CLASS TEST

S.No.: 01_JP_CS_26082025

MADE EASY

Leading Institute for IES, GATE & PSUs

Delhi | Bhopal | Hyderabad | Jaipur | Pune | Kolkata

Web: www.madeeasy.in | **E-mail:** info@madeeasy.in | **Ph:** 011-45124612

ALGORITHMS

COMPUTER SCIENCE & IT

Date of Test : 26/08/2025

ANSWER KEY

1.	(b)	7.	(d)	13.	(b)	19.	(a)	25.	(a)
2.	(d)	8.	(a)	14.	(a)	20.	(b)	26.	(c)
3.	(b)	9.	(d)	15.	(b)	21.	(d)	27.	(b)
4.	(a)	10.	(a)	16.	(d)	22.	(b)	28.	(a)
5.	(c)	11.	(c)	17.	(a)	23.	(c)	29.	(c)
6.	(c)	12.	(d)	18.	(d)	24.	(d)	30.	(c)

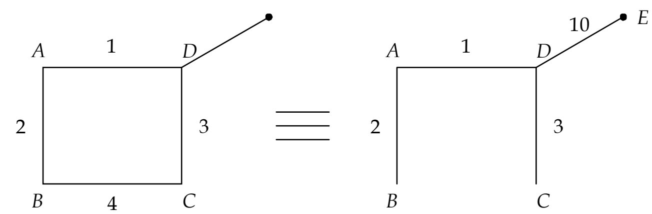
DETAILED EXPLANATIONS

1. **(b)**
I is false, II and III are true because I is not symmetric.

2. **(d)**
Only option (d) is correct.

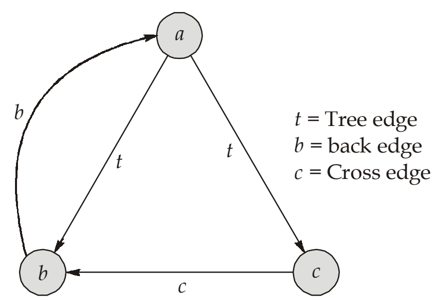
3. **(b)**

4. **(a)**
If the weights are unique, then every minimum spanning tree V contains minimum edge weight. But statement S_2 is false.

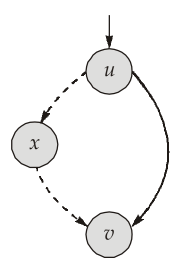


Clearly, maximum edge weight is in MST.

5. **(c)**
Following graph has a 3 types of edges:



Forward edge not possible for given condition. If we assume that a forward edge (u, v) in graph, then this diagram shows this condition.

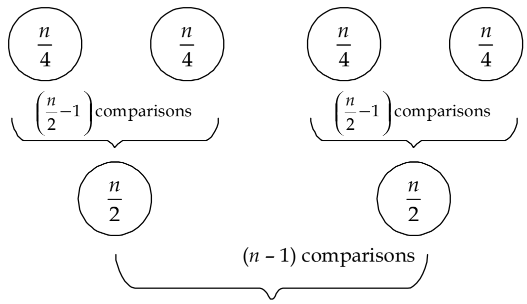


If (u, v) is forward edge then, there is another path from u to v ($u \rightsquigarrow v$), with atleast one intermediate node, then this forward edge (u, v) will actually become tree edge, contradiction, so no forward edge.

6. **(c)**
Longest common subsequence = "*ADH*".
Hence, length of LCS = 3.

7. **(d)**

Let us consider n-elements where each of the 4 list are having $\frac{n}{4}$ elements.



$$\text{Total comparison} = 2\left(\frac{n}{2}-1\right)+n-1 = 2n - 3$$

Here, $n = 4 \times 8 = 32$

So, Total comparison $= 2 \times 32 - 3 = 61$

8. **(a)**

Insertion sort time complexity

Best case $= \theta(n)$

Worst case $= \theta(n^2)$

$$\text{Average case} = \frac{n+n^2}{2} = \theta(n^2)$$

9. **(d)**

- Havel-Hakimi algorithm is used when degree sequence is given.
  There are n vertices. For every vertices we will have to sort the degree sequence for each vertices, i.e., n times.
  So to sort best algorithm takes $0(n \log n)$ times.
  So, for n times it will be $0(n^2 \log n)$.

10. **(a)**

Quick sort for n elements,

Time complexity $\rightarrow O(n^2)$

Quick sort for $n\log n$ elements,

Time complexity $\rightarrow O((n \log n)^2)$

Merge sort: In case of inplace merge takes $O(n^2)$.

Recurrence relation $2T\left(\frac{n}{2}\right)+n^2$

Using Master's theorem for n elements $O(n^2)$

Therefore for $n \log n$ elements $n^2(\log n)^2$

11. **(c)**

For topological order, the node should have 0 unselected incoming edges only then it is picked for ordering.

