $S' \to S$. The augmented grammar is $S' \to S$

$S' \to S$

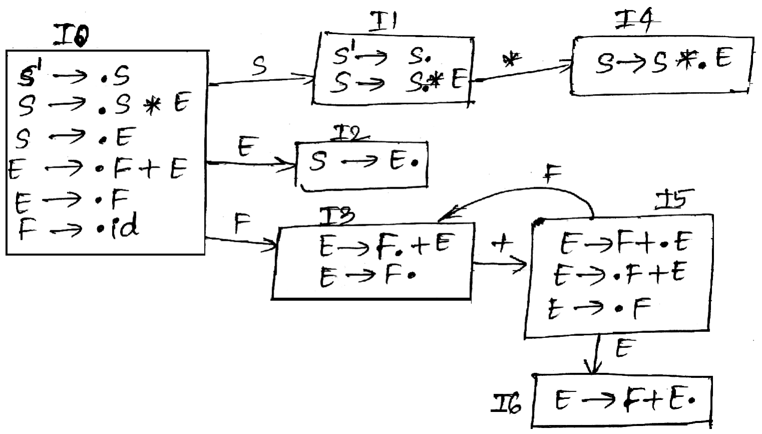
$S \to S * E$

$S \to E$

$E \to F + E$

$E \to F$

$\mathrm{F} \to \mathrm{id}$



From the LR(0) item sets constructed so far, we can see that the given items belong to the sets I4, I3 and I5 respectively i.e. none of them appear in the same set

(D) is the answer.

8. (A) gives output with the same phase as f.

(B) is inverting f.

In (C) the output is activated by $\overline{\text{CLK}}$. Also the CLK has 50% duty cycle, i.e. the output will have a phase difference of 180°.

(D) is also inverting f.

(C) is the answer.

9. Instruction size = 24 bits

$$= \frac{24}{8} \text{ bytes}$$

$$= 3 \text{ bytes}$$

One instruction takes 3 bytes i.e. the program counter jumps 3 bytes to fetch the next instruction address. So, the program the counter value must be divisible by 3.

Of the given options, only 600 is divisible by 3 and hence it is a legal program counter value.

(C) is the answer.

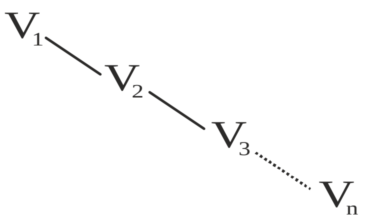
10. In a max heap, the smallest element is always present at a leaf node. So we need to check all the leaf nodes to find the minimum value.

$\therefore$ Worst case time complexity will be O(n)

(A) is the answer.

11. Minimum spanning tree of such a graph will contain adjacent edges of the graph as they have minimum weight.

For n vertices the minimum spanning tree will be



$\therefore$ Weight of the spanning tree

$= 2|2 - 1| + 2|3 - 2| + \ldots 2|n - (n - 1)|$

$= 2 + 2 + 2$ .... $(n - 1)$ times

$= 2(n - 1)$

$= 2n - 2$

(B) is the answer.

12. To find shortest path in an unweighted graph, we can use Breadth First Search (BFS) which runs in O(E +V) time i.e. linear with respect to edges and vertices. BFS uses queue data structure.

(A) is the answer.

13. Since we can store any binary tree, worst case happens for the right skewed binary tree. So the amount of space required by array X is same as the space required for complete binary tree having all its levels filled.

    We know that for a complete binary tree with n levels, we have $2^n - 1$ nodes. So, minimum size of array X is $2^n - 1$. Consider for 3 nodes, if root goes to index 1, its right child will have index 3, further its right child will have index 7. Hence we require 7 memory locations for 3 nodes i.e. $(2^3 - 1)$ space aray.

    (D) is the answer.

14. Selection sort has the minimum number of swaps. In worst case, selection sort performs O(n) swaps because we repeatedly find the minimum element from the unsorted part of the array and place it at the beginning (i.e. the swap) So, even in worst case we will have (n – 1) swaps which is the minimum among all the sorting algorithms mentioned.

    (C) is the answer.

15. $\text{val (j)} = \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \ldots\ldots + \frac{n}{2^k}$

    The loop will terminate when $n = 2^k$ or $k = \log n$

    $$\therefore \text{val(j)} = n\sum_{i=1}^{k} \frac{1}{2^i}$$

    $$= n\left(1 - \frac{1}{2^k}\right)$$

    $$= n - \frac{n}{2^k}$$

    $$= n - 1 \qquad (\text{as } 2^k = n\ )$$

    $\therefore$ val (j) = $\theta(n)$

    (C) is the answer.

16. Since Q and R are not known to be in NP so Q and R cannot be NP-complete (as NP-complete problem has to be in both NP and NP-hard)

    $\therefore$ (A) and (C) are false.

    - A problem H is NP–hard when every problem L in NP can be reduced in polynomial time to H. It is given that S is polynomial time reducible to R, and S is an NP-complete problem

    i.e. S is in NP

    $\therefore$ R is NP-hard

    (B) is true

    (D) is false because there is no NP problem that is polynomical time reducible to Q.

    (B) is the answer.

17. All the leaders in an array can be found in O(n) time using a right to left pass of the array.

We can start from the rightmost element keeping a track of the maximum elements. (The rightmost element is always a leader). Now if an element is greater that the maximum element so far, it will also be considered as a leader. Hence, we can find all the leaders in an array in linear time by using right to left pass of the array.

(B) is the answer.

18. The probability of selection of each element in the set X is independent with probability = $\frac{1}{2}$

Now, probability that X_1 is smallest $= \frac{1}{2}$

Probability that X_2 is smallest = probability that X_1 is not small and X_2 is smallest

$$= \left(1 - \frac{1}{2}\right) \times \frac{1}{2}$$

$$= \left(\frac{1}{2}\right)^2$$

Similarly, probability that X_n is smallest = $\left(\frac{1}{2}\right)^n$

$\therefore$ Required expectation = $\sum_{i=1}^{n} 2^i \times \left(\frac{1}{2}\right)^i$

$$= 2 \times \frac{1}{2} + 2^2 \times \frac{1}{2^2} + ...2^n \times \frac{1}{2^n}$$

$= 1 + 1 + 1............+1$ (n times)

= n

(D) is the answer

19. L_1 is context free. Initially push all the 0s and then for all 1s followed by 0s, pop all the initially pushed 0s.

L_2 is not context free. We cannot ensure the count of 0s followed by 1s to be less than the count of previous 0s as well as 1s using a single stack.

