POSTAL Book Package 2023

Computer Science & IT

Objective Practice Sets

Algorithms

Contents

MADE EASY Publications

1 CHAPTER Asymptotic Analysis of Algorithms

Multiple Choice Questions

Q.1 The concept of order (Big O) is important because
(a) It can be used to decide the best algorithm that solves a given problem
(b) It determines the maximum size of a problem that can be solved in a given amount of time
(c) It is the lower bound of the growth rate of algorithm
(d) Both (a) and (b) above

Q.2 Let $f(n) = \Omega(n)$ and $g(n) = \Omega(n^2)$. Then $f(n) + g(n)$ is
(a) $\Omega(n)$ (b) $\theta(n)$
(c) $\Omega(n^2)$ (d) $O(n)$

Q.3 The order of an algorithm that finds whether a given Boolean function of 'n' variables, produces a 1 is
(a) Constant (b) Linear
(c) Logarithmic (d) Exponential

Q.4 $f(n) = 5n^2 + 6n + 10$
Which will be the exact value for $f(n)$?
(a) $\theta(n^2)$ (b) $O(n^2)$
(c) $o(n^2)$ (d) $\Omega(n^2)$

Q.5 Match the following groups:

Group-I ($n > 0$)	Group-II
A. $3n + 4n^2 + 5n \log n$	1. $O(1)$
B. $n + \log n + \log \log n$	2. $O(\log n)$
C. $10 + n + n \log n + \log n$	3. $O(n)$
D. $10 + 10000 + 100000$	4. $O(n \log n)$
	5. $O(n^2)$

Codes:

	A	B	C	D
(a)	5	3	4	2
(b)	5	4	3	1
(c)	5	3	4	1
(d)	5	4	3	2

Q.6 Let $f(n) = \Omega(n)$ and $g(n) = \Omega(n^2)$. Then $f(n) + g(n)$ is
(a) $\Omega(n)$ (b) $\theta(n)$
(c) $\Omega(n^2)$ (d) $O(n)$

Q.7 Find which of the following is not correct?
(a) $\sum_{i=1}^{n} \sqrt{i} = O(n^{3/2})$
(b) $n^2 \log n = \Theta(n^2)$
(c) $100n^3 + 2n^2 = \Omega(n^2)$
(d) $n! = O(n^n)$

Q.8 Unrestricted use of Goto is harmful, because it
(a) increase running time of programs.
(b) makes debugging difficult.
(c) Results in the compiler generating longer machine code.
(d) Increase memory requirement of programs.

Q.9 Each of the function $2^{\sqrt{n}}$ and $n^{\log n}$ has a growth rate .... than that of any polynomial.
(a) Greater (b) Less
(c) Equal to (d) Uncertain

Q.10 Consider the following function:

```
void f(int n)
{
    int i, j, k, m;
    for (i = 0; i < 100; i ++)
    {
        for (j = 0; j < n; j++)
        {
            for (k = 0; k < j; k++)
                printf("%d", k);
        }
    }
}
```

What is the worst case running time of the function f for any positive value of n?
(a) $O(1)$ (b) $O(n)$
(c) $O(n^2)$ (d) $O(n^3)$

Q.11 Consider the following function:

```
int unknown (int n)
{
    int i, j, k = 0;
    for (i = n/2; i <= n; i++)
        for (j = 2; j <= n; j = j* 2)
            k = k + n/2;
    return (k);
}
```

The return value of the function is

(a) $\Theta(n^2)$ (b) $\Theta(n^2 \log n)$
(c) $\Theta(n^3)$ (d) $\Theta(n^3 \log n)$

Q.12 Consider the following segment of c-code:

```
int j, n;
j = 1;
while (j <= n)
```

The number of comparisons mode in the execution of the loop for any $n > 0$ is

(a) $\lceil \log_2 n \rceil + 1$ (b) n
(c) $\lceil \log_2 n \rceil$ (d) $\lfloor \log_2 n \rfloor + 1$

Q.13 In the following C function, let $n \geq m$.

```
int gcd (n, m)
{   if (n% m = = 0} return m;
    n = n% m;
    return gcd (m, n);
}
```

How many recursive calls are made by this function?

(a) $\Theta(\log_2 n)$ (b) $\Omega(n)$
(c) $\Theta(\log_2 \log_2 n)$ (d) $\Theta\left(\sqrt{n}\right)$

Q.14 What is time complexity of following code:

```
int a = 0;
for (i = 0; i < N; i++)
{
    for (j = N; j > i; j--)
    {
        a = a + i + j;
    }
}
```

(a) O(N) (b) O(N * (log N)
(c) O(N * Sqrt(N)) (d) $O(n^2)$

Q.15 What does it mean when we say that one algorithm X is asymptotically more efficient than Y?

(a) X will always be a better choice for small inputs.
(b) X will always be a better choice for large inputs.
(c) Y will always be a better choice for small inputs.
(d) X will always be a better choice for all inputs.

Q.16 What is time complexity of following code:

```
int a = 0, i = N;
while (i > 0) {
    a = a + i;
    i/ = 2;
}
```

(a) O(N) (b) O(Sqrt(N))
(c) O(N/2) (d) O (log N)

Q.17 What is time complexity of fun( )?

```
int fun (int n)
{
    int count = 0;
    for (int i = n; i > 0; i/ = 2)
      for (int j = 0; j < i; j++)
      count + = 1;
    return count;
:
```

(a) $O(n^2)$ (b) O(n)
(c) O(n log n) (d) $O(n (\log n)^2)$

Q.18 Consider the following recursive function:

```
int F (int array [ ], int n)
{
    int S = 0;
    if (n = = 0)
        return 0;
    S = F (array, n – 1)
    if (array [n – 1] < 0)
        S = S + 100;
    return S;
}
```

What is the worst case time complexity of the above function?

(a) O(n) (b) O(nlogn)
(c) $O(n^2)$ (d) O(logn)

Q.19 What is time complexity of following program?

```
Void fun (int n){
int i, j, counter = 0;
for (i = 1; i ≤ n; i++) {
for (j = 1; j × j ≤ n; j++) count++;}}
```

(a) $O(n^2)$ (b) $O(n^{3/2})$
(c) $O(n \log n)$ (d) $O(n \log \log n)$

Q.20 Consider the following functions:

1. $n!$
2. a^n, a is constant, $a > 0$
3. n^n
4. n^k, k is a constant, $k > 0$
5. e^n

Choose the correct statement which ranks all functions by order of growth.

(a) $2 < 4 < 5 < 1 < 3$
(b) $4 < 2 < 3 < 5 < 1$
(c) $4 < 5 < 2 < 1 < 3$
(d) $4 < 2 < 5 < 1 < 3$

Q.21 Consider the following functions:

$n, \log n, \sqrt{n}, \log(\log n), \frac{n}{\log n}, (\log n)^2,$

$\sqrt{n}\log n, n\log n$

Identify the functions in increasing order of growth.

(a) $\frac{n}{\log n}, \log(\log n), (\log n), (\log n)^2,$

$\sqrt{n}, \sqrt{n} \cdot \log n, n$

(b) $\log(\log n), \log n, (\log n)^2, \sqrt{n},$

$\sqrt{n}\log n, \frac{n}{\log n}, n, n \log n$

(c) $\log(\log n), \log n, (\log n)^2,$

$\sqrt{n}, \frac{n}{\log n}, \sqrt{n}\log n, n, n \log n.$

(d) None of these

Q.22 Which one of the following is true?

1. $an = O(n^2)$ (small oh) $a \geq 0$
2. $an^2 = O(n^2)$ (big oh) $a > 0$
3. $an^2 \neq O(n^2)$ (small oh) $a > 0$

(a) Only 1 and 2 are correct
(b) Only 1 is correct
(c) 1 and 3 are correct only
(d) All are correct

Q.23 Consider the following statements:

1. Any two functions f, g are always comparable under big-oh that is $f = O(g)$ or $g = O(f)$
2. If $f = O(g)$ and $f = O(h)$ then, $g(n) = \theta(h)$

Select correct option:

(a) 1 is true and 2 is false
(b) 1 is false and 2 is true
(c) Both are false
(d) Both are true

Q.24 1. $\frac{1}{2}n^2 = \omega(n)$, 2. $\frac{1}{2}n^2 = \omega(n^2)$

Which of the following is true?

(a) 1 is correct
(b) 2 is correct
(c) 1 and 2 both are correct
(d) None of these

Q.25 Consider the following functions:

$f(n) = 2^{\log_2 n}$
$f(n) = n^{\log_2 n}$
$h(n) = n^{1/\log_2 n}$

Which of the following statements about the asymptotic behaviour of $f(n)$, $g(n)$ and $h(n)$ is true?

(a) $f(n) = \Omega(g(n))$ and $g(n) = O(h(n))$
(b) $g(n) = \Omega(h(n))$ and $f(n) = O(f(n))$
(c) $f(n) = O(g(n))$ and $g(n) = \Omega(h(n))$
(d) $g(n) = O(h(n))$ and $h(n) = O(g(n))$

Q.26 Suppose $f, g, h, k : N \to N$.
If $f = O(h)$ and $g = O(k)$, then

(a) $f + g = O(h + k)$
(b) $fg = (hk)$
(c) Both (a) and (b) above
(d) None of the above

Q.27 Let $g(n) = \Omega(n)$, $f(n) = O(n)$ and $h(n) = \theta(n)$ then what is the time complexity of $[g(n)\ f(n) + h(n)]$

(a) $O(n)$ (b) $\theta(n)$
(c) $\Omega(n)$ (d) $\theta(n^2)$

Q.28 Consider an array of n element with sorted order, if any element i appear more than half the number of element, what is the time complexity to count the number of occurrences of i?

(a) $O(\log n)$ (b) $O(1)$
(c) $O(n)$ (d) $O(\log \log n)$

Q.29 Find the time complexity of the following summation. Assume that k is a constant, $k > 0$

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

(a) $O(n)$ (b) $O(n^2)$
(c) $O(n^3)$ (d) None of these

Q.30 What will be the worst case time complexity for the following code segment?

```
int count = 0, N;
for (i = 0; i < N * 2; i++)
{
    for (j = 0; j < i/3; i++)
    {
        for (k = 0; k < j * j; k++)
        {
            count ++;
        }
    }
}
```

(a) $O(N^4)$ (b) $O(N^3)$
(c) $O(N^2)$ (d) $O(N)$

Q.31 Which of these functions grows fastest with n?

(a) $\frac{e^n}{n}$ (b) $e^{n-0.9 \log n}$
(c) 2^n (d) $(\log n)^{n-1}$

Q.32 Let $f(n) = n$ and $g(n) = n^{(1+\sin n)}$, where n is a positive integer. Which of the following statements is/are correct?

I. $f(n) = O(g(n))$
II. $f(n) = \Omega(g(n))$

(a) Only I (b) Only II
(c) Both I and II (d) Neither I nor II

Q.33 Consider two natural-values functions $f: N \to N$ and $g: N \to N$. Which of the following statements cannot be true?

(a) $f \in O(g)$ and $g \in O(f)$
(b) $f \in \theta(g)$ and $g \in \theta(f)$
(c) $f \in \Omega(g)$ and $g \in \Omega(f)$
(d) $f \in O(g)$ but $g \notin \Omega(f)$

Q.34 Consider the following code segment:

```
int f00 (int n)
{
    if (n > 10000)
        return 1:
    int sum = 0, 1;
    for (i = 0; i < n; i++)
    {
        sum + = i;
    }
    return sum;
}
```

The value returned by above function is

(a) $\theta(n^2)$ (b) $\theta(n)$
(c) $\theta(1)$ (d) $\Omega(n^2)$

Q.35 Let's look about the algorithm:

```
int temp, j, i;
for (i = 1; i < n; i++)
{
    temp = A[i];
    for (j = i - 1; j ≥ && (A(j] > temp); j--)
    A[j + 1] = A[ j];
    A[ j] = temp;
    }
}
```

If the array is sorted the complexity is

(a) $O(n)$ (b) $O(n^2)$
(c) $O(n \log n_2 n)$ (d) $O(\log_2 n)$

Q.36 In above question if the array is in reverse sorted order then time complexities will be

(a) $O(n)$ (b) $O(n \log_2 n)$
(c) $O(n^2)$ (d) $O(\log_2 n)$

Q.37 For each group of functions, sort the functions in increasing order of asymptotic (big-O) complexity:

$f_1(n) = 2^{2^{1000000}}$

$f_2(n) = 2^{100000n}$

$f_3(n) = \binom{n}{2}$

$f_4(n) = n\sqrt{n}$

(a) $f_1(n)$, $f_4(n)$, $f_3(n)$, $f_2(n)$
(b) $f_1(n)$, $f_2(n)$, $f_3(n)$, $f_4(n)$
(c) $f_4(n)$, $f_1(n)$, $f_3(n)$, $f_2(n)$
(d) $f_4(n)$, $f_3(n)$, $f_1(n)$, $f_2(n)$

Q.38 If $f(n) = O(g(n))$ for f, g are non-decreasing function, then $f(n) * \log_2(f(n)^c) = O(g(n) * \log_2 g(n))$, where C = power of $f(n)$ is

(a) Always true
(b) Never
(c) Sometimes true, sometimes not depending on f, g.
(d) Sometimes true, sometimes false depending on C.

Q.39 Which of the following set is empty?

(a) $o(g(n)) \cap \omega(g(n))$ (b) $O(g(n)) \cap \Omega(g(n))$
(c) $o(g(n)) \cap O(g(n))$ (d) $\omega(g(n)) \cap \Omega(g(n))$

Q.40 Consider the following C function:

```
void dosomething (int n)
{
        int m, j, k;
    for (j = 0; j < 200; j++)
    {
        for (k = 0; k < n; k++)
        {
            for (m = 0; m < j, m++)
                printf("%d", i + m);
        }
    }
}
```

What is the time complexity of the above function?

(a) $O(n^2)$ (b) $O(n\log n)$
(c) $O(n)$ (d) $O(n^2\log n)$

Q.41 Consider the following code segment of C-code:

```
{
    for(int i = 1; i < = n; i++)
    {
        int p = pow(i, k)
        for (int j = 1; j < = p; j ++)
        {
            // some O(1) work
        }
    }
}
```

Calculate time complexity of above code. The time complexity will be

(a) $O(n^k)$ (b) $O(n^{k+1})$
(c) $O(n^2)$ (d) $O(n^3)$

Q.42 You are given two sorting algorithms *A* and *B* that work in time $O(n \log n)$ and $O(n^2)$ respectively. Consider the following statements:

1. Algorithm *A* will sort any array faster than algorithm *B*.
2. On an average, algorithm *A* will sort a given array faster than algorithm *B*.
3. If we need to implement one of the two as the default sorting algorithm in a system, algorithm *A* will always be preferable to algorithm *B*.

Which of the statements above are true?

(a) 1, 2 and 3 (b) 1 and 3
(c) 2 and 3 (d) None of them

Q.43 Suppose $A = \log^k n$; $B = n^\epsilon$
Assume that $k \geq 1$ and $\epsilon > 0$
What is relation between the asymptotic time complexities of *A* and *B*.

(a) $A = O(B)$ (b) $A = o(B)$
(c) $A = \Omega(B)$ (d) $A = \omega(B)$

Q.44 Consider the following program segments:

```
main( )
{
    int i = 0, m = 0;
for (i = 1; i < n; i++) {
        for (j = 1; j < i * i; j++) {
            if ((j % i) = = 0)
        for (k = 1; k < j; k++) {
            m = m + 1; }
        }
    }
}
```

What is the time complexity of above program?

(a) $O(n^2\log n)$ (b) $O(n^3)$
(c) $O(n^4)$ (d) $O(n\log n)$

Q.45 What is the time complexity for the following C module? Assume that $n > 0$.

```
int module (int n)
{   if (n == 1) return 1;
    else return (n + module (n – 1));
}
```

(a) $O(n)$ (b) $O(\log n)$
(c) $O(n^2)$ (d) $O(n!)$

Q.46 Consider the following function:

```
Void func (int n) {
    int k = n;
    int i = 0;
    for(; i < n; i++){
    while (k > 1)q
    k >> = 1;
    }
}
```

What is worst case time complexity of the function

(a) $O(n)$ (b) $O(\log n)$
(c) $O(n^2)$ (d) $O(n \log n)$

Q.47 Among the following asymptotic expression which of these functions grows slowest (as a function of *n*) asymptotically?

(a) $\left(\sqrt{\log n}\right)^{\log^2 n}$ (b) $(\log n)^{\sqrt{\log n}}$
(c) n^{10} (d) $2^{2^{\sqrt{\log\log n}}}$

Q.48 Consider the following segment of C-code:

```
int j, n;
j = 1;
while (j <= n)
    j = j * 2
```

The number of comparisons made in the execution of the loop for any $n > 0$ is

(a) $\lceil \log_2 n \rceil * n$ (b) n

(c) $\lceil \log_2 n \rceil$ (d) $\lceil \log_2 n \rceil + 1$

Q.49 Consider the following functions:

$f(n) = 2^n$

$g(n) = n!$

$h(n) = n^{\log n}$

Which of the following statements about the asymptotic behaviour of $f(n)$, $g(n)$ and $h(n)$ is true?

(a) $f(n) = O(g(n)); g(n) = O(h(n))$

(b) $f(n) = \Omega(g(n)); g(n) = O(h(n))$

(c) $g(n) = O(f(n)); h(n) = O(f(n))$

(d) $h(n) = O(f(n)); g(n) = \Omega(f(n))$

Q.50 What is time complexity of this code:

```
int A(int)
{
    if (n < = 2) return 1;
    else
    return (A(√n) + n);
}
```

(a) O(log log(n)) (b) O(log n)

(c) O(n) (d) O(log n log n)

Q.51 What will be the time complexity of the following code?

```
A( )
{
    int i, j, k, n;
    for (i = 1, i < = N; i++)
    {
        for (j = 1; j < = i; j++)
        {
            for (k = 1; k < = 100; k++)
            {
                Printf ("Hello");
            }
        }
    }
}
```

(a) $\theta(n^2)$ (b) $\theta(n^3)$

(c) $\theta(n^4)$ (d) $\theta(n)$

Q.52 What is the time complexity for the following C module? Assume that $n > 0$.

```
int module (int n)
{   if (n == 1) return 1;
    else return (n + module (n – 1));
}
```

(a) O(n) (b) O(log n)

(c) $O(n^2)$ (d) O(n!)

Q.53 What is relation between the asymptotic time complexities of $f(n)$ and $g(n)$?

$f(n) = n * 2^n$

$g(n) = e^n$

(a) $e^n = O(n * 2^n)$ (b) $e^n = \theta(n * 2^n)$

(c) $e^n = \omega(n * 2^n)$ (d) $e^n = o(n * 2^n)$

Q.54 Arrange the following functions is increasing order of growth rate (with $g(n)$ following $f(n)$ in your list if and only if $f(n) = O(g(n))$.

$\sqrt{n}, 10^n, n^{1.5}, 2^{\sqrt{\log_2 n}}, n^{5/3}$

(a) $\sqrt{n}, 10^n, n^{1.5}, 2^{\sqrt{\log_2 n}}, n^{5/3}$

(b) $n^{1.5}, n^{5/3}, \sqrt{n}, 2^{\sqrt{\log_2 n}}, 10^n$

(c) $\sqrt{n}, 2^{\sqrt{\log_2 n}}, n^{1.5}, n^{5/3}, 10^n$

(d) None of these

Q.55 Determine the asymptotic running time of $f(n)$ for the following C program:

```
int f(int n)
{
    int x;
    if (n == 0) x = 1;
    else x = f(n – 1)*2;
    g(x);
    return x;
}
void g(int m)
{
    int y;
    for (y = m; y > 0; y/ = 2);
}
```

(a) $\Theta(n)$ (b) $\Theta(n^2)$

(c) $\Theta(2^n)$ (d) $\Theta(n \log n)$

Q.56 Sort the following by increasing order of growth. Assume *lg* means log base 2.

$\sqrt{n}\, 2^n \cdot 2^{\log n} \lg n$

(a) $\sqrt{n} \log n\, 2^{\log n}\, 2^n$ (b) $\sqrt{n} \log n\, 2^n 2^{\log n}$

(c) $\log n \sqrt{n}\, 2^n\, 2^{\log n}$ (d) $\log n \sqrt{n}\, 2^{\log n}\, 2^n$

Q.57 For each group of functions, sort the functions in increasing order of asymptotic (big-O) complexity:

$f_1(n) = n^{\sqrt{n}}$

$f_2(n) = 2^n$

$f_3(n) = n^{10} \cdot 2^{n/2}$

$f_4(n) = \sum_{i=1}^{n} (i+1)$

(a) $f_4(n), f_1(n), f_3(n), f_2(n)$
(b) $f_1(n), f_2(n), f_3(n), f_4(n)$
(c) $f_1(n), f_4(n), f_2(n), f_3(n)$
(d) $f_4(n), f_1(n), f_2(n), f_3(n)$

Q.58 Consider the equality $\sum_{i=0}^{n} i^3 = X$ and the following choices for X.

1. $\theta(n^4)$
2. $O(n^5)$
3. $\theta(n^5)$
4. $\Omega(n^3)$

The equality above remains correct of X is replaced by:

(a) Only 1
(b) Only 2
(c) 1 or 3 or 4 but not 2
(d) 2 or 3 or 4 but not 1

Q.59 Consider the following pairs of functions:

(i) $f(n) = \sqrt{n}, g(n) = \log(n^2)$

(ii) $f(n) = \log(n^2), g(n) = \log n$

(iii) $f(n) = 2n, g(n) = 10n^2$

(iv) $f(n) = 2n, g(n) = 3n$

Which of the following is true?

(a) (ii) $g(n) = O(f(n))$ and (iv) $g(n) = O(f(n))$
(b) (i) $f(n) = O(g(n))$ and (iii) $g(n) = O(f(n))$
(c) (iii) $f(n) = O(g(n))$ and (iv) $f(n) = O(g(n))$
(d) (i) $g(n) = O(f(n))$ and (ii) $g(n) = O(f(n))$

Q.60 Consider the following code:

```
Sum = 0;
for (i = 1 to n)
  for (j = i to 2n)
    Sum = Sum + 1;
```

Find the time complexity of above code in terms of n.

(a) $O(n)$ (b) $O(n \log n)$
(c) $O(n^2)$ (d) $O(n^2 \log n)$

Q.61 Consider the following C code segment:

```
int f(int x)
{
    if (x < 1) return 1;
    else return (f(x – 1) + g(x));
}
int g(int x)
{
    if (x < 2 ) return 2;
    else return (f(x – 1) + g(x/2));
}
```

Of the following, which best describes the growth of $f(x)$ as a function of x?

(a) Linear (b) Exponential
(c) Quadratic (d) Cubic

Q.62 Consider the following code segment. Assume n to be a positive integer

```
for (i = 1; i < = n; i++) {
for (j = 1; j < = n; j + i) {
printf("Hi");
} }
```

Let $T(n)$ represent the total running time of the above case segment. Specify which of the following statements are correct.

(a) $T(n) = O(n^2)$ (b) $T(n) = O(n \log n)$
(c) $T(n) = O(n^2 \log n)$ (d) $T(n) = O(\log n)$

Q.63 Consider the following two functions:

$f(n) = n^3$, if $0 \le n < 10{,}000$
$\quad\quad n^2$, otherwise

$g(n) = n$, if $0 \le n < 100$
$\quad\quad n^2 + 5n$, otherwise

Which of the following is/are true?

1. $f(n)$ is $O(n^3)$
2. $g(n)$ is $O(n^3)$
3. $O(f(n))$ is same as $O(g(n))$
4. $g(n)$ is $O(n^2)$

(a) 3 and 4 (b) 1, 2 and 3
(c) 1, 2 and 4 (d) None of these

Q.64 Consider the following C function:

```
find (int n)
{   if (n < 2) then return;
    else
    {   sum = 0;
        for (i = 1; i ≤ 4; i++)
        find (n/2);
        for (i = 1; i ≤ n * n; i ++)
        sum = sum +1;
    }
}
```

Assume that the division operation takes constant time and "sum" is global variable. What is the time complexity of "find (n)"?

(a) $\Theta(n^2)$ (b) $\Theta(n^2 \log n)$
(c) $\Theta(n^3)$ (d) None of these

Q.65 How many times the following loop runs for given value of n?

```
for (int i = 2; i < = n; i = pow(i, 2))
{
  printf("%", i);
}
```

(a) O(n) (b) O(log n)
(c) $O(n^2)$ (d) O(log log n)

Q.66 What is the running time of the program?

```
Void fun (int n, int arr[ ]){
int i = 0, j = 0;
for (; i < n; ++i)
while (j < n && arr[i] < arr[j])
j++;
}
```

(a) $O(n^2)$ (b) O(n)
(c) O(n log n) (d) O(log n)

Q.67 You are given a set of n points on the number line. They are given in arbitrary order. The task is to find the points that are closest to each other. To solve the problem you decide to take one point and compute its distance to all other points and repeat this process for all points. During the process you track the closest pair. What is running time of this algorithm?

(a) O(n) (b) $O(n^2)$
(c) Runtime is independent of n
(d) O(log n)

Q.68 Let $f(n) = \Omega(n)$, $g(n) = O(n)$ and $h(n) = \Theta(n)$. Then $[f(n) \cdot g(n)] + h(n)$ is _____.

(a) $\Omega(n)$ (b) $O(n)$
(c) $\theta(n)$ (d) None of these

Q.69 Consider the following recursive function find:

```
int find (int A[ ], int n)
{
    int sum = 0;
    if (n == 0) return 0;
    sum = find (A, n – 1)
    if(A[n – 1] < 0) sum = sum + 1;
    return sum;
}
```

What is worst case running time of above function find (A[ ], n) when Array A has 0 to n – 1 elements?

(a) O(1) (b) O(log n)
(c) O(n) (d) $O(n^2)$

Q.70 Look at the following algorithm:

```
int n;
int A [100];
void x ( )
{   int i;
    for (i = n/2; i > = 1; i – –)
    y(i);
    while (n > 1) {
    y(1); }
}
void y(int i)
{   .....;
    .....;
    .....;
}
```

Let complexity of y is $O(\log_2 n)$. Then the complexity of x will be

(a) $O(n^2)$ (b) $O(n^2 \log_2 n)$
(c) $O(n \log_2 n)$ (d) O(n)

Q.71 Consider the following pseudo-code:

```
while (m < n)
  if (x > y) and (a < b) then
     a = a + 1
     y = y – 1
  end if
  m = m + 1
end while
```

What is cyclomatic complexity of the above pseudo-code?

(a) 2 (b) 3
(c) 4 (d) 5

Q.72 Consider the following C function:

```
Void dosomething (in n)
{
    int m, j, k;
    for (j = 0; j < 200; j++)
    {
        for (k = 0; k < n; k++)
        {
            for (m = 0, m < j; m++)
            Printf ("%d", i + m);
        }
    }
}
```

What is time complexity of above function?

(a) $O(n^2)$ (b) $O(n \log n)$

(c) $O(n)$ (d) $O(n^2 \log n)$

Multiple Select Questions (MSQ)

Q.73 Consider the following pseudo code:

```
sum = 0;
for (i = 0, i < n; i++) {
    for (j = 0, j < i; j++) {
        for (k = 0, k < j; k++) {
            sum = sum + 1;
        }
    }
}
```

What of the following options is/are correct with respect to the time complexity of the pseudo code given above?

(a) $\theta(n^3(\log n))$ (b) $\Omega(n^2)$

(c) $O(n^3(\log n))$ (d) $O(n^3)$

Q.74 Consider the following pseudocode for calculating ab (where a and b are positive integers)

```
FastPower (a, b) :
if b = 1
        return a
    otherwise
    c := a*a
    ans := FastPower(c, ⌊b/2⌋)
    if b is odd
        return a*ans
    otherwise return ans
end
```

Here $\lfloor x \rfloor$ denotes the floor function, that is, the largest integer less than or equal to x.

Now assuming that you use a calculator that supports multiplication and division (i.e., you can do multiplications and divisions in constant time), Which of the following statement(s) is/are correct?

(a) The overall asymptotic running time of the above algorithm is $O(\log n)$.

(b) The overall asymptotic running time of the above algorithm is O(n).

(c) The overall asymptotic running time of the above algorithm is O(1).

(d) The overall asymptotic running time of the above algorithm is $O(\log(\log n))$.

Q.75 Consider the three functions $f(n) = n^2 + n + 234$, $g(n) = 12n^2 + 34$ and $h(n) = 123n + 4$.

Which of the following statements is/are true?

(a) $f(n) = O(g(n))$ (b) $g(n) = O(f(n))$

(c) $f(n) = O(h(n))$ (d) $h(n) = O(f(n))$

Q.76 An algorithm is made up of two independent modules M_1 and M_2. If order of M_1 is $f(n)$ and M_2 is $g(n)$ then the order of algorithm is/are

(a) $\max(f(n), g(n))$ (b) $\min(f(n), g(n))$

(c) $f(n) + g(n)$ (d) $f(n) \times g(n)$

■■■■

Answers Asymptotic Analysis of Algorithms

1.	(d)	2.	(c)	3.	(d)	4.	(b)	5.	(c)	6.	(c)	7.	(b)	8.	(b)	9.	(a)
10.	(c)	11.	(b)	12.	(d)	13.	(a)	14.	(d)	15.	(b)	16.	(d)	17.	(b)	18.	(a)
19.	(b)	20.	(d)	21.	(b)	22.	(c)	23.	(c)	24.	(a)	25.	(c)	26.	(c)	27.	(c)
28.	(a)	29.	(b)	30.	(a)	31.	(d)	32.	(d)	33.	(d)	34.	(c)	35.	(a)	36.	(c)
37.	(a)	38.	(a)	39.	(a)	40.	(c)	41.	(b)	42.	(c)	43.	(b)	44.	(c)	45.	(a)
46.	(a)	47.	(d)	48.	(d)	49.	(d)	50.	(a)	51.	(a)	52.	(a)	53.	(c)	54.	(d)
55.	(b)	56.	(d)	57.	(a)	58.	(c)	59.	(d)	60.	(c)	61.	(b)	62.	(b)	63.	(a)
64.	(b)	65.	(d)	66.	(b)	67.	(b)	68.	(a)	69.	(c)	70.	(c)	71.	(c)	72.	(c)
73.	(c, d)		74.	(a, b)		75.	(a, b, d)		76.	(a, c)							

Explanations Asymptotic Analysis of Algorithms

1. (d)

Big O notation gives worst case limit for a given problem. Also find out the least upper bound of problem.

2. (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.

3. (d)

In the worst case it has to check all the 2^n possible input combinations, which is exponential.

4. (b)

$$f(n) = 5n^2 + 6n + 10$$

So, $O(n^2)$ (Big–oh) is exact value of $f(n)$.

So, answer is (b).

5. (c)

$3n + 4n^2 + 5n \log n = O(n^2)$

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

$10 + n + n \log n + \log n = O(n \log n)$

$10 + 10000 + 100000 = O(1)$

6. (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.

7. (b)

$n^2 \log n \leq k.n^2$ will not satisfy for any constant k.

$\therefore$ Option (c) is not correct.

8. (b)

Unrestricted we of goto statement is harmful because it makes more difficult to verifying programs i.e., use of goto can results in unstructured code and there can be blocks with multiple entry and exit which can cause difficulty which debugging of program.

9. (a)

$2^{\sqrt{n}}$ and $n^{\log n}$ grows exponentially which have growth rate greater than any polynomial.

10. (c)

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

11. (b)

Outer loop execute for $\frac{n}{2}+1$ iterations. Inner loop executes for $\log_2 n$ iterations. In every iteration of inner loop $\frac{n}{2}$ is added to k.