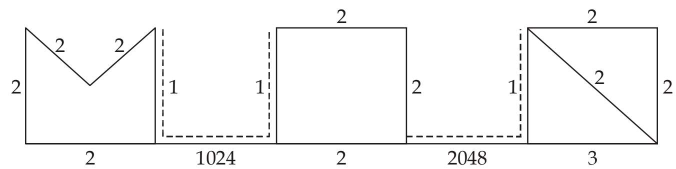
Node 8 cannot be written just after model because of above reason.

(a) → 8, 4 incorrect

(b) → 10, 1 incorrect

(d) → 10, 2 incorrect

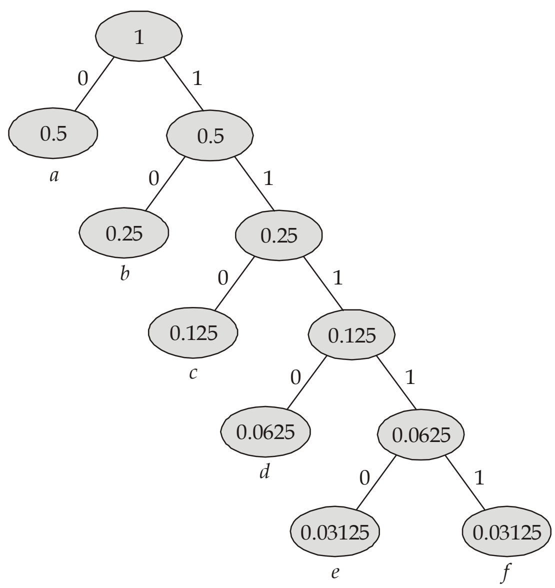
12. **(d)**



(Dotted edge are those included in every MST)

$$\text{Number of MSTs} = ({}^4C_3 \times {}^3C_2 \times {}^3C_2)$$
$$= (4 \times 3 \times 3) = 36$$

13. **(b)**



Codes:

$$a = 0, \quad b = 10, \quad c = 110$$
$$d = 1110, \quad e = 11110, \quad f = 11111$$
$$\text{Average length} = 0.5 \times 1 + 0.25 \times 2 + 0.125 \times 3 + 0.0625 \times 4 + 0.03125 \times 5 + 0.03125 \times 5$$
$$= 1.9375$$

14. **(a)**

For n, inner loop execute for $\frac{n}{2}$ times.

For $\frac{n}{2}$, inner loop execute for $\frac{n}{4}$ times.

For $\frac{n}{4}$, inner loop execute for $\frac{n}{8}$ times.

So, $\text{T.C.}(n) = \frac{n}{2}+\frac{n}{4}+\frac{n}{8}+.....+1 = n\left(\frac{1}{2}+\frac{1}{4}+.....+1\right) = 0(n)$

15. (b)

O(V) $\rightarrow$ Time taken to build the vertices min heap.

O(V logV) $\rightarrow$ Extract minimum vertex and heapify.

O(E logV) $\rightarrow$ For each updates have to be mode in the min heap before next extraction.

16. (d)

Insertion sort: 15, 12, 7, 20, 25, 42, 6, 2

After 1st iteration: 12, 15, 7, 20, 25, 42, 6, 2

After 2nd iteration: 7, 12, 15, 20, 25, 42, 6, 2

After 3rd iteration: 7, 12, 15, 20, 25, 42, 6, 2

After 4th iteration: 7, 12, 15, $\boxed{20}$, 25, 42, 6, 2

17. (a)

18. (d)

1. Delete the beginning *b* from *x*.*x* becomes *cde*.
2. Replace character e in *x* by character *g*.*x* becomes *cdg*.
3. Insert character *f* at the end of *x*.*x* becomes *cdgf*.

19. (a)

for $(i = 1; i \leq m; i++) \Rightarrow 0(m)$

for $(J = 1; J \times J \leq m; J++) \Rightarrow 0(\sqrt{m})$

$\text{T.C.} = 0(m \times m^{1/2}) = 0(m^{3/2})$

20. (b)

(a) $100\, n \log n = O\left(\frac{n \log n}{100}\right)$

Multiplication by a constant does not change the rate of growth of functions.

(b) $\sqrt{\log n} = (\log n)^{0.5}$

$\log\log n = \log(\log n)$

Since any positive polynomial function grows faster then any logarithmic, this is false.

(c) Since, $y > x,\ n^x = O(n^y)$

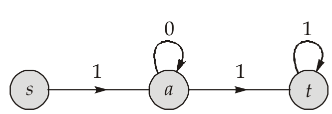
(d) Growth of an exponential function is higher than a linear function.