L_3 is not context free. It is same as $0^p1^p0^p$ where p = n + m, which is a CSL but not a CFL.
(D) is the answer.

20. Note that here we aren't using checkpoints. A checkpoint writes the current in–memory modified pages (known as dirty pages) and transaction log information from memory to disk, and also records information about the transaction log.

When a system crashes, we do the following :

1. Use two lists of transaction maintained by the system , the committed transactions since the last checkpoint and the active transactions.
2. Undo all the write-item operations of the active (uncommitted) transaction, using the UNDO procedure. The operations should be undone in reverse of the order in which they were written into the log.
3. Redo all the write-item operations of the committed transactions from the log, in the order in which they were written into the log.

   Here, the system crashes before log record 7 is written. So, we must undo log record 6 as T2 is active uncommitted transaction. And we must redo log records 2 and 3 as T1 is committed.

(B) is the answer

(Q. NO. 21 – 85) 2 MARKS

21. We have to find the probability of n heads among 2n coin tosses.

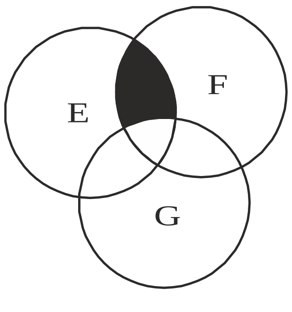
Required probability $= {}^{2n}C_n \left(\frac{1}{2}\right)^n \left(\frac{1}{2}\right)^n$

$= {}^{2n}C_n \left(\frac{1}{2}\right)^{2n}$

$= {}^{2n}C_n \frac{1}{4^n}$

(A) is the answer.

22. Using Venn diagram, both X and Y give the same region as below.



$\therefore$ X = Y

(C) is the answer.

23. Since it is given that Fu = b and Fv = b,

so we get

Fu = Fv

$\therefore$ Fu – Fv = 0

$\therefore$ F(u–v) = 0

Also it is given that $u \neq v$, so F = 0 i.e. F is a singular matrix. So F_x = b will either have no solution or infinite solution. But as we are already given two solutions u and v for x, so F_x = b must have infinitely many solutions.
So (A), (B) and (C) are true. (D) is false because it may not be necessary the two rows are identical, instead two columns can be identical and even then we can get F as a singluar matrix.
(D) is the answer.

24. Total number of permutations = n!
Consider N = { 1, 2, 3, 4}
π = {4, 3, 1, 2}
A = { 1, 2, 3} and B = {1, 3}
π (A) = {2, 1, 3}
π (B) = {3, 1}
Here, min (π(A)) = min (π(B)) = 1

Now, we can choose permutations for $A \cup B$ in ${}^{n}C_{|A \cup B|}$ ways.

To do the mapping, we can choose any element from $A \cup B$ such that min (π (A)) = min (π (B)) and since one minimum is fixed, we have $|A \cup B\text{-}1|!$ ways of permutation

We can permute the remaning elements out of n in $\left(n - |A \cup B\text{-}1|\right)!$ ways

$\therefore$ Total permutation = ${}^{n}C_{|A \cup B|} * |A \cap B| * |A \cup B - 1|! * \left(n - |A \cup B - 1|\right)!$

$$= n!\frac{A \cap B}{A \cup B}$$

(C) is the answer.

25. Total elements in S = m
Total number of subsets of size 3 = ${}^{m}C_3$.
Now, consider an element in the set S, suppose 2.
Number of subsets in which the element 2 won't be there = ${}^{(m-1)}C_3$
$\therefore$ The element 2 will be there in ${}^{m}C_3 - {}^{(m-1)}C_3$ subsets

$$= \frac{m!}{(m-3)!3!} - \frac{(m-1)!}{3!(m-1-3)!}$$

$$= \frac{(m-1)!}{(m-4)!3!}\left[\frac{m}{m-3} - 1\right]$$

$$= \frac{(m-1)}{(m-4)!3!} \times \frac{3}{(m-3)}$$

$$= \frac{(m-1)(m-2)(m-3)}{6} \times \frac{3}{(m-3)}$$

$$= \frac{(m-1)(m-2)}{2}$$

$$\sum_{i=1}^{m} f(i) = \sum_{i=1}^{m} \frac{(m-1)(m-2)}{2}$$

$$\frac{m(m-1)(m-2)}{2}$$

Now, we know that ${}^{m}C_3 = n$

i.e. $\frac{m(m-1)(m-2)}{6} = n$

$$\therefore \frac{m(m-1)(m-2)}{2} = 3n$$

(D) is the answer.

26. The given statement means that if an animal is either tiger or lion, then if it is hungry or threatened, it will attack.

Since it is given that "Tigers and lions" it does not translate to tiger (x) $\wedge$ Lion (x) because there is no such animal x which is both tiger as well as lion.

So the given statement can be written in first order predicate calculus as :

$$\forall x\Big[\big(\text{tiger}(x) \vee \text{lion}(x)\big) \rightarrow \big\{\big(\text{hungry}(x) \vee \text{threatened}(x)\big) \rightarrow \text{attacks}(x)\big\}\Big]$$

(D) is the answer

27. For P1,

$$\text{LHS} = \big((A \wedge B) \rightarrow C\big)$$
$$= \overline{AB} + C$$
$$= \overline{A} + \overline{B} + C$$

$$\text{RHS} = (A \rightarrow C) \wedge (B \rightarrow C)$$
$$= \big(\overline{A} + C\big)\big(\overline{B} + C\big)$$
$$= \overline{A}\,\overline{B} + \overline{A}C + \overline{B}C + C$$
$$= \overline{A}\,\overline{B} + C\,\big(\overline{A} + \overline{B} + 1\big)$$
$$= \overline{A}\,\overline{B} + C$$

$\therefore$ LHS $\neq$ RHS

So, P1 is not a tautology.

For P2,

LHS $= ((A \vee B) \to C)$

$= \overline{(A+B)} + C$

$= \overline{A}\,\overline{B} + C$

RHS $= (A \to C) \vee (B \to C)$

$= \overline{A} + C + \overline{B} + C$

$= \overline{A} + \overline{B} + C$

$\therefore$ LHS $\neq$ RHS

So, P2 is also not a tautology.