Return value = $\frac{n}{2}$ × number of outer loops × number of inner loops

$$= \frac{n}{2} \times \left(\frac{n}{2} + 1 \right) (\log n)$$

$$= O(n^2 \log n)$$

12. (d)

Let the increment of j is $2^0, 2^1, \ldots 2^i$ for some value of i so, according to the question for while loop; $2i \le n$ or $i \le \log_2 n$.

One extra comparison required for the termination of while loop.

So, total number of comparisons

$$= i + 1 = \lfloor \log_2 n \rfloor + 1.$$

13. (a)

Let, $T(m, n)$ be the total number of steps.

So, $T(m, 0) = 0$, $T(m, n) = T(n, m \bmod n)$ on average

$$T_n = \frac{1}{n} \sum_{0 \le k \le n} T(k, n)$$

$$T_n \approx 1 + \frac{1}{n}(T_0\, T_1 + \ldots + T_{n-1})$$

$$T_n \approx S_n$$

$$S_n = 1 + \frac{1}{n}(S_0\, S_1 + \ldots + S_{n-1})$$

$$S_n = 1 + \frac{1}{n+1}(S_0\, S_1 + \ldots + S_n)$$

$$= 1 + \frac{1}{n+1}(n(S_{n-1}) + S_n)$$

$$= 1 + \frac{1}{n+1}$$

$$= S_n + \frac{1}{n+1}$$

So, $T_n \approx \Theta(\log_2 n) + 0(1)$

$$T \approx \Theta(\log_2 n)$$

14. (d)

The above code runs total number of times

$= N + (N-1) + (N-2) + \ldots\ldots + 1 + 0$

$= N * (N+1)/2$

$= \frac{1}{2} * N \wedge 2 + \frac{1}{2} * N = O(N \wedge 2)$ times

15. (b)

In asymptotic analysis, we consider growth of algorithm in terms of input size. An algorithm X is said to be asymptotically better than Y if X takes smaller time than Y for all input sizes n larger than a value n_0 where $n_0 > 0$.

16. (d)

We have to find smallest x such that $N/2 \wedge X\, N$

$$X = \log(N)$$

So, O(log N) is time complexity.

17. (b)

For n time, inner loop will execute for n times.

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

For $\frac{n}{4}$ time, inner loop will execute for $\frac{n}{4}$ times

and do on ...

So, time complexity:

$$T(n) = O\left(n + \frac{n}{2} + \frac{n}{4} + \ldots + 1 \right) = O(n)$$

18. (a)

Recurrence relation of function F

$$F(n) = 0 \text{ if } n = 0$$

$$F(n) = F(n-1) + 1,\ n > 0$$

Time complexity = $O(n)$

19. (b)

for $(i = 1; i \le n; i{+}{+}) \Rightarrow O(n)$

for $(j = 1; j \times j \le n; j{+}{+}) \Rightarrow O(\sqrt{n})$

count ++;

Total time complexity = $O(n \times n^{1/2}) = O(n^{3/2})$

20. (d)

$n^k < a^n < e^n < n! < n^n$

n^n will take maximum asymptotic time.

$n! = O(n^n)$

$e < n \quad \therefore e^n < n^n$

$k < n \quad \therefore n^k < a^n$

21. (b)

$\log(\log n) < \log n < (\log n)^2 < \sqrt{n} < \sqrt{n}\log n < \dfrac{n}{\log n} < n < n\log n$

So option (b) is correct.

22. (c)

1. $an = O(n^2)$ for $a \geq 0$
   True
2. $an^2 = O(n^2)$ for $a > 0$
   False, big-*oh* needed.
3. $an^2 \neq O(n^2)$ for $a > 0$
   True, big-*oh* needed

So, 1 and 3 are correct.

23. (c)

Both are false.

Statement 1 is false.

Reason: Consider $f(n) = 0.5$ and $g(n) = \sin(n)$ or, consider $f(n) = n$ and $g(n) = 1$ when n is even; n^2 when n is odd.

In both the above cases neither $f(n) = Og(n)$ nor $g(n) = Of(n)$

Statement (2) is false.

Reason: Consider $g(n) = 2n$ and $h(n) = n$ and $f(n) = 10n$

24. (a)

1. $\frac{1}{2}(n^2) = \omega(n)$

   $\frac{1}{2}(n^2) \geq C_1(n)$

   So true always

2. $\frac{1}{2}(n^2) = \omega(n^2)$

   $\frac{1}{2}(n^2) \geq C_1(n^2)$

   So false, since $\Omega(n^2)$ is true.

25. (c)

$f(n) = 2^{\log_2 n} = n^{\log_2 2} = n$

$g(n) = n^{\log_n}$

$h(n) = n^{\frac{1}{\log n}} = \sqrt[\log n]{n}$

$[n > \sqrt[\log n]{n}$ for all large value of $n]$

[it is less than n since max power of n is always less than 1 for large value of n.]

So, $g(n) \geq f(n) \geq h(n)$

So, $f(n) = O(g(n))$ and $g(n) = \Omega(h(n))$

So, option (c) is correct.

26. (c)

If $f = O(h)$

and $g = O(k)$

Then, $(f + g) = O(n + k)$

and $(f \times g) = O(nk)$

27. (c)

$g(n) = \Omega(n) \quad g(n) \geq C_1 \cdot n$

$f(n) = O(n) \quad f(n) \leq C_2 n$

$h(n) = \theta(n) \; C_3 \cdot n \leq h(n) < C_4 \cdot n$

$g(n) \cdot f(n) + h(n)$

$\geq C \cdot n \quad \theta(n)$

$= \Omega(n)$

28. (a)

Time complexity is O(log$n$) to count number of occurrences of i.

29. (b)

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

$$= \frac{1}{k}\sum_{i=1}^{n}[1+1+1+\ldots+n-(i+1)+1 \text{ times}]$$

$$= \frac{1}{k}\sum_{i=1}^{n}[n-i]$$

$$= \frac{1}{k}\left[n\sum_{i=1}^{n}(1) - \sum_{i=1}^{n}(i)\right] = \frac{1}{k}\left[n\cdot n - \frac{n(n+1)}{2}\right]$$

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

30. (a)

```
ing count = 0, N; ....O(1)
for (i = 0; i < N * 2; i++) .....O(N)
{
    for(j = 0; j < i/2; i++) .....O(N/3)
    {
        for(k = 0; k < j * j; k++).....O((n * n)/3)
        {
            count ++;
```

```
    }
  }
}
```

Here, we using the for loop so that

$O\left(n*\left(\frac{n}{3}\right)*(n*n)/3\right)$ so that $O(n \wedge 4)$.

31. (d)

Assuming that base of the log in the question is e.

Let us try to rewrite each of these functions in the form $e^{something}$, to comparison easier.

(i) $e^n/n = \frac{e^n}{e \wedge (\log n)} = e^{(n-\log n)}$

(ii) $e^{n-0.9\log n} = e^{(n-0.9\log n)}$

(iii) $2^n = (e^{\log 2})^n = e^{(n\log 2)}$

(iv) $(\log n)^{n-1} = (e^{\log\log n})^{n-1} = e^{(n\log\log n - \log\log n)}$

Now, if we just compare the exponents of all, we can clearly see that (n log log n – log log n) grows faster than rest.

Note that in option (c), the multiplicative log2 is a constant, and hence grows slower than the multiplicative log log n from option (d).

This implies that $e \wedge (n \log\log n - \log\log n)$ grows the fastest and hence $(\log n) \wedge (n-1)$ grows the fastest. Thus, option (d) is correct.

32. (d)

When ever the value of 'sin n' value is –1, then $g(n) = 1$. {sin n value ranges from –1 to +1}.

Hence $f(n) = O(g(n))$ is violated. Hence I is false.

and when ever the value of 'sin n' is +1, then $g(n) = n^2$

Hence $f(n) = \Omega(g(n))$ is violated. Hence II is false.

33. (d)

$f \in O(g) = f \geq C \cdot g(n)$ $g \in O(f) = g \leq C_2 \times f(G)$

if we take same function for both f and g then a, b and c can be possible for some constant.

(a) $f \in O(g)$ and $g \in O(f)$

(b) $f \in \theta(g)$ and $g \in \theta(f)$

(c) $f \in \Omega(g)$ and $g \in \Omega(f)$

(d) $f \in O(g)$ but $g \notin \Omega(f)$

$f \in O(g) = f \leq C.g(n)$ for some constant and it also implies that $g \in \Omega(f)$ because for some constant $g \leq C.f(n)$ is possible.

So, option (d) is right

34. (c)

For loop will run 10000 times.

Sum is sum of 1st 9999 natural numbers.

sum = 9000 * 10000/2

sum = 49995000 = $\theta(1)$

So, answer is option (c).

35. (a)

In this programme first for loop will run n times and second for loop also run n times, because our array is sorted then second loop will not run and the total time complexity for sorted order will be $O(n)$.

36. (c)

In this programme first for loop will run n times and second for loop also run n times, because our array is reverse sorted then second loop will also run and the total time complexity for reverse sorted order will be $O(n^2)$.

37. (a)

The correct order of these functions is $f_1(n)$, $f_4(n)$, $f(3)(n)$, $f_2(n)$.

The variable n never appears in the formula for $f_1(n)$. So despite the multiple exponentials $f_1(n)$ is constant. Hence, it is asymptotically smaller than $f_4(n)$, which does grow with n. We may rewrite the formula for $f(4)(n)$ to be $f_4(n) = n\sqrt{n} = n^{1.5}$. The value of $f_3(n) = \binom{n}{2}$ is given by the formula $\frac{n(n-1)}{2}$, which is $\theta(n^2)$.

Hence, $f_4(n) = n^{1.5} = O(n^2) = O(f_3(n))$. Finally, $f_2(n)$ is exponential, while $f_3(n)$ is quadratic, meaning that $f_3(n)$ is $O(f_2(n))$.

38. (a)

$f(n) * \log_2(f(n)^c) = O(g(n) * \log_2 g(n))$,

$\Rightarrow$ $C * f(n) * \log_2(f(n)) = O(g(n) * \log_2 g(n))$,

$\Rightarrow$ $f(n) * \log_2(f(n)) = O(g(n) * \log_2 g(n))$,

Even we can check if it always true or not.

(i) $1/n^2 = 0(1/n) \Rightarrow$ True in this (f, g are non-decreasing functions, so, i should not include this example, but even if we use, it is true.)

(ii) $n = O(n^2) \Rightarrow$ Even true for this. So, always true.

39. (a)
$f(n) = O(g(n))$ if $f(n) \le C.g(n)$
$f(n) = \Omega(g(n))$ if $f(n) \ge C.g(n)$
therefore, there is possibility of intersection, therefore option b, c, d will not be empty but,
$f(n) = o(g(n))$ if $f(n) < C.g(n)$
$f(n) = \omega(g(n))$ if $f(n) > C.g(n)$
there is no possibility of intersection therefore it will be empty.
So, answer is (a).

40. (c)

$$\text{dosomething}(n) = \sum_{j=0}^{199}\sum_{k=0}^{n-1}\sum_{m=0}^{i-1} j$$

$$= \sum_{j=0}^{199}\sum_{k=0}^{n-1} j = \sum_{j=0}^{199} j(n-1)$$

$$= n \cdot \frac{199(199+1)}{2} = O(n)$$

41. (b)
We need to calculate upper bound of this series summations.

$$S = 1^k + 2^k + 3^k + \ldots\ldots n^k$$

for $k = 1$, $S = \frac{n*(n+1)}{2} = O(n^2)$

for $k = 2$, $S = \frac{n*(n+1)*(2n+1)}{6} = O(n^3)$

for $k = 3$, $S = \left(\frac{n*(n+1)}{2}\right)^2 = O(n^4)$

So, in this question answer will be $O(n^{k+1})$.
Another solution
Here, inner loop is running for P times which is equal to i^k.
So, total time complexity for all $i(1 \le i <= n)$ will be,

$$T(n) = 1k + 2k + 3k + \ldots\ldots n^k$$
$$= (n+1)k$$
$$= O(n+1)^k)$$

42. (c)
A has sorting time faster than B, generally but not always because quick sort in worst case running time than merge sort an heap sort. Again bubble sort, insertion sort has $O(n)$ time complexity in best case, where in other case has $O(n^2)$ complexity.
So, we cannot say array A sort any array faster than B.
So, option (c) is correct.

43. (b)
Given $A = \log^k n$, $k \ge 1$; $B = n^b$, $p > 0$
$\log A = k * \log(\log n)$
$\log B = p * \log n$
Now, $\log B > \log A$, for large value of n and also log is an increasing function.
So, we can say that $B > A$, for large values of n, hence $A = O(B)$
PS: A = small – oh(B) is best choice.

44. (c)
for ($i = 1$, $i < n$, i++) – n time
{
for ($j = 1$; $j < i * i$, j++)

$$1^2 + 2^2 + 2^3 \ldots\ldots n^2 = \frac{n(n+1)(2n+1)}{6} = O(n^3)$$

for ($k = 1$; $k < j$; k++)
{ $m = m + 1$; } $O(n)$

Total = $O(n) \times O(n^3) = O(n^4)$

45. (a)
Recurrence relation for above program:

$$T(n) = T(n-1) + 1$$
$$= T(n-2) + 1 + 1$$
$$= T(n-3) + 1 + 1 + 1$$

By doing in this way we get
$= T(n-k) + k \times c$ where $n - k = 1$
So, $k = n - 1$

$$= T(n-n+1) + (n-1) \times c$$
$$= T(1) + (n-1) \times c$$
$$= O(n)$$

46. (a)
$k = n$
for $i = 0$
while loop will run for ($\log_2 N$) times as in every iteration k becomes half.
Now, k becomes 0 and while loop becomes false.
for $i = 1$
As $k = 0$, while will run 1 times.
for $i = 2$,
Same 1 times.
.....
.....
for $i = n$, 1 time
$T(n) = (\log n) + 1 + 1 + 1 + \ldots\ldots (n$ + times 1)
$T(n) = O(n)$

47. (d)

For layer value of n, put $n = 2^{64}$

(a) $\left(\sqrt{\log n}\right)^{\log^2 n} = 8^{64\times 64} = 2^{3\times 64\times 64} = 2^{12288}$

(b) $(\log n)^{\sqrt{\log n}} = 64^{\sqrt{8}} = 2^{6\sqrt{8}} = 2^{16.97}$

(c) $n^{10} = 2640$

(d) $2^{2^{\sqrt{\log\log n}}} = 2^{2^{\sqrt{8}}} = 2^{7.10}$

So, option (d) is slowest.

48. (d)

Since $j = j * 2$, from $j = 1$ to $j = n$ run for $\log_2 n$ times and 1 more time condition check when it fails.

So, it will be $\lfloor \log_2 n \rfloor + 1$. It can be check by taking 2-3 example.

49. (d)

$f(n) = 2^n \Rightarrow f(n) = O(2^n)$

$g(n) = n! \Rightarrow g(n) = O(n!)$

$h(n) = n^{\log n} \Rightarrow h(n) = O(n^{\log n})$

[$n < n^2$ means n grows more slowly than n^2]

The asymptotic order of function is as follows:

$1 < \log\log n < \log n < n^{\varepsilon} < n^c < n^{\log n} < c^n < n^n < c^{c^n} < n!$

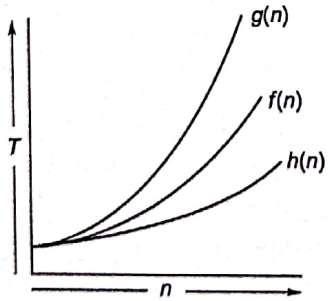
Where $0 < \varepsilon < 1 < c < n$

$\Rightarrow n^{\log n} < c^n < n!$

$\Rightarrow n^{\log n} < 2^n < n!$

Assume $c = 2$

$h(n) < f(n) < g(n)$



$h(n) = O(f(n)) \Rightarrow h(n) \le k.f(n)$ is true

$g(n) = \Omega(f(n)) \Rightarrow g(n) \ge k.f(n)$ is also true

50. (a)

Recurrence relation is

$T(n) = T\left(\sqrt{n}\right) + C$ where C = constant

So, time complexity is O(log log n)

51. (a)

Loop 1 executes: for ($i = 1$; $i <= n$; i++) so, $(n + 1)$ times.

Loop 2 executes: for ($j = 1$; $j <= i$; f++) so,

$$\frac{(n+1)(n+2)}{2} - 1$$

Loop 3 executes: for ($k = 1$; $k <= 100$; k++) so,

$$101 * \frac{n*(n+1)}{2}$$

Total time complexity = $\theta(n^2)$.

52. (a)

Recurrence relation for above program:

$$T(n) = T(n-1) + 1 = T(n-2) + 1 + 1$$
$$= T(n-3) + 1 + 1 + 1$$

By doing in this way we get

$$= T(n-k) + k \times c$$

Where $n - k = 1$

So, $k = n - 1 = T(n - n + 1) + (n-1) \times c$

$= T(1) + (n-1) \times c = O(n)$

53. (c)

Lets see the nature of $\frac{e^n}{n+2^n}$

$$\Rightarrow \frac{e^n}{n+2^n} = e^{\log(e^n) - \log(n+2^n)}$$

We find that,

$$\lim_{x\to\infty} \log(e^n) - \log(n+2^n)$$

$$\Rightarrow \lim_{x\to\infty} 0.7n - \log n$$

$$\Rightarrow \lim_{x\to\infty} n * \lim_{x\to\infty}\left(0.7 - \frac{\log n}{n}\right)$$ is infinite as n approaches to infinite.

$$\Rightarrow \frac{e^n}{n*2^n} = \infty$$

$$\Rightarrow e^n = \omega(n*2^n)$$

54. (d)

$\sqrt{n} = n^{1/2} = n^{0.5}$

$2^{\sqrt{\log_2 n}}$

$n^{1.5}$

$n^{1.6}$

10^n

$2^{\sqrt{\log n}} \le n^{0.5} \le n^{1.5} \le n^{1.6} \le 10^n$

55. (b)

Time complexity of $g(m)$ is : $\Theta(\log m)$

[Note: $f(n)$ returns the value 2^n]

$$g(x) = g(2^n) = g(m) \Rightarrow m = 2^n$$

$$\therefore \quad g(m) = \Theta(\log m) = \Theta(\log 2^n) = \Theta(n)$$

$$\therefore \quad g(m) = \Theta(n)$$

Recurrence relation for $f(n)$:

$$f(n) = f(n-1) + g(m)$$
$$= f(n-1) + \Theta(n) = \Theta(n^2)$$

56. (d)

$$2^{\log_2 n} = n$$

$$\log n \le \sqrt{n} \le n \le 2^n$$

57. (a)

The correct ordering of these functions is $f_4(n)$, $f_1(n)$, $f_3(n)$, $f_2(n)$.

To see why, we first we the rules of arithmetic series to derive a simpler formula for $f_4(n)$:

$$f_4(n) = \sum_{i=1}^{n}(i+1) = \frac{n((n+1)+2)}{2} = \frac{n(n+3)}{2} = \theta(n^2)$$

This is clearly asymptotically smaller than $f_1(n) = n^{\sqrt{n}}$. Next, we want to compare $f_1(n)$, $f_2(n)$ and $f_3(n)$.

To do so, we transform both $f_1(n)$ and $f_3(n)$ so that they look more like $f_3(n)$:

$$f_1(n) = n^{\sqrt{n}} = \left(2^{\log n}\right)^{\sqrt{n}} = 2^{\sqrt{n}\log n}$$

$$f_3(n) = n^{10} \cdot 2^{n/2} = 2^{\lg(n^{10})} \cdot 2^{n/2} = 2^{n/2+10\log n}$$

The exponent of the 2 in $f_1(n)$ is a function that grows more slowly than linear time; the exponent of the 2 in $f_3(n)$ is a function that grows linearly with n. Therefore, $f_1 = O(f_3(n))$.

Finally, we wish to compare $f_3(n)$ with $f_2(n)$. Both have a linear function of n in their exponent, so its tempting to say that they behave the same asymptotically, but they do not, if C is any constant and $g(x)$ is a function, then $2^{cg(x)} = (2^c)^{g(x)}$ hence, changing the constant of the function in the exponent is the same as changing the base of exponent, which does affect the asymptotic running time. Hence, $f_3(n)$ is $O(f_2(n))$, but $f_2(n)$ is not $O(f_3(n))$.

58. (c)

X = Sum of cubes of {1, 2, 3, ...n}

$X = n^2(n+1)^2/4$

1. $X = \theta(n^4)$ True

$$C_1 \cdot (n^4) <= \frac{n^2(n+1)^2}{4} <= C_2 \cdot (n^4)$$

2. $X = \theta(n^5)$ False

$$C_1 \cdot (n^5) <= \frac{n^2(n+1)^2}{4} \le C_2(n^5) \text{ not possible}$$

3. $X = O(n^5)$ True $\dfrac{n^2(n+1)^2}{4} <= C \cdot n^5$

4. $X = \Omega(n^3)$ True $\dfrac{n^2(n+1)^2}{4} \ge = C(n^3)$

59. (d)

Option (d) is correct because

(i) $g(n) = O(f(n))$ since $\log n^2 \le n^{1/2}$

(ii) $g(n) = O(f(n))$ since $\log n \le \log (n^2)$

(ii) $g(n) = O(f(n))$ since $10n^2 \le 2^n$

(iv) $f(n) = O(g(n))$ since $2n^2 \le 3^n$

60. (c)

Assume that each simple statement executes in constant time.

$$= \sum_{i=1}^{n}\sum_{j=i}^{2n}(1) = \sum_{i=1}^{n}(1+1+1+...+(2n-i+1)\text{times})$$

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

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

$$= \frac{\left(4n^2 - n^2 - n + 2n\right)}{2} = \frac{\left(3n^2+n\right)}{2} = O(n^2)$$

61. (b)

$f(x)$ grows exponentially as a function of x.

Hence option (b) is correct.

62. (b)

In the i^{th} iteration, the inner loop run $O(n/i)$ times, and therefore, the overall complexity is

$O\left(n + \frac{n}{1} + \frac{n}{2} + ... + 1\right)$ and this is $O(n \log n)$.

63. (a)

In this question total possibility is four but only two possibility will give solution and other two possibilities will not give solutions.

$g(n) = O(f(n))$ in case of $0 \le n < 100$ and $0 \le n < 10{,}000$ (it is also not possible)

$g(n) = \Theta(f(n))$ otherwise

From the above two equations the conclusion is $f(n)$ is same as $O(g(m))$ and $g(n)$ and $f(n)$ will be $O(n^2)$.

64. (b)

$$T(n) = 4.T\left(\frac{n}{2}\right) + n^2 + 1$$

$$= \Theta(n^2 \log n)$$

Using master's theorem

$$n^{\log_b a} = n^{\log_2 4} = 2$$

$$n^2 = n^2$$

$\therefore$ $O(n^2 \log n)$

Case 2 is applied. So, option (b) is correct.

65. (d)

This loop iterates as follows:

$i = 2$

$i = 2 \wedge 2$

$i = (2 \wedge 2) \wedge 2$

.....

.....

$i = 2 \wedge (2 \wedge k) <= n$

$k = \log \log n$

So, complexity = $O(\log \log n)$

66. (b)

Since the variable j is not initialized for each value of variable i. So, the inner loop runs atmost n times.

67. (b)

Note that there are nC_2 many pairs of points that the algorithm takes. This is equal to $O(n^2)$ running time.

68. (a)

$$f(n) = \Omega(n) \Rightarrow f(n) \ge c \cdot n$$

$$g(n) = O(n) \Rightarrow g(n) \le c \cdot n$$

$$h(n) = \Theta(n) \Rightarrow c_1 \cdot n \le h(n) \le c_2 \cdot n$$

$$f(n) \cdot g(n) = c \cdot n$$

$[\because f(n) \ge c \cdot n \text{ and } g(n) \le c \cdot n]$

$$\underbrace{f(n).g(n)}_{\ge c.n} + \underbrace{h(n)}_{\theta(n)} = \Omega(n)$$

69. (c)

Recurrence relation for the function find:

$f(n) = 0$; if $n = 0$

$= f(n-1) + 1$; if $n > 0$

Time complexity of $f(n) = O(n)$

70. (c)

In this program, for loop run $n/2$ times and in this 'for loop' $y(i)$ have $(\log_2 n)$ time complexity.

In the above we will select maximum time complexity i.e. $\frac{n}{2}\log_2 n$.

Then the total time complexity for the 'for loop' will $O(n \log n)$

71. (c)

Cyclomatic complexity of program = Predicate statement + 1

$= 3 + 1 = 4$

72. (c)

$$\text{dosomething}(n) = \sum_{j=0}^{199} \sum_{k=0}^{n-1} \sum_{m=0}^{i-1} j$$

$$= \sum_{j=0}^{199} \sum_{k=0}^{n-1} j = \sum_{j=0}^{199} j(n-1)$$

$$\cong \frac{n \cdot 199(199+1)}{2} = O(n)$$

73. (c, d)

The time complexity of the above code will be $O(n^3)$.

Thus, (c) and (d) are both true.

74. (a, b)

Denote the time to run this algorithm $T(n)$, (we use n here instead of b to avoid name clash with the master method variables), so we have:

$$T(n) = T(n/2) + O(1)$$

Using the master method terminologies, we have time complexity $O(\log n)$.

75. (a, b, d)

Only option (c) is false since $f(n)$ is quadratic function which is obviously greater than $h(n)$ which is linear function of n.

76. (a, c)

Since, both the modules are independent thus both option (a) and (c) would be correct.

2 CHAPTER

Recurrence Relations

Multiple Choice Questions & NAT Questions

Q.1 Let $T(n) = [n(\log(n^3) - \log n) + \log n]n + \log n$
Find complexity of $T(n)$.
(a) $O(n \log n)$ (b) $O(n^2)$
(c) $O(n^2 \log n)$ (d) $O(n^3)$

Q.2 Let T_n be the recurrence relation is defined as follows:

$$T_n = \begin{cases} 0, & n = 0 \\ T_{n-1} + n, & n \geq 1 \end{cases}$$

Find the value of T_n?
(a) n (b) n^2
(c) $\frac{n^2 + n}{2}$ (d) $\frac{n^2 - n}{2}$

Q.3 What is time complexity of recurrence equation

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

(a) $O(\log n)$ (b) $O(n^2)$
(c) $O(n \log \log n)$ (d) $O(\log \log n)$

Q.4 What is complexity of recurrence equation
$T(n) = \text{sqrt}(2) + (n/2) + \text{sqrt}(n)$
(a) $\theta(n(\log n + 1))$ (b) $\theta(n^2(\log n + 1))$
(c) $\theta(\sqrt{n}(\log n + 1))$ (d) $\theta(n^3(\log n + 1))$

Q.5 $T(n) = T\left(\frac{2n}{3}\right) + 1$ then $T(n)$ is equal to
(a) $\Theta(\log_2 n)$ (b) $\Theta(n \log_2 n)$
(c) $\Theta(n^2)$ (d) $\Theta(n)$

Q.6 The running time of an algorithm is given by

$$T(n) = T(n-1) + T(n-2) - T(n-3), \text{ if } n > 3$$
$$n, \text{ otherwise}$$

The order of this algorithms is
(a) $O(n)$ (b) $O(\log n)$
(c) $O(n^n)$ (d) $O(n^2)$

Q.7 What should be the relation between $T(1)$, $T(2)$ and $T(3)$, so that in above question gives an algorithm whose order is constant?
(a) $T(1) = T(2) = T(3)$ (b) $T(1) + T(3) = 2T(2)$
(c) $T(1) - T(3) = T(2)$ (d) $T(1) + T(2) = T(3)$

Q.8 What is complexity of recurrence eq.

$$T(n) = 2T(n/4) + \sqrt{3}$$

(a) $O(\sqrt{n})$ (b) $O(n)$
(c) $O(n^2)$ (d) $O(n \log n)$

Q.9 Which recurrence relation satisfy the sequence: 2, 3, 4, ..., for $n \geq 1$.
(a) $T(N) = 2T(N-1) - T(N-2)$
(b) $T(N) = 2T(N-1) + T(N-2)$
(c) $T(N) = N + 1$
(d) None of these

Q.10 Which one of the following correctly determines the solution of the recurrence relation with $T(1) = 1$?

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

(a) $\Theta(n)$ (b) $\Theta(n \log n)$
(c) $\Theta(n^2)$ (d) $\Theta(\log n)$

Q.11 The time complexity of computing the transitive closure of a binary relation on a set of n elements is known to be
(a) $O(n \log n)$ (b) $O(n^{3/2})$
(c) $O(n^3)$ (d) $O(n)$

Q.12 Let $T(n) = T(n-1) + \frac{1}{n}$; $T(1) = 1$; Then $T(n) = ?$
(a) $O(n^2)$ (b) $O(n \log n)$
(c) $O(\log n)$ (d) $O(n^2 \log n)$

Q.13 A certain problem is having an algorithm with the following recurrence relation.

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

How much time would the algorithm take to solve the problem?
(a) $O(\log n)$ (b) $O(n \log n)$
(c) $O(\log \log n)$ (d) $O(n)$

Q.14 Consider the following recurrence relation:

$T(n) = 8T\left(\frac{n}{2}\right) + n^2 \log n, n > 1$ and

$T(1) = 1$

Find the time complexity of $T(n)$

(a) $O(n^2)$ (b) $O(n^2 \log n)$
(c) $O(n^3)$ (d) $O(n^3 \log n)$

Q.15 What is complexity of recurrence (n is a natural number)

$$T(n) = \begin{cases} 7T(n/3) + n^3; & n > 2 \\ 1 & ; n \le 2 \end{cases}$$

(a) $O(n)$ (b) $O(n^2)$
(c) $O(n^3)$ (d) $O(n \log n)$

Q.16 What is time complexity of the following recurrence relation and step to derive the same.

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

(a) $\log \log n$ (b) n
(c) $(\log \log n)^2$ (d) n^2

Q.17 Let $T(n) = T(n/5) + T(7n/10) + c.n$, where c is a constant. Find running time of $T(n)$.

(a) $\theta(n)$ (b) $\theta(n \log n)$
(c) $\theta(n^2)$ (d) None of these

Q.18 Let $T(n) = T(n-1) + \frac{1}{n}$. Then $T(n)$ is

(a) $\Theta(1)$ (b) $\Theta(\log n)$
(c) $\Theta(\log \log n)$ (d) $\Theta(n)$

Q.19 Given $T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{n}{2}\right) + n^2$. Then

(a) $T(n) = \theta(n^3)$ (b) $T(n) = \theta(n^2 \log n)$
(c) $T(n) = \theta(n^2)$ (d) $T(n) = \theta(n^3 \log n)$

Q.20 What is the value of $T(n)$ for given recurrence relation

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

when n is power of 2

(a) $2(\log n + 1)$ (b) $2 \log n$
(c) $\log n + 1$ (d) $2 \log n + 1$

Q.21 What is the time complexity of the following recurrence relation $T(k) = T\left(\frac{k}{2}\right) + T\left(\frac{k}{2}\right) + k^2$?

(a) $O(k)$ (b) $O(k^2)$
(c) $O(k \log k)$ (d) $O(k^2 \log k)$

Q.22 Which of the following function given by their recurrences grows the fastest asymptotically?

(a) $T(n) = 8T\left(\frac{n}{4}\right) + 100n^2$

(b) $T(n) = 81T\left(\frac{n}{9}\right) + 10n^2$

(c) $T(n) = 16T\left(\frac{n}{4}\right) + 100(n \log n)^{1.99}$

(d) $T(n) = 100T\left(\frac{n}{100}\right) + n \log^2 n$

Q.23 Let $T(n)$ be the function defined by $T(1) = 1$,

$T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right)\sqrt{n}$ for $n \ge 2$

Which of the following statements is true?