21. (d)

Since the largest element should remain in bottom of the tree, any element less than it will trigger a heapify operation. So the largest element will definitely come to leaf node level.

22. (b)

- The parent pointers may not lead back to the source node if a zero length cycle exists. Take an example [π means parent]

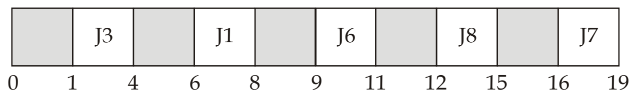


Relaxing the (S, a) edge will set $d[a] = 1$ and $\pi[a] = S$. Then relaxing the (a, a) edge will set $d[a] = 1$ and $\pi[a] = a$.

Following the π-pointers from t will no longer give a depth to S. So, it is no correct algorithm.

- Option (b) correctly states about Radix sort.

23. **(c)**



Sort the entries according to their finish times.

So units of time for which no job is executed = 6 units.

24. **(d)**

Prim's algorithm always gives connected whenever a spanning tree is constructed.

(c, e), (e, b), (b, a), (b, d), (b, f)

25. **(a)**

First let's find the height of the tree (say h).

$$\frac{(\log m)}{2^h} = 1$$

$$h = O(\log \log m)$$

The time to merge from level i to level $i + 1 = O(\log n)$.

So, the total time to merge log m sorted lists into a single list of log n elements

$$= O(\log n . \log \log m)$$

26. **(c)**

$$|S_1| = 98$$
$$|S_2| = 49$$
$$|S_3| = 49$$

If graph is undirected and we apply DFS then every edge is either tree edge or back edge.

So, $y = 0, z = 0$

And $w + x$ = Number of edges in graph

$= |S_1| + |S_2| + |S_3|$

$= 98 + 49 + 49 = 196$

27. **(b)**

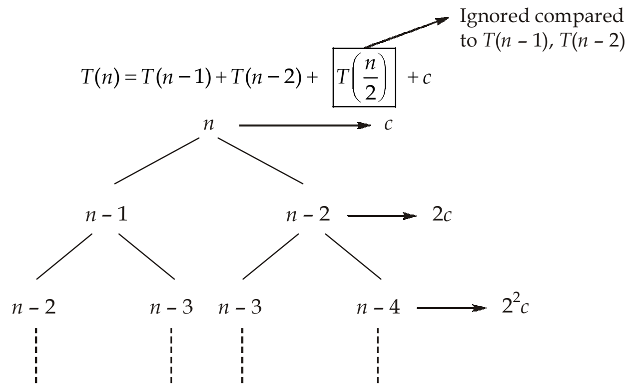
The order in which shortest path calculated using Dijkstra's algorithm.

Node	*a*	*c*	*b*	*e*	*d*	*g*	*f*	*h*	*i*
Shortest path length	0	3	5	7	9	9	12	17	20

Interchangeable (*d*, *g*)

From above, for 6 nodes shortest path has calculated before node 'f'.

28. (a)



In worst case, there will be n-levels in the above recursive tree and hence,

$$\begin{aligned} \text{Time complexity} &= c + 2c + 2^2 c + \dots + 2^n c \\ &= c[1 + 2 + 2^2 + \dots 2^n] \\ &= \frac{c[2^{n+1} - 1]}{2 - 1} = c \cdot 2^{n+1} \\ &= O(2^n) \end{aligned}$$

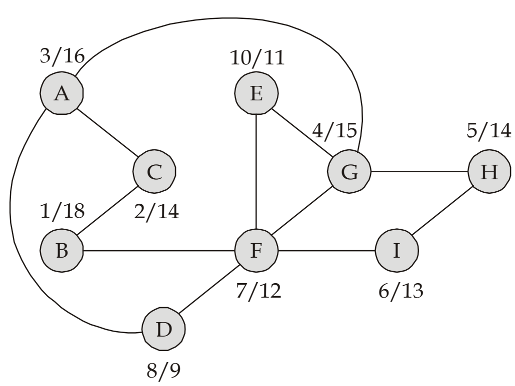
29. (c)

$$(n + 1)! \approx n! \approx O(n^n) \quad \text{[Sterling's approximation]}$$

$$\left.\begin{aligned} 4^n &= O(4^n) \\ e^n &= O(e^n) \\ (\log n)^{\log n} &= O((\log n)^{\log n}) \end{aligned}\right] \text{Compare by taking log function since these belong to different function families}$$

e^n	$(\log n)^{\log n}$
Taking log	Taking log
$n \log e$	$(\log n)(\log \log n)$
$= O(n)$	$= O((\log n)^2)$
Taking log	Taking log
$\log n$ $>$	$2 \log \log n$

30. (c)



■■■■