S.No. : 06 SK_CS_C_141119

Algorithms

MADE EASY
India's Best Institute for IES, GATE & PSUs

Delhi | Noida | Bhopal | Hyderabad | Jaipur | Lucknow | Indore | Pune | Bhubaneswar | Kolkata | Patna

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

CLASS TEST 2019-2020

COMPUTER SCIENCE & IT

Date of Test : 14/11/2019

ANSWER KEY ➤ Algorithms

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

Detailed Explanations

1. **(a)**

Median: Middle element in the sorted array called median.

Since both are sorted array of size N.

Time taken to combine = O(2N) = O(N) using merge procedure (outplace)

Now we get sorted array.

Find median of that array take O(1) i.e., $\frac{\text{Start index} + \text{Last index}}{2} = \text{Mid}$

Total time = O(N) + O(1) = O(N)

Or

Since both the arrays are shorted so we can apply binary search tree. To find the n/2 smallest element in the array will take log n time.

So, best answer will be O(log n).

2. **(c)**

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

$$\text{Put } n = 2^m, \qquad \because \quad m = \log n$$

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

$$\underset{S}{\boxed{T(2^m)}} = \underset{S}{\boxed{T(2^{m/2})}} + 1$$

$$S(m) = S(m/2) + 1$$

Using Master's Theorem

$$S(m) = \log m \qquad (\because \quad m = \log n)$$

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

3. **(a)**

To create binary tree inoder always neccessory. Here binary search tree is created whose inoder always in increasing order.

So we can apply binary search i.e. n time binary search that takes n × log n = n log n.

Or

To construct binary search tree when preorder and postorder of the tree is given will take O(n) time.

So best answer will be option (a).

4. **(c)**

Given $f(n) = \Omega(n)$

i.e. $f(n) \geq c_1(n)$

$f(n)$ can be anything but atleast (n) not less than (n)

Given $g(n) = \Omega(n^2)$

i.e. $g(n) \geq c_2(n^2)$

$g(n)$ can be anything but atleast (n^2) not less than (n^2)

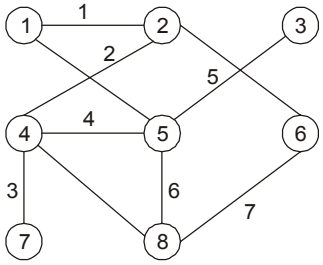
$$f(n) + g(n) = \Omega((n) + (n^2))$$

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

Here we can not comment about upper bound.

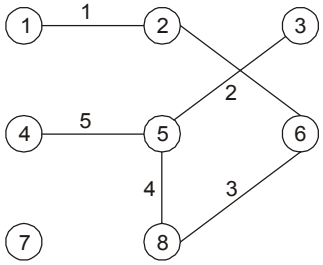
5. **(b)**

(a)



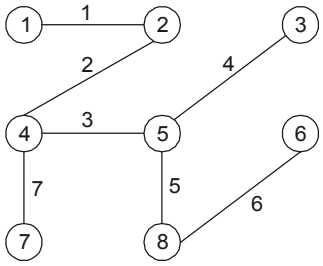
DFS safe sequence

(b)



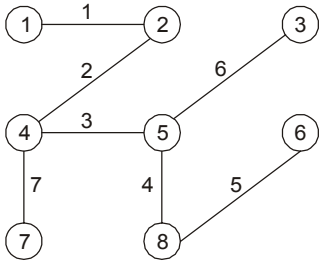
4 to 3 not follow DFS so not DFS sequence because have to go 4 to 7 first then 7 to 3.

(c)



DFS safe sequence

(d)



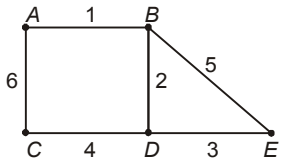
DFS safe sequence

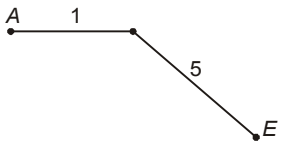
6. **(a)**

Adding a constant to every edge weight in a directed graph can change the set of edges that belongs to shortest path tree. Assume unique weights.

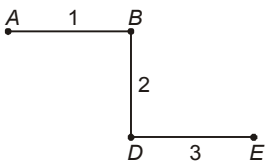
7. **(b)**

(i) Graph can have more than one shortest path between two vertex where all edge weights are distinct.