(a) $T(n) = O(\sqrt{n})$ (b) $T(n) = O(n)$
(c) $T(n) = O(\log n)$ (d) None of these

Q.24 Solve the recurrence relation to find:

$T(n),\ T(n) = 4T\left(\frac{n}{2}\right) + n$

(a) $\Theta(\log_2 n)$ (b) $\Theta(n^2)$
(c) $\Theta(n^2 \log_2 n)$ (d) None of these

Q.25 Solve $T(n) = 2T\left(\frac{n}{2}\right) + n^3$

(a) $\Theta(n^3)$ (b) $(n^3 \log_n n)$
(c) $\Theta(n \log n_2 n)$ (d) None of these

Q.26 What is complexity of following recurrence relati

$T(n) = \sqrt{n}T(\sqrt{n}) + 100n$

(a) $\theta(n \log \log n)$ (b) $\theta(n \log n \log n)$
(c) $\theta(n)$ (d) $\theta(n \log n)$

Q.27 What is complexity of recurrence relation

$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + T\left(\frac{n}{8}\right) + n$

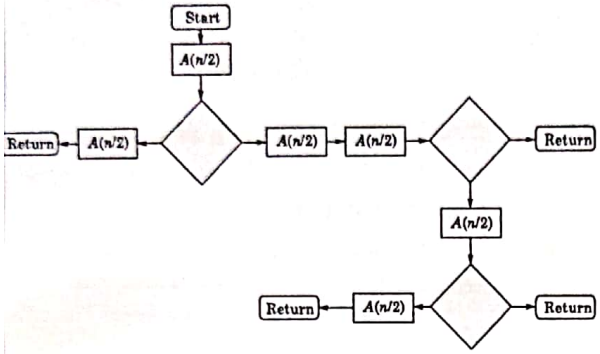
(a) $\theta(n)$ (b) $\theta(n^2)$
(c) $\theta(n \log n)$ (d) $\theta(n^3)$

Q.28 The solution of the Recurrence relation $T(m) = T(3m/4) + 1$ is

(a) $\theta(\log m)$ (b) $\theta(m \log m)$
(c) $\theta(m)$ (d) $\theta(\log \log m)$

Q.29 The given diagram shows the flow chart for a recursive function $A(n)$. Assume that all statements, except for the recursive calls, have O(1) time complexity. If the worst case time complexity of this function is $O(n^{\alpha})$, then the least possible value (accurate up to two decimal positions) of α is _____.

Flow chart for Recursive Function $A(n)$



Q.30 What is complexity of recurrence relation

$$T(n)=T\left(\frac{n}{4}\right)+7\left(\frac{3n}{4}\right)+n$$

(a) $\theta(n\log_3 n)$ (b) $\theta(n\log\log_{4/3} n)$

(c) $\theta(n^2\log_{4/3} n)$ (d) $\theta(n\log_{4/3} n)$

Q.31 Which of the following can be solved using Master theorem?

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

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

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

(d) None of these

Q.32 Consider the following recurrence relation:

$$T(n)=7T\left(\frac{n}{2}\right)+an^2$$

What will be the solution of above recurrence?

(a) $O(n^2)$ (b) $O(n^3)$

(c) $O(n^2\log n)$ (d) $O(n^{2.81})$

Q.33 What is complexity of recurrence relation

$T(n) = T(n-1) + 2n$ where $T(n) = 1$

(a) $\theta(n)$ (b) $\theta(n^2)$

(c) $\theta(n^3)$ (d) $\theta(n^2\log n)$

Q.34 The running time of an algorithm $T(n)$, where n is the input size, is given by following:

$$T(n)=\begin{cases}8T(n/2)+qn & \text{if } n>1\\ p & \text{if } n=1\end{cases}$$

where p and q are constants, the order of algorithm is

(a) n^2 (b) n^3

(c) n (d) n^n

Q.35 We solve TOH problem recursively breaking the task in three sections, which of the following recurrence will accord will with the approach, that is shows correct order of work done on each recursive step.

(a) $T(n) = T(n-1) + 1 + T(n-1)$

(b) $T(n) = T(n-1) + 1T(n-1) + 1$

(c) $T(n) = 1 + T(n-1) + T(n-1)$

(d) $T(n) = T(n-1) + T(n-1) + 2$

Q.36 What is complexity of recurrence relation

$$T(n)=\sqrt{2}T\left(\frac{n}{2}\right)+C, \text{ for } n>1$$

$= a$ for $n = 1$

(a) $\theta(\sqrt{n})$ (b) $\theta(\log n)$

(c) $\theta(n)$ (d) $\theta(\sqrt{n}\log n)$

Q.37 Let $T(n)$ be defined by $T(0) = T(1) = 4$ and

$$T(n)=T\left(\left\lfloor\frac{n}{2}\right\rfloor\right)+T\left(\left\lfloor\frac{n}{4}\right\rfloor\right)+Cn$$ for all integers $n >= 2$, where C is a positive constant. What is asymptotic growth of $T(n)$?

(a) $\theta(n)$ (b) $\theta(n\log n)$

(c) $\theta(n^2)$ (d) $\theta(n^{\log_3 4})$

Q.38 The recurrence equation:

$T(1) = 1$

$T(n) = 2T(n-1) + n, n \geq 2$

Evaluates to

(a) $2^{n+1} - n - 2$ (b) $2^n - n$

(c) $2^{n+1} - 2n - 2$ (d) $2^n + n$

Q.39 What will be time complexity for the following recurrence relation

$T(n) = 8\sqrt{n}\,T(\sqrt{n}) + (\log n)^2$

(a) $O(n\log n)$ (b) $O(n(\log n)^2)$

(c) $O(n(\log n)^3)$ (d) $O(n(\log n)^4)$

Q.40 Suppose $T(n) = 2T\left(\frac{n}{2}\right) + n$, $T(0) = T(1) = 1$

Which one of the following is FALSE?

(a) $T(n) = O(n^2)$ (b) $T(n) = \Theta(n \log n)$

(c) $T(n) = \Omega(n^2)$ (d) $T(n) = O(n \log n)$

Q.41 Solve the recurrence relation and find complexity of it

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

(a) $O(n^3)$ (b) $O(n)$

(c) $O(n^2)$ (d) $O(n^2 \log n)$

Q.42 An algorithm is made up of 2 modules $m1$ and $m2$. If order of $M1$ is $f(n)$ and $M2$ is $g(n)$ then the order of the algorithm is

(a) $\max(f(n), g(n))$

(b) $\min(f(n), g(n))$

(c) $f(n) + g(n)$

(d) $f(n) \times g(n)$

Q.43 Counting combinations can be also solved using divide and conquer approach

$$\left[\binom{n}{r} = \binom{n-1}{r-1} + \binom{n-1}{r}\right]$$

Consider the following function to compute n things chosen r at a time.

```
function X(n, r)
{   if (r = 0 or n = r) then return 1;
    else
    return (X(n − 1, r − 1) + X(n − 1, r));
}
```

Find the worst case time complexity of $X(n, r)$?

(a) $O(n)$ (b) $O(n^2)$

(c) $O(2^n)$ (d) None of these

Q.44 The recurrence relation:

$T(1) = 2$

$T(n) = 3T\left(\frac{n}{4}\right) + n$, has the solution then $T(n)$ equals to

(a) $O(n)$

(b) $O(\log n)$

(c) $O(n^{3/4})$

(d) None of these

Q.45 The time complexity of the following C function (assume $n > 0$)

```
int recursive (int n)
{   if (n == 1)
        return (1);
    else
        return (recursive (n − 1) + recursive (n − 1));
}
```

(a) $O(n)$ (b) $O(n \log n)$

(c) $O(n^2)$ (d) $O(2^n)$

Q.46 What is the time complexity of the following recursive function:

```
int DoSomething (int n)
{
    if (n < = 2)
        return 1;
    else
        return DoSomething (floor (sqrt (n))) + n;
}
```

(a) $\Theta(n^2)$ (b) $\Theta(n \log_2 n)$

(c) $\Theta(\log_2 n)$ (d) $\Theta(\log_2 \log_2 n)$

Q.47 Solve the recurrence:

$$T(n) = T\left(\frac{n}{2} - 1\right) + T\left(n - \frac{n}{2}\right) + \theta(n)$$

(a) $\theta(n \log n)$ (b) $\theta(n)$

(c) $\theta(n^2 \log n)$ (d) $\theta(n^2)$

Q.48 When $n = 2^{2k}$ for some $k \geq 0$, the recurrence relation $T(n) = \sqrt{2}\,T\left(\frac{n}{2}\right) + \sqrt{n}$, $T(1) = 1$ evaluates to

(a) $\sqrt{n}(\log n + 1)$ (b) $\sqrt{n} \log n$

(c) $\sqrt{n} \log \sqrt{n}$ (d) $n \log \sqrt{n}$

Q.49 What is complexity of recurrence relation

$$T(n) = 3T\left(\frac{n}{2} + 47\right) + 2n^2 + 10n - \frac{1}{2}$$

(a) $O(n^2)$ (b) $O(n^{3/2})$

(c) $O(n \log n)$ (d) None of these

Q.50 Consider the following recurrence relation:

$$T(n) = 7T\left(\frac{n}{2}\right) + an^2$$

What will be the solution of above recurrence?

(a) $O(n^2)$ (b) $O(n^3)$

(c) $O(n^2 \log n)$ (d) $O(n^{2.81})$

Q.51 In Tower of Hanoi there are three towers: left, right and middle. There are n discs in left tower in a particular sequence (ascending order), we have to transfer these discs into right tower having same sequence using middle tower (consider that disc with smaller sequence number is always at top of large sequence number i.e. 1 is above 2 but 2 above 1 is not allowed). If Towers of Hanoi is implemented with recursive function then total discs moves to move 9 discs from left to right are _____.

Q.52 Let m, n be positive integers. Define $Q(m, n)$ as

$Q(m, n) = 0$, if $m < n$

$Q(m - n, n) + p$, if $m \geq n$

Then $Q(m, 3)$ is (a div b, gives the quotient when a is divided by b)

(a) a constant (b) $p \times (m \bmod 3)$

(c) $p \times (m \text{ div } 3)$ (d) $3 \times p$

Q.53 Find complexity of recurrence relation

$T(n) = T(n-1) + T(n-2) + C$

(a) $O(2^n)$ (b) $O(n^2)$

(c) $O(n)$ (d) $O(n^n)$

Q.54 The recurrence relation capturing the optimal execution time of the Towers of Hanoi problem with n discs is

(a) $T(n) = 2T(n-2) + 2$

(b) $T(n) = 2T(n-1) + n$

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

(d) $T(n) = 2T(n-1) + 1$

■■■■

Answers Recurrence Relations

1.	(c)	2.	(c)	3.	(d)	4.	(c)	5.	(a)	6.	(a)	7.	(a)	8.	(a)	9.	(a)
10.	(a)	11.	(c)	12.	(b)	13.	(c)	14.	(c)	15.	(b)	16.	(b)	17.	(a)	18.	(b)
19.	(c)	20.	(d)	21.	(b)	22.	(a)	23.	(a)	24.	(b)	25.	(a)	26.	(a)	27.	(a)
28.	(a)	29.	(2.32)	30.	(d)	31.	(b)	32.	(d)	33.	(b)	34.	(b)	35.	(a)	36.	(a)
37.	(a)	38.	(a)	39.	(c)	40.	(c)	41.	(c)	42.	(a)	43.	(c)	44.	(a)	45.	(d)
46.	(d)	47.	(a)	48.	(a)	49.	(a)	50.	(d)	51.	(511)	52.	(c)	53.	(a)	54.	(d)

Explanations Recurrence Relations

1. (c)

$[n(\log(n^3) - \log n) + \log n]\, n + \log n$

$= \left[n\left(\log\frac{n^3}{n}\right) + \log n\right] n + \log n$

$= [n \log n^2 + \log n]n + \log n$

$= n^2 . 2 \log n + n \log n + \log n$

$= 2n^2 \log n + n \log n + \log n = O(n^2 \log n)$

So option (c) is correct.

2. (c)

$T(n) = T(n-1) + n$

By Repetitive Substitution

$T(n) = [T(n-2) + n - 1] + n$

$= [(n-2) + T(n-3] + (n-1) + n$

$= 0 + 1 + 2 + \ldots + n$

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

So option (c) is correct.

3. (d)

Since, we have a square root term, considering only perfect squares and those which are multiple of 2 as that can take care of log.

$T(2) = 1$ //Assume

$T(2^2) = T(2) + 1 = 2$

$T(2^{2^2}) = T(4) + 1 = 3$

$T(2^{2^3}) = T(16) + 1 = 4$

$T(2^{2^4}) = T(256) + 1 = 5$

So, we are getting

$$T(n) = \log \log n + 1$$

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

4. (c)

From Master theorem, case 2

Here, $a = \sqrt{2}$, $b = 2$, $k = 1/2$, $p = 0$

$a = b \wedge k$ which is true.

$p > -1$

So, complexity is theta ($n \wedge$ (log a base to b) log $\wedge (p+1)n$

$$= \theta(\sqrt{n}(\log n + 1))$$

5. (a)

$$T(n) = T\left(\frac{2n}{3}\right) + 1$$

By using recursive tree method:

$$T(n)$$

$$\downarrow$$

$$T\left(\frac{2n}{3}\right) = O(1)$$

$$\downarrow$$

$$T\left(\frac{2^2 n}{3^2}\right) = O(1)$$

$$\downarrow$$

$$T\left(\frac{2^3 n}{3^3}\right) = O(1)$$

$$\vdots$$

$$T\left(\frac{2^k n}{3^k}\right) = O(1)$$

So, height of tree is $\log_{(3/2)} n$.

So, $T(n)$ = Height × Work at each level

$$= \log_{3/2} n \times O(1) = O(\log_{3/2} n)$$

6. (a)

$$T(n) = \begin{cases} T(n-1) + T(n-2) - T(n-3) & n > 3 \\ n & \text{otherwise} \end{cases}$$

$T(4) = T(3) + T(2) - T(1)$

$= 3 + 2 - 1 = 4$

$T(5) = T(4) + T(3) - T(2)$

$= 4 + 3 - 2 = 5$

So by induction,

$T(n) = T(n-1) + T(n-2) - T(n-3)$

$= n - 1 + n - 2 - (n - 3)$

$= 2n - 3 - n + 3 = n$

So, $T(n) = O(n)$

7. (a)

Let, $T(1) = T(2)$

$= T(3) = k$ (say)

Then, $T(4) = k + k - k = k$

$T(5) = k + k - k = k$

By mathematical induction it can be proved tha $T(n) = k$, a constant.

8. (a)

In these type of questions, where master theorem does not apply you can do the following

$$T(n) = 2T\left(\frac{n}{4}\right) + \sqrt{3}$$

Reframe it like this, by putting i in the specified positions.

$$T(n) = 2^i T\left(\frac{n}{4^i}\right) + i\sqrt{3}$$

Now, for $\frac{n}{4^i} = 1$, put $i = \log_4 n$

So, the equation becomes

$$T(n) = 2^{\log_4 n} + \log_4 n\sqrt{3}$$

or

$$T(n) = n^{\log_4 2} + \log_4 n\sqrt{3}$$

$$T(n) = \sqrt{n} + \log_4 n\sqrt{3}$$

and therefore, $T(n) = O(\sqrt{n})$

9. (a)

Take base conditions as $T(0) = 1$ and $T(-1) = 0$.

$T(-1) = 0$

$T(1) = 2T(0) - T(-1) = 2$

$T(2) = 2T(1) - T(0)$

$= 3$ ... and so on

So, option (a) is correct.

10. (a)

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

[By Master's Theorem case1]

$$= \Theta(n)$$

11. (c)

The time complexity of computing the transitiv closure of a binary relation on a set of n-elemer will take $O(n^3)$ using Floyd-Warshall's algorithm.

12. (b)

$$T(n) = T(n-1)+\frac{1}{n}$$

$$T(n-1) = T(n-2)+\frac{1}{n-1}$$

$$T(n-2) = T(n-3)+\frac{1}{n-2}$$

$$T(n-3) = T(n-4)+\frac{1}{n-3}$$

$$T(n) = T(n-2)+\frac{1}{n-1}+\frac{1}{n}$$

$$= T(n-3)+\frac{1}{(n-2)}+\frac{1}{(n-1)}+\frac{1}{n}$$

$$T(n) = T(n-4)+\frac{1}{(n-3)}+\frac{1}{(n-2)}+\frac{1}{(n-1)}+\frac{1}{n}$$

$$T(n) = T(n-k)+\frac{1}{(n-(k-1))}+\frac{1}{(n-(k-1))}$$

$$+\frac{1}{(n-(k-3))}+\frac{1}{(n-(k-4))}+...+\frac{1}{n}$$

$$T(n) = T(n-k)+\frac{1}{(n-k+1)}+\frac{1}{(n-k+2)}$$

$$+\frac{1}{(n-k+3)}+...+\frac{1}{n-1}+\frac{1}{n}$$

$$T(n) = T(n-k)+\frac{1}{n-k+1}+\frac{1}{n-k+2}+\frac{1}{n-k+3}+...$$

$$...+\frac{1}{n-3}+\frac{1}{n-2}+\frac{1}{n-1}+\frac{1}{n}$$

Thing that the one condition is given if $(n > 1)$
So, we can substitute $(k = n-1)$

$$T(n) = T(n-(n-1))+\frac{1}{n-(n-1)+1}+...+\frac{1}{n}$$

$$T(n) = T(1)+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}+...+\frac{1}{n}$$

Assume that $T(1) = 1$

$$T(n) = \frac{1}{1}+\frac{1}{2}+\frac{1}{3}+....+\frac{1}{n}$$

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

13. (c)

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

Put $n = 2^m$, $\because m = \log n$

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

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

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

Using Master's Theorem

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

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

14. (c)

Using Master Theorem:

$$T(n) = aT\left(\frac{n}{b}\right)+f(n)$$

$$n^{\log_b^a} = n^{\log_2^8} = n^3$$

$$f(n) = n^2 \log n$$

If $n^{\log_b^a} > f(n)$ then

$$T(n) = O(n^{\log_b^a})$$

$\Rightarrow$ $n^3 > n^2$

$\Rightarrow$ $T(n) = O(n^3)$

So option (c) is correct.

15. (b)

According to Master theorem
$a = 7$, $b = 3$, $k = 2$
So, its case of $a < b^k$
So, $O(N^2)$ will be complexity
So, option (b) is correct.

16. (b)

Take $n = 2^m$ then,

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

Solve using Master's theorem

$$S(m) = O(\log^2 m) = T(2^m)$$

Put $m = \log n$ $\Rightarrow$ $T(n) = (\log \log n)^2$
So, option (b) is correct.

17. (a)

$$T(n) = T\left(\frac{n}{5}\right)+T\left(\frac{7n}{10}\right)+n$$

$$= \sum_{i=0}^{\log_{\frac{10}{9}}^n}\left(\frac{9}{10}\right)^i \cdot n = \Theta(n)$$

18. (b)

$$T(n) = T(n-1)+\frac{1}{n} \quad ...(1)$$

$$= \left[T(n-2)+\frac{1}{n-1}\right]+\frac{1}{n}$$

$$= T(n-2)+\frac{1}{n-1}+\frac{1}{n} \quad ...(2)$$

$$= T(n-3)+\frac{1}{n-2}+\frac{1}{n-1}+\frac{1}{n} \quad ...(3)$$

$$= T(1) + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{n-2} + \frac{1}{n-1} + \frac{1}{n} \quad \ldots(n-1)$$

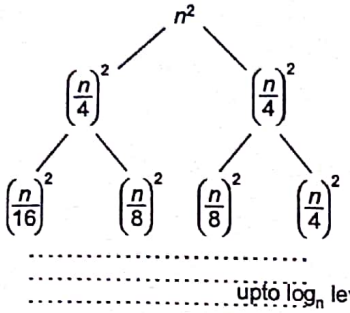
$$= T(1) - 1 + \left[1 + \frac{1}{2} + \frac{1}{3} + \ldots\ldots + \frac{1}{n}\right]$$

$$= T(1) - 1 + O(\log n)$$

$$= O(1) + O(\log n) = O(\log n)$$

19. (c)

By Recursion Tree method,



$$T(n) = n^2\left(1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 \ldots\ldots \log n \text{ times}\right)$$

$T(n) = n^2(1)$ (it is decreasing GP so it will be close to 1)

$T(n) = \theta(n^2)$

20. (d)

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

by substitution method,

$$T\left(\frac{n}{2}\right) = T\left(\frac{n}{4}\right) + 2$$

$$T\left(\frac{n}{4}\right) = T\left(\frac{n}{8}\right) + 2$$

$$T(n) = T\left(\frac{n}{8}\right) + 2 + 2 + 2$$

$$T(n) = T\left(\frac{n}{(2)^3}\right) + 3 * 2$$

$$T(n) = T\left(\frac{n}{(2^k)}\right) + k * 2$$

Let $2k = n$ i.e., $k = \log n$

$$T(n) = T\left(\frac{n}{(2^{\log n})}\right) + \log n * 2$$

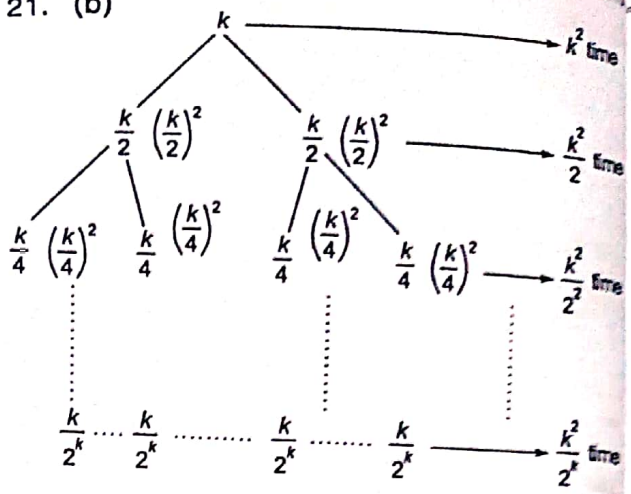
$$T(n) = T(1) + 2 * \log n$$

Since, $T(1) = 1$

$$T(n) = 2 * \log_2 n + 1$$

So, option (d) correct.

21. (b)



$$T(n) = \frac{k^2}{2^0} + \frac{k^2}{2^1} + \frac{k^2}{2^2} + \ldots \frac{k^2}{2^k}$$

$$T(k) = k^2\left[1 + \frac{1}{2} + \frac{1}{4} + \ldots + \frac{1}{2^k}\right]$$

$$= k^2 \times O(1) = O(k^2)$$

22. (a)

A function F is asymptotically larger than function g means for large values all numbers greater than some finite number) $f(n) > g(n)$

$$T(n) = 81T\left(\frac{n}{9}\right) + 10n^2$$

This recurrence will asymptotically grow faster than all the other options given.

By applying Master's theorem

(a) $T(n) = 8T\left(\frac{n}{4}\right) + 100n^2$

Recursive part, $n^{\log_b a} = n^{\log_4 8}$ and $f(n) = 100n^2$

Since, $f(n)$ is polynomially (by a factor of some n^{ϵ}, $\epsilon > 0$) larger, and $f(n)$ is regular, at per Master theorem.

Case 3. $T(n) = \theta(f(n)) = \theta(n^2)$

(b) $T(n) = 81T\left(\frac{n}{9}\right) + 10n^2$

$\therefore$ $T(n) = O(n^2 \log n)$

Recursive part,

$n^{\log_{10} a} = n^{\log_9 81} = n^2$ and $f(n) = 10n^2$

Since, $f(n) = \theta(n^{\log_b a})$ as per Master theorem case 2,

$T(n) = \theta(f(n) \log n) = \theta(n^2 \log n)$

(c) $T(n) = 16T\left(\frac{n}{4}\right) + 100(n\log n)^{1.99}$

and $f(n) = 100\,(n \log n)^{1.99}$

Since $f(n)$ is polynomially (by a factor of some n^{ϵ}, $\epsilon > 0$) smaller, as per Master theorem case 1.

$$T(n) = \theta(n^{\log_b a}) = \theta(n^2)$$

(d) $T(n) = 100T\left(\frac{n}{100}\right) + n\log^2 n$

Recursive part, $n^{\log_b a} = n^{\log_{100} 100} = n$

and $f(n) = n \log^2 n$

Here $f(n)$ and $n^{\log_b a}$ are not asymptotically equal and they don't have a polynomial difference. Any polynomial is n is asymptotically larger than any polynomial in log n). So, Master theorem cannot be applied as such. But we can apply the extended Master theorem case 2, which states that

If $f(n) = \theta(n^{\log_b a} \log^k n), k > -1$,

Then, $T(n) = \theta(n^{\log_b a} \log^{k+1} n)$

Here, $k = 2$ so, we get

$$T(n) = \theta(n \log^3 n)$$

23. (a)

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

Put, $n = 2^K$

$$T(2^K) = T\left(\frac{2^K}{2}\right) + \sqrt{2^K}$$

$$S(K) = S\left(\frac{K}{2}\right) + 2^{K/2}$$

$$S(K) = O(2^{K/2})$$

Put, $K = \log_2 n$

So, $T(n) = O(\sqrt{n})$

24. (b)

$$T(n) = n + 4T(n/2)$$
$$= n + 4(n/2 + 4T(n/2^2))$$
$$= \ldots + \frac{4^{i-1}}{2^{i-1}} n + 4^i T(n/2^i)$$
$$= n + 2n + 4n + \ldots\ldots + 4\log^n T(1)$$
$$= n \sum_{i=0}^{\log_2 n - 1} 2^i + \Theta(4^{\log n})$$
$$= \frac{n(2^{\log n} - 1)}{(2-1) + \Theta(2^{\log n})^2}$$

Since,

$$(2^2)^{\log n} = (2^{\log n})^2 = \Theta(n^2) + \Theta(n^2) = \Theta(n^2)$$

25. (a)

$$T(n) = 2T\left(\frac{n}{2}\right) + n^3$$

Applying Master theorem:

$$n^{\log_2 2} < n^3$$

So, $T(n) = \Theta(n^3)$

26. (a)

$$T(x) = \sqrt{n}T(\sqrt{n}) + 100n$$

Let $n = 2^k \Rightarrow k = \log n$

$$T(2k) = (2)^{k/2}\, T(2^{k/2}) + 100 \times 2^k$$

(dividing by 2^k)

$$\frac{T(2^k)}{2^k} = \frac{2^{k/2} T(2^{k/2})}{2^k} + 100$$

$$\frac{T(2^k)}{2^k} = \frac{T(2^{k/2})}{2^{k/2}} + 100$$

Let $y(k) = \frac{T(2^k)}{2^k}$, then ...(i)

$$y(k) = y\left(\frac{k}{2}\right) + 100$$

Now, applying Master theorem [$a = b^k = 1$]

$$y(k) = \log k$$

from (i) we also know that $T(2^k) = 2^k y(k)$, then

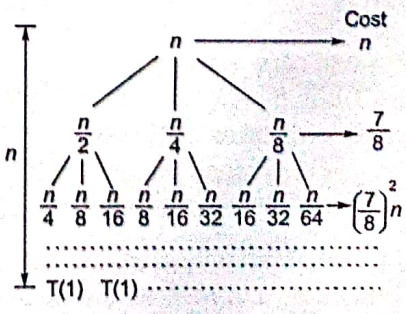
$$T(2^k) = 2k \log k = T(n)$$
$$= n \log \log n$$

(because $n = 2^k$ and $k \log n$)

Finally, $T(n) = \theta(n \log \log n)$

27. (a)

By recursion tree method,



↓

$$\text{Height} = \left(\frac{n}{2^i}\right) = 1$$

$$n = 2i$$

$$i = \log_2 n$$

$$T(n) = n + \left(\frac{7}{8}\right)n + \left(\frac{7}{8}\right)^2 n + \ldots . \log_2 n \text{ times}$$

$$\text{Sum of GP} = a\left(\frac{r^n - 1}{r-1}\right) = n\left(\frac{(7/8)^{\log_2 n} - 1}{(7/8) - 1}\right) \quad \ldots(1)$$

Without solving the eq. (1)

Let $\log n \to \infty$

$$\therefore \quad S_\infty = n\left(\frac{1}{1 - \frac{7}{8}}\right) = n\left(\frac{8}{1}\right)$$

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

So, option (a) is correct.

28. (a)

Using Master theorem,

$$T(m) = aT\left(\frac{m}{b}\right) + \theta(n^k \log^p n)$$

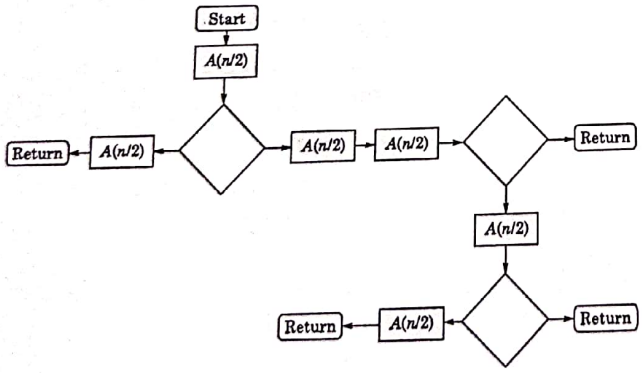
Here, $a = 1$, $b = \frac{4}{3}$, $k = 0$, $p = 0$

Hence, $a = b^k$ and $p > -1$

Then, $T(m) = \theta(n^{\log a} \log^{p+1} m)$

$= \theta(n^0 \log^1 m) = \theta(\log m)$

29. (2.32)



In worst case scenario,

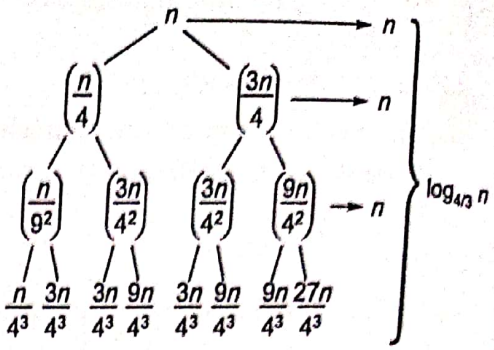
we have 5 $A(n/2)$ function calls and other statements takes O(1) time.

So, the recurrence relation will be,

$$A(n) = 5\,A(n/2) + O(1)$$

By using Master theorem, we get $n^{\log_2 5}$ which is equivalent to $n^{2.32}$. So the value of α is 2.32.

30. (d)



$$\Rightarrow \quad \frac{n + 3n + 3n + 9n}{116} = \frac{7n + 9n}{16} = \frac{16n}{16} = n$$

$$\Rightarrow \quad n + n + n \ldots . (n * \log_{4/3} n)$$

31. (b)

$$T(n) = 2.T(n/2) + \log n$$

$\because \quad \log n = O(n^{\log_b a - \epsilon})$ for some $\epsilon > 0$

$$T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_2 2}) = \Theta(n)$$

$\therefore \quad T(n) = \Theta(n)$

So option (b) can be solved using master theorem

(a) $\quad T(n) = 2 \cdot T\left(\frac{n}{2}\right) + \frac{n}{\log n}$

Try case 1: $\frac{n}{\log n} \neq O(n^{1-\epsilon})$ for some $\epsilon > 0$,

because $\frac{1}{\log n} \neq O(n^{-\epsilon})$

Try case 2: $\frac{n}{\log n} \neq \Theta(n^1)$, because $\frac{n}{\log n} = o(n$

Try case 3: $\frac{n}{\log n} \neq \Omega(n^{1+\epsilon})$ for some $\epsilon > 0$,

because $\frac{n}{\log n} = o(n)$

All cases failed using master theorem for optic (a), similarly option (c) is also not applicable.