(D) is the answer

28. A□B is same as $B \to A$ i.e. $\sim B \vee A$. Now, to get $A \wedge B$, we check individually for each of the options

(A) $\sim A \,\square\, B$ $= B \to \sim A$

$= \sim B \vee \sim A$

$\neq A \wedge B$

(B) $\sim (A \,\square \sim B)$ $= \sim (\sim B \to A)$

$= \sim (B \vee A)$

$= \sim A \wedge \sim B$

$\neq A \wedge B$

(C) $\sim (\sim A \,\square \sim B)$ $= \sim (\sim B \to \sim A)$

$= \sim (B \vee \sim A)$

$= A \wedge \sim B$

$\neq A \wedge B$

(D) $\sim (\sim A \,\square \sim B)$ $= \sim (B \to \sim A)$

$= \sim (\sim B \vee \sim A)$

$= A \wedge B$

$\therefore \sim (\sim A \square B)$ is equivalent to $(A \wedge B)$

(D) is the answer

29. (A) is regular as there are finite three digit prime numbers and any finite set is regular.

    (B) is also a regular language. We need 6 states to recognize L.

    1. #0 - #1 = 0
    2. #0 - #1 = 1
    3. #0 - #1 = 2
    4. #0 - #1 = -1
    5. #0 - #1 = -2
    6. Dead state if the difference between number of 0's and 1's in the prefixes is greater than 2 or less than -2.

    (C) is not a regular language. Here we need to maintain the count of 0's and 1's which is not possible in a finite automata.

    (D) is a regular language. Its DFA has $5 \times 7 = 35$ states.

    (C) is the answer.

30. Let L1 = d(s) mod 5 and L2 = d(s) mod 7

    L1 is regular with its DFA having 5 states.

    L2 is regular with its DFA having 7 states.

    We have to find $L = L1 \cap L2'$ which is regular as regular languages are closed under complementation and intersection.

    So, L is regular

    (D) is the answer

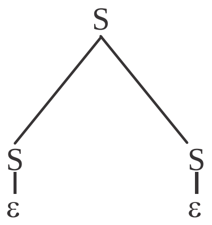
31. The problem of finding whether there exists a Hamiltonian cycle or not in a graph is NP-complete (i.e. it belongs to NP as well as NP-hard)

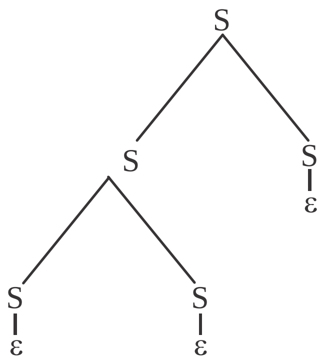
    The problem of finding a Hamiltonian cycle in a graph G = (V,E) with V divisible by 3 is also NP-complete.

    $\therefore$ Bothe DHAM$_3$ and SHAM$_3$ are in NP as well as NP hard.

    (A) is the answer

32. I is true as we have two different parse trees for the string 'ε' itself.





II is false. It does not produce the strings of the type aabb i.e. equal number of a's followed by equal number of b's.

III is true. The given grammar G generates the language (ab+ba)*, which is regular. So, it is also a DCFL and we can have a DPDA for the language accepted by G.

(B) is the answer

33. (A) is true as DCFLs are closed under intersection with regular language.

(B) is false. L_1 is a regular language and L_3 is recursively enumerable $L_1 \cap L_3$ is REL but not recursive.

(C) is true. L_1 is regular and hence a CFL. L_2 is a DCFL, hence a CFL and CFLs are closed under union.

(D) is true. L_1 is regular and hence REL. L_2 is a DCFL and hence a REL. L_3 is REL.

$\therefore$ $L_1 \cap L_2 \cap L_3$ is also a REL as recursively enumerable languages are closed under intersection.

(B) is the answer.

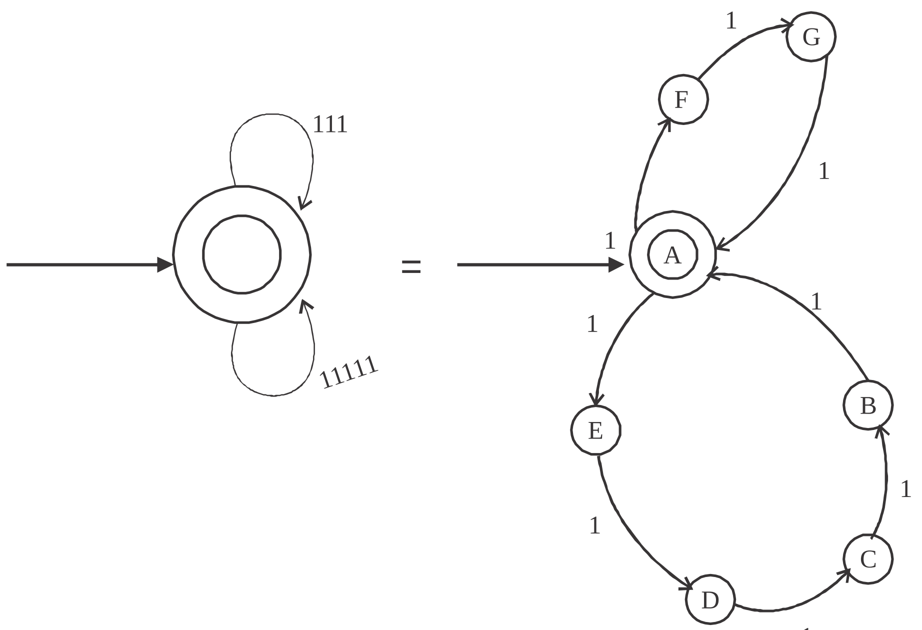
34. We have to design a DFA for the language

L=(111 + 11111)*

It is same as (a+b)*

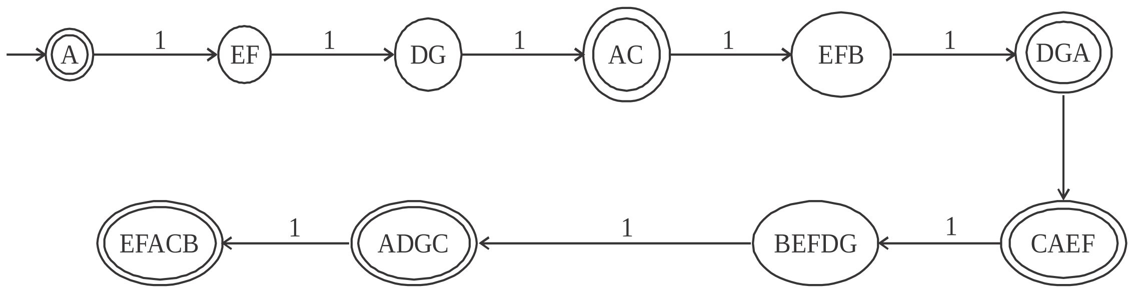
Here a = 111 and b =11111

$\therefore$ FA is



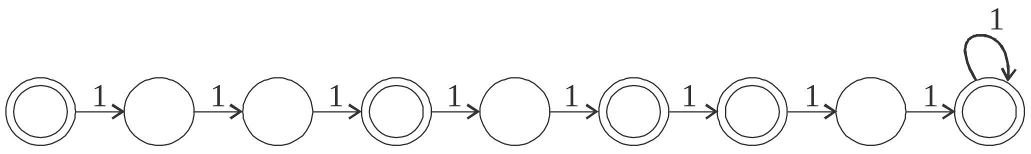
But this is a NFA and we need a DFA

$\therefore$ DFA is as follows :