Cost = 6



Cost = 6

(ii) Multiplying all edge weight by a positive number should not effect the minimum spanning tree. So option (b) is invalid.

8. **(d)**

The comparisons required in the worst case is O(M + N) using merge procedure. Similarly all (M + N) elements need to be moved in to new array using merge procedure.

Therefore movements are of O(M + N).

9. (b)

Size m $\Rightarrow$ To delete the root : O(log m)
Size log n $\Rightarrow$ To delete the root : O(log (log n)) = O(log log n)

10. (b)

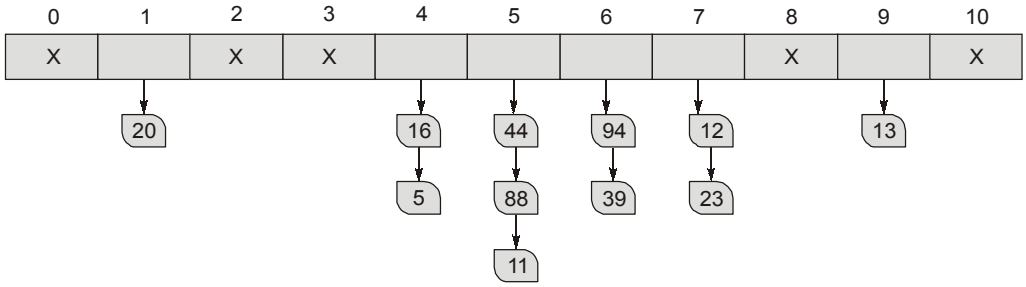
If A[j] is smaller than A[j – 1] then exchanges A[j] and A[j – 1] to get the smaller first.

11. (c)

Minimum spanning tree by using Dijkstra's is

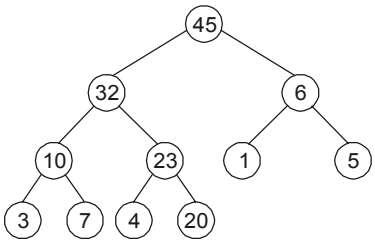
E_{max} = 0 + 2 + 1 + 3 + 6 + 7 + 9 + 8 + 10 + 12 = 58
E_{sp} = 1 + 5 + 1 = 7
Value of $E_{max} - E_{sp}$ = 58 – 7 = 51

12. (b)



Number of free indexes = 5
So, probability will be = 5/11 = 0.45

13. (a)



min_0 = Minimum element present at 0 level = 45
min_1 = Minimum element present at 1 level = 6
min_2 = Minimum element present at 2 level = 1
min_3 = Minimum element present at 3 level = 3

$$\text{Sum} = \min_0 + \min_1 + \min_2 + \min_3$$
$$= 45 + 6 + 1 + 3 = 55$$

14. (a)

Time to find maximum number of inversions in array of n elements is $O(n^2)$

Here $$n = \frac{(16 \times 15)}{2} = 120$$

Time to find minimum number of inversions in array of n element is zero (array in increasing order).

Difference = 120 – 0 = 120

15. (d)

Sort the task inorder by their start time.

{(1, 3), (2, 4), (3, 5), (2, 7), (4, 6), (5, 6), (3, 7), (6, 10), (7, 9)}

Now we schedule the task inorder, on machines. So that they don't conflict. We will put the job first on the lowest numbered machine if that is possible.

m_1 : (1, 3), (3, 5), (5, 6), (6, 10)

m_2 : (2, 4), (4, 6), (7, 9)

m_3 : (2, 7)

m_4 : (3, 7)

4 machines are required.

16. (c)

Pass 1:

10**0**, 00**1**, 11**0**, 01**1**, 01**0** = 100, 110, 010, 001, 011

Pass 2: ⇓

1**0**0, 1**1**0, 0**1**0, 0**0**1, 0**1**1 = 100, 001, 110, 010, 011

Pass 3: ⇓

100, **0**01, **1**10, **0**10, **0**11 = 001, 010, 011, 100, 110

3rd binary number is: 011

17. (c)

$$f(n) = \sum_{i=0}^{99} \sum_{j=0}^{n-1} \left(\sum_{k=0}^{j-1} 1 \right)$$
$$= O(n^2)$$

18. (b)

The given recurrence relation is denoted for matrix chain multiplication problem and the following are the conditions for recurrence relation.

If $i = j$; $m[i, j] = 0$

If $i < j$; $m[i, j] = \min_{i \le k < j} \{m[i,k] + m[k+1, j] + P_{i-1} P_k P_j\}$

19. (d)

In level order traversal of the tree, we visit nodes on the current level and then goes to the next level.