32. (d)

$$T(n) = 7T\left(\frac{n}{2}\right) + an^2$$

Using Master's method: $a = 7$, $b = 2$, $f(n) = an$

$$h(n) = \frac{f(n)}{n^{\log_b a}} = \frac{an^2}{n^{\log_2 7}} = \frac{an^2}{n^{2.81}} = an^{0.}$$

So $\quad U(n) = 0(1)$

$$T(n) = n^{\log_b a} U(n) = n^{\log_2 7} 0(1)$$

$$T(n) = 0(n^{2.81})$$

3. (b)

$$T(n) = T(n-1) + 2n \quad ...(1),\ T(1) = 1$$
$$T(n-1) = T(n-2) + 2(n-1) \quad ...(2)$$
$$T(n-2) = T(n-3) + 2(n-2) \quad ...(3)$$
$$T(n) = T(n-2) + 2(n-1) + 2n$$
$$= T(n-3) + 2(n-2) + 2(n-1) + 2n$$
$$= T(n-k) + 2(n-k+1) + 2(n-k+2) + ... + 2n$$

Let $n-k = 1$

Then, $k = n-1$

$$T(n) = T(n-(n-1)) + 2(n-(n-1)+1) + 2(n-(n-1)+2) + ... + 2n$$
$$= T(1) + 4 + 6 + ... + 2n$$
$$= 1 + 4 + 6 + ...+ 2n \quad (\because T(1) = 1)$$
$$= 1 + 4 + 6 + ... + 2n + 2 - 2$$
$$= 1 + (2 + 4 + 6 +... +2n) - 2$$
$$= 1 + 2(1 + 2 + 3 +... +n) - 2$$
$$= 1 + 2\left(\frac{n(n+1)}{2}\right) - 2$$
$$T(n) = n(n+1) - 1$$

So, $T(n) = \theta(n^2)$

4. (b)

$$T(n) = 8T(n/2) + qn \text{ if } n > 1$$
$$= p \text{ if } n = 1$$

p and q are constants.

Now, according to Master theorem,

$$T(m) = aT(n/b) + n^c \text{ if } n > 1$$
$$= d \text{ if } n = 1$$

and if $\log_b a > c$ then,

$$T(n) = \theta(\log_b a)$$

In this recurrence relation,

$$a = 8,\ b = 2,\ c = 1$$
$$\therefore \quad \log_b a = \log_2 8 = 3$$

Here, $\log_b a = c$

So, order will $T(n) = \theta(n^{\log_b a}) = \theta(n^3)$

35. (a)

Tower of Hanoi function works as:

```
TOH (n, x, y, z)
{
    if (n > = 1)
    {
        1. TOH (n, x, z, y)  //...T(n-1)
        2. move x → y  //...T(1)
        3. TOH (n, z, y, x)  //...T(n-1)
    }
}
```

Thus, TOH function is recursively called and step 2 is just move which takes O(1) time so as it is asked order of work, option (a) is correct.

36. (a)

$$T(n) = \sqrt{2} * T\left(\frac{n}{2}\right) + C$$
$$= \sqrt{2}\left[\sqrt{2}T(n/2^2) + C\right] + C$$
$$= \sqrt{2^2} * T(n/2^2) + \sqrt{2}C + C$$
$$= \sqrt{2^3} * T(n/2^3) + \sqrt{2}^2 C + \sqrt{2}C + C$$
$$= \sqrt{2^k}T(n/2^k) + \sqrt{2^{k-1}}C + \sqrt{2^{k-2}}C + ...\sqrt{2}C + C$$

Put $\frac{n}{2^k} = 1$

$\Rightarrow \quad k = \log_2 n$

$$= \left(\sqrt{2}\right)^{\log n} * T(1) + C\left[1 * (\sqrt{2}^{\log n} - 1)/(\sqrt{2} - 1)\right]$$

[By applying and series]

$$= n^{\log_2 \sqrt{2}} * a + C\left[(n^{\log_2 \sqrt{2}} - 1)/(\sqrt{2} - 1)\right]$$
$$= a * n^{1/2} + C[(n^{1/2} - 1)/(\sqrt{2} - 1)]$$

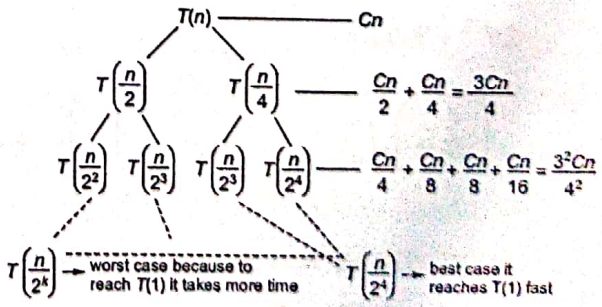
[Ignoring constant]

$= \theta(n^{1/2})$

So, option (a) is correct.

37. (a)

$$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + Cn$$



$$T(n) = Cn + \frac{3Cn}{4} + \left(\frac{3}{4}\right)^2 Cn + ... + \left(\frac{3}{4}\right)^k Cn$$

$\frac{n}{2^k} = 1$, $k = \log_2 n$ levels

$$= Cn + \frac{3}{4}Cn + \left(\frac{3}{4}\right)^2 Cn + \ldots + \left(\frac{3}{4}\right)^{\log_2 n} C_n$$

$$= Cn\left[\left(\frac{3}{4}\right)^0 + \left(\frac{3}{4}\right)^1 + \left(\frac{3}{4}\right)^2 + \ldots + \left(\frac{3}{4}\right)^{\log_2 n}\right]$$

$$= Cn\left[\frac{1-\left(\frac{3}{4}\right)^{\log_2 n}}{1-\frac{3}{4}}\right] \cong Cn \cong \theta(n)$$

38. (a)

$T(1) = 1$

$T(n) = 2T(n-1) + n,\ n \geq 2$

$T(2) = 2T(1) + 2 = 2.1 + 2 = 4$

$T(3) = 2T(2) + 3 = 2.4 + 3 = 11$

$T(4) = 2T(3) + 4 = 2.11 + 4 = 26$

$\vdots$

$T(n-1) = 2T(n-2) + n$

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

So, $T(n) = 2^{n+1} - n - 2$

39. (c)

$$T(n) = 8\sqrt{n} \cdot T(\sqrt{n}) + (\log n)^2$$

$$= 8n^{1/2}\left\{8n^{1/4}T(n^{1/4}) + (\log n^{1/2})^2\right\} + (\log n)^2$$

$$= 8^2 n^{1/2+1/4}T(n^{1/4}) + 2n^{1/2}(\log n)^2 + (\log n)^2$$

$$= 8^2 n^{1/2+1/2^2}\left\{8n^{1/8} + T(n^{1/8}) + (\log n^{1/4})^2\right\}$$

$$+2n^{1/2}(\log n)^2 + (\log n)^2$$

$$= 8^3 n^{1/2+1/2^2+1/2^3}T(n^{1/2^3}) + (2^2 n^{1/2+1/4} +$$

$$2n^{1/2} + 1)(\log n)^2$$

$$= 8^k\left(n^{\sum_{i=1}^{k}\frac{1}{2k}}\right)T\left(n^{\frac{1}{2^k}}\right) + \left(\sum_{i=1}^{k-1} 2^i\left(n^{\sum_{j=1}^{i}\frac{1}{2j}}\right) + 1\right)$$

$$(\log n)^2 \leq (\log n)^3 n$$

Let, $n^{1/2k} = 2$

Then, $2k = \log n$

$\leq (\log n)^3 n \cdot 1$

$\cong O(n(\log n)^3)$

40. (c)

$T(0) = T(1) = 1$

$T(n) = 2T(n/2) + n$

$T(n)$ be computed as follows:

$$T(n) = n\sum_{i=0}^{\log n}(2/2)^i$$

$$T(n) = n\sum_{i=0}^{\log n}(1)^i$$

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

[$T(n)$ can also proved by Master Theorem]

If $T(n) = \Theta(n \log n)$ then it is also $O(n \log n)$ and $O(n^2)$ but it is not $\Omega(n^2)$.

41. (c)

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

$$T\left(\frac{n}{2}\right) = \left[T\left(\frac{n}{2^2}\right) + \left(\frac{n}{2}\right)^2\frac{1}{2} + \frac{n}{2}\right] + \frac{n^2}{2} + n$$

$$T\left(\frac{n}{2^2}\right) = T\left(\frac{n}{2^3}\right) + \left(\frac{n}{2^2}\right)^2\frac{1}{2} + \frac{n}{2^2} + \left(\frac{n^2}{2}\right)^*$$

$$\frac{1}{2} + \frac{n}{2} + \frac{n^2}{2} + n$$

$$= T\left(\frac{n}{2^{k+1}}\right) + \left[\left(\frac{n}{2^k}\right)^2\frac{1}{2} + \ldots + \left(\frac{n}{2}\right)^2\frac{1}{2} + (n^2)\left(\frac{1}{2}\right)\right]$$

$$+\left[\frac{n}{2^k} + \frac{n}{2^{k+1}} + \ldots + \frac{n}{2} + \frac{n}{2^0}\right]$$

Terminating condition: $\frac{n}{2^{k+1}} = 1$

$n = 2k + 1$

$k = \log_2 n - 1$

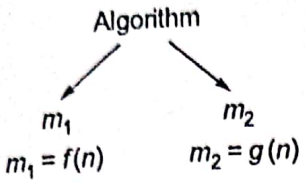
$$= T(1) + \frac{n^2}{2}\left[\frac{1}{2^{2k}} + \frac{1}{2^{2(k-1)}} + \ldots + \frac{1}{2^2} + \frac{1}{2^0}\right] +$$

$$n\left[\frac{1}{2^k} + \frac{1}{2^{k-1}} + \ldots + \frac{1}{2^1} + \frac{1}{2^0}\right]$$

$$\Rightarrow 1 + \frac{n^2}{2}\left[\frac{1-\left(\frac{1}{4}\right)^k}{1-\frac{1}{4}}\right] + n\left[\frac{1-\left(\frac{1}{2}\right)^k}{1-\frac{1}{2}}\right]$$

$\Rightarrow O(n^2)$ ignoring other as it is less than 1.

2. (a)



So, order of algorithm

$= m_1 + m_2$

$= \text{Max}(f(n)+g(n))$

3. (c)

$$T(n) = T(n-1) + T(n-1) + d$$
$$= 2T(n-1) + d$$
$$= 2^i T(n-i) + d\sum_{j=0}^{i-1} 2^j$$

[Solving by Substitution $\because\ i = n-1$]

$$= 2^{n-1}T(1) + d\sum_{j=0}^{n-2} 2^j$$

$$\left[\because \sum_{r=0}^{n-1} a^r = \frac{a^n - 1}{a-1}\right]$$

$$= 2^{n-1}\cdot c + d\frac{(2^{n-1}-1)}{2-1}$$
$$= 2^{n-1}\cdot c + d.2^{n-1} - d$$
$$= (c+d)2^{n-1} - d = O(2^n)$$

4. (a)

Let $4^k = n$

$T(n) = 3T(n/4) + n$

$$= 3^2 T\left(\frac{n}{4^2}\right) + n + \frac{n}{4}$$

$$= 3^3 T\left(\frac{n}{4^3}\right) + n + \frac{n}{4} + \frac{n}{4^2}$$

$$= 3^4 T(n/4^4) + n + \frac{n}{4} + \frac{n}{4^2} + \frac{n}{4^3}$$

$$= 3^k T(n/4^k) + n + \frac{n}{4} + \frac{n}{4^2} + \ldots \frac{n}{4^{k-1}}$$

$$= 3^k T(1) + n\left[1 + \frac{1}{4} + \frac{1}{4^2} + \frac{1}{4^3} + \ldots\right]$$

$$= 3^k \cdot 2 + n \times \left(\frac{\left(\frac{1}{4}\right)^k - 1}{\frac{1}{4} - 1}\right) \cong O(n)$$

45. (d)

$T(n) = T(n-1) + T(n-1) + 1$

[recursive $(n-1) = T(n-1)$]

$= 2T(n-1)+1$

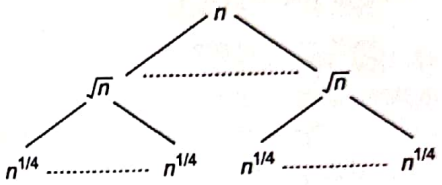
$= O(2^n)$ [using substitution method]

46. (d)

The given function is recursive so the equivalent recursion equation is

$T(n) = 1;\ n \le 2$

$T(n) = \lfloor\sqrt{n}\rfloor + n;\ n > 2$



All the level sums are equal to n. The problem size at level k of the recursion tree is n^{2-k} and we stop recursing when this value is a constant. Setting $n^{2-k} = 2$ and solving for k gives us

$2^{-k}\log_2 n = 1$

$\Rightarrow\quad 2^k = \log_2 n$

$\Rightarrow\quad k = \log_2 \log_2 n$

So $\quad T(n) = \Theta(\log_2 \log_2 n)$

47. (a)

In recurrence relation, floor, ceil, addition of constant terms etc. are not required for solving as we need asymptotic solution. So,

$$T(n) = T\left(\frac{n}{2} - 1\right) + T\left(n - \frac{n}{2}\right) + \theta(n)$$

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

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

$\Rightarrow\quad T(n) = \theta(n \log n)$

[by using 3rd case of Master theorem]

48. (a)

$$T(n) = \sqrt{2}\cdot T(n/2) + \sqrt{n} \quad \ldots(1)$$

$$= \sqrt{2}\left[\sqrt{2}\cdot T(n/4) + \sqrt{n/2}\right] + \sqrt{n}$$

$$= \left(\sqrt{2}\right)^2 \cdot T\left(n/2^2\right) + 2\sqrt{n} \quad \ldots(2)$$

$$= \left(\sqrt{2}\right)^3 \cdot T\left(n/2^3\right) + 3\sqrt{n} \quad \ldots(3)$$

$$\vdots$$

$$= \left(\sqrt{2}\right)^{2k} \cdot T\left(n/2^{2k}\right) + 2k\sqrt{n}$$

$$= \left(\sqrt{2}\right)^{\log n} \cdot 1 + \log n \cdot \sqrt{n} \qquad [\because \text{Put } 2k = \log n]$$

$$= \sqrt{n} + \log n \cdot \sqrt{n}$$

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

49. (a)

For higher values of n, $\frac{n}{2} >> 47$. So, we can ignore 47, Now $T(n)$ will be

$$T(n) = 3T\left(\frac{n}{2}\right) + 2n^2 + 10n - \frac{1}{2}$$

$$= 3T\left(\frac{n}{2}\right) + O(n^2)$$

Apply Master theorem, it is case 3 of Master theorem $T(n) = O(n^2)$

So, option (a) correct.

50. (d)

$$T(n) = 7T\left(\frac{n}{2}\right) + an^2$$

Using Master's method

$$a = 7,\ b = 2,\ f(n) = an^2$$

$$h(n) = \frac{f(n)}{n^{\log_b a}} = \frac{an^2}{n^{\log_2 7}} = \frac{an^2}{n^{2.81}}$$

$$= an^{-0.81}$$

So $\quad U(n) = O(1)$

$$T(n) = n^{\log_b a} U(n) = n^{\log_2 7} O(1)$$

$$T(n) = O(n^{2.81})$$

51. (511)

The recurrence relation for Tower of Hanoi problem is:

$$T(n) = 2T(n-1) + 1$$

$\Rightarrow$ Number of moves

$$2^n - 1 = 2^9 - 1 = 511$$

52. (c)

Two cases: $Q(m, 3)$

(i) $m \geq n$: here $m \geq 3$

For $\quad m = 3,\ Q(3, 3)$

$$= Q(3-3, 3) + P$$

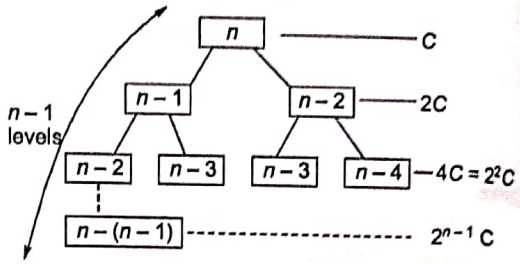
$$= Q(0, 3) + P$$

$\because$ Since 3 is divide by 3 only 1 time $= 0 + P = P$

So this recursive program generate P, m/n time until $m < n$.

So, $\quad Q(m, 3) = P \times (m \text{ div } 3)$

53. (a)

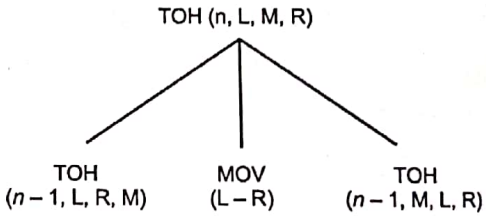


$T(n) = C + 2C + 4C + 8C + \ldots + 2^{n-1}$

$= C.2n \rightarrow$ Considering the sum of geometric series.

$$T(n) = O(2^n)$$

54. (d)



$\Rightarrow \quad T(n) = 2T(n-1) + 1$

3 CHAPTER

Divide and Conquer

Multiple Choice Questions & NAT Questions

.1 Sorting is useful for
(a) Report generation
(b) Minimizing the storage needed and responding to queries easily
(c) Making searching easier and efficient
(d) All of the above

.2 Which of the following sorting methods sorts a given set of items that is already in sorted order or in reverse sorted order with equal speed?
(a) Heap sort (b) Quick sort
(c) Insertion sort (d) Selection sort

.3 A sorting technique is called stable if
(a) It takes O(n log n) time.
(b) It maintain the relative order of occurrence of non-distinct elements.
(c) It uses divide and conquer paradigm.
(d) It takes O(n) space.

.4 Quick sort gives O(n log$n$) worst case performance if the pivot is selected as:
(a) First element of array
(b) Median of first, last and middle elements
(c) Arithmetic mean of the elements
(d) None of above

.5 A sorting technique that guarantees, that records with the same primary key occurs in the same order in the sorted list as in the original unsorted list is said to be
(a) Stable (b) Consistent
(c) External (d) Linear

.6 Randomized quicksort is an extension of quicksort where the pivot is chosen randomly. What is the worst case complexity of sorting n numbers using randomized quicksort?
(a) O(n) (b) (n log$n$)
(c) O(n^2) (d) O(nl)

Q.7 Consider the following cases for quick sort to sort an array of n element $a[0.....n-1]$.
1. Choosing the pivot element randomly from the given array.
2. Choosing median element as pivot.
3. Choosing middle element as pivot.

For which of the above cases quick sort always gives O(n log n) time complexity?
(a) 1 and 3 (b) 2 and 3
(c) 1 and 2 (d) None of these

Q.8 In quick sort for sorting of n numbers, the 75th greatest element is selected as pivot using O(n^2) time complexity algorithm than what is the worst case time complexity of quick sort.
(a) O(n^2) (b) O(n log n)
(c) O(n^3) (d) O(n)

Q.9 What is recurrence relation for merge sort?
(a) $T(n) = 2T\left(\frac{n}{3}\right) + O(n)$
(b) $T(n) = 2T\left(\frac{n}{2}\right) + O(n)$
(c) $T(n) = 3T\left(\frac{n}{2}\right) + \Omega(n)$
(d) $T(n) = 2T\left(\frac{n^2}{2}\right) + \Omega(n)$

Q.10 Given two sorted double linked list L_1 and L_2 of n elements each, which of the following are true?
(a) L_1 and L_2 can be merged into single sorted list in $\theta(n)$ time.
(b) L_1 and L_2 can be merged into single sorted list in $\theta(1)$ time.
(c) L_1 and L_2 can be merged into single sorted list in $\theta(n \log n)$ time.
(d) L_1 and L_2 can be merged into single sorted list in $\theta(n^2)$ time.

Q.11 What is time complexity of matrix multiplication (Strassen's) using divide and conquer approach?
(a) $O(n^{\log 5})$ (b) $O(n^{\log 6})$
(c) $O(n^{\log 7})$ (d) None of these

Q.12 The time complexity of multiplying two long integers of n-digits each using divide and conquer method is :
(a) $O(n^2)$ (b) $O(n)$
(c) $O(n^3)$ (d) $O(n \log n)$

Q.13 Suppose there are 4 sorted lists of $\frac{n}{2}$ elements each. If we merge these lists into a single sorted list of n elements, how many key comparisons are needed in the worst case using an efficient algorithm?
(a) $\frac{6}{4}n-3$ (b) $\frac{7}{4}n-3$
(c) $\frac{8}{4}n-3$ (d) $\frac{9}{4}n-3$

Q.14 Consider the following sorting algorithms
A. Merge sort B. Quick sort
C. Heap sort D. Insertion sort
Find the correct pair such that their best, average and worst case time complexities are same. (Assume that all elements are distinct)
(a) A, B (b) A, C
(c) A, D (d) None of these

Q.15 For merging two sorted lists of sizes m and n into a sorted list of size $m + n$. Find out the time complexity of this merging process.
(a) $O(m)$ (b) $O(n)$
(c) $O(m + n)$ (d) $O(\log(m) + \log(n))$

Q.16 A list of n strings, each of length n, is sorted into lexicographic order using the merge sort algorithm. The worst case running time of this computation is
(a) $O(n \log n)$ (b) $O(n^2 \log n)$
(c) $O(n^2 + \log n)$ (d) $O(n^2)$

Q.17 Which of the following is/are true?
S_1: A sorting algorithm is in place if the relative order of common element is maintained after sorting.
S_2: A sorting algorithm is stable if it requires very little additional space besides the initial array holding the elements that are to be sorted.
(a) Only S_1 is true
(b) Only S_2 is true
(c) Both S_1 and S_2 are true
(d) Both S_1 and S_2 are false

Q.18 The problem of long-integer multiplication (LIM is defined as: Given two long integers having n-digits each; it is required to multiply them Assuming that the numbers are represented in an array of size 'n'. The time complexity to multiply them using traditional divide and conquer method is
(a) $O(n)$ (b) $O(n^2)$
(c) $O(n \log n)$ (d) $O(\log n)$

Q.19 Given n integers in the range of 0 to k we want to preprocess the input in such away that to any query about how many of the n-integers fall in the range $a....b$ in $O(1)$. Then what will be the processes time?
(a) $O(n + k)$ (b) $O(n)$
(c) $O(\log n)$ (d) $O(k)$

Q.20 Consider an array containing 'n' elements. The elements present in an array are in arithmetic progression, but one element is missing in that order. What is the time complexing to find the position of the missing element using divide and conquer (by binary search).
(a) $O(n)$ (b) $O(\log n)$
(c) $O(n^2)$ (d) $O(n \log n)$

Q.21 How many term will be computed to determine the value of ${}^{10}C_8$ using a divide and conquer algorithms?

Q.22 Merging 4 sorted files having 200, 50, 125, 25 Records will take 0 (–) time?

Q.23 If one uses straight two-way merge sort algorithm to sort the following elements in ascending order

20, 47, 15, 8,9, 4, 40, 30, 12, 17

Then the order of these elements after the 2nd pass of the algorithm is
(a) 8, 9, 15, 20, 47, 4, 12, 17, 13, 40
(b) 8, 15, 20, 47, 4, 9, 30, 40, 12, 17
(c) 15, 20, 47, 4, 8, 9, 12, 30, 40, 17
(d) 4, 8, 9, 15, 20, 47, 12, 17, 30, 40

Q.24 You have n plates in your kitchen, and you want to put them on a dining table. You want to put your plates on a dining table sorted by colour.

The colour of each plates is specified by tuple of red, green and blue values, each in the range 1 to 16. Place your plates on dining table sorted first by red, then green then blue value. Which of the following sorting algorithm is best suited to put the plates on dining table sorted by colour?

(a) Insertion sort (b) Radix sort
(c) Counting sort (d) None of these

Q.25 Given an array A of elements:

80, 70, 55, 90, 71, 72, 73, 77, 52, 50

What will be the output after 1 pass of 3 way merge sort?

(a) 70, 80, 55, 90, 71, 72, 73, 77, 50, 52
(b) 55, 70, 80, 71, 72, 90, 52, 73, 50, 77
(c) 50, 52, 55, 70, 71, 72, 73, 77, 80, 90
(d) 55, 70, 80, 71, 72, 90, 52, 73, 77, 50

Q.26 Given an array of n elements, each of which is at most k positions from its target position, if merge sort is used to sort such an array what is the time complexity?

(a) $O(n \log n)$ (b) $n \log k$
(c) $n \times k$ (d) n^k

Q.27 Given two sorted list of size 'm' and 'n' respectively. The number of comparisons needed in the worst case by the merge sort algorithm will be

(a) $m \times n$ (b) maximum of m, n
(c) minimum of m, n (d) $m + n - 1$

Q.28 Which of the following algorithms cannot be designed without recursion?

(a) Fibonacci series (b) Tower of Hanoi
(c) Quick sort (d) None of these

Q.29 The average number of comparisons performed by the merge sort algorithm, in merging two sorted lists of length 2 is

(a) $\frac{8}{3}$ (b) $\frac{8}{5}$
(c) $\frac{11}{7}$ (d) $\frac{11}{6}$

Q.30 Consider an online sorting algorithm which is based on insertion sort. At some point of time, we have n elements which are already in sorted order. Now knew elements arrive and they need to be added to the list using the above mentioned online sorting algorithm. The worst-case time complexity to get the new sorted list with $(n + k)$ elements is:

Note: Initial elements are already sorted.

(a) $O(n(n + k))$ (b) $O((n + k)^2)$
(c) $O(k(n + k))$ (d) None of these

Q.31 You want to check whether a given set of items is sorted. Which of the following sorting methods will be the most efficient if it is already in sorted order?

(a) Bubble sort
(b) Selection sort
(c) Insertion sort
(d) Merge sort

Q.32 Which one of the following does not represent the array if bubble sort to sort the elements in ascending order is run after one iteration

(a) 1, 2, 5, 7, 8, 6, 4, 3 9
(b) 3, 2, 5, 9, 8, 6, 4, 1, 7
(c) 1, 2, 5, 6, 7, 8, 4, 3, 9
(d) None of the above

Q.33 Which of the following algorithms exhibits the unnatural behaviour that, minimum number of comparisons are needed if the list to be sorted is in the reverse order and maximum number of comparisons are needed if they are already in sorted order?

(a) Heap sort
(b) Radix sort
(c) Binary insertion sort
(d) There can't be any such sorting method

Q.34 The worst case running times of Insertion sort, Merge sort and Quick sort, respectively, are:

(a) $\Theta(n \log n)$, $\Theta(n \log n)$, and $\Theta(n^2)$
(b) $\Theta(n^2)$, $\Theta(n^2)$, and $\Theta(n \log n)$
(c) $\Theta(n^2)$, $\Theta(n \log n)$, and $\Theta(n \log n)$
(d) $\Theta(n^2)$, $\Theta(n \log n)$, and $\Theta(n^2)$

Q.35 Which of the following sorting algorithms performs efficiently to sort a singly linked list containing $\log n$ nodes and the corresponding time complexity is _____?

(a) Insertion sort, $O(\log^2 n)$
(b) Merge sort, $\theta(\log n) \log(\log(n))$
(c) Heat sort, $\theta(\log^2)(\log n))$
(d) Quick sort, $O(\log^2 \log n)$

Q.36 Consider the following scenario during insertion sort when the array looks like the following:

{25, 75, 95, 125, 80, 5, 10}

The number of comparisons that it will further take for the array to be completely sorted are _____.

Q.37 What is time complexity to find the k^{th} largest element in a min heap?

(a) $O(n)$ (b) $O(\log n)$
(c) $O(n \log n)$ (d) $O(n^2)$

Q.38 Given "log n" sorted list each of size "$\frac{n}{\log n}$", what is time required to merge them into one single list?

(a) $O(n \log n)$ (b) $O(n \log \log n)$
(c) $O(n \log n \log n)$ (d) $O(n)$

Q.39 What is total number of comparisons needed in the best case to find minimum and maximum of 300 elements?

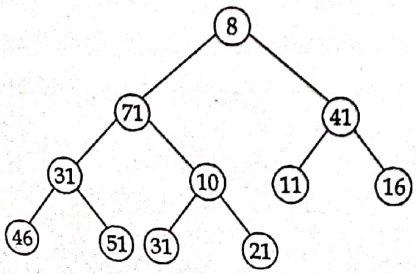
Q.40 If Radix sort is used to sort an array of n integers which are in range $(n^{\log_2 d}, n^{d^2})$, where d = some function of input size, the time taken would be?

(a) $O(nd^2)$
(b) $O(n^2 d \log n + n^2 \log_2 d \log_2 n)$
(c) $O(n^2 d^2 + n^2 \log_2 n)$
(d) $O(nd^2 \log_2 n)$

Q.41 "n" strings each of length "n" are given them, what is time taken to sort them, 2-way merge?

(a) $O(n)$ (b) $O(n \log n)$
(c) $O(n^2 \log n)$ (d) $O(\log n)$

Q.42 A **min-max heap** has the following property: "A almost complete binary tree where each node at an even level in the tree is less than all of its descendants and each of the node at an odd level in the tree is greater than all of its descendants".



What is the worst case time complexity "deleting minimum element" and "deletin maximum element" on above data structure?

(a) $O(\log n)$ and $O(\log n)$ (b) $O(n)$ and $O(\log n)$
(c) $O(\log n)$ and $O(n)$ (d) $O(n)$ and $O(n)$

Q.43 Given a binary-max heap. The elements are store in an array as 25, 14, 16, 13, 10, 8, 12. What is th content of the array after two delete operations?

(a) 14,13,8,12,10 (b) 14,12,13,10,8
(c) 14,13,12,8,10 (d) 14,13,12,10,8

Q.44 What is the worst case time complexity to delet middle element from min heap of n distinc element?

(a) $O(\log n)$ (b) $O(n)$
(c) $O(n \log n)$ (d) $O(n^2)$

Q.45 Consider the following statements:

S_1: In a connected undirected graph $G = (V, E$ with distinct edge costs, the cheapest edg belongs to every minimum spanning tree.

S_2: In a connected undirected graph $G = (V, E$ with distinct edge costs, the most expensiv edge is excluded from every minimun spanning tree.

Which of the following is true?

(a) Only S_1 (b) Only S_2
(c) Both S_1 and S_2 (d) Neither S_1 nor S_2

Q.46 The minimum number of comparisons to find th biggest element in an unsorted list of size 15 _____.

Q.47 Of the following sorting algorithms, which has running time that is least dependent on the initi ordering of the input?

(a) Merge Sort (b) Insertion Sort
(c) Selection Sort (d) Quick Sort

Q.48 Which of the following represent correct recurrenc for worst case of Binary search with n element.