However, we can combine the last two states into a single state as there is no back transition

$\therefore$ Minimized DFA is :



#states in minimum DFA = 9

(D) is the answer

35. Let the output of the upper MUX be f1

Now,

$$f = \bar{y}f1 + yx$$
$$= \bar{y}\left(\bar{z}x + z\bar{y}\right) + xy$$
$$= x\bar{y}\bar{z} + \bar{y}z + xy$$
$$= \bar{y}\left(x\bar{z} + z\right) + xy$$
$$= \bar{y}\left(z + x\right) + xy$$
$$\therefore f = \bar{y}z + \bar{y}x + xy$$

(A) is the answer.

36. For a carry look ahead adder, we have two functions-carry generate G (A,B) and carry propagate P (A, B)

G(A, B) = A.B

P(A, B) = A + B

The carry C_{i+1} is given as

$C_{i+1} = G_i + P_iC_i$

$\therefore C_3 = a_2b_2 + a_1b_1 (a_2 + b_2) + a_0b_0 (a_1 + b_1) (a_2 + b_2)$

$\therefore C_3 = a_2b_2 + a_1a_2b_1 + a_1b_2b_1 + a_0b_0 (a_1a_2 + a_1b_2 + b_1a_2 + b_1b_2)$

$\therefore C_3 = a_2b_2 + a_1a_2b_1 + a_1b_2b_1 + a_2a_1a_0b_0 + a_1a_0b_2b_0 + a_2a_0b_1b_0 + a_0b_2b_1b_0$
(A) is the answer.

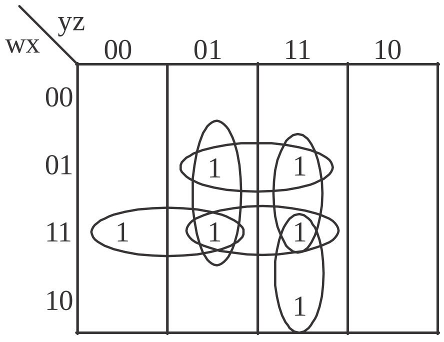
37. Analyzing the given circuit,

$q_{1(new)} = q_{0(old)} \oplus q_{2(old)}$
$q_{2(new)} = q_{1(old)}$
$q_{0(new)} = \text{data}$
initially, $q_0 = q_1 = q_2 = 0$

Clock Cycle	Data	q_0	q_1	q_2
1	1	1	0	0
2	0	0	1	0
3	0	0	0	1
4	1	1	1	0
5	1	1	1	1
6	0	0	0	1
7	0	0	1	0
8	0	0	0	1
9	0	0	1	0

At the end of ninth clock cycle, values of $q_2\, q_1\, q_0$ are 010
(C) is the answer

38. We know that adjacent cells in K-map differ in exactly 1 bit position.
And it is given that the function remains true as input changes from i_1 to i_2 where i_1 and i_2 are input vectors that differ in exactly 1 bit position. So we group the cells in K- map as below :
f (w,x,y,z) = $\sum$ (5,7,11,12,13,15)



So, we get six pairs
$\therefore$ f (w,x,y,z) = $\overline{w}xz + wxz + x\overline{y}z + xyz + wx\overline{y} + wyz$
(D) is the answer

39. Overflow is detected if :
(1) both operands are positive and result is negative
(2) both operands are negative and result is positive
(3) Overflow is relevant only for signed numbers and we use carry for unsigned numbers

(4) when the carry out bit and the carry in to the most significant differs

The question is actually asking for the condition of overflow of signed numbers when we use an adder which is meant to work for unsigned numbers.

Option (B) is correct as it takes care of both case (1) as well as case(2)
The rest options don't work for all the cases.
(B) is the answer

40

Decimal n	Binary n	H(x) = gray(n)	G(x) = gray[(n+1) mod 16]
0	0000	0000	0001
1	0001	0001	0011
2	0010	0011	0010
3	0011	0010	0110
4	0100	0110	0111
5	0101	0111	0101
6	0110	0101	0100
7	0111	0100	1100
8	1000	1100	1101
9	1001	1101	1111
10	1010	1111	1110
11	1011	1110	1010
12	1100	1010	1011
13	1101	1011	1001
14	1110	1001	1000
15	1111	1000	0000

Now to determine min terms for g_3, g_2, g_1 and g_0 check for corresponding 1s in the columns
$g_3\,(h_3,h_2,h_1,h_0) = \sum(4,12,13,15,14,10,11,9)$
$g_2\,(h_3,h_2,h_1,h_0) = \sum(2,6,7,5,4,12,13,15)$
$g_1\,(h_3,h_2,h_1,h_0) = \sum(1,3,2,6,13,15,14,10)$
$g_0\,(h_3,h_2,h_1,h_0) = \sum(0,1,6,7,12,13,10,11)$
$\therefore$ from the given options, only g_2 is given correctly.
(C) is the answer

41. Cache block size = 64 B
The main memory has 24 banks, each of which is 2 bytes wide
In one iteration, we get 24*2 = 48B
$\therefore$ To access 64B, we require 2 iterations

Time taken for one parallel access (i.e. 1 iteration) (T) = decoding time + latency time

$$= \frac{k}{2} + 2$$
$$= 12 + 80$$
$$= 92\,\text{ns}$$

$\therefore$ Time taken for 2 iterations $= 2 \times 92$
$= 184\text{ns}$

(D) is the answer

42. Clock time $= \dfrac{1}{\text{clock frequency}}$

$= \dfrac{1}{10^9\,\text{Hz}}$

$= 1\,\text{nsec}$

Total #instructions = 10^9

20% out of 10^9 are conditional branches

Stall cycles = branch frequency $\times$ branch penally
$= 0.2 \times (3-1)$
$= 0.4$

Total execution time = (1+stall cycles) $\times$ cycle time
$= (1+0.4) \times 10^{-9}$
$= 1.4$ nsec

$\therefore$ Execution time for 10^9 instructions $= 1.4 \times 10^{-9} \times 10^9$ s
$= 1.4$ sec

(C) is the anwer

43. Here we have to find if the bit is set or not at the position "pos" in given register "reg". So, we perform the AND operation of that bit with 1.