Similarly in breadth first search visits all the neighbors first and then visits all the neighbors of each neighbor one by one.

20. (c)

Fractional Knapsack using greedy algorithm.

(*i*) Find ratio of profit to weight (P/W) for all objects $\Rightarrow O(n)$.

(*ii*) Sort the above ratio (P/W) in decreasing order for all objects by using merge sort $\Rightarrow O(n \log n)$.

(*iii*) Take array of n objects. Place objects in array until capacity of Knapsack becomes zero $\Rightarrow O(n)$.

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

21. (a)

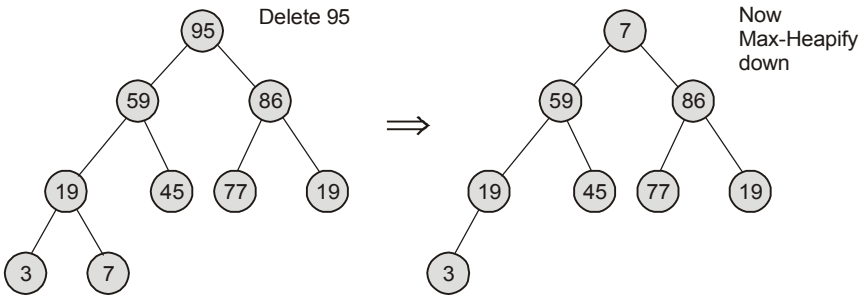
1. $X = i > 0$; 1st loop run until i is greater than 0 i.e. reach for 1st element from last element.
2. $Y = K > 0$; 2nd run $(n - i)$ time every time where $i = 1, 2, 3, 4, \ldots n$.
3. $Z = a[K] > a[\max]$; if condition checks $a[K]$ greater than $a[\max]$ them update max = K.

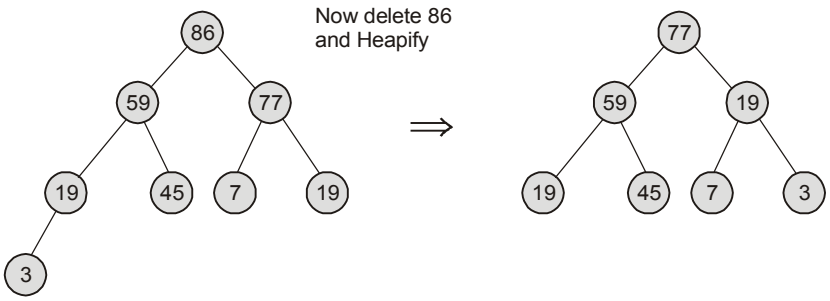
Atleast swap ($a[i]$ and $a[\max]$).

22. (b)

$$\sum_{i=1}^{2n}\sum_{j=1}^{n} 1 - \sum_{j=1}^{n} j = \sum_{i=1}^{2n} n - \frac{n(n+1)}{2} = 2n^2 - \frac{n^2}{2} - \frac{n}{2} = \frac{3}{2}n^2 - \frac{n}{2} = \frac{3n^2 - n}{2}$$

23. (b)

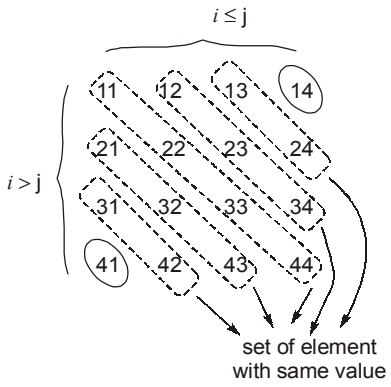




$\therefore$ 77, 59, 19, 19, 45, 7, 3

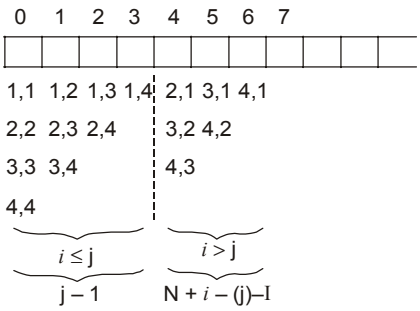
24. (d)

Consider an array A with an example of 4 × 4 size.



every A [i, j]= =A [i – 1, j – 1]

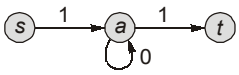
Hence we need to store only once. Which can be done by storing entire first row and first column in row major order into an array B.