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

(b) $T(n) = 2T\left(\frac{n}{2}\right) + O(1)$

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

(d) $T(n) = T\left(\frac{n}{2}\right) + O(1)$

Q.49 Assume an array of n integers and a function Max ( ) given below:
Max ( ) = {find an element which is higher than its left and right element if any}
What is the worst case time complexity for Max( ) on an input array?
(a) $O(\log n)$ (b) $O(n)$
(c) $O(n \log n)$ (d) $O(n^2)$

Q.50 What is worst case time complexity to delete an middle element from min heap of n distinct element?
(a) $O(\log n)$ (b) $O(n \log n)$
(c) $O(n)$ (d) $O(n^2)$

Q.51 Which one of the following is the tightest upper bound that represents the time complexity of inserting an object into a binary search tree of n nodes?
(a) $O(1)$ (b) $O(\log n)$
(c) $O(n)$ (d) $O(n \log n)$

Q.52 In the standard merge sort algorithm on a list of size n, what is the maximum number of times an item can be compared?
(a) 2 (b) $\log n$
(c) $n-1$ (d) $n \log n$

Q.53 A modified version of merge sort is devised with a modified merge method is used where the merge takes $O(n^2)$ time to merge lists that sums up to size n, the time complexity for such a version of merge sort is
(a) $O(n \log n)$ (b) $O(n)$
(c) $O(n^2 \log n)$ (d) $O(\sqrt{n})$

Q.54 The running time of any comparison based algorithm for sorting an n element sequence in the worst case is
(a) $\Omega(n \log n)$ (b) $\Omega(n^2)$
(c) $\Omega(n^{1.5})$ (d) $O(n \log n)$

Q.55 An e-commerce site on search for a particular key word returns the search results in order to popularity by default also the user is allowed to sort by price (low to high) or (high to low) however if two or more products are of the same price then among the same price products should be presented to the user in the same order of popularity. Which of the following sorting algorithm cannot be used for such a scenario?
(a) Insertion sort (b) Bubble sort
(c) Quick sort (d) None of above

Q.56 Consider a sorted array of n numbers. What would be the time complexity of the best known algorithm to find a pair "a" and "b" such that $|a-b| = k$, k being a positive integer.
(a) $O(\log n)$ (b) $O(n \log n)$
(c) $O(n)$ (d) $O(n^2)$

Q.57 Assume that the algorithms considered here sort the input sequences in ascending order. If the input is already in ascending order, which of the following are TRUE?
I. Quick sort runs in $\Theta(n^2)$ time
II. Bubble sort runs in $\Theta(n^2)$ time
III. Merge sort runs in $\Theta(n)$ time
IV. Insertion sort runs in $\Theta(n)$ time
(a) I and II only
(b) I and III only
(c) II and IV only
(d) I and IV only

Q.58 The number of swappings needed to sort the numbers 8, 22, 7, 9, 31, 5, 13 in ascending order, using bubble sort is
(a) 11 (b) 12
(c) 13 (d) 10

Q.59 Assume that a CPU can process 10^8 operations per second. Suppose you have to sort an array with 10^6 elements. Which of the following is true?
(a) Insertion sort will always take more than 2.5 hours while merge sort will always take less than 1 second.
(b) Insertion sort will always take more than 2.5 hours while quick sort will always take less than 1 second.
(c) Insertion sort could take more than 2.5 hours while merge sort will always take less than 1 second.
(d) Insertion sort could take more than 2.5 hours while quick sort will always take less than 1 second.

Q.60 Array of n elements with first 10 and last 50 elements unsorted. Find an algorithm with runs faster.
(a) Merge sort (b) Quick sort
(c) Insertion sort (d) Bubble sort

Q.61 A list has n-strings where each string is of length n. Using divide and conquer approach, what is the worst case running time (tightest upper bound) to find the first string?

Note: First string is a string which appears first when all strings are sorted into the lexicographical order.

(a) $O(n^2)$ (b) $O(n)$
(c) $O(n \log n)$ (d) $O(n^2 \log n)$

Q.62 Match **List-I** (Divide Conquer) with **List-II** (Recurrence Relation (Worst-Case) and select the correct answer using the codes given below the lists:

List-I	List-II
A. Binary search	1. $T(n) = 2T\left(\frac{n}{2}\right) + C$
B. Merge sort	2. $T(n) = T\left(\frac{n}{2}\right) + 1$
C. Quick sort	3. $T(n) = T(n-1) + n - 1$
D. Max Min	4. $T(n) = 2T\left(\frac{n}{2}\right) + kn$

Codes:

	A	B	C	D
(a)	1	3	4	2
(b)	2	3	4	1
(c)	1	4	3	2
(d)	2	4	3	1

Q.63 True or false for internal sorting algorithms:

1. Internal sorting are applied when the entire collection if data to be sorted is small enough that the sorting can take place within main memory.
2. The time required to read or write is considered to be significant in evaluating the performance of internal sorting.

(a) 1-true; 2-true (b) 1-true; 2-false
(c) 1-false; 2-true (d) 1-false; 2-false

Q.64 What is best case run time of heap sort?

(a) $O(1)$ (b) $O(n)$
(c) $O(n \log n)$ (d) $O(\log n)$

Q.65 What are the worst-case complexities of insertion and deletion of a key in a binary search tree?

(a) $\Theta(\log n)$ for both insertion and deletion
(b) $\Theta(n)$ for both insertion and deletion
(c) $\Theta(n)$ for insertion and $\Theta(\log n)$ for deletion
(d) $\Theta(\log n)$ for insertion and $\Theta(n)$ for deletion

Q.66 Which of the following algorithm will give bes performance when items in array in reverse orde

(a) Insertion sort (b) Merge sort
(c) Quick sort (d) Heap sort

Q.67 Given a sequence of n integers in the order 9, 8 7, 6, 5, 4, 3, 2, 1, 0, after which iteration, the resulting sequence would be 1, 2, ,3 4, 5, 6, 7, 8 9, 0 if you apply insertion sort? _____th iteration

Q.68 Which one of the following is the tightest upper bound that represents the time complexity o inserting an object into a binary search tree o n nodes?

(a) $O(1)$ (b) $O(\log n)$
(c) $O(n)$ (d) $O(n \log n)$

Q.69 A sort method is said to be stable if the relative order of keys is the same after the sort as it wa before the sort. In which of the following pairs both sorting algorithms are stable?

(a) Quick sort and Insertion sort
(b) Insertion sort and Bubble sort
(c) Quick sort and Heap sort
(d) Quick sort and Bubble sort

Q.70 Give the result of partitioning the keys.

THISCOURSEISOVER

after the 1st pass of quick sort. Choose the las elements as pivot element (R). Also fo duplicates, adopt the convention that both pointers stop.

(a) E H I O C O I E R R U S S V T S
(b) E H I S C O I E R R U S O V T S
(c) E H I O C O U E S R T S S V T R
(d) E H I O O C I E R R U S S V T S

Q.71 Consider an array consisting of the following elements in unsorted order (placed randomly), bu 60 as first element.

60, 80, 15, 95, 7, 12, 35, 90, 55

Quick sort partition algorithm is applied b choosing first element as pivot element. How many total number of arrangements of array integers is possible preserving the effect of first pass o partition algorithm.

.72 The usual $\Theta(n^2)$ implementation of Insertion Sort to sort an array uses linear search to identify the position where an element is to be inserted into the already sorted part of the array. If instead, we use binary search to identify the position, the worst case running time will

(a) remain $\Theta(n^2)$
(b) become $\Theta(n(\log n)^2)$
(c) become $\Theta(n \log n)$
(d) become $\Theta(n)$

.73 Find the number of comparisons that will be needed in worst case to merge the following sorted files into a single sorted file by merging together two files at a time?

Files	f1	f2	f3	f4
Number of records	20	21	22	23

.74 Assume 5 buffers pages are available to sort a file of 105 pages. The cost of sorting using m-way merge sort is _____?

.75 In quick sort, for sorting n elements the $\left(\frac{n}{4}\right)^{th}$ smallest elements is selected as pivot using an $O(n)$ time algorithm. What will be the time complexity?

(a) $\theta(\log n)$ (b) $\theta(n)$
(c) $\theta(n^2)$ (d) $\theta(n \log n)$

.76 A machine needs a minimum of 100 sec to sort 1000 names by quick sort. The minimum time needed to sort 100 names will be approximately

(a) 50.2 sec (b) 6.7 sec
(c) 72.7 sec (d) 11.2 sec

Q.77 In quick sort, for sorting n elements, the $\left(\frac{n}{4}\right)^{th}$ smallest element is selected as pivot using an $O(n)$ time algorithm. What is the worst case time complexity of the quick sort?

(a) $\Theta(n)$ (b) $\Theta(n \log n)$
(c) $\Theta(n^2)$ (d) $\Theta(n^2 \log n)$

Q.78 Which of the following sorting methods will be the best if number of swapping done is the only measure of efficiency?

(a) Bubble sort (b) Selection sort
(c) Insertion sort (d) Quick sort

Q.79 Which of the following algorithms is not a comparison-based algorithm?

(a) Heap sort (b) Quick sort
(c) Radix sort (d) Topological sort

Q.80 Which of the following is true about heap data structure?

1. Priority queues can be efficiently implements using binary heap.
2. Heap data structure can be used to efficiently find the k^{th} smallest (or largest) element in an array.

(a) Only 1 (b) Only 2
(c) Both 1 and 2 (d) Neither 1 nor 2

Q.81 You have to sort a list L, consisting of a sorted list followed by a few 'random' elements which of the following sorting method would be most suitable for such a task?

(a) Bubble sort (b) Selection sort
(c) Quick sort (d) Insertion sort

Q.82 Which of the following cases is definite to give $O(n \log n)$ time complexity always for quick sort to sort the array $A[0...n]$, for any set of values of array A.

Case 1: Choosing middle element as pivot.
Case 2: Choosing pivot element as $(2^0)^{th}$ element initially followed by $(2^1)^{th}$ element, followed by $(2^2)^{th}$ element of array A and so on.
Case 3: Choosing median element as pivot.
Case 4: Choosing the pivot element randomly from the array A.
Case 5: Choosing pivot such that the array is partitioned into almost two equal subarrays.

(a) Only 1 and 4 (b) Only 1, 2 and 3
(c) Only 3 and 5 (d) Only 1, 3 and 4

Q.83 The best case of quick sort helps Vivek to sort a particular data set of size 'n' is 2048 msec. Gaurav also tried the same algorithm on similar data set and it took him 324 msec in worst case to sort a file of size 18. The file size of Vivek is_____.

Q.84 Suppose you are provided with the following function declaration in the C programming language.

```
int partition (int a[ ], int n);
```

The function treats the first element of a[ ] as a pivot, and rearranges the array so that all elements less than or equal to the pivot is in the left part of the array, and all elements greater than the pivot is in the right part. In addition, it moves the pivot so that the pivot is the last element of the left part. The return value is the number of elements in the left part.

The following partially given function in the C programming language is used to find the k^{th} smallest element in an array a[ ] of size n using the partition function. We assume $k \le n$.

```
int k^th_smallest (int a[ ], int n, int k)
{
    int left_end = partition (a, n);
    if (left_end+1==k)
    {
        return a [left_end];
    }
    if (left_end+1 > k)
    {
        return k^th_smallest (_____);
    }
    else
    {
        return k^th_smallest (_____);
    }
}
```

The missing argument lists are respectively

(a) (a, left_end, k) and (a + left_end + 1, n–left_end–1, k–left_end–1)

(b) (a, left_end, k) and (a, n–left_end–1, k–left_end–1)

(c) (a + left_end + 1, n–left_end–1, k–left_end–1) and (a, left_end, k)

(d) (a, n–left_end–1, k–left_end–1) and (a, left_end, k)

Q.85 Design an algorithm for this purpose that minimizes the numbers of swaps required for above question?

(a) Insertion sort (b) Merge sort

(c) Selection sort (d) Quick sort

Q.86 Derive an expression for the number of swaps needed by your algorithm in the worst case for above algo.

(a) n (b) $n-1$

(c) $n \log n$ (d) $\log n$

Q.87 Which of the following can make for an improve version of bubble sort?

(a) Traverse the array left to right during o passes and right to left during even passe in bubble sort.

(b) Divide the array into smaller arrays appl quick sort on each of the small arrays an apply the bubble sort on the entire array.

(c) Store the addresses to be swapped in separate array and make all the swaps at on go.

(d) None of these

Q.88 Merge sort makes two recursive calls whic statement is true after these two recursive cal finish, but before the merge sort step?

(a) The array element form a heap.

(b) Element in each half of the array are sorte among themselves.

(c) Elements in the first half of the array are les than the elements in second half of the arra

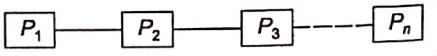
(d) All of the above

Q.89 A machine needs a minimum of 100 sec to so 1000 names by quick sort. The minimum tim needed to sort 100 names will be approximatel

(a) 50.2 sec (b) 6.7 sec

(c) 72.7 sec (d) 11.2 sec

Q.90 Suppose n processors are connected in a linea array as shown below. Each processor has number. The processors need to exchang numbers so that the numbers eventually appear in ascending order (the processor P_1 should hav the minimum value and the processor in shoul have the maximum value)



The algorithm to be employed is the following od numbered processors and even numbered processors are activated. When a processor i activated, the number it holds is compared wit the number held by its right-hand neighbour (one exists and the smaller of the two numbers i retained by the activated processor and the bigge stored in its right hand neighbour.

How long does it take for the processors to sor the values?

(a) $n \log n$ steps (b) n^2 steps

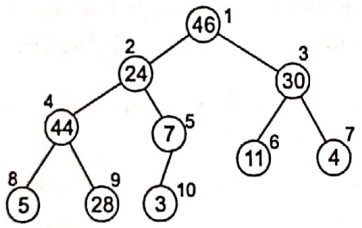
(c) n steps (d) $n^{1.5}$ steps

.91 The worst case running time to search for an element in a balanced binary search tree with $n\,2^n$ elements is
(a) $\Theta(n \log n)$ (b) $\Theta(n 2^n)$
(c) $\Theta(n)$ (d) $\Theta(\log n)$

.92 Heap sort is an implementation of _____ using a descending priority queue.
(a) Insertion sort (b) Selection sort
(c) Bubble sort (d) Merge sort

.93 Which of one the following is false?
(a) Heap sort is an inplace algorithm.
(b) Heap sort has $O(n \log n)$ average case time complexity.
(c) Heap sort is a stable sort.
(d) Heap sort is a comparison-based sorting algorithm.

.94 The essential part of heap sort is construction of max-heap. Consider the tree shown below, the node 24 violates the max-heap property. Once heapify procedure is applied to it, which position will it be in?



(a) 4 (b) 5
(c) 8 (d) 9

.95 The descending heap property is _____.
(a) $A[\text{Parent}(i)] = A[i]$
(b) $A[\text{Parent}(i)] <= A[i]$
(c) $A[\text{Parent}(i)] >= A[i]$
(d) $A[\text{Parent}(i)] > 2 * A[i]$

.96 How many elements can be sorted in $O(\log n)$ time using heap sort.
(a) $O\left(\frac{n}{2}\right)$ (b) $O(1)$
(c) $O(\log n/\log(\log(n)))$ (d) $O(\log n)$

Q.97 Given an array where numbers in range from 1 to n^6, which sorting algorithm can be used to sort these numbers in linear time?
(a) Radix sort
(b) Quick sort
(c) Counting sort
(d) Not possible to sort in linear time

Q.98 State true or false for internal sorting algorithms:
1. Internal sorting are applied when the entire collection if data to be stored in small enough that the sorting can take place within main memory.
2. The time required to read or write is considered to be significant in evaluating the performance of internal sorting.
(a) 1 true; 2 true
(b) 1 true; 2 false
(c) 1 false; 2 true
(d) 1 false; 2 false

Multiple Select Questions (MSQ)

Q.99 With respect to Quicksort, which of the following is/are true?
(a) Ideal for sorted array.
(b) Works best when input is randomly placed.
(c) worst case is $O(n^2)$.
(d) Better than insertion sort for every type of input.

100. Which of the following changes to typical Quicksort improves its performance on average and are generally done in practice.
(a) Randomly picking up to make worst case less likely to occur.
(b) Calling insertion sort for small sized arrays to reduce recursive calls.
(c) QuickSort is tail recursive , so tail call optimizations can be done.
(d) A linear time median searching algorithm is used to pick the median, so that the worst case time reduces to $O(n \log n)$.

101. Given an array that represents elements of arithmetic progression in order. It is also given that one element is missing in the progression, the worst case time complexity to find the missing element efficiently is:
(a) $\Theta(n)$ (b) $\Theta(n \log n)$
(c) $\Theta(\log n)$ (d) $\Theta(1)$

102. The asymptotic running time of Randomized QuickSort on arrays of length n.
(a) In expectation is $\theta(n \log n)$
(b) In expectation is $\theta(n^2)$.
(c) In the worst case is $\theta(n \log n)$
(d) In the worst case $\theta(n^2)$.

103. Define the recursion depth of QuickSort to be the maximum number of successive recursive calls before it hits the base case --- equivalently, the number of the last level of the corresponding recursion tree. Note that the recursion depth is a random variable, which depends on which pivots get chosen.
(a) The maximum possible depth would be $\theta(\log n)$.
(b) The minimum possible depth would be $\theta(\log n)$.
(c) The maximum possible depth would be $\theta(n)$.
(d) The minimum possible depth would be $\theta(n)$.

104. Which of the following is/are true about Merge Function?
(a) Minimum number of comparisons are Minimum (m, n).
(b) Minimum number of comparisons are $m + n - 1$.
(c) Maximum number of comparisons are Minimum (m, n).
(d) Maximum number of comparisons are $m + n - 1$.

105. Which of the following statement(s) is/are correct?
(a) Consider an array A in which upto some index I, integers are stored and after that NULL values are stored. Let the size of array be n, Then time taken to find the value I would be $O(\log n)$.
(b) Consider an array A in which upto some index I, Real numbers are stored and after that NULL values are stored. Let the size of array be n, Then time taken to find the value I would be $O(\log n)$.
(c) Consider an array A in which upto some index I, integers are stored and after that NULL values are stored. Let the size of array be n, Then time taken to find the value I would be $O(n)$.
(d) Consider an array A in which upto some index I, Real numbers are stored and after that NULL values are stored. Let the size of array be n, Then time taken to find the value I would be $O(n)$.

106. Which of the following sorting algorithm(s) does not have a worst case running time of $O(n^2)$?
(a) Insertion sort (b) Merge sort.
(c) Quick sort (d) Bubble sort

107. Which of the following is/are NOT divide and conquer approach?
(a) Insertion Sort (b) Merge Sort
(c) Shell Sort (d) Heap Sort

■■■■

Answers Divide and Conquer

1.	(d)	2.	(b)	3.	(b)	4.	(b)	5.	(a)	6.	(c)	7.	(d)	8.	(c)	9.	(c)
10.	(a)	11.	(c)	12.	(a)	13.	(c)	14.	(b)	15.	(c)	16.	(b)	17.	(d)	18.	(b)
19.	(a)	20.	(b)	21.	(89)	22.	(400)	23.	(b)	24.	(b)	25.	(d)	26.	(b)	27.	(d)
28.	(d)	29.	(a)	30.	(b)	31.	(c)	32.	(b)	33.	(c)	34.	(d)	35.	(b)	36.	(17)
37.	(a)	38.	(b)	39.	(448)	40.	(d)	41.	(b)	42.	(a)	43.	(c)	44.	(a)	45.	(a)
46.	(14)	47.	(a)	48.	(d)	49.	(b)	50.	(a)	51.	(c)	52.	(c)	53.	(b)	54.	(a)
55.	(c)	56.	(b)	57.	(d)	58.	(d)	59.	(c)	60.	(c)	61.	(c)	62.	(d)	63.	(b)
64.	(c)	65.	(b)	66.	(d)	67.	(8)	68.	(c)	69.	(a)	70.	(a)	71.	(720)	72.	(a)
73.	(169)	74.	(840)	75.	(d)	76.	(b)	77.	(b)	78.	(b)	79.	(c)	80.	(c)	81.	(d)
82.	(c)	83.	(256)	84.	(a)	85.	(c)	86.	(b)	87.	(d)	88.	(b)	89.	(b)	90.	(c)
91.	(c)	92.	(b)	93.	(c)	94.	(d)	95.	(c)	96.	(c)	97.	(a)	98.	(b)	99.	(b, c)
100.	(c)	101.	(c)	102.	(a, d)	103.	(b, c)	104.	(a, d)	105.	(a, b)	106.	(b)	107.	(a, c, d)		

Explanations Divide and Conquer

1. **(d)**
Sorting is useful for making search easier and efficient and in report generation but not minimize the space needed.

2. **(b)**
Quick sort has two worst cases, when input is in either ascending or descending order, it takes same time $O(n^2)$.

3. **(b)**
A sorting algorithm is called stable if it keeps element with equal keys in the same relative order in the output as they were in the input.
For example: in the following input the two 4's are indistinguishable. 1, 4*a*, 3, 4*b*, 2 and so the output of a stable sorting algorithm must be:
1, 2, 3, 4*a*, 4*b*
Bubble sort, merge sort, counting sort, insertion sort are stable sorting algorithms.

4. **(b)**
Quick sort will give best case performance if pivot is selected as median of 1st, last and middle element.

5. **(a)**
A sorting algorithm is said to be stable sorting algorithm if two objects/records with equal primary key appears in the same order in the sorted list as in the original unsorted list.

6. **(c)**
Randomized quick sort worst case time complexity = $O(n^2)$.

7. **(d)**
Choosing median element as pivot divide the array into two equal half and time complexity $O(n \log n)$.
For 1 and 2 choosing pivot randomly or middle element does not guaranty $O(n \log n)$ time complexity.

8. **(c)**
Generally, PARTITION process take $O(n)$ time complexity in quick sort.
But here, IMPROVED-PARTITION process take $O(n^2)$ time complexity.
The worst case occurs when the partition process always picks greatest or smallest element as pivot. If we consider above partition strategy where 75th greatest element is selected as pivot, the worst case would occur when the array is already sorted in increasing or decreasing order.
$$T(n) = T(n-1) + O(n^2)$$
The solution of the above recurrence by substitution method is $O(n^3)$.

9. (c)

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

because first merge sort for half array (1 to m) that takes $T\left(\frac{n}{2}\right)$.

Second merge sort for last half array ($m + 1$ to n)
Final phase merging them in $O(n)$ time

So, total time complexity = $2T\left(\frac{n}{2}\right) + O\left(\frac{n}{2}\right)$

$= O(n \log n)$

10. (a)
We can use same as merging procedure which will take $O(m + n)$.
Both link list have n element then time complexity should be $O(n + n) = O(2n) = O(n)$.
So, option (a) is right.

11. (c)

$$T(n) = 7T\left(\frac{n}{2}\right) + k \cdot n^2$$

Using Master Theorem $[n^{\log_2 7} > kn^2]$

$$T(n) = O(n^{\log_2 7})$$

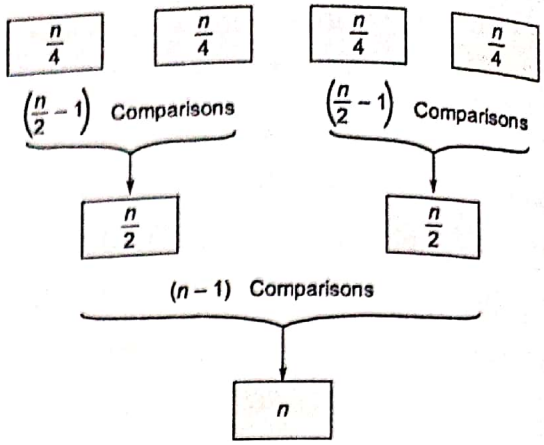
12. (a)
Let x and y be two n bit numbers divide x in two $\frac{n}{2}$ bit no say a, b similarly y in c, d.

To get $x * y$ we need to perform bd, ad, bc, ca that is $4\frac{n}{2}$ bit multiplication and some addition and shift operation in $O(1)$ so, recurrence relation can be written as

$$T(n) = 4T\left(\frac{n}{2}\right) + O(1)$$

Solve using Master theorem,
We get, $T(n) = O(n^2)$

13. (c)



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

$= 2n - 3 = \frac{8}{4}n - 3$

14. (b)
Merge sort and Heap sort have $O(n \log n)$ time complexity.

15. (c)
Each comparison will append one item to the existing merge list. In the worst case one needs $m + n - 1$ comparisons which is of order $m + n$.

16. (b)
The recurrence tree for merge sort will have height n and $O(n^2)$ work will be done at each level of the recurrence tree (Each level involves n comparisons and a comparison takes $O(n)$ time in worst case). So time complexity of this merge sort will be $O(n^2 \log n)$.

17. (d)
Stable Sorting : A sorting algorithm is stable if the relative order of common element is maintained after sorting.
In-place sorting : A sorting algorithm is inplace if it requires very little additional space besides the initial array holding the elements that are to be sorted.
For Ex. Quick sort is not stable sorting but it is in place sorting.

18. (b)
Divide the long int in multiple parts and multiply with the other one.

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

multiplication having n^2 time complexity.

$a = 2, b = 2, k = 2$ by Master theorem

So, $T(n) = O(n^2)$

19. (a)

For this Pseudo code is

(a) Construct a count array such that count[m] = number of elements with value m //$O(n)$

(b) Convert count to a cumulative count array such that count [m] = number of elements upto and including m. //$O(k)$

(c) Range Query [a, b] = count [b] – count[$a - 1$] //assuming $b >= a$

(d) Total processing time = $O(n + k)$

So, option (a) $O(n + k)$ is correct.

20. (b)

It takes O(log n) time by using binary search modification.

The idea is to go to the middle element. Check if the difference between middle and next to middle is equal to difference or net, if net then the missing element dies between mid and (mid + 1). If the middle element is equal to $\frac{n}{2}^{th}$ term in arithmetic series (Let n be the number of elements in input array), then missing element lies in right half. Else element lies in left half.

21. (69)

Let X = value N_{C_r} (any value in Pascal's triangle)

Y = number of terms to be computed to determine the value of X.

Then, $Y = 2 * X - 1$

i.e., $X = N_{C_r} = 10_{C_5} = 45$

and $Y = 89$

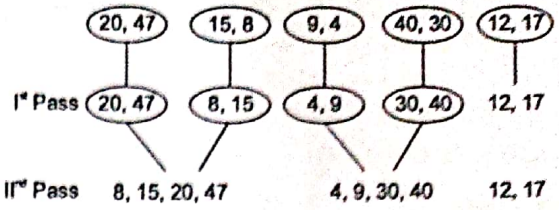
So, 89 terms computed in worst case.

22. (400)

Two sorted file of size m and n take $O(m + n)$ time for merging so total time

$= 200 + 50 + 125 + 25 = 400$

23. (b)



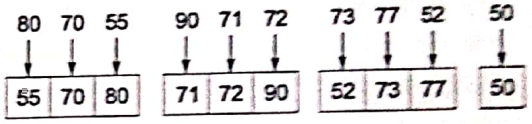
Hence (b) is the correct option.

24. (b)

Radix sort the plates by colour, using counting sort to sort first by blue value, placing the books into at most 10 stacks, then sort by green, then red value. This is very efficient as each book is moved four times.

25. (d)

3 way merge sort will combine 3 sub arrays at a time. After division elements becomes.



26. (b)

Divide the elements into $\frac{n}{k}$ groups of size k and sort each piece of $O(k \log k)$ time. This preserves the property that no element is more than k element out of position. Now merge each block of k elements with the block to its left.

27. (d)

Each comparison of both the list 'm' and 'n' result one element in sorted array. So in worst case number of comparison will be $m + n - 1$ but for best case min of m and n.

28. (d)

All can be implemented without recursion.

29. (a)

Size sorted list has size 2.

So, number of comparison is between min (2, 2) to (2 + 2 – 1) = 3

So, 2, 3, 3 are number of comparisons.

So average number of comparison

$$= \frac{2+3+3}{3} = \frac{8}{3}$$

30. (b)

$T(n) = n + (n+1) + (n+2) + ... + (n+k-1)$
$= nk + (1 + 2 + ... + (k-1))$
$= nk + (k-1)k/2$
$= O(nk + k^2)$
$= O(k(n+k))$

31. (c)

Insertion sort in best case = $O(n)$
and bubble sort = $O(n^2)$;
Selection sort = $O(n^2)$;
Merge sort = $O(n \log n)$

32. (b)

After one iteration the greatest element reaches its correct position i.e., last position, out of all the options only in option (b), the largest element is not present in the last position.
In option (a) and (c) the largest element that is 9 is present in the last position but in case of the (b) option it is not the case, therefore (b) option is not a possible outcome after one iteration of bubble sort.
Hence, option (b) is correct.

33. (c)

Binary insertion sort take minimum comparison if the list to be sorted is in reverse order and maximum number of comparison if they already in sorted order.

34. (d)

The worst case time complexity of algo given are:
Insertion sort = $\Theta(n^2)$
Merge sort = $\Theta(n \log n)$
Quick sort = $\Theta(n^2)$

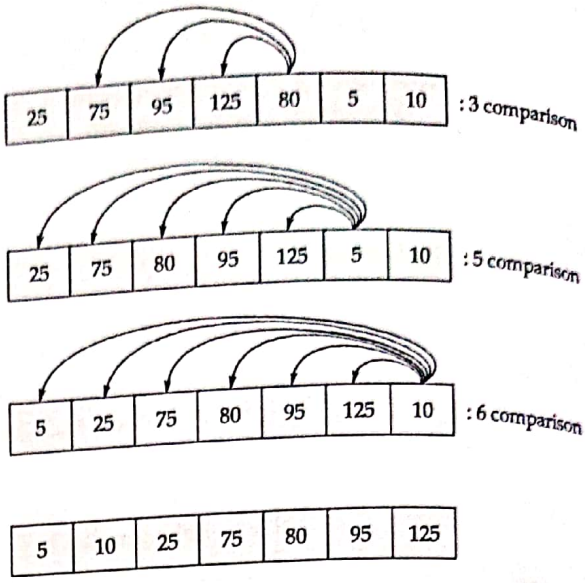
35. (b)

Merge sort is the best known algorithm to sort a linked list in $\theta(n \log n)$ time, and this is even in place sorting algorithm unlike array sorting by merge which is not in place.
There are $\log n$ elements in the linked list hence the overall T.C. will be $\theta(\log n * \log \log n)$.

36. (17)

$$17 = 3 + 5 + 6 + 3$$

3 comparison, one each for 75, 95, 125



37. (a)

The n^{th} largest element will be present in one of the leaf nodes and will take $O\left(\frac{n}{2}\right)$ time to find.

Therefore, $O(n)$ is the required complexity.

38. (b)

We have $\log n$ lists each of size $\frac{n}{\log n}$. And now, if we merge them by 2-ways, then at each level there are : $\frac{\log * n}{\log n} = n$ elements.

And total number of levels are = log (number of sorted list) = log log n
Thus, total time complexity = $n \log \log n$

39. (448)

- Here $n = 300$ elements.
- Compare each pair, you will get 150 numbers which are minimum in their respective pairs.
- Repeat above step now on remaining 150 numbers.
- Finally you will get minimum number in $(n-1)$ comparisons i.e., 299 comparisons.
- After first step we get 150 numbers which were minimum in their respective pairs. At the same time we got 150 numbers that were maximum in their respective pairs. Out maximum number can only belong to this remaining 150 numbers.

- So, now applying same algorithm for 150 numbers to find the maximum number.
- This will require $\frac{n}{2}-1$ comparisons i.e., 149 comparisons.
- In total, number of comparisons to find maximum and minimum with best case complexity is

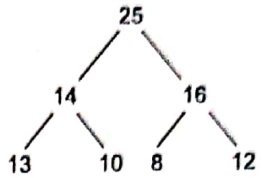
$$\left((n-1)+\left(\frac{n}{2}-1\right)\right)=\frac{3n}{2}-2=448$$

40. (d)
Let number of digit present in maximum array = d.
The complexity of radix sort = (number of digits in the maximum number * apply counting sorts.)
$= d * O(n + b)$
Here, large number is given = nd^2
Relationship between base and number of digits
$= \lfloor \log bn \rfloor + 1$
Therefore,
$\log_{10} nd^2 + 1 = d^2 \log n + 1$
$\cong (d^2 \log n + 1) * (O(n + 10)$
$= O(n * d^2 \log n)$

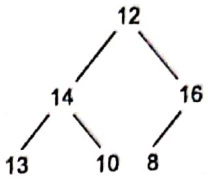
41. (b)
Merge sort for n letters in string will take $O(n \log n)$ time so for n strings its $O(n^2 \log n)$.
By 2-ways merge question is specifying that sorting is done using merge sort by dividing recursively the array into size of 2 and performing bottom up merging. So, option (b) is correct.