If the result is 1, the bit is set else the bit is not set.

Now if the "pos" is not 0, then we need to left shift 1 by pos i.e. 0 X 1<< pos.

(A) is the answer

44. Round trip time (RTT) is the length of time it takes for a signal to be sent plus the length of time it takes for an acknowledgment of that signal to be received.

For optimal window size, we need maximum utilization i.e. the sender should have used full bandwidth during this time.

$\therefore$ Number of packets $= \dfrac{\text{BW} \times \text{RTT}}{\text{Packet size}}$

$= \dfrac{128\text{Kbps} \times 80\text{ms}}{32 \times 8\text{b}}$

$= 40$

(B) is the answer

45. Subnet mask of C1 = 255.255.128.0

Network id of C1 :

$$\begin{array}{r} 203.197.2.53 \\ \wedge\, 255.255.128.0 \\ \hline 203.197.0.0 \end{array}$$

and Network id of C2 :

$$\begin{array}{r} 203.197.75.201 \\ \wedge\ \underline{255.255.128.0} \\ 203.197.0.0 \end{array}$$

$\therefore$ Both belong to same network

Subnet mask for C2 = 255.255.192.0

$\therefore$ Network id of C1

$$\begin{array}{r} 203.197.2.53 \\ \wedge\ \underline{255.255.192.0} \\ 203.197.0.0 \end{array}$$

Network id of C2

$$\begin{array}{r} 203.197.75.201 \\ \wedge\ \underline{255.255.192.0} \\ 203.197.64.0 \end{array}$$

So they are on different networks.

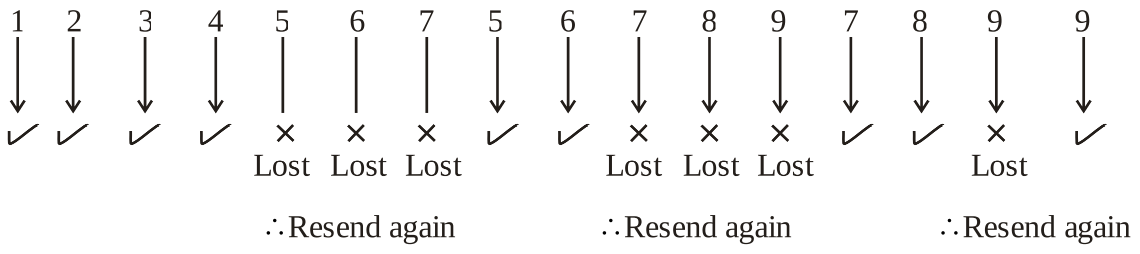
(C) is the correct answer

46. Using Go-Back-n error control strategy,

    # packets to be transferred = 9

    Window size = 3

    Every 5th packet is lost



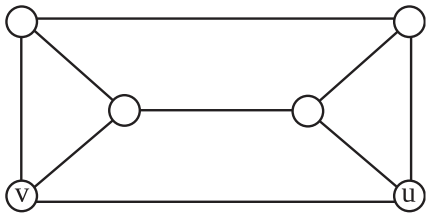
So if we count the total number of transmissions (successful + lost), we get 16 total transmissions.

(C) is the answer

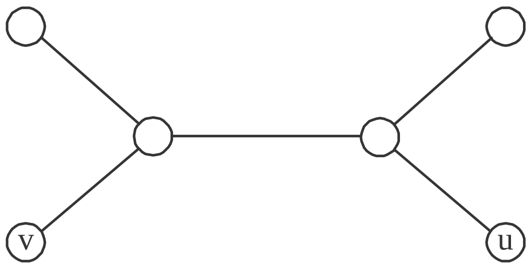
47. In Kruskal's algorithm, the edges are added in non decreasing order of their weights. But in option (D), edge d-e (weight = 3) is added before edge d-c (weight = 2) which is not possible in Kruskal's algorithm.

    (D) is the answer

48. Consider the graph



DFS tree is :



This proves that (A), (B) and (C) are false :

Also there is a cycle in G containing u and all its neigbours

(D) is the answer

49. • Here the best case will be when insert and delete operations are performed alternatively. For each delete operation, 2 pop and 1 push operations are performed. And for each insert operation, 1 push is perfomed.

$\therefore$ Total push operations = m + n [as we have m delete and n insert operations]

Total pop operations = 2m

• The worst case occurs when first n elements are inserted and then m elements are deleted. For first delete operation, we will require (n+1) pop operations. Other than the first, in all delete operations, 1 pop operations is performed.

$\therefore$ Total pop operations = m + n

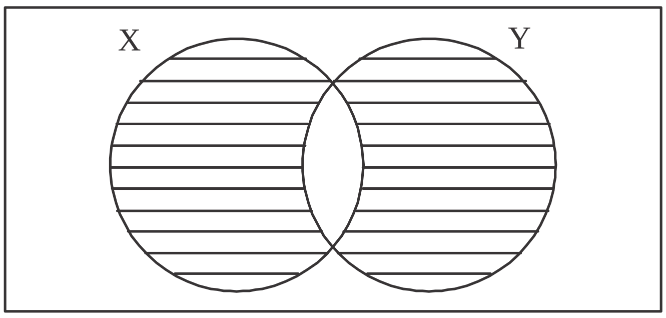
Total push operations = 2n

(A) is the answer.

50. The z is complete as:

$Z = \left(X \cap \overline{Y}\right) \cap (\overline{X} \cap Y)$

Using set theroy we get Z as :



which is $(X \cup Y) - (X \cap Y)$

This is same as option (D)

$(X - Y) \cup (Y - X)$ gives the same shaded portion.

(D) is the answer.

51. $T(n) = 2T\left(\lceil \sqrt{n} \rceil\right) + 1$

$T(1) = 1$

Consider

$T(n) = 2T\left(\sqrt{n}\right) + 1$

Let $n = 2^m$

$T(2^m) = 2T(2^{m/2}) + 1$

i.e. $S(m) = 2S\left(\frac{m}{2}\right) + 1$

Using Master's Theorem,

$a = 2,\ b = 2,\ p = 0,\ k = 0$

$a = 2 \qquad b^k = 2^0 = 1$

$\therefore a > b^k$

$\text{So}, S(m) = \theta\left(m^{\log_b a}\right)$

$= \theta\left(m^{\log_2 2}\right)$

$= \theta(m)$

But we know $m = \log n$ as $n = 2^m$

$\therefore T(n) = \theta(\log n)$

(B) is the answer

52. When median is chosen as a pivot, it divides the given set of elements into two equal halves. So the resultant recurrence relation is :

$T(n) = 2T\left(\frac{n}{2}\right) + \theta(n)$

On solving this, we get

$T(n) = O(n \log n)$

(B) is the answer

53. The given code is merging of two sorted arrays into a single sorted array.

The while loop terminates when either $i = n$ or $j = m$

When either of these conditions is true all the elements of either of the arrays a and b are added to the array c.