25. (a)

- In Radix Sort, all input orderings give the worst-case running time, the running time does not depend on the order of the inputs in any significant way.

- The parent pointers may not lead back to the source node if a zero length cycle exists.
  In the example below, 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 the algorithm is incorrect.



26. (b)

(a) $((A_1\ A_2)A_3)\ A_4$ requires $(4 \times 6 \times 8 + 4 \times 8 \times 4 + 4 \times 4 \times 5) = 400$ multiplications.

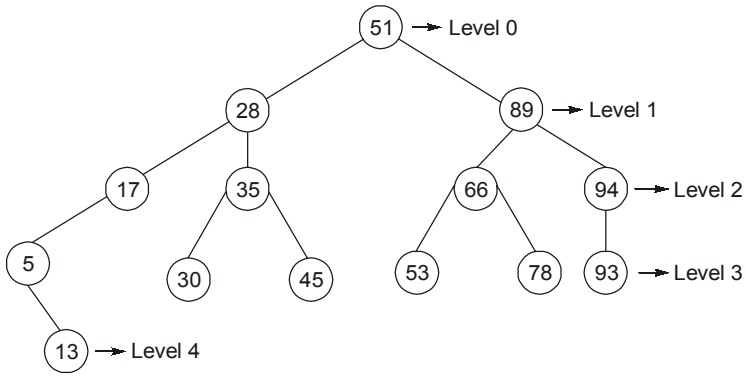
(b) $(A_1\ (A_2\ A_3))A_4$ requires $(6 \times 8 \times 4 + 4 \times 6 \times 4 + 4 \times 4 \times 5) = 368$ multiplications.

(c) $A_1\ ((A_2\ A_3)\ A_4)$ requires $(6 \times 8 \times 4 + 6 \times 4 \times 5 + 4 \times 6 \times 5) = 432$ multiplications.

(d) $(A_1\ A_2)\ (A_3\ A_4)$ requires $(4 \times 6 \times 8 + 8 \times 4 \times 5 + 4 \times 8 \times 5) = 512$ multiplications.

27. (b)

Preorder: 51, 28, 17, 5, 13, 35, 30, 45, 89, 66, 53, 78, 94, 93



Number of nodes at level 3 = 6

28. (b)

Best case of quick sort is $O(n \log n)$ and it takes 2048 msec.

$$2048 = c_1 \cdot n \log n \qquad \ldots(1)$$

Worst case of quick sort is $O(n^2)$ and it takes 324 msec.

$$324 = c_1 \cdot n^2 \qquad [n = 18]$$
$$324 = c_1 \cdot 18^2$$
$$\boldsymbol{c_1 = 1}$$

Substitute $c_1 = 1$ in equation (1)

$$c_1 \cdot n \log n = 2048 \qquad [c_1 = 1]$$
$$n \log n = 2048$$
$$\text{Put } \boldsymbol{n = 2^K}$$
$$2^K \times K = 2048 \qquad \ldots(2)$$

The value of $K = 8$ which satisfies equation (2)

$\therefore$ Vivek's file size = $2^K = 2^8 = 256$

29. **(c)**

Fractional Knapsack problem:

Select all of item '*a*', '*d*', '*e*', '*j*' and 1/3 of item 'g'

Total weight = 3 + 1 + 12 + 1 + 1/3 + 9 = 20

Total profit = 7 + 3 + 26 + 4 + 1/3 × 18 = 46

0/1 Knapsack problem:

Select all of item *j*, *d*, *a*, *e* and *c*.

Total weight = 2 + 1 + 1 + 3 + 12 = 19

Total profit = 7 + 3 + 26 + 4 + 3 = 43

Difference = Total profit (using fractional Knapsack – Using 0/1 Knapsack)

= 46 – 43 = 3

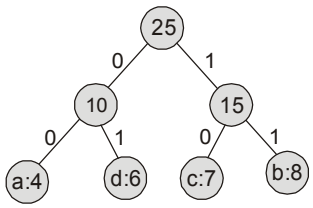
30. **(a)**

Number of a's = 4

Number of b's = 8

Number of c's = 7

Number of d's = 6



a = 00,

b = 11,

c = 10,

d = 01,

Number of bits for given message = 4 × 2 + 8 × 2 + 7 × 2 + 6 × 2 = 50 bits

■■■■