42. (a)
deletemin( ): Root element is the smallest element. Remove it and swap it with the rightmost leaf node to maintain heap property. Then search for the minimum element out of the remaining elements by looking in the lowermost min (even) level. Let it be at index '*m*'. Swap this element at index *m* with the root. Do this recursively for the subtree rooted at index m to maintain the min-max property.
deletemax( ): The max element is one out of the two children of root. Find it and then follow the same steps as deletemin( ) operation, this time choosing the max element out of the remaining to be swapped at every stage.

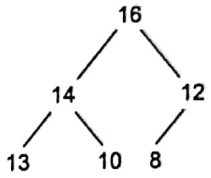
43. (c)
Given max heap is,



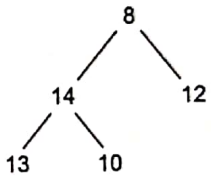
1. Deleting 25,



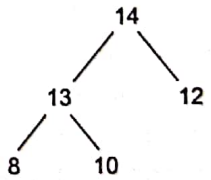
- Apply max heapify on root,



2. Deleting 16,



- Apply max heapify on root,



So the final contents of the array will be 14, 13, 12, 8, 10
Hence option (c) is correct.

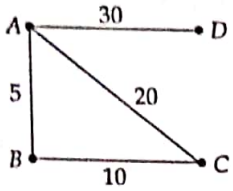
44. (a)
We can find the middle element of the heap, by simply accessing the $\frac{n}{2}^{th}$ element of the array which can be done in O(1) time. Further that element is replaced by the last element of the array in O(1) time. Now, perform the heapify operation which will take $O(\log n)$ time.
Time taken = $O(1) + O(1) + O(\log n)$
$= O(\log n)$

45. (a)

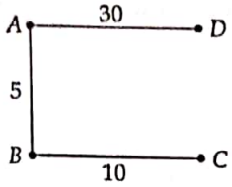
Since edges costs are distinct, so cheapest edge must be present in every minimum spanning tree while expensive edge is may not excluded from every minimum spanning tree.

Statement S_2 is false

Example:



MST will be:



So, most expensive edge is not excluded.

46. (14)

$(n-1)$ comparisons are required to find the biggest element in unsorted list of size n.

i.e., $(15-1) = 14$

47. (a)

Merge sort $\Rightarrow$ requires $\theta(n \log n)$ irrespective of the ordering of input.

Quick sort $\Rightarrow$ required $\theta(n^2)$ is worst case, input, $\theta(n \log n)$ in best case, so varies according to input. So option (b) is wrong.

Insertion sort $\Rightarrow$ best case $O(n)$ when input is sorted, requires $O(n^2)$ when input is sorted in decreasing order. So option (c) is wrong.

Although the "standard" selection sort algorithm requires $\theta(n^2)$ in every case, strongly the answer given in key in option (a). It should be (a) and (c) instead, if we see to it conceptually.

48. (d)

In Binary search we first compare the given element with middle element of the array. If the element matches with middle element, then return middle element index. Otherwise, go to left half of array or right half of array.

49. (b)

Traverse the array element wise and for each element, compare its value with its left adjacent and right adjacent element (if it exists). If the condition is satisfied, print the elements.

50. (a)

We can find the middle element of the heap, by simply accessing the $\frac{n}{2}^{\text{th}}$ element of the array which can be done in O(1) time. Further, that element is replaced by the last element of the array in O(1) time. Now, perform the heapify operation which will take O(log n) time.

$$\text{Time taken} = O(1) + O(1) + O(\log n)$$
$$= O(\log n)$$

51. (c)

In the worst case length of binary search tree can be $O(n)$.

So insertion of an object will take $O(n)$ time in worst case.

52. (c)

if $n = 2^k$

let at level 1, L_1 and L_2 is merged.

An item in L_1 is compared maximum times when all the items in L_2 is less than item in L_1.

Stay at leaf level k, an item is compared only once when merging.

At level $k-1$, an item is compared twice when merging.

At level 2, it is compared $\frac{n}{2}$ times and after merging we get the sorted array.

Total comparison of an item

$$= \frac{n}{2} + \frac{n}{(2)^2} + \ldots + \frac{n}{(2)^k}$$

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

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

53. (b)

The recurrences relation for such an implementation can be given by

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

By using Master theorem, we get

$a = 2$, $b = 2$ and $k = 2$

$$T(n) = \theta(n^2)$$

The most appropriate option is as we do not have $\theta(n^2)$ the most appropriate option is $O(n^2 \log n)$

54. (a)

(a) $f(n) = \Omega(n \log n)$ means $f(n) > C * n \log n$

(d) $f(n) = O(n \log n)$ means $f(n) <= C * n \log n$ which means that no comparison based algo takes more than $n \log n$ time. But we know there are algo like bubble sort, worst case of quick sort etc. that takes n^2 time which is more than $n \log n$.

So, option (a) is correct.

55. (c)

Quick sort is not a stable sorting algorithm the popularity acts as satellite data and the order of satellite data is preserved only by stable sorting algorithms.

56. (b)

If array is sorted in ascending order you need to keep two pointer one on left and other on right side. Now, subtract these two numbers and check whether its equal to k or not, if not then two possibilities comes whether some in less than k or greater than k.

If sum is greater than k, decrease right pointer and if sum less than k than increase left pointer so, max. It will take $O(n)$ item find whether such two number exists or not.

So, option (b) is correct.

57. (d)

Quick sort takes $O(n^2)$ time to sort the input (ascending order) into ascending order by using divide and conquer.

Insertion sort take $O(n)$ time to sort input (ascending order) into ascending order since need one comparison every time and no swapping.

Merge sort take $O(n \log n)$ and bubble sort can take $O(n)$ time.

58. (d)

Applying bubble sort on given input:

Step 1	8	22 ↔	7	9	31	5	13
Step 2	8	7	22 ↔	9	31	5	13
Step 3	8	7	9	22	31 ↔	5	13
Step 4	8	7	9	22	5	31 ↔	13
Step 5	8 ↔	7	9	22	5	13	31
Step 6	7	8	9	22 ↔	5	13	31
Step 7	7	8	9	5	22 ↔	13	31
Step 8	7	8	9 ↔	5	13	22	31
Step 9	7	8 ↔	5	9	13	22	31
Step 10	7 ↔	5	8	9	13	22	31
	5	7	8	9	13	22	31

59. (c)

Merge sort complexity is $\theta(n \log n)$, and so we need $10^6 \log 10^6 \approx 20 \times 106$ and assuming the constant factor is less than 5, number of instructions would be less than 10^8 and can be run within a second.

Worst case complexities of insertion sort is $O(n^2)$ and to sort 10^6 elements it needs 10^{12} operations and with 10^8 operations per second this would need 10^4 seconds = 2 hours 46 minutes.

Now, best case complexity of insertion sort is $O(n)$.

So, always in option 1 and 2 make them false.

Similarly, worst case of quick sort is $O(n^2)$ and this makes option (d) false.

So, option (c) is right.

60. (c)

To sort the first 10 element – maximum $O(10^2) = 0(1)$ time complexity.

Next, $n-60$ elements take $\theta(n-60)$ time as these would involve no shifting among themselves.

Final, 50 elements takes $\theta(50\ n) = \theta(n)$ time in worst case to place each of the elements in position. So, totally $\theta(n)$.

So, insertion sort is correct answer.

61. (c)

To find the first string, algorithm is similar to MaxMin algorithm using divide and conquer.

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

{$O(n)$ is to find the first string from any two subproblems}

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

62. (d)

Binary Search: $T(n) = T\left(\frac{n}{2}\right) + 1$ [Avg, Worst]

Merge Sort: $T(n) = 2T\left(\frac{n}{2}\right) + kn$ [Avg, Best, Worst]

Quick Sort: $T(n) = T(n-1) + n - 1$ [Worst Case]

Max Min: $T(n) = 2T\left(\frac{n}{2}\right) + C$ [Best, Avg, Worst]

63. (b)

Internal sorting: When the elements to be sorted are all in main memory while in external sort all are not present in memory during sorting, as in internal sort all elements are in main memory the time to read write is not that much significant as compared to external sorting.

64. (c)

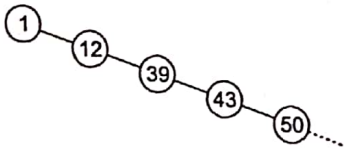
Best, worst and average time complexity of heap sort is $O(n \log n)$. So, option (c) is correct.

65. (b)

In worst case the BST may be Skewed BST. This case occurs when we insert the elements in increasing order or decreasing order.

Example: Consider we insert elements, in the order 1, 12, 39, 43, 50, ...

The BST would be



To find 'n' is the worst case we may have to traverse to bottom of tree which takes $O(n)$ time. Hence for both insertion and deletion worst case goes to $\Theta(n)$.

66. (d)

Best in terms of complexity:

Time complexity filter out and we have choice of merge and heap sort.

Space complexity selects heap sort as best due to extra space required by merge sort.

So, best is heap sort option (d) is correct.

67. (8)

Input sequence : 9, 8, 7, 6, 5, 4, 3, 2, 1, 0

I[st] iteration: 8, 9, 7, 6, 5, 4, 3, 2, 1, 0

2[nd] iteration: 7, 8, 9, 6, 5, 4, 3, 2, 1, 0

3[rd] iteration: 6, 7, 8, 9, 5, 4, 3, 2, 1, 0

4[th] iteration: 5, 6, 7, 8, 9, 4, 3, 2, 1, 0

5[th] iteration: 4, 5, 6, 7, 8, 9, 3, 2, 1, 0

6[th] iteration: 3, 4, 5, 6, 7, 8, 9, 2, 1, 0

7[th] iteration: 2, 3, 4, 5, 6, 7, 8, 9,, 1, 0

6[th] iteration: 1, 2, 3, 4, 5, 6, 7, 8, 9, 0

Hence, answer should be 8[th].

68. (c)

In the worst case length of binary search tree can be $O(n)$.

So insertion of an object will take $O(n)$ time in worst case.

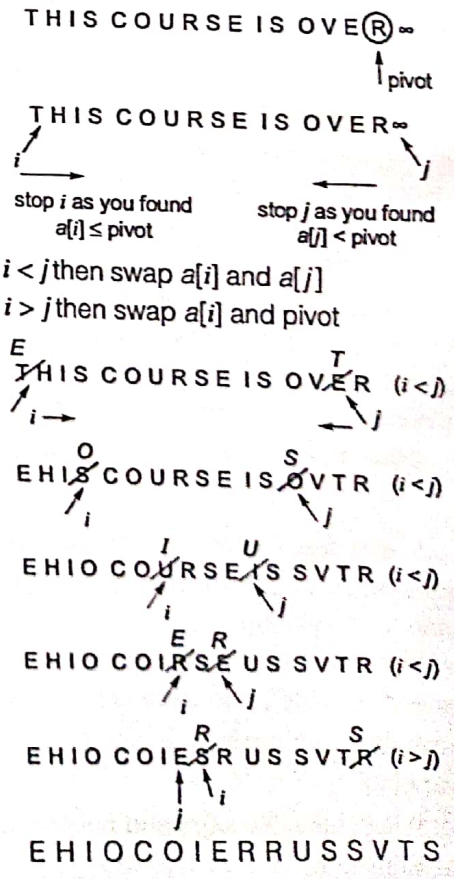
69. (a)

A sorting algo is said to be stable if it preserves the order of equal items.

There are algorithms which are stable naturally like bubble sort and insertion sort. And there are algorithm which need modification to exhibit this stable sorting property, example of such sorting algo is quick sort.

So, answer is (a) option.

70. (a)



71. (720)

We have to choose first element as pivot.

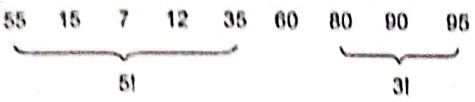
Here 60 is given as first element.

After first pass, the pivot element goes to it's exact location.

Here 60 goes to 6th place.

All the elements less than 60 go to left of 60 and all the elements greater than 60 go to right of 60.

After 1st pass



$\Rightarrow 5! \times 3!$

$\Rightarrow$ 720 possible arrangements.

72. (a)

The usual $O(n^2)$ implementation of insertion sort to sort an array uses binary search to identify the position, the worst case running time will remain $O(n^2)$. The time required to insert an element x into a binary search tree is bounded by a constant times the number of comparisons made between x and elements already in the tree. Thus we can measure time in terms of the number of comparisons made. In worst case implementation of insertion sort takes $O(n^2)$. So in the worst case adding n elements to a binary search tree could require $O(n^2)$ time. In expected case means element are sorted in increasing order adding n elements to a binary search tree could require $O(n \log n)$ time.

73. (169)

Given files

20, 21, 22, 23 } 41 – 1 = 40 comparisons

41, 22, 23 } 45 – 1 = 44 comparisons

$\left.\begin{matrix}41,\ 45\\ 85\end{matrix}\right\}$ 86 – 1 = 85 comparisons

40 + 44 + 85 = 169

74. (840)

$\left\lfloor \frac{105}{5} \right\rfloor = 21$

4 way merge-sort because last one is extra use to store infinity.

$$\frac{21}{4} = 6$$

$$\frac{6}{4} = 2$$

2 are merged so, total = $4 \times 2 \times 105 = 840$

75. (d)

At each level, the array of size n is getting divide into 2 sub arrays of size $\left(\frac{n}{4}\right)$ and $\left(\frac{3n}{4}\right)$ along with an extra work for choosing pivot which require $O(n)$ time.

So, recurrence will be form:

$$T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{3n}{4}\right) + O(n)$$

This can be solved using recursion tree.

Lower bound for this recurrence will be $\Omega(n \log_4 n)$ and upper bound will be $O(n \log_{4/3} n)$ which gives $T(n) = \theta(n \log n)$

So, option (d) is correct.

76. (b)

Running time of quick sort = $n \log_2 n$

Hence, $n = 1000$

$100 \text{ sec} = c \times 1000 \times \log 1000$

$c = 0.01$

For 100 names,

Time = $0.01 \times 100 \times \log 100 = 6.7$ sec

77. (b)

The relation $T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{3n}{4}\right) + n$

The pivot element is selected in such a way that it will divide the array into $\frac{1}{4}^{\text{th}}$ and $\frac{3}{4}^{\text{th}}$ always solving this relation give $\Theta(n \log n)$.

78. (b)

Selection sort is best if efficiency is in terms of swaps.

Number of swaps in selection sort is of $O(n)$.

79. (c)

Along with Radix sort two other sorting algo fall in this categories which are counting sort and bucket sort, so option (c) is right.

80. (c)

1. Priority queue can be efficiently implemented using binary heap because it supports insert( ), delete( ), extract max( ), decrease key( ) in $O(\log n)$ time.

2. Order statistics: Heap data structure can be used to efficiently find k^{th} smallest (or layers) element in array.

Method for getting k^{th} largest

- Build Maxheap $\rightarrow O(n)$
- Use Extract max( ) k times to get k max elements from the max heap $\rightarrow O(k \log n)$

Time complexity : $O(n + k \log n)$

Hence, option (c) is correct.

81. (d)

From the given description, the list is "almost sorts".

(a) Bubble sort: sorted array as input is best case for this sort, best case complexity is $O(n^2)$.

(b) Selection sort: Sorted array as input is best case for this sort. Best case complexing is $O(n^2)$.

(c) Quick sort: Sorted array as input is worst case for this sort. Worst case complexity is $O(n^2)$.

(d) Insertion sort: Sorted array as input is best case for this sort. Best case complexity is $O(n)$.

So, correction option is (d) insertion sort.

82. (c)

Only case 3 and 5 ensures $O(n \log n)$ time complexity always.

Case 1: Choosing middle element doesn't mean that pivots location is in middle, after the partition algorithm. In worst case pivot may go either first or last location ($O(n^2)$). Similarly Case 2, Case 4 does not guarantee $O(n \log n)$.

But cases 3 and 5 are similar and makes sure that pivot always goes to middle location, after the partition algorithm.

83. (256)

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

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

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

$$324 = c_1 \cdot n^2 \quad [n = 18]$$
$$324 = c_1 \cdot 18^2$$
$$c_1 = 1$$

Substitute c_1 $n = 1$ in equation (1)

$$c_1 \cdot n \log n = 2048 \quad [c_1 = 1]$$
$$n \log n = 2048$$

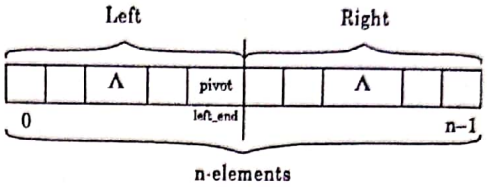
Put $n = 2^K$

$$2^K \times K = 2048 \quad \ldots(2)$$

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

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

84. (a)



left_end (is pivot position) = partition (a, n);

Note: k^{th} smallest is same as an element at index $(k-1)$ in the sorted array.

if (left_end+1 = = k) // k^{th} smallest found.

return a[left_end]; // return pivot element.

if (left_end +1 > k) // k^{th} smallest is before pivot.

return k^{th}_smallest (a, left_end, k); // search before pivot

else // k^{th} smallest is after pivot (right part)

return kth_smalest (a+left_end+1, n−(left_end−1), k−(left_end+1));

(a+left_end+1): address of first element after pivot (right part)

n−(left_end−1): Number of elements in right part

85. (c)

Selection sort will be answer. It uses minimum swaps.

86. (b)

$(n-1)$ swaps in all cases.

87. (d)

Bubble sort can be improved by using inner loop while required line swapping element while required. Otherwise inner loop should skip.

(a) Doing both way sort is not an efficient solution, as if we do left to right increasing order sorting then right to left decreasing order.

(b) Through quick sort has time complexity ($O(n \log n)$ in average case which is better than bubble sort $O(n^2)$, but that is not needed forgetting better version of bubble sort. Actually here we are using quick sort and then again bubble sort, which is not improving time or space complexity of bubble sort.

(c) Swapping in separate array will increase space complexity of bubble sort.

So, answer will be (d).

88. (b)

Merge sort uses divide and conquer so on array is divided into two parts in such a way that we recursively apply merger sort in two halves.
Once both halves are sorted then we compare and copy and merge them to form a single sorted array. So, answer should be (b).

89. (b)

In the best case quick sort algorithm makes $n \log (n)$ comparisons. So $1000 \times \log (1000) = 9000$ comparisons, which takes 100 sec. To sort 100 names a minimum of 100 (log 100) = 600 comparisons are needed.

This takes $100 \times \frac{600}{9000} = 6.7$ sec.

90. (c)

N = 8 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |

Pass 1 : Even number activated:	8	7	6	5	4	3	2	1
7 < 8 so, Swap	7	8	5	6	3	4	1	2
Pass 2 : Odd number activated:	7	8	5	6	3	4	1	2
5 < 8 so, Swap	7	5	8	3	6	1	4	2
Pass 3 : Even number activated	7	5	8	3	6	1	4	2
	5	7	3	8	1	6	2	4
Pass 4 : Even number activated	5	7	3	8	1	6	2	4
	5	3	7	1	8	2	6	4

91. (c)

We know that in balanced BST there are $\log_2 n$ levels in both worst as well as best case where 'n' is number of elements

$\therefore \quad \log_2 (n 2^n) = \log_2 n + \log_2 2^n$

$\log_2 n + n \log_2 2 = \log_2 n + n = \Theta(n)$

92. (b)

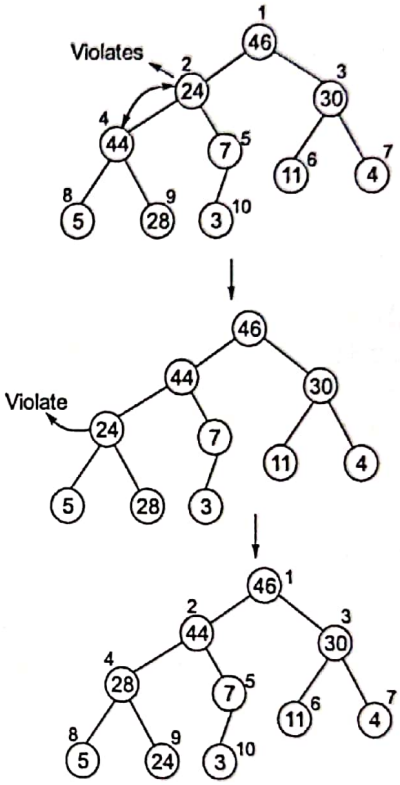
Heap sort is an implementation of selection sort using the input array as a heap representing a descending priority queue. Heap sort algorithm is divided into two phase. In the first phase the max heap is created and the second phase (selection phase) deletes the elements from the priority queue using sift down operation.
So, option (b) correct.

93. (c)

Heap sort is a comparsion based sorting algorithm and has time complexity $O(n \log n)$ in the average case. Heap sort in an inplace algorithm as it need $O(1)$ of auxiliary space. Heap sort uses heap and operations on heap can change the relative order of items with the same key values.
Therefore, heap sort is not a stable sort.

94. (d)

In max-heap element at each node is smaller than or equal to the element at its parent node. On applying the heapify procedure on item at position 2, it will be in position 9 a shown below.



So, it will on 9 position.

95. (c)

The max heap is also known as descending heap. Max heap of link n is an almost complete binary true of n nodes such that the element at each node is less than or equal to the element at its parent node. So, option (c) is correct.

96. (c)

The time complexity of heap sort is
$O(k \log k)$ for k input elements,
for $k = \log n / \log (\log n)$
$O(k \log k) = O(\log n / \log (\log n) * \log \log n / \log \log n))))$

$= O(\log n)$

Hence correct option is (c).

2. Order statistics: Heap data structure can be used to efficiently find k^{th} smallest (or layers) element in array.

Method for getting k^{th} largest

- Build Maxheap $\rightarrow O(n)$
- Use Extract max( ) k times to get k max elements from the max heap $\rightarrow O(k \log n)$

Time complexity : $O(n + k \log n)$

Hence, option (c) is correct.

81. (d)

From the given description, the list is "almost sorts".

(a) Bubble sort: sorted array as input is best case for this sort, best case complexity is $O(n^2)$.

(b) Selection sort: Sorted array as input is best case for this sort. Best case complexing is $O(n^2)$.

(c) Quick sort: Sorted array as input is worst case for this sort. Worst case complexity is $O(n^2)$.

(d) Insertion sort: Sorted array as input is best case for this sort. Best case complexity is $O(n)$.

So, correction option is (d) insertion sort.

82. (c)

Only case 3 and 5 ensures $O(n \log n)$ time complexity always.

Case 1: Choosing middle element doesn't mean that pivots location is in middle, after the partition algorithm. In worst case pivot may go either first or last location ($O(n^2)$). Similarly Case 2, Case 4 does not guarantee $O(n \log n)$.

But cases 3 and 5 are similar and makes sure that pivot always goes to middle location, after the partition algorithm.

83. (256)

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

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

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

$$324 = c_1 \cdot n^2 \quad [n = 18]$$
$$324 = c_1 \cdot 18^2$$
$$c_1 = 1$$

Substitute c_1 $n = 1$ in equation (1)

$$c_1 \cdot n \log n = 2048 \quad [c_1 = 1]$$
$$n \log n = 2048$$

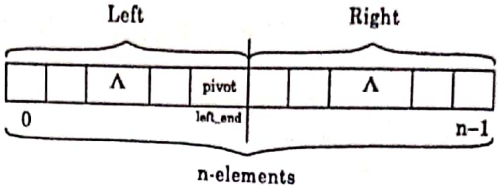
Put $n = 2^K$

$$2^K \times K = 2048 \quad \ldots(2)$$

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

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

84. (a)



left_end (is pivot position) = partition (a, n);

Note: k^{th} smallest is same as an element at index $(k-1)$ in the sorted array.

```
if (left_end+1 = = k) // kth smallest found.
return a[left_end]; // return pivot element.
if (left_end +1 > k) // kth smallest is before pivot.
return kth_smallest (a, left_end, k); // search before pivot
else // kth smallest is after pivot (right part)
return kth_smalest (a+left_end+1, n-(left_end-1), k-(left_end+1));
```

address of first element after pivot (right part)

Number of elements in right part

85. (c)

Selection sort will be answer. It uses minimum swaps.

86. (b)

$(n-1)$ swaps in all cases.

87. (d)

Bubble sort can be improved by using inner loop while required line swapping element while required. Otherwise inner loop should skip.

(a) Doing both way sort is not an efficient solution, as if we do left to right increasing order sorting then right to left decreasing order.

(b) Through quick sort has time complexity ($O(n \log n)$ in average case which is better than bubble sort $O(n^2)$, but that is not needed forgetting better version of bubble sort. Actually here we are using quick sort and then again bubble sort, which is not improving time or space complexity of bubble sort.

(c) Swapping in separate array will increase space complexity of bubble sort.

So, answer will be (d).

88. (b)

Merge sort uses divide and conquer so on array is divided into two parts in such a way that we recursively apply merger sort in two halves.

Once both halves are sorted then we compare and copy and merge them to form a single sorted array. So, answer should be (b).

89. (b)

In the best case quick sort algorithm makes $n \log(n)$ comparisons. So $1000 \times \log(1000) = 9000$ comparisons, which takes 100 sec. To sort 100 names a minimum of $100 (\log 100) = 600$ comparisons are needed.

This takes $100 \times \frac{600}{9000} = 6.7$ sec.

90. (c)

N = 8 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |

Pass 1 : Even number activated: 7 < 8 so, Swap

8	7	6	5	4	3	2	1
7	8	5	6	3	4	1	2

Pass 2 : Odd number activated: 5 < 8 so, Swap

7	8	5	6	3	4	1	2
7	5	8	3	6	1	4	2

Pass 3 : Even number activated

7	5	8	3	6	1	4	2
5	7	3	8	1	6	2	4

Pass 4 : Even number activated

5	7	3	8	1	6	2	4
5	3	7	1	8	2	6	4

91. (c)

We know that in balanced BST there are $\log_2 n$ levels in both worst as well as best case where 'n' is number of elements

$\therefore \quad \log_2(n\,2^n) = \log_2 n + \log_2 2^n$

$\log_2 n + n \log_2 2 = \log_2 n + n = \Theta(n)$

92. (b)

Heap sort is an implementation of selection sort using the input array as a heap representing a descending priority queue. Heap sort algorithm is divided into two phase. In the first phase the max heap is created and the second phase (selection phase) deletes the elements from the priority queue using sift down operation.

So, option (b) correct.

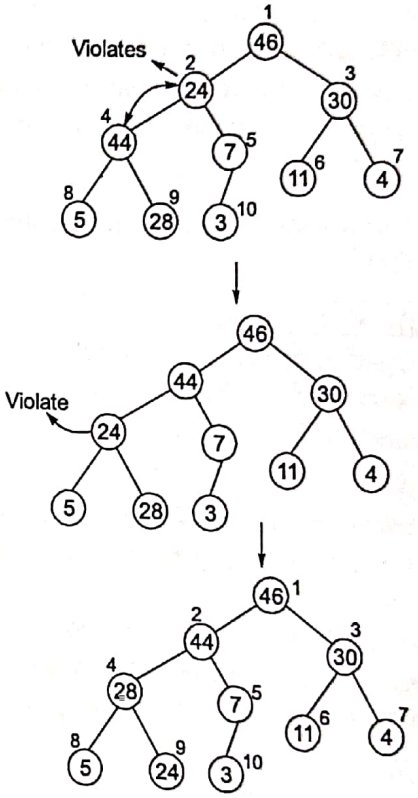
93. (c)

Heap sort is a comparsion based sorting algorithm and has time complexity $O(n \log n)$ in the average case. Heap sort in an inplace algorithm as it need $O(1)$ of auxiliary space. Heap sort uses heap and operations on heap can change the relative order of items with the same key values.

Therefore, heap sort is not a stable sort.

94. (d)

In max-heap element at each node is smaller than or equal to the element at its parent node. On applying the heapify procedure on item at position 2, it will be in position 9 a shown below.



So, it will on 9 position.

95. (c)

The max heap is also known as descending heap. Max heap of link n is an almost complete binary true of n nodes such that the element at each node is less than or equal to the element at its parent node. So, option (c) is correct.

96. (c)

The time complexity of heap sort is $O(k \log k)$ for k input elements,

for $k = \log n / \log(\log n)$

$O(k \log k) = O(\log n / \log(\log n) * \log\log n / \log\log n))))$

$= O(\log n)$

Hence correct option is (c).

97. (a)
Radix sort used to sort number in linear time.

98. (b)
First statement is true and second is false.

99. (b, c)
Quicksort gives the worst case for the sorted array. Quicksort is not useful when we apply for sorted array compare to insertion sort which gives O(n) when we have sorted array.

100. (c)
The 4th optimization is generally not used, it reduces the worst case time complexity to O(n Log$n$), but the hidden constants are very high.

101. (c)
We can use Binary Search to find the missing element.

102. (a, d)
It is again just memorizing standard results:
The expected result is $\theta(n \log n)$, and the worst case result is $\theta(n^2)$.

103. (b, c)
The minimum possible depth occurs when we pick median all the time, and the minimum depth would be $\log n = \theta(\log n)$.
The maximum possible depth occurs when we pick the smallest element all the time, and the maximum depth would be $n = \theta(n)$.

104. (a, d)
Example for best case sub array
A: 10, 20, 30
B: 40, 50, 60, 70
Worst Case
A: 11, 21, 31, 41
B: 10, 20, 30, 40, 50

105. (a, b)
We can use binary search.

106. (b)
Merge sort - $n \log n$
Quick sort - $n*n$
Bubble sort - $n*n$
Insertion sort - $n*n$
Answer is merge sort.

107. (a, c, d)
Among the options, only Merge sort divides the list in sub-list, sorts and then merges them together.

■■■■

4 CHAPTER

Greedy Techniques

Multiple Choice Questions & NAT Questions

Q.1 Dijkstra's algorithm is used to
(a) Create LSAs
(b) Flood an internet with information
(c) Calculate the routing tables
(d) Create a link state database

Q.2 The space complexity of Dijkstra's algorithm is
(a) $O(V^2)$ (b) $O(V \log E)$
(c) $O(E)$ (d) $O(V^2 \log E)$

Q.3 Single shortest path problem can be implemented by Greedy algorithms using
(a) Singly linked list (b) Min Heap
(c) AVL tree (d) All of the above

Q.4 Consider the following steps:
S_1: Characterize the structure of an optimal solution.
S_2: Compute the value of an optimal solution in bottom-up fashion.

Which of the following step(s) is/are common to both dynamic programming and Greedy algorithm?
(a) Only S_1
(b) Only S_2
(c) Both S_1 and S_2
(d) Neither S_1 or S_2

Q.5 If Kruskal's algorithm is used for finding a minimum spanning tree of a weighted graph G with n vertices and m edges and edge weights are already given in a sorted list, then, what will be the time complexity to compute the minimum cost spanning tree given that union and find operations amortized O(1)?
(a) $O(m \log n)$ (b) $O(n)$
(c) $O(m)$ (d) $O(n \log m)$

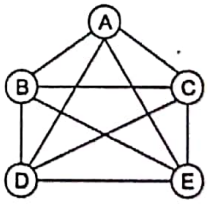
Q.6 Which of the following is not true about a Huffman tree?
(a) The most frequently used character always appear near the top of the tree.
(b) Normally, decoding a message involves repeated following a path from root to a leaf.
(c) In coding a character you typically start at a leaf and work upward.
(d) The tree can be generated by removal and insertion operations on a priority queue.

Q.7 Consider the following instance of Knapsack problem.