When $i = n$, c will contain n elements from array a, but it will also contain j elements from array b

$\therefore$ Array c will have n + j elements

and since k starts from 0, after termination of the while loop, k = n + j

Similarly for j = m, k = m + i

So neither of the given conditions hold

(D) is the answer

54. Since there are total n elements in the array and the array is binary, maximum sum is n for both the arrays. However the difference between sum of two arrays will vary from – n to + n. So there are 2n + 1 possible values difference.

Here, we make use of two arrays of size 2n + 1 that will store starting and ending indices of each possible sum. So, we need to find maximum {ending_index [i] – starting_index [i] } given that both have assigned values.

Thus the given problem can be solved in θ (n) time and space

(C) is the answer.

55. S1 is false as the output will be different in both the cases when j = = i + 2.

S2 is also false as just by adding extra temporary variables, we cannot say anything about performance improvement of the code

(A) is the answer.

56. There won't be any runtime error in the given code. The statements S1 and S4 are correct.

(B) is the answer.

57. S2 is true. Depending on the arguments passed, the code may generate a segmentation fault.

S4 is true. The code works correctly only for some inputs. It does not work if we call swap (x,x) as here we should get the value of x to be swapped but the code returns 0 in both px and py. So the code does not check if both pointers point to the same address or not. $\therefore$ S3 is false

S1 and S5 are also false.

(C) is the answer

58. First (S) = {id}

First (R)={id, ε }

First (F)={id}

Follow(S)={\$}

Follow(R)={\$}

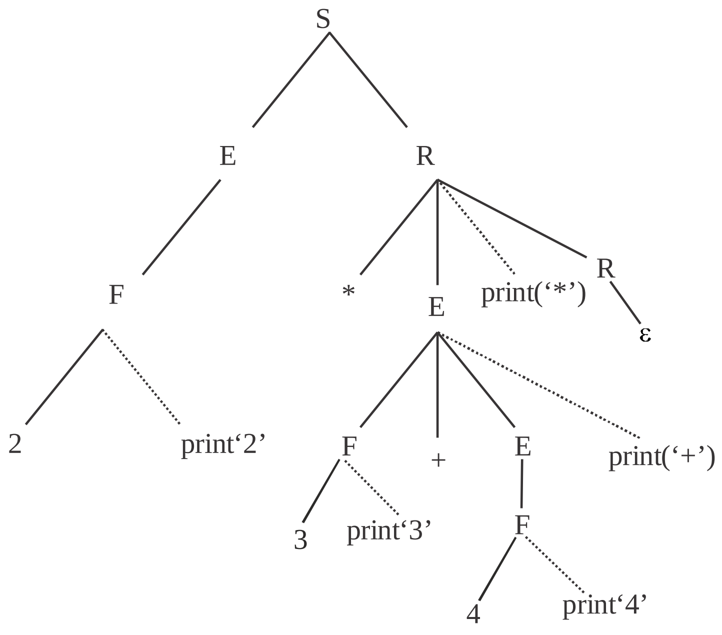
Follow(F)={id,\$}

Construct the predictive parsing table as follows:

	id	$
S	$S \rightarrow FR$	Error
R	$R \rightarrow S$	$R \rightarrow \varepsilon$
F	$F \rightarrow id$	Error

(A) is the answer

59. The input is given as : 2 * 3 + 4



Now, traverse the above tree in DFS order to print the output.

So, the output printed by the given translation scheme is : 2 3 4 + *

(D) is the answer.

60. 4*j is the common sub-expression in the code. So, (B) is true.

i % 2 is the loop invariant for inner loop

5*i is also loop invariant for inner loop

$\therefore$ (A) is true

(C) is also true

Consider the statement

x += (4*j + 5*i);

Here 5*i can be replaced by k (k = 0 initially)

i.e. x += 5*i; becomes

x += k

Thus, replacing * by + results in strength reduction.

So, (D) is the answer

61. (A) is true because if context switching is disabled in P, then the process which is currently blocked may not relinquish control to the process which might eventually execute the V operation.

(A) is the answer.

62. Virtual address = 32 bit

page size (d) = 4KB

$= 2^2 2^{10}$ B

$= 2^{12}$ B

$\therefore$ d = 12 bits

So, p = (32-12) = 20 bits

$$\text{Number of sets} = \frac{128}{4} = 32$$

$$\therefore \text{Set offset} = \log_2 \lceil 32 \rceil = 5\,\text{bits}$$

$$\text{so, tag bits} = 20 - 5 = 15\text{bits}$$

(C) is the answer

63. Memory management unit (MMU) is a computer hardware unit having all memory references passed through itself, primarily performing the translation of virtual memory address to physical memory address.

Thus when we decide to get rid of virtual memory entirely, we do not need any hardware support for MMU.

(C) is the answer

64.

Process	A.T.	B.T.	C.T.	TAT[CT – AT]
P0	0	2	12	12
P1	0	4	13	13
P2	0	8	14	14

Using LRTF, the Gantt chart is :

P2	P1	P2	P1	P2	P0	P1	P2	P0	P1	P2

0 4 5 6 7 8 9 10 11 12 13 14

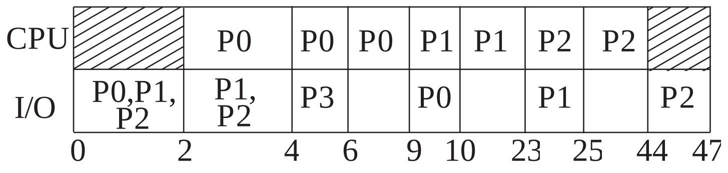
$$\therefore \text{Average TAT} = \frac{39}{3} = 13\text{units}$$

(A) is the answer.

65. Let the tree processes be P0, P1 and P2

Process	BT	t_{io}	t_{cpu}	t_{io}
P0	10	2	7	1
P1	20	4	14	2
P2	30	6	21	3

Using SRTF,



$$\therefore \%\text{time CPU idle} = \frac{2+3}{47} \times 100$$
$$= 10.638\%$$

(B) is the answer

66. The requirments of p and q can be done in ($x_p + x_q$) instances.
Deadlock can be avoded only if
$x_p + x_q >= \min_{k \neq p,q} Y_k$
However this condition only ensures the system is not apporaching a deadlock from the present state and not for all the processes present.
(B) is the answer.

67. Neither Queryl nor Query2 are correct implementations. Consider the case when all the customers have same balance, the rank assigned should be 1. But instead, the Query1 assigns them rank as n (n = number of tuples)
Whereas for the same case, Query2 will return a null set as the condition A.balance < B.balance can never be true so by looking at the options, we can eliminate (A), (B) and (D).
(C) is the answer.

68. Queryl and Query3 will give the same result.
In enrolled table, student is not a key i.e. duplicate values of student might be present which will be retrieved in the final result.
Query2 and Query4 will output the same result.
In paid table, student is a key. So it is guaranteed to give unique values of student.
(B) is the answer

69. Plan1 executes faster than Plan2 as in Plan1, we are selecting amount > x first and then performing nested loop join which will considerably reduce the time needed for query execution, because we are reducing the number of tuples initially by performing the select operation.
However, Plan 2 takes longer time to execute as we are performing nested loop join first and then selecting amount > x, so we need to check for every tuple inside the join which indeed will cause delay in producing the result
(C) is the answer.

70. Consider each of the options.

(A) $\{CF\}^+ = \{CFGEAD\}$

So, (A) is true

(B) $\{BG\}^+ = \{BGACD\}$

$\therefore$ (B) is also true

(C) $\{AF\}^+ = \{AFDE\}$

$\therefore$ (C) is false

(D) $\{AB\}^+=\{ABCDG\}$

(C) is the answer

71. Two vertices are adjacent if and only if the corresponding sets intersect in exactly two elements.

$\therefore$ Number of vertices with zero degree will be same as number of subsets having elements less than or equal to 1.

Number of such subsets = n + 1

For n = 6, the vertices of degree 0 will be

ϕ, 1, 2, 3, 4, 5 and 6 which is 7 in number.

(C) is the answer.

72. Let the vertex having maximum degree contain 'm' elements. For two vertices to be adjacent, we need the corresponding sets to intersect in exactly two elements.

So, 2 elements out of m can be chosen in ${}^mC_2$ ways.

Now these two-element pairs can have an edge to a vertex having (n – m) elements.

So, we will have a total of 2^{n-m} possible subsets (as 2 elements must always be present i.e. the common elements)

So we get maximum degree as ${}^mC_2\, 2^{n-m}$

Now, put different values of m > = 2 or differentiate the expression w.r.t. m and equate it to 0

We get the maximum degree for m = 3

$$\therefore \text{Maximum degree} = {}^3C_2\, 2^{n-3} = 3.2^{n-3}$$

(C) is the answer

73. All the vertices i.e. the subsets of n elements having > = 2 elements will form one connected component. Also, we have found out that the graph will have n + 1 vertices of degree 0. These will form n + 1 connected components.

So, total number of connected components = n + 1 + 1

= n + 2

(B) is the answer

74. For 2- way set associative memory,

tag	set offset	block offset

$\longleftarrow 32 \longrightarrow$

block offset $= \log_2 \lceil 32 \rceil$

$= 5 \text{ bits}$

$\#\text{cache lines} = \dfrac{\text{cache size}}{\text{block size}}$

$= \dfrac{32\text{KB}}{32\text{B}}$

$= 2^{10}$

$\#\text{sets} = \dfrac{\#\text{cache lines}}{\text{P} - \text{way}}$

$= \dfrac{2^{10}}{2}$

$= 2^{9}$

$\therefore \text{ set offset} = \log_2 \lceil 2^9 \rceil$

$= 9 \text{ bits}$

$\therefore \text{Tag bits} = 32 - (5 + 9)$

$= 32 - 14$

$= 18$

For set associative memory, we need to consider latency of multiplexer as well as comparator for calcutating the hit latency.

$$\therefore \quad h_1 = \left(0.6 + \frac{k}{10}\right)\text{ns}$$

Here, we will need 18 - bit comparator as tag is of 18 bits

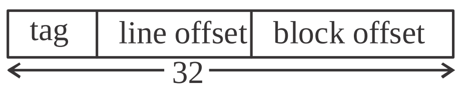
$$\therefore \quad h_1 = \left(0.6 + \frac{18}{10}\right)\text{ns}$$

$= (0.6 + 1.8) \text{ ns}$

$= 2.4 \text{ ns}$

(A) is the answer.

75. For direct mapped cache,



Block size = 32B

$\therefore \text{Block offset} = \log_2 \lceil 32 \rceil$

$= 5 \text{ bits}$

$\#\text{cache lines} = \dfrac{\text{cache size}}{\text{block size}}$

$= \dfrac{32\text{KB}}{32\text{B}}$

$= 2^{10}$

$\therefore \text{ line offset} = \log_2 \lceil 2^{10} \rceil$

$= 10 \text{ bits}$

$\therefore \text{ Tag bits} = 32 - (5 + 10)$

$= 32 - 15$

= 17 bits

For hit latency in direct memory, we need to consider the latency of only tag comparator. And here we need 17 bit tag comparator

$$\therefore\ h_2 = \frac{k}{10}\text{ns}$$

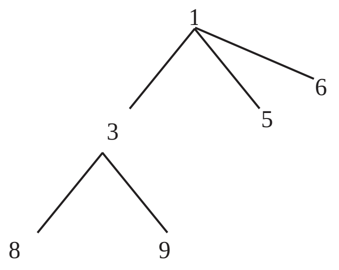
$$= \frac{17}{10}\text{ns}$$

$$= 1.7\text{ ns}$$

(D) is the answer

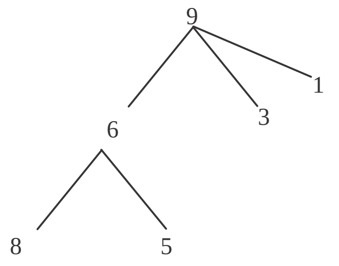
76. For a max heap, every parent $\geq$ its children Here, we are looking for a 3-ary max heap check for each of the options.