Item	x_1	x_2	x_3	x_4	x_5
Profit	15	12	9	16	17
Weight	2	5	3	4	6

The maximum weight of 12 is allowed in the Knapsack. Find the value of maximum profit with the optimal solution of the fractional Knapsack problem.
(a) 31 (b) 40.2
(c) 48.5 (d) None of these

Q.8 The number of spanning tree possible for the following graph is _____.



Q.9 Consider the following code segment to find the n^{th} Fibonacci number.

```
Fib (n)
{
    if(n == 0) {return 0;}
    if (n == 1) {return 1;}
    else{return(fib(n – 1) + fib(n – 2));}
}
```

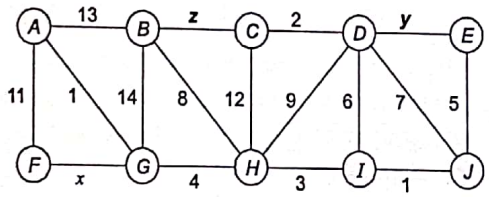
The time complexity of the above code and time complexity of the same problem solved using dynamic programming is _____.

(a) $O(n^2), O(n)$ (b) $O(2^n), O(n)$
(c) $O(2^n), O(n^2)$ (d) None of above

Q.10 Suppose that you implement Dijkstra's algorithm using a priority queue algorithm that requires $O(V)$ time to initialize, worst case $F(V, E)$ time for each EXTRACT-MIN operation and worst case $g(V, E)$ time for each DECREASE-KEY operation. How much (worst-case) time does it take to run Dijkstra's on an input graph $G = (V, E)$?
(a) $O(V \cdot f(V, E) + E \cdot g(V, E))$
(b) $O(E)$
(c) $O(V + E)$
(d) $O(V^2 + (V, E) + E \cdot g(V, E))$

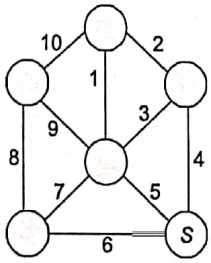
Q.11 The number of spanning trees of an undirected complete graph with 7 nodes is _____.

Q.12 Suppose that minimum spanning tree of the following edge weighted graph contains the edges with weights x, y and z



What is the maximum value of $x + y + z$?

Q.13 Consider the following graph:



Assume that node 'S' is the starting vertex for prim's algorithm. Which of the following can be the correct order of edges in which they are added to construct the minimal spanning tree?
(a) 4, 2, 1, 6, 8 (b) 4, 1, 2, 6, 8
(c) 4, 2, 1, 7, 10 (d) None of these

Q.14 Assume coins with denominator 20, 15, 5 and 1 are available, we are required to make a sum of 33, using minimum number of coins. Difference between answer using greedy technique and correct answer will be _____?

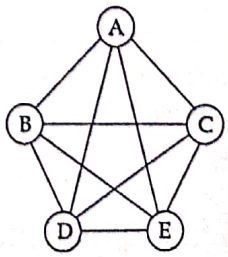
Q.15 Kruskal's algorithm for finding a minimum spanning tree of a weighted graph G with n vertices and m edges has the time complexity of
(a) $O(n^2)$ (b) $O(mn)$
(c) $O(m + n)$ (d) $O(m \log n)$

Q.16 The number of spanning trees for a complete graph with seven vertices is
(a) 2^5 (b) 7^5
(c) 3^5 (d) $2^{2 \times 5}$

Q.17 Consider the problem of merging (n) sorted files f_1, f_2, f_n of lengths z_1, z_2 .... z_n records into one file F of length $\sum_{i=1}^{n} z_i$ while minimizing the total number of records moves. Only two files can be merged at once. Given the following files and their lengths find the minimum number of record movements needed to merge the files into one file. $\{F_1; 10\}, \{F_2; 30\}, \{F_3; 15\}, \{F_4; 20\}, \{F_5; 5\}, \{F_6; 16\}, \{F_7; 4\}$
(a) 253 (b) 259
(c) 254 (d) 260

Q.18 Let w be the minimum weight among all edge weights in an undirected connected graph. Let e be a specific edge of weight w. Which of the following is FALSE?
(a) There is a minimum spanning tree containing e.
(b) If e is not in a minimum spanning tree T, then in the cycle formed by adding e to T, all edges have the same weight.
(c) Every minimum spanning tree has an edge of weight w.
(d) e is present in every minimum spanning tree.

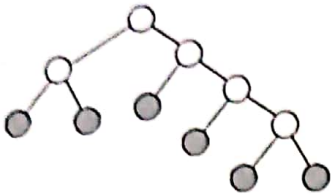
Q.19 Consider a graph 'G' given below:



If edges are having weight 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10. Then maximum possible weight that a

minimum weight spanning tree of G can have is _____.

Q.20 Which of the following frequency lists generates the Huffman tree as shown in figure.



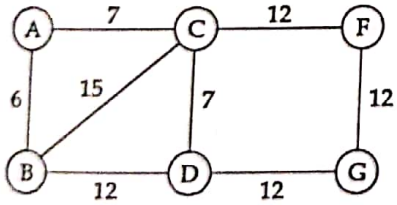
(a) 2, 2, 3, 5, 8, 14 (b) 2, 2, 3, 5, 5, 5
(c) 1, 2, 2, 2, 5, 7 (d) None of the above

Q.21 Given a sequence of ten jobs where every job has a deadline and associated profit if the job is finished before the deadline. It is also given that every job takes single unit of time.

Job	Deadline	Profit
A	4	100
B	1	19
C	3	15
D	2	50
E	3	45
F	4	46
G	1	25
H	5	70
I	2	27
J	1	26

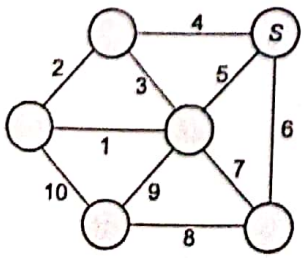
What is maximum total profit if and one job can be scheduled at a time _____?

Q.22 Consider the following graph G:



The total number of minimum spanning trees using Prim's or Kruskal's algorithm are _____.

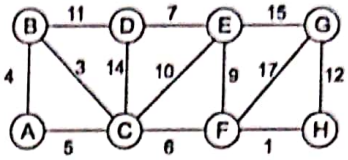
Q.23 Consider the following graph:



Assume node 'S' is the starting vertex for prim's algorithm. Which of the following can be the correct order of edges in which they are added to construct the minimal spanning tree?

(a) 4, 2, 1, 6, 8 (b) 4, 1, 2, 6, 8
(c) 4, 2, 1, 7, 10 (d) None of these

Q.24 Consider the following graph:



Apply single source shortest path algorithm on the given graph using vertex 'A' as the source. What is the maximum possible distance between vertex A to vertex G (Assume exclude infinity) _____?

Q.25 Consider message made up entirely of vowels (A, E, I, O, U). The below table gives the probabilities for each of the vowels.

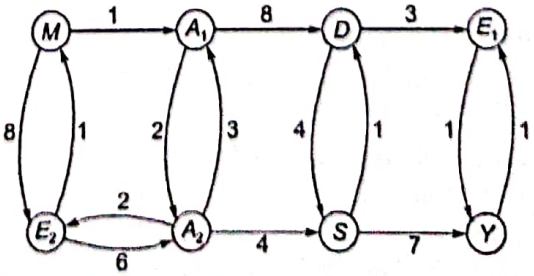
Character	Probability
A	0.24
E	0.32
I	0.21
O	0.15
U	0.08

Using Huffman algorithm, the number of bits of an encodes message transmitting 100 vowels is _____?

Q.26 Consider a weighted complete graph G on the vertex set $\{v_1, v_2, \ldots, v_n\}$ such that the weight of the edge (v_i, v_j) is $2|i-j|$. The weight of a minimum spanning tree of G is

(a) $n-1$ (b) $2n-2$
(c) $\left(\frac{n}{2}\right)$ (d) n^2

Q.27 Consider the following graph:



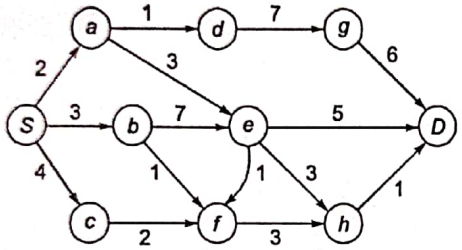
Find which of the following edge is never used to compute the shortest path from single source(M) to every other vertex using Dijkstra's algorithm.

(a) (M, A_1) (b) (D, E_1)
(c) (A_2, S) (d) (S, Y)

Q.28 Find the running time of Dijkstra's algorithm on complete graph of n-vertices.
(a) $O(n)$ (b) $O(n^2)$
(c) $O(n^2 \log n)$ (d) $O(n^3)$

Q.29 Suppose that the probabilities of searching for certain words in a document were: BAT (18%), CAT(22%), DOG (18%), EGG (20%), HAT (22%). Complete the average search cost if we use Greedy Approach to insert these words in an optimal Binary tree? (Upto 2 decimal) _____?

Q.30 Consider the following graph G:



What is minimum distance from S to D?
(a) 8 (b) 16
(c) 9 (d) 15

Q.31 Let $G(V, E)$ an undirected graph with positive edge weights. Dijkstra's single source-shortest path algorithm can be implemented using the binary heap data structure with time complexity?
(a) $O(|V|^2)$ (b) $O(|E|+|V|\log|V|)$
(c) $O(|V|\log|V|)$ (d) $O((|E|+|V|)\log|V|)$

Q.32 To implement Dijkstra's shortest path algorithm on unweighted graphs so that it runs in linear time, then data structure to be used is
(a) Queue (b) Stack
(c) Heap (d) B-Tree

Q.33 Which of the following statement(s) is/are correct regarding Bellman-Ford shortest path algorithm?
P. Always finds a negative weighted cycle, if one exists.
Q. Finds whether any negative weighted cycle is reachable from the source
(a) P only (b) Q only
(c) Both P and Q (d) Neither P nor Q

Q.34 Let G be a weighted connected undirected graph with distinct positive edge weights. If every edge weight is increased by the same value, then which of the following statements is/are TRUE?
P. Minimum spanning tree of G does not change
Q. Shortest path between any pair of vertices does not change
(a) P only (b) Q only
(c) Neither P nor Q (d) Both P and Q

Q.35 Let $G = (V, E)$ be any connected undirected edge-weighted graph. The weights of the edges in E are positive and distinct. Consider the following statements:
I. Minimum Spanning Tree of G is always unique.
II. Shortest path between any two vertices of G is always unique.
Which of the above statements is/are necessarily true?
(a) I only (b) II only
(c) Both I and II (d) Neither I nor II

Q.36 Consider the following instance of the Knapsack problem: $n = 3$, $w = 50$, $(V_1, V_2, V_3) = (60, 100, 120)$ and $(W_1, W_2, W_3) = (10, 20, 30)$.
Solve the given Knapsack problem applying Greedy algorithm _____?

Q.37 A thief is robbing a store and can carry a maximum weight of 18 into his knapsack. These are 6 items available in the store and weight of i^{th} item is W_i and its profit is P_i. The items can be broken into smaller places, hence the thief can select fractions of items. The items should be selected in such a way that the thief will carry those items for which he will gain maximum profit. What is the maximum profit the thief can gain?

Item	1	2	3	4	5	6
Profit	7	10	3	3	26	19
Weight	3	5	2	1	12	10

Q.38 Consider the following statements:
I. For every weighted graph and any two vertices s and t, Bellman-Ford algorithm starting at s will always return a shortest path to t.
II. At the termination of the Bellman-Ford algorithm, even if graph has negative weight cycle, a correct shortest path is found for a vertex for which shortest path is well-defined.
Which of the above statements are true?
(a) Only I (b) Only II
(c) Both I and II (d) None of these

Q.39 Assume a graph G has negative weight edges. Which of the following statement is correct when the Dijkstra's algorithm is applied on G?
(a) Always produces correct results
(b) Always produces incorrect results
(c) It always terminates but may produce incorrect results
(d) None of the above

Q.40 Consider a weight complete graph G on the vertex set $\{V_1, V_2, ...V_n\}$ such that the weight of edge (V_i, V_j) is $4|i-j|$. The weight of minimum cost spanning tree of G is:
(a) $4n^2$ (b) n
(c) $4n-4$ (d) $2n-2$

Q.41 Assume that letters p, q, r, s, t and u have probabilities $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}$ and $\frac{1}{32}$ respectively. What is the average length of the message using Huffman's coding?

Q.42 Which of the following is true about Huffman coding?
(a) Huffman coding may become lossy in some cases
(b) In Huffman coding no code is prefix of any other code
(c) Huffman codes may not be optimal lossless codes in some cases.
(d) All of the above

Q.43 Let k_n be the complete graph on "n" vertices labelled $\{1, 2, ...n\}$ with $m = \frac{n(n-1)}{2}$ edges. What is the number of spanning trees of k_n?
(a) $\frac{m}{n-1}$ (b) m^{n-1}
(c) n^{n-2} (d) n^{n-1}

Q.44 Let $G = (V, E)$ be an undirected graph which is connected simple (P.e. no parallel edge or self loop) graph with the weight function $W: E \to R$ on its edge set. Let $W(e_1) < W(e_2) < ....< W(e_m)$ where $E = \{e_1, e_2, e_3 .... e_m\}$.
(a) The tree T has to contain the edge e_1.
(b) The minimum weight edge incident on each vertex has to be present in T.
(c) T is unique minimum spanning tree in G.
(d) If we replace each edge weight $W_i = W(e_i)$ by its square W_i^2. Then T must still be a minimum spanning tree of this new instance.

Q.45 Find the maximum possible value for fractional Knapsack problem?

Item	a	b	c	d	e	f	g	h	i	j
Weight	3	5	2	1	12	10	9	9	4	1
Value	7	10	3	3	26	19	18	17	5	4

Knapsack capacity (W) = 20.

Q.46 Are all tasks completed in the schedule that gives maximum profit?
(a) All tasks are completed
(b) T_1 and T_6 are left out
(c) T_1 and T_8 are left out
(d) T_4 and T_6 are left out

Q.47 What is the maximum profit earned?
(a) 147 (b) 165
(c) 167 (d) 175

Q.48 Given a set of tasks with deadlines and total profit earned on completion of the task. Assume any task will take one unit of time and any task can't execute beyond its deadline. What is optimal sequence of tasks that can be executed to maximize the total profit?

Tasks	Deadline	Profit
T_1	10	1
T_2	11	4
T_3	6	5
T_4	9	3
T_5	12	4
T_6	13	1
T_7	7	3
T_8	5	5
T_9	4	2
T_{10}	8	1

(a) T_1, T_6, T_4, T_5, T_7 (b) T_6, T_4, T_2, T_5, T_3
(c) T_6, T_7, T_2, T_5, T_8 (d) $T_6, T_7, T_5, T_2, T_{10}$

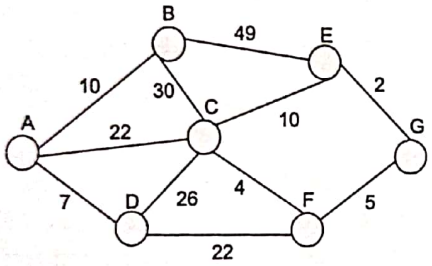
Q.49 How many bits may be required for encoding the string "APPLIED COURSE" using Huffman coding?

Q.50 Which of the following statements is invalid?
(a) A graph where all edge weights are distinct can have more than one shortest path between two vertices.

(b) Multiplying all edge weights by a positive number might change the cost of minimum spanning tree.
(c) Both (a) and (b)
(d) Neither (a) nor (b)

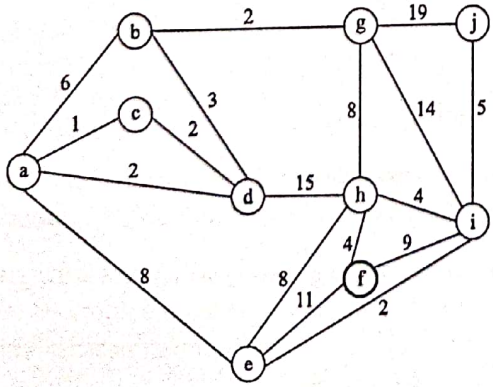
Q.51 Let G be an undirected connected graph with distinct edge weights. Let e_{max} be the edge with maximum weight and e_{min} the edge with minimum weight. Which of the following statements is false?
(a) Every minimum spanning tree of G must contain e_{min}
(b) If e_{max} is in a minimum spanning tree, then its removal must disconnect G
(c) No minimum spanning tree contains e_{max}
(d) G has a unique minimum spanning tree

Q.52 Consider the undirected graph below:



Using Prim's algorithm to construct a minimum spanning tree starting with node A, which one of the following sequences of edges represents a possible order in which the edges would be added to construct the minimum spanning tree?
(a) (E,G), (C,F), (F,G), (A,D), (A,B), (A,C)
(b) (A,D), (A,B), (A,C), (C,F), (G,E), (F,G)
(c) (A,B), (A,D), (D,F), (F,G), (G,E), (F,C)
(d) (A,D), (A,B), (D,F), (F,C), (F,G), (G,E)

Q.53 What is the weight of a minimum spanning tree of the following graph?



(a) 29 (b) 31
(c) 38 (d) 41

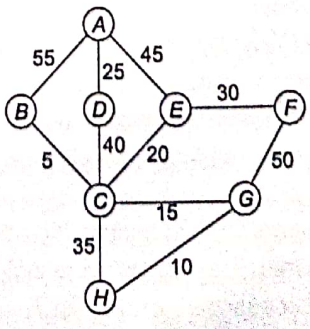
Q.54 Consider the following statements:
S_1: In a connected undirected graph $G = (V, E)$ with distinct edge costs, the cheapest edge belongs to every minimum spanning tree.
S_2: In a connected undirected graph $G = (V, E)$ with distinct edge costs, the most expensive edge is excluded from every minimum spanning tree.

Which of the following is true?
(a) Only S_1 (b) Only S_2
(c) Both S_1 and S_2 (d) Neither S_1 nor S_2

Q.55 Consider G = (V, E) be a connected graph with n vertices and m edges with distinct positive edge weight. T = (V, E') be a spanning tree of G and bottleneck edge of T is the edge with greatest cost in T. A spanning tree T of G is a minimum bottleneck spanning tree if there is no spanning tree T' of G with cheaper bottleneck edge. Which of the following statement true?
S_1: Every minimum bottleneck tree of G is a minimum spanning tree of G.
S_2: Every minimum spanning tree of G is a minimum bottleneck tree of G.
(a) Only S_1 (b) Only S_2
(c) Both S_1 and S_2 (d) Neither S_1 nor S_2

Q.56 Let the weight of minimum spanning tree of given graph G as per Kruskal's algorithm is w_K and as per prim's algorithm is w_P then
(a) $w_K \leq w_P$
(b) $w_K \geq w_P$
(c) $w_K = w_P$
(d) Any of the above is possible depends on graph structure

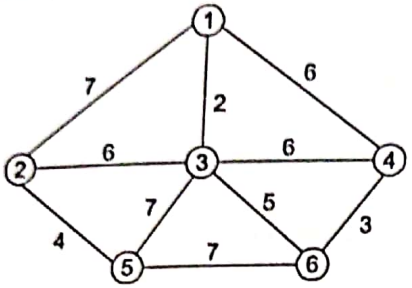
Q.57 Given the graph (starting with A)



What is the cost of minimum cost spanning tree.

(a) 195 (b) 150
(c) 145 (d) 140

Q.58 Consider the undirected weighted graph in figure. The minimum cost spanning tree for this graph has the cost.



(a) 18 (b) 20
(c) 24 (d) 22

Q.59 The total number of spanning trees that can be drawn using five labelled vertices is _____?

Q.60 G is a graph on n vertices and $2n-2$ edges. The edges of G can be partitioned into two edge-disjoint spanning trees. Which of the following is NOT true for G?

(a) For every subset of k vertices, the induced subgraph has a most $2k-2$ edges.
(b) The minimum cut in G has a least two edges
(c) There are two edges-disjoint paths between every pair of vertices
(d) There are two vertex-disjoint paths between every pair of vertices.

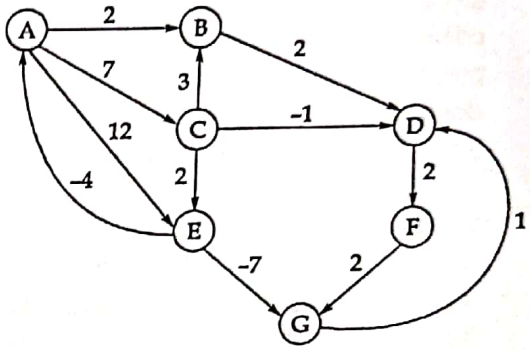
Q.61 An inter school basketball tournament is being held at Olympic sports complex. There are multiple basketball courts. Matches are scheduled in parallel, with staggered timing, to ensure that spectators always have some match or other available to watch. Each match requires a team of referees and linesman. Two matches that overlap require disjoint teams of referees and linesman. The tournament organizers would like to determine how many teams of referees and linesman they need to mobilize to effectively conduct the tournament. To determine this, which graph theoretic problem do the organizers have to solve?

(a) find minimal colouring
(b) find a minimal cut
(c) find a vertex cover
(d) find a minimal spanning tree

Q.62 Choose the invalid statement from the following?

(a) A graph can have more than one shortest path between two vertices.
(b) A graph where all edge weights are distinct can have more than one shortest path between two vertices.
(c) Multiplying all edge weights by a positive number might change the graph's minimum spanning tree.
(d) Both (b) and (c)

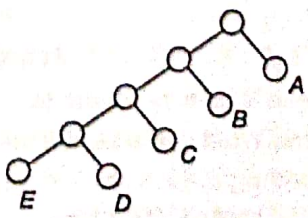
Q.63 While applying Dijkstra algorithm on given graph, it is already known that it is computing wrong path to some of the vertices. The number of wrong path computed are _____.



Q.64 Which of the following sets of code cannot be Huffman code?

(a) 0, 10, 1100, 1101, 1110, 1111
(b) 1, 01, 001, 0001, 00001, 00000
(c) 1, 01, 0010, 0011, 0001, 0000
(d) 1, 01, 0100, 0011, 0001, 0000

Q.65 Consider the following decode tree (Called Huffman codes) corresponding to codes for message *A*, *B*, *C*, *D*, *E*. The left and right branch of the tree is interpreted as 1 and 0 receptively. The distance of the external node is counted as bit sequence in decimal from the root node. Then, what will be expected decode time if message transmitted is *DBACABEED*?



(a) 9 units (b) 31 units
(c) 45 units (d) 68 units

Q.66 Consider the following statements with respect to a directed graph G in which edge can have positive or negative edge length but that has no negative cycles:

S_1: The Bellman-ford algorithm correctly computes shortest path lengths from a given origin 's' to every other vertex 'v'.

S_2: The Floyd-Warshall algorithm correctly computes shortest path lengths between every pair of vertices.

(a) S_1 True S_2 True (b) S_1 True S_2 False
(c) S_1 False S_2 True (d) S_1 False S_2 False

Q.67 US Portal denominations are 1, 10, 21, 34, 70, 100 and 350. Device a method to pay the following amounts to customer using fewest number of coins using greedy strategy. At each iteration, greedy always add coin of the largest value that does not take us past the amount to be paid. (Note: Out goal is to pay the amount using fewest number of coins).

Case A: An amount of 140 cents.

Case B: An amount of 182 cents.

(a) Greedy strategy works in I but fails in II.
(b) Greedy strategy works in II but fails in I.
(c) Greedy strategy works in both I and II.
(d) Greedy strategy does not work in both I and II.

Q.68 A text is made up of the characters *a*, *b*, *c*, *d*, *e* each occurring with the probability 0.12, 0.4, 0.15, 0.08 and 0.25 respectively. The optimal coding technique will have the average length of

(a) 2.15 (b) 3.01
(c) 2.3 (d) 1.78

Q.69 In the previous questions, which of the following characters will have codes of length 3?

(a) Only *c* (b) Only *b*
(c) *b* and *c* (d) Only *d*

Q.70 The characters *a* to *h* have the set of frequencies based on the first 8 Fibonacci numbers as follows:

a : 1, *b* : 1, *c* : 2, *d* : 3, *e* : 5, *f* : 8, *g* : 13, *h* : 21

A Huffman code is used to represent the characters. What is the sequence of characters corresponding to the following code?

110111100111010

(a) *fdheg* (b) *ecgdf*
(c) *dchfg* (c) *fehdg*

Q.71 Given a set $A = \{A_1, A_2, ..., A_n\}$ of n activities with start and finish time (Si, fi), $1 \le i \le n$, select maximal set S of "non-overlapping" activities. If the given problem is solved using greedy algorithm, then what will be the time complexity?

(a) $O(n^3)$ (b) $O(n^2)$
(c) $O(n \log n)$ (d) $O(n^n)$

Q.72 6 files *F1*, *F2*, *F3*, *F4*, *F5* and *F6* have 100, 200, 50, 80, 120, 150 records respectively. In what order should they be stored so as to optimize act. Assume each file is accessed with the same frequency.

(a) *F3*, *F4*, *F1*, *F5*, *F6*, *F2*
(b) *F2*, *F6*, *F5*, *F1*, *F4*, *F3*
(c) *F1*, *F2*, *F3*, *F4*, *F5*, *F6*
(d) Ordering is immaterial as all files are accessed with the same frequency.

Q.73 The following are the starting and ending times of activities A, B, C, D, E, F, G, and H respectively in chronological order:

"$a_s\ b_s\ c_s\ a_e\ d_s\ c_e\ e_s\ f_s\ b_e\ d_e\ g_s\ e_e\ f_e\ h_s\ g_e\ h_e$"

Here, x_s denotes the starting time and x_e denotes the ending time of activity X. W need to schedule the activities in a set of rooms available to us. An activity can be scheduled in a room only if the room is reserved for the activity for its entire duration. What is the minimum number of rooms required?

(a) 3 (b) 4
(c) 5 (d) 6

Q.74 Match the following algorithms in **List-I** to optimal strategy used from **List-II**.

List-I

A. Dijkstra's Algorithm
B. Prim's Algorithm
C. Floyd Warshall
D. Huffman coding
E. Bellman-Ford
F. 0/1 Knapsack

List-II

1. Greedy strategy
2. Dynamic strategy

(a) A-2, B-1, C-2, D-1, E-1, F-2
(b) A-1, B-1, C-2, D-1, E-2, F-2
(c) A-2, B-1, C-1, D-2, E-1, F-1
(d) A-1, B-1, C-1, D-1, E-2, F-2

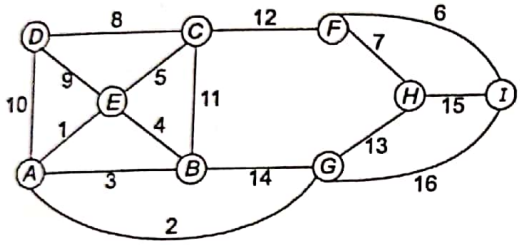
Q.75 Consider the following statements:

1. Dijkstra's algo will terminate even if there is a –ve edge or –ve cycle.
2. At the termination of Bellman Ford, even if graph has –ve cycle, a correct shortest path is found for a vertex for which shortest path is well-defined.

Choose correct option:

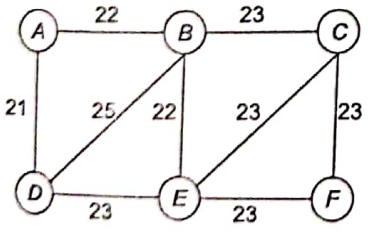
(a) 1-True; 2-True (b) 1-True; 2-False
(c) 1-False; 2-True (d) 1-False; 2-False

Q.76 Consider the weighted undirected graph below:



Assume prim's algorithm and Kruskal's algorithm are executed on the above graph to find the minimum spanning tree. For a particular edge (e_i) which is included in minimum spanning tree and the position of an edge in minimum spanning tree is denoted by e_{p_i}. Where $1 \le e_{p_i} \le 8$ (where position defines the order in which edges are included in the MST). Then what is the maximum value of $\left|(e_{p_i})_{\text{Prim's}} - (e_{p_i})_{\text{Kruskal's}}\right|$?

Q.77 Consider the following graph G:



Find the number of minimum cost spanning trees using Kruskal's algorithm or prim's algorithm.

(a) 3 (b) 5
(c) 7 (d) 4

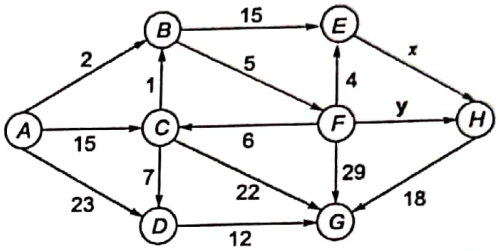
Q.78 A source sends a message using 4 symbols (A, B, C, D) with frequencies 1000, 500, 250, 250 respectively. The symbols are encoded by four different encoding schemes and the corresponding codes are shown below:

Scheme	A	B	C	D
1	00	01	10	11
2	0	10	110	111
3	00	100	1100	1101
4	111	110	10	0

Which of the above schemes gives the best coding efficiency?

(a) Scheme 1 (b) Scheme 2
(c) Scheme 3 (d) Scheme 4

Q.79 Suppose that you are running Dijkstra's algorithm on the edge-weighted digraph below, starting from vertex *A*.



The table gives 'Distance' and 'Parent' entry of each vertex after vertex *E* has been deleted from the priority queue and relaxed.

Vertex	Distance	Parent
A	0	NULL
B	2	*A*
C	13	*F*
D	23	*A*
E	11	*F*
F	7	*B*
G	36	*F*
H	19	*E*

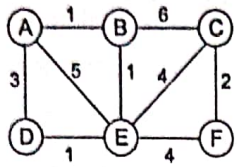
What could be the possible values of x and y?

(a) $x = 11, y = 7$ (b) $x = 8, y = 12$
(c) $x = 6, y = 10$ (d) $x = 10, y = 14$

Q.80 Which of the following statements is true?

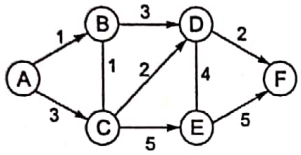
(a) Adding a constant to every edge weight in a directed graph can change the set of edges that belongs to shortest path tree. Assume unique weights.
(b) Adding a constant to every edge weight in an undirected graph can change the set of edges that belongs to the minimum spanning tree.
(c) Rerunning Dijkstra's algorithm on a graph V times will result in the correct shortest paths tree, even if there are negative edges (but no negative cycles).
(d) None of these

Q.81 Consider the weighted, undirected, connected graph below. What is the minimum cost spanning tree and maximum cost spanning tree respectively?



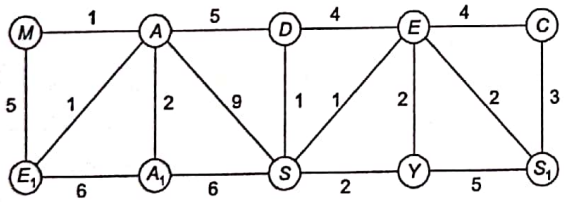
(a) 9, 22 (b) 10, 22
(c) 9, 20 (d) 10, 20

Q.82 Suppose we run Dijkstra's single source shortest path algorithm on the following edge weighted directed graph with vertex *A* as the source. What is the order of the vertices in which the shortest path is found by the algorithm?



(a) *A, B, C, D, F, E* (b) *A, B, C, D, E, F*
(c) *A, B, D, C, E, F* (d) *A, B, C, E, D, F*

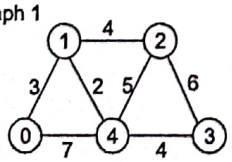
Q.83 Assume that *G* be a graph with (*v*, *e*) where *v* is the set of vertices and *e* is the set of edges in *G*. Assume that E_{sp} be the cost of edges of shortest path from *M* to *S*, which is computed by Dijkstra's algorithm. The graph *G* is given below.