(A)



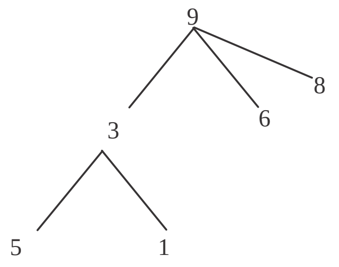
This is a min heap as every parent $\leq$ its children

(B)



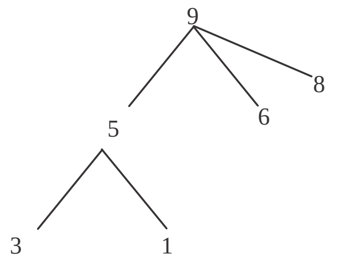
This is not a 3-ary max heap as 8 is a child of 6

(C)



This is not a 3-ary max heap as 5 is a child of 3

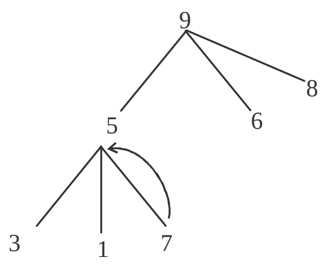
(D)



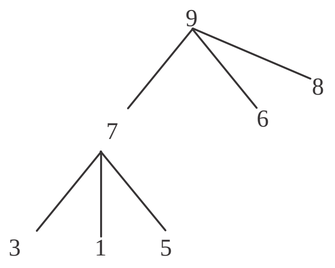
This satisfies the property of a 3-ary max heap and every parent $\geq$ its children

(D) is the answer.

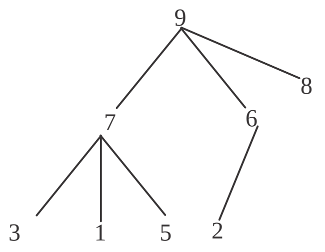
77. We know that heaps are stored in complete binary trees.
After inserting the elements 7, 2, 10 and 4 in that order, the 3 ary max heap will be as follows :
Insert 7 :



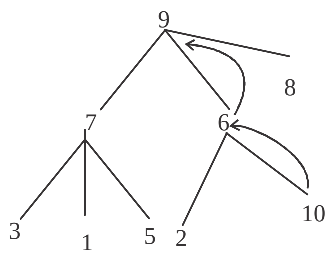
$\xrightarrow[\text{heapify}]{\text{Reverse}}$



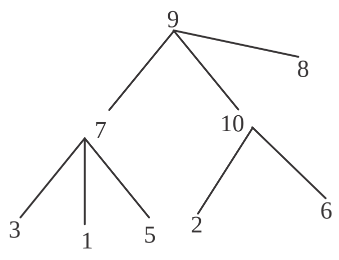
Insert 2 :

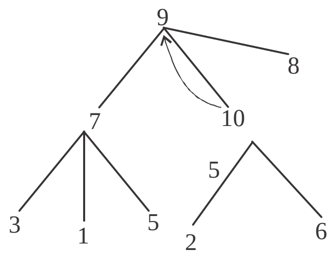


Insert 10

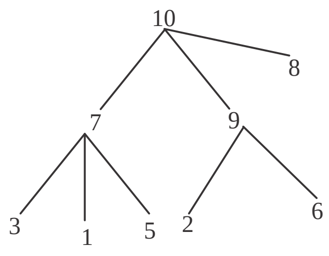


$\xrightarrow[\text{heapify}]{\text{Reverse}}$

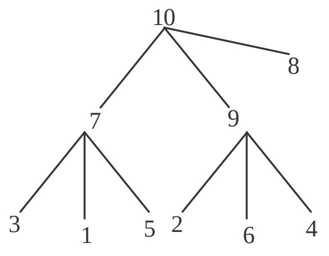




$\xrightarrow[\text{heapify}]{\text{Reverse}}$



Insert 4 :



It satisfies the property of 3-ary max heap

$\therefore$ It is the resultant heap.

Perform the level order traversal of the resultant heap to store it in an array.

$\therefore$ Sequence of elements in the array is :

10, 7, 9, 8, 3, 1, 5, 2, 6, 4

(A) is the answer

78. It is possible that more than three processes arrive and it may happen that process_arrived is always greater that 3 and it cannot become 3 again. In that case, a deadlock will occur.

(B) is the answer

79. We have to ensure that when a process enters the barrier for the second time, step 2 is not executed till other two processes have not executed step 7. This is done so that the variable process_arrived does not become greater than 3 and hence deadlock will be resolved. Only option (B) rectifies the problem in the implementation.

(B) is the answer

80. For P1 the array is accessed in row major order.

$$\#\text{Cache blocks} = \frac{\text{cache size}}{\text{block size}} = \frac{32\text{KB}}{128\text{B}} = 256$$

Each element of the array occupis 8 bytes

$\therefore$ Number of elements in 1 block $= \frac{128}{8}$

$= 216$

So, when A[0][0] is fetched, there won't be a cache miss for another 15 references.

$\therefore$ Number of cache misses for P1 (M_1)

$= \frac{512}{16} \times 512$

$= 32 \times 512$

$= 16384$

$\therefore M_1 = 16384$

(C) is the answer

81. For calculating $\frac{M_1}{M_2}$, we need to find the numbers of misses in P2

we already have calculated $M_1 = 16384$ for the previous question

For code P2, the array is accessed in column major order

Here, every element is a miss because the array is stored in row major order by default and we are trying to access it in column major order.

When A[0][0] is accessed, 128 consecutive bytes are moved to cache. But the next access is

A[1] [0] which is after 512*8 memory locations. But our cache can hold only 256 blocks and so A[0][1] will also result in a miss.

$\therefore$ Number of cache missess for P2 (M_2)

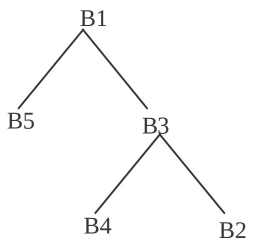
$= 512 \times 512$

$= 2^{18}$

So, $\frac{M1}{M2} = \frac{16384}{2^{18}} = \frac{2^{14}}{2^{18}} = \frac{1}{2^4} = \frac{1}{16}$

(B) is the answer

82. Using all the given constraints, the spanning tree for the bridges is :
DFS traversal of the tree is : B1, B5, B3, B4, B2
(A) is the answer:



83. According to the generated spanning tree, only option (A) matches.
(A) is the answer.

84. The given language is L= $\{a^i b^j \mid i \neq j\}$
$\therefore$ L = {a,b,abb,aab,bb,aaa,abbb,....}
(A) and (C) both generate the string ε which is not accepted by the given language.
(D) is the answer.

85. Consider the string w = abb, l = 1 and m = 2
Derivation :

$$\begin{aligned} S &\longrightarrow CB \\ &\longrightarrow aCbB \\ &\longrightarrow abB \\ &\longrightarrow abb \end{aligned}$$

$\therefore$ Length of the derivation = 4
So, only option (A) satisfies
max(l, m) + 2 = max (1, 2) + 2
= 2 + 2
= 4
which is the length of the derivation.
We can verify by taking examples of other strings generated by the correct grammar.
(A) is the answer.