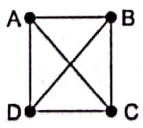
Let E_{max} is the cost of all edges which show shortest path from *M* to every other vertices. What is the value of $E_{max} - E_{sp}$?

Q.84 Let *A* be the cost of minimum cost spanning tree of graph 1 and *B* be the number of spanning trees possible for Graph 2, find the value of $4A + 3B$?

Graph 1



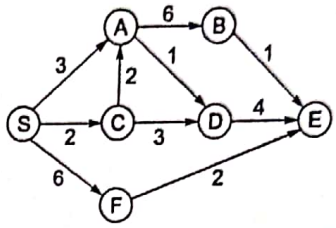
Graph 2



Q.85 Achint is travelling from Point *A* to Point *B* and there are *n* toll posts along the way. Before starting the journey, Achint is given for each post $1 \le i < j \le n$, the fee $f_{ij}, j \in N$ to travel from post *i* to post *j*. The goal is to minimize the toll paid. The most efficient algorithm to solve this problem has the time complexity as:

(a) $\theta(n)$ (b) $\theta(n \log n)$
(c) $\theta(n^2)$ (d) $\theta(n^3)$

Q.86 Run Dijkstra's algorithm on the following directed graph, starting at vertex S. What is order in which vertices get removed from the priority queue?



(a) *S, C, A, D, F, E, B* (b) *S, C, D, A, F, E, B*
(c) *S, C, A, D, E, F, B* (d) *S, C, A, D, E, B, F*

Q.87 Consider following statements about Prim's algorithm to compute an MST (minimum spanning tree) for a given graph.

S_1: An edge is added to the tree, at every step which belongs to a cut, connecting to some node in the graph.

S_2: An edge with the maximum weight in the graph is always avoided in order to get a minimum spanning tree of the graph.

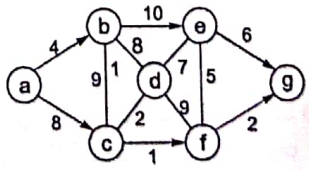
S_3: At any point of time while the construction of the trees is in progress, edges of the tree always from one connected component.

S_4: An edge is added to the tree, if its weight is the minimum of any edge crossing the cut, connecting it to some node of the graph.

Which of the following is true?

(a) S_1 and S_2 (b) S_1, S_2 and S_3
(c) S_1, S_3 and S_4 (d) S_1, S_2, S_3 and S_4

Q.88 Which of the following edge is added at last to the minimum spanning tree using Prim's algorithm?



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

Answers Greedy Techniques

1.	(c)	2.	(b)	3.	(d)	4.	(a)	5.	(c)	6.	(c)	7.	(c)	8.	(125)	9.	(b)
10.	(a)	11.	(16807)	12.	(25)	13.	(d)	14.	(1)	15.	(d)	16.	(b)	17.	(b)	18.	(d)
19.	(14)	20.	(b)	21.	(251)	22.	(3)	23.	(a)	24.	(28)	25.	(223)	26.	(b)	27.	(d)
28.	(b)	29.	(2.22 to 2.34)			30.	(a)	31.	(b)	32.	(c)	33.	(b)	34.	(a)	35.	(a)
36.	(240)	37.	(40)	38.	(b)	39.	(c)	40.	(c)	41.	(1.9375)	42.	(b)	43.	(c)	44.	(d)
45.	(46)	46.	(d)	47.	(a)	48.	(b)	49.	(45)	50.	(b)	51.	(b)	52.	(c)	53.	(b)
54.	(a)	55.	(b)	56.	(c)	57.	(c)	58.	(b)	59.	(125)	60.	(d)	61.	(a)	62.	(c)
63.	(3)	64.	(d)	65.	(d)	66.	(a)	67.	(d)	68.	(a)	69.	(a)	70.	(a)	71.	(c)
72.	(a)	73.	(b)	74.	(b)	75.	(a)	76.	(2)	77.	(b)	78.	(b)	79.	(b)	80.	(a)
81.	(a)	82.	(a)	83.	(51)	84.	(100)	85.	(c)	86.	(a)	87.	(c)	88.	(a)		

Explanations Greedy Techniques

1. (c)
Calculation of routing tables, as Dijkstra algorithm calculates shortest path for all. The node in link state routing protocol.

2. (b)
In worst case scenario, i.e., in a complete graph,

no of edges becomes $\frac{V(V-1)}{2}$ i.e., V^2.

Now, even with adjacency list implementation, which takes $O(V+E)$ as $E \sim V^2$, $O(V+E) \sim O(V^2)$.
So, option (a) is correct.

3. (d)
Lets consider the efficient one, it will be done either by using Dijkstra or Bellman, as we know that Dijkstra is better. So, now in Dijkstra's we keep on choosing the node or vertex with minimum weight, this operation can be mapped to extract minimum operation of min heap which takes $O(\log n)$ or $O(\log V)$ times and then we are relaxing the edges i.e., $O(E \log V)$.
So, it is best option we have.
So, answer is (d) option.

4. (a)
Both dynamic and Greedy algorithm find optimal substructure in the problem but only dynamic programming uses the bottom-up approach, whereas greedy algo uses top-down approach.
So, S_1 is true so, option (a) is correct.

5. (c)
$O(m)$ because the edge weights are sorted already, now you may have check m of the edges in the worst case as intermediate cycles could be formed when adding an edge.
So, option (c) is correct.

6. (c)
In coding a character we start at root and remember the path it took to reach a character in Huffman tree and assign Huffman code based on it.

7. (c)

Item	x_1	x_2	x_3	x_4	x_5
P_i	15	12	9	16	17
W_i	2	5	3	4	6
P_i/W_i	7.5	2.4	3	4	2.8

Decreasing order of $\frac{P_i}{W_i}$ is:

x_1, x_4, x_3, x_5, x_2

$x_1 \Rightarrow \quad P = 15, W = 2$

$\{x_1, x_4\} \Rightarrow \quad P = 15 + 16 = 31$

$W = 2 + 4 = 6$

$\{x_1, x_4, x_3\} \Rightarrow P = 31 + 9 = 40$

$W = 6 + 3 = 9$

$\{x_1, x_4, x_3, x_5/2\}$

$\Rightarrow \quad P = 40 + \frac{17}{2} = 48.5$

$W = 9 + \frac{6}{2} = 12$

$\therefore \quad P = 48.5$

8. (125)

For a complete graph k_n, number of spanning tree $= n^{n-2}$.

Here, $n = 5$

So, number of spanning tree $= 5^{5-2} = 5^3 = 125$

9. (b)

First part is easy we can solve recurrence relation

$T(n) = T(n-1) + T(n-2) + C = O(2^n)$

For second part we can make a program like this

```
int Fibonacci (int n)
{
    // Declare an array to store Fibonacci numbers
    int fib[n + 2];
    int i;
    // 0th and 1st number of the series are 0 and 1
    fib[0] = 0;
    fib[1] = 1;
    for(i = 2; i <= n; i++)
    {
    //Add the previous 2 numbers in the series and store it
    fib[i] = fib[i - 1] + fib[i - 2];
    }
    return fib[n];
}
```

Time complexity $= \theta(n)$

10. (a)

O((number of vertex) time for each EXTRACT-MIN operation) + (number of edges)(time for each DECREASE-KEY operation))

i.e., $O(V \cdot f(V, E) + E \cdot g(V, E))$

11. (16807)

Number of spanning trees for an undirected complete graph with nodes $n = n^{n-2}$.

12. (25)

$x \le 11, \quad y \le 6, \quad z \le 8$

13. (d)

In a connected graph, a vertex v is said be an articulation point if by removing that vertex together with its edges the graph become disconnected. In the given graph there is no articulation point.

14. (1)

Using Greedy : 20 + 5 + 5 + 1 + 1 + 1

Total number of coins = 6

Dynamically: 15 + 15 + 1 + 1 + 1

Difference = 6 – 5 = 1

15. (d)

Kruskal's algorithm time complexity

$= O(e \log v)$

$= O(m \log n)$

16. (b)

Number of spanning tree possible with n-node $= n^{n-2}$.

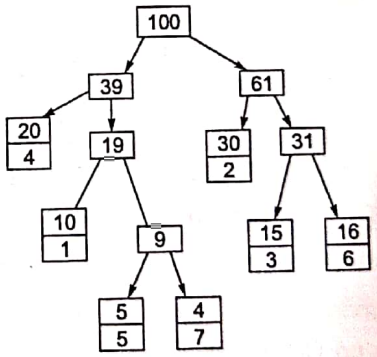
Given, $n = 7$

Total number of spanning trees $= 7^5$

17. (b)

Files 1.....7 will be in the levels.

$L = \{3, 2, 3, 2, 4, 3, 4\}$ Multiplying the level for a file by its length and summing over all files, we get a minimum of 259 total record moves to merge all the files into one file.

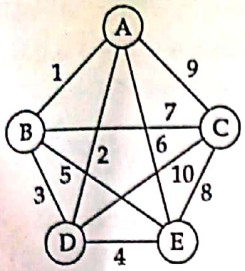


So, answer is 259 option (b) is correct.

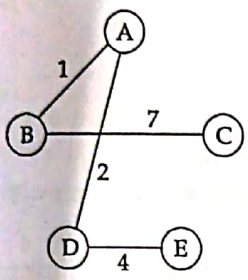
18. (d)

w be the minimum weight among all edge weights in an undirected connected graph. e is the specific edge of weight w. It may be possible that another edge in the graph having weight w which had been added to minimum spanning tree and when we add e to minimum spanning tree it form a simple circuit. So we can't include e in every minimum spanning tree.

19. (14)
Consider edges weights are numbered as below:

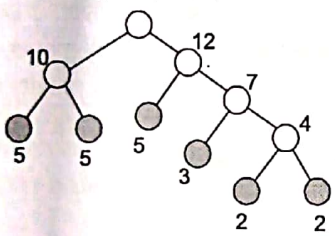


So minimum cost spanning tree will be:



In worst case, cost of MST will be 1 + 2 + 4 + 7 = 14.

20. (b)
Frequencies: 2, 2, 3, 5, 5, 5

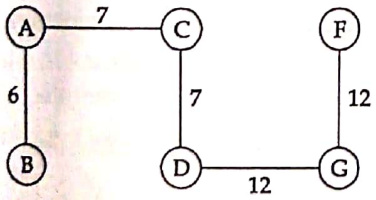


21. (251)

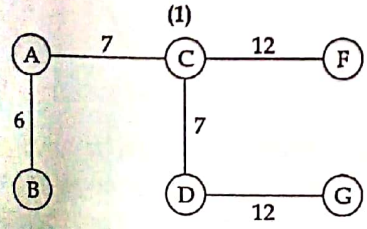
Timeline	1	2	3	4	5
Job done	E	D	F	A	H
Profit	45	50	46	100	10

The maximum total profit
= 45 + 50 + 46 + 100 + 10 = 251

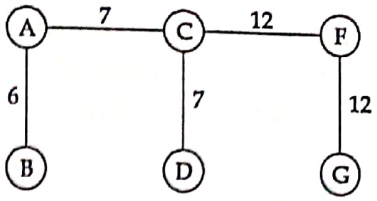
22. (3)



(1)

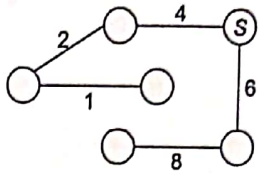


(2)



(3)

23. (a)



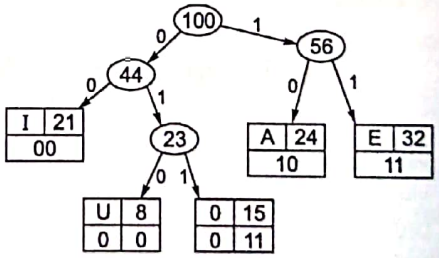
Starting vertex is *S*:
Selecting of edges in MST : 4, 2, 1, 6, 8

24. (28)

Queue	B	C	D	E	F	G	H	
{A}	4	5	∞	∞	∞	∞	∞	Select B
{A, B}	–	5	15	∞	∞	∞	∞	Select C
{A, B, C}	–	–	15	15	11	∞	∞	Select F
{A, B, C, F}	–	–	15	15	–	28	12	Select H
{A, B ,C, F, H}	–	–	15	15	–	24	–	Select D
{A, B, C, F, H, D}	–	–	–	15	–	24	–	Select E
{A, B, C, F, H, D, E}	–	–	–	–	–	24	–	Select G

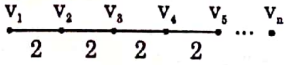
From above table, we found that the max distance to reach *G* is possibly by applying single shortest path algorithm costs 28.
So, answer is 28.

25. (223)



For transmitting 100 vowels, the number of bits required are:
2 * 24 + 2 * 32 + 2 * 21 + 3 * 8 + 3 * 15 = 223

26. (b)
Given vertex set of G = $\{v_1, v_2, \ldots, v_n\}$
Weight of an edge = $2|i - j| : (v_i, v_j) \in G$



is MST.

In the case of minimum spanning tree of a graph G we will add upto $(n-1)$ vertices, so the weight of a minimum spanning tree of G

$$= \sum_{i=1}^{n-1} 2|V_i - V_{i-1}|$$

$$= 2\sum_{i=1}^{n-1}|V_i - V_{i-1}| = 2\sum_{i=1}^{n-1}|1|$$

$$= 2|n-1| = 2(n-1) = 2n-2$$

27. (d)

M to A_1: $M \to A_1$
M to E_2: $M \to A_1 \to A_2 \to E_2$
M to A_2: $M \to A_1 \to A_2$
M to D: $M \to A_1 \to A_2 \to S \to D$
M to S: $M \to A_1 \to A_2 \to S$
M to E_1: $M \to A_1 \to A_2 \to S \to D \to E_1$
M to Y: $M \to A_1 \to A_2 \to S \to D \to E_1 \to Y$
(S, Y) Edge not used in any shortest path.

28. (b)

Time complexity: $O(E + V\log E)$ using Fibonacci heap.

$$|V| = n$$

$$|E| = \frac{n(n+1)}{2} = \frac{n^2+n}{2}$$

$$T(n) = O\left(\frac{n^2+n}{2} + n\log n\right)$$

$$= O(n^2)$$

OR

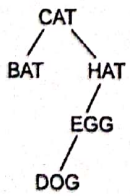
Time complexity : $O(E + V^2)$ using array.

Therefore,

$$T(n) = O\left(n^2 + \frac{n^2-n}{2}\right) = O(n^2)$$

29. (2.22 to 2.34)

Using a greedy approach, we insert the higher probability word first, taking into consideration the key alphabetical order:



The total cost = 1 * 22 + 2 * (10 + 22) + 3 * 20 + 4 * 18 = 234

i.e., it is 2.34 per search.

30. (a)

$$S \xrightarrow{3} b \xrightarrow{1} f \xrightarrow{3} h \xrightarrow{1} D$$

Minimum distance = 3 + 1 + 3 + 1 = 8

31. (b)

Dijkstra's implementation for single source shortest path

(i) Using binary heap takes $O((|E|+|V|)\log|V|)$

(ii) Using Fibonacci heap takes $O(|E|+|V|\log|V|)$

32. (c)

Heap and priority queue are very neat data structures allowing:

- Add an element to heap with an associated priority.
- Remove the element from the heap or priority queue that has the highest priority, and return it.
- Peak at the element with highest priority without removing it.

A simple way to implement a heap or priority queue data type is to keep a list of elements, and search through the list for the highest priority which gives $O(n)$ time to implement Dijkstra's shortest path algorithm on unweighted graph.

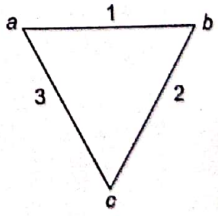
33. (b)

Bellman-Ford shortest path algorithm always finds any negative weighted cycle which is reachable from the source. If the negative weighted cycle is not reachable then, Bellman Ford algorithm not able to find that cycle.

34. (a)

Statement P: Since every edge weight is positive and we increase the value of every edge weight by same constant values. So minimum spanning tree of G does not change.

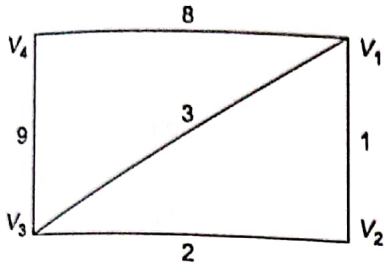
Statement Q: Taking an example:



First path from 'a' to 'c' via 'b' have path value 3. But here path can be change 'a' to 'e' direct since paths value is same but path can be change. So statement is wrong.

35. (a)

- Since all the edge weights are unique, hence the minimum spanning tree of the graph will be unique.
- Shortest path between the two vertices need not to be unique. A counter example for the statement can be,



The path from $V_1 \rightarrow V_3$ can be,

I. $V_1 \rightarrow V_2 \rightarrow V_3 : 1 + 2 = 3$

II. $V_1 \rightarrow V_3 : 3$

Hence the path is not unique.

36. (240)

Objects	O_1	O_2	O_3
Weights	10	20	30
Profits	60	100	120
Profit/Weight	6	5	4

Sort out objects in decreasing order of Profit/weight ratio which is : O_1, O_2, O_3.

This is the order, items are being put in the bag.

30 profit O_3 O_2 O_1 } 60

$W = \cancel{50}\ \cancel{40}^{01}_{(-10)}\ 20^{02}_{(-20)}$

30 $wt \rightarrow$ 120 profit

$$20\ wt \rightarrow \frac{120 \times 20}{30} = 80 \text{ profit}$$

Total profits = 60 + 100 + 80 = 240

37. (40)

In case of fractional Knapsack we apply greedy technique. Calculate $\frac{P_i}{W_i}$ for each item.

$$\frac{P_1}{W_1} = \frac{7}{3} = 2.3$$

$$\frac{P_2}{W_2} = \frac{10}{5} = 2$$

$$\frac{P_3}{W_3} = \frac{3}{2} = 1.5$$

$$\frac{P_4}{W_4} = \frac{3}{1} = 3$$

$$\frac{P_5}{W_5} = \frac{26}{12} = 2.1$$

$$\frac{P_6}{W_6} = \frac{19}{10} = 1.9$$

Now, we select the object by giving priority to the one having larger profit by weight ratio.

P_4 is selected first

Profit = 3

Weight remaining = 18 – 1 = 17

Then, P_1 is selected,

Profit = 3 + 7 = 10

Weight remaining = 17 – 3 = 14

Then, P_5 is selected.

Profit = 10 + 26 = 36

Weight remaining = 14 – 12 = 2

Then P_2 is selected.

But, remaining weight = 2 and its weight = 5,

So, we consider the profit of $\left(\frac{2}{5}\right)^{th}$ part only.

So, Profit = $36 + 10 \times \frac{2}{5} = 36 + 4 = 40$

Hence, the maximum profit achievable using fractional Knapsack = 40.

38. (b)

I. If the graph contains a negative weight cycle then no shortest path exist.

II. If shortest path is well defined then it contains at most V-1 edges. Running V-1 iterations of Bellman ford will find the path.

39. (c)

Dijkstra's algorithm always terminates after $|E|$ relaxations and $|V| + |E|$ priority queue operations, but may produce incorrect results.

So, option (c) is correct.

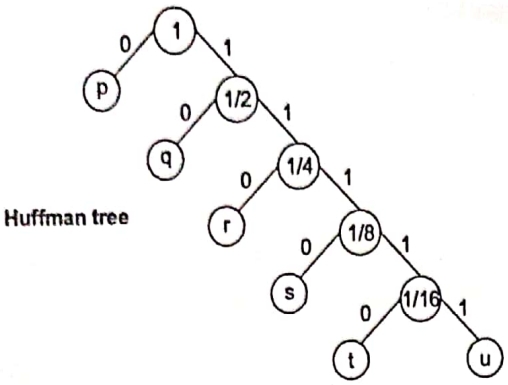
40. (c)

For minimum spanning tree $(n - 1)$ edges are enough and graph is complete graph. So each adjacent vertex is connected by an edge of weight 4.

So, cost of minimum spanning tree using Prim algo = $(n - 1) * 4 = 4n - 4$

So, option (c) is correct.

41. (1.9375)



Huffman tree

p : 0
q : 10
r : 110
s : 1110
t : 11110
u = 11111

Average length

$$= \frac{1}{2}\times 1+\frac{1}{4}\times 2+\frac{1}{8}\times 3+\frac{1}{16}\times 4+\frac{1\times 5}{32}+\frac{1}{32}\times 5$$

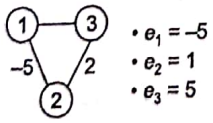
$$\Rightarrow \frac{1}{2}+\frac{1}{2}+\frac{3}{8}+\frac{1}{4}+\frac{10}{32}$$

$$\Rightarrow \frac{16+16+12+8+10}{32} \Rightarrow \frac{62}{32} \Rightarrow 1.9375$$

42. (b)
It is never lossy and it is always optimal. The codes assigned to input characters are prefix code i.e. the codes are assigned in such a way that code assigned to one character is not prefix of code assigned to any other character.

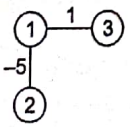
43. (c)
Number of spanning trees of a complete graph k_n is given by Cayley's theorem = n^{n-2}.

44. (d)
The catch here is edge-weight belongs to real number. Therefore, edge weight can be negative. In that case the minimum spanning tree may be different.

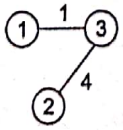


- $e_1 = -5$
- $e_2 = 1$
- $e_3 = 5$

T is:



New edge weights:
- $e_1 = 25$
- $e_2 = 1$
- $e_4 = 4$



Here, every edge weight is distinct, therefore MST is unique.
Option (a): is true. If apply Kruskal's algo it will choose e_1.
Option (b): is also true. If we apply Prim's algorithm also on any vertex (let U), it chooses minimum weight edge incident on vertex U.
Option (c): is true. Because every edge weight is distinct.
Option (d): is false because weights belongs to real numbers. Therefore edge weights can be negative.
So, option (d) is correct.

45. (46)

Select all of items a, d, e, j and select $\frac{1}{3}$ of item g.

Total weight = $3 + 1 + 12 + 1 + \frac{1}{3}\times 9 = 20$

Total profit = $7 + 3 + 26 + 4 + \frac{1}{3}\times 18 = 46$

46. (d)
The given problem is job scheduling problem 9 tasks are $T_1, T_2, \ldots, T_9$.
J is initially empty then according to deadlines it includes $\{T_1, T_2, T_3, T_5, T_7, T_8, T_9\}$.
So T_4 and T_6 can't be included in J.

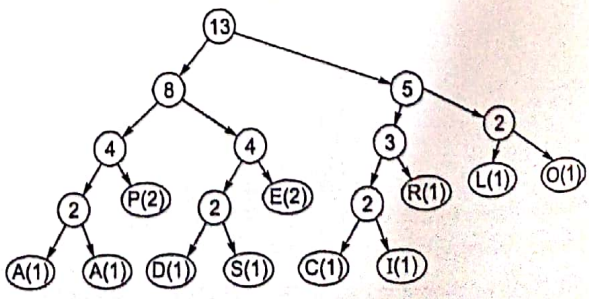
47. (a)
Total profit earn by algorithm
= 15 + 20 + 30 + 18 + 23 + 16 + 25 = 147

48. (b)

Timeline	1	2	3	4	5
Task	T_6	T_4	T_2	T_5	T_3
Profit	13	9	11	12	6

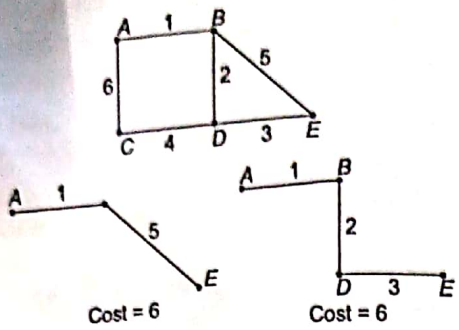
So, option (b) is correct.

49. (45)



50. (b)

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



(*ii*) Multiplying all edge weight by a positive number should not effect cost of minimum spanning tree.

So option (b) is invalid.

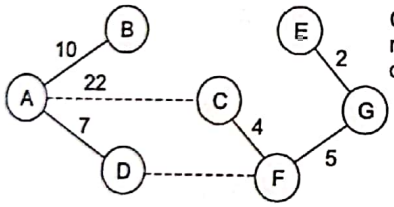
51. (b)

All edges are distinct weights.

Removal of e_{max} may not disconnect G, because other edges may cover the vertices incident with e_{max}.

52. (c)

Prim's algorithm having special property which is, while finding minimum cost spanning tree, graph always be connected.



Choice of edges represented by dashed line

Minimum cost spanning tree using Prim's

Now for correct options, we need to check while entry for any edge except for 1st edge only one vertex match from all left side (i.e. visited edges) edges.

(a) (E, G), (C, F) : here for edge (C, F) no vertex match to (E, G). So not possible using Prim's.

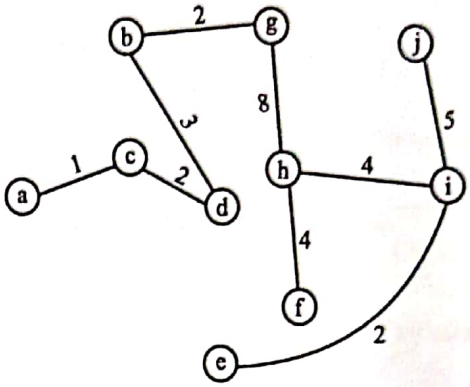
(b) (A, D), (A, B), (A, C), (C, F), (G, E) : here for edge (G, E) no vertex match to any edge present left side of it so not possible using Prim's.

(c) (A, B), (A, D), (D, F), (F, G), (G, E), (F, C) not violated rule i.e. always connected in construction of MST.

So True.

53. (b)

Number of vertices in the graph $n = 10$ so after adding $n-1 = 9$ edges which contains all vertices and doesn't form a circuit in the minimum spanning tree of G.



$$\text{Total weight} = \sum_{i=1}^{9} w(i)$$

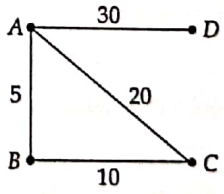
$$= 1 + 2 + 2 + 2 + 3 + 4 + 4 + 5 + 8 = 31$$

54. (a)

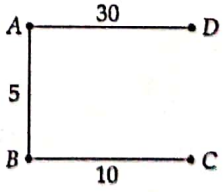
Since edges costs are distinct, so cheapest edge must be present in every minimum spanning tree while expensive edge is may not excluded from every minimum spanning tree.

Statement S_2 is false.

Example:



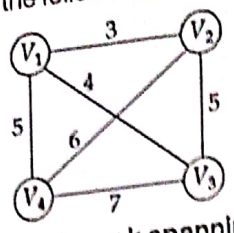
MST will be:



So, most expensive edge is not excluded.

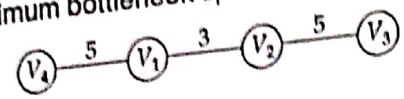
55. (b)

S_1: Consider the following graph:

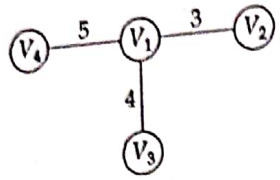


Minimum bottleneck spanning tree:

(A):



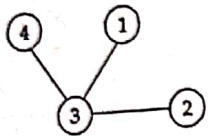
(B):



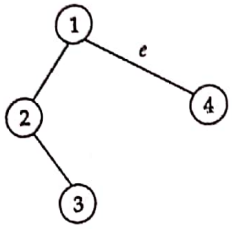
However only (b) is MST therefore S_1 is false.

S_2: Consider 2 MST's:

(A):



(B):

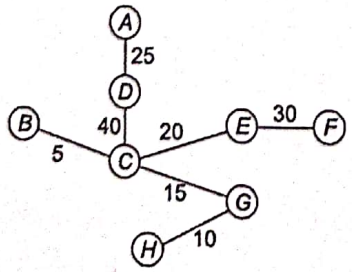


Where A has a lighter bottleneck edge. This means that B has an edge 'e' that is heavier than every edge of A. If we include this edge in A, it will form a cycle in which e would be heaviest. This contradicts the definition of MST. Thus, 'e' cant be present in B, meaning that both A and B have same bottleneck edge. Thus, S_2 is true.

56. (c)

There will be unique minimum cost for any minimum spanning tree in a graph irrespective of algorithm applied. i.e. more than one minimum cost spanning trees are possible but their cost will be same.

57. (c)



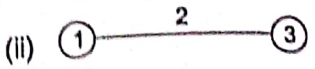
5 + 10 + 15 + 20 + 25 + 30 + 40 = 145

58. (b)

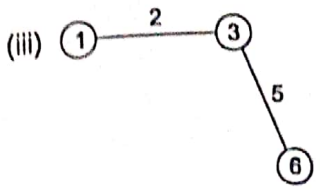
Using the Kruskal's algorithm

(i)

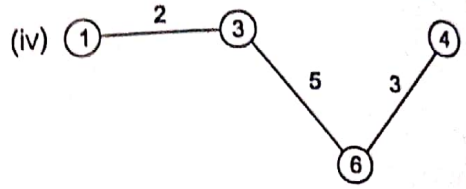
(ii)



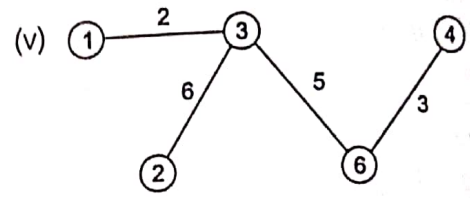
(iii)



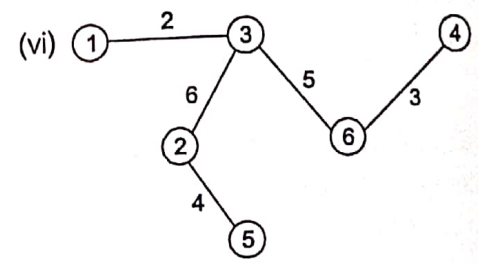
(iv)



(v)



(vi)



∴ Minimum cost is 20.

59. (125)

According to Cayley's formula for counting spanning trees, for a complete graph k_n.

$T(K_n) = n(\text{pow}(n-2)$ where n = number of vertices

$T(K_5) = 5^{5-2} = 5^3 = 125$

60. (d)

Given graph G contains n vertices and $2n-2 = 2(n-1)$ edges. So spanning tree must contains $2n-4$ edges. If G can be partitioned into two edge disjoint spanning tree. Then for any tree $2e = \sum_{i=1}^{n} d_i$ belongs to a tree if each d_i is positive and $e = n-1$ but in given problem $e = 2n - 4 = 2(n-2)$. So there is no two-vertex-disjoint paths between every pair of vertices.

61. (a)
Option (a) is correct.
Lets represent this situation in the form of a graph. Consider matches as the nodes in which the same edges represents that the different matches are overlapping. So, for this situation, we need different referees and linesman.
So, how would you solve this?
Definitely by taking care that the nodes which have the same edge do not have the referee or linesman (think of referee and lineman as the colour of nodes now).
So, basically we just have to solve the minimal colouring problem now, which will be the required answer to above problem.
So, option (a) is correct.

62. (c)
Multiplying all edge weights by a positive number does not change the graph's minimum spanning tree.
∴ Option (c) is not valid.

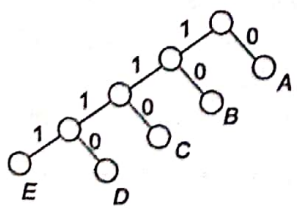
63. (3)

	Using Dijkstra algorithm		Actual Shortest path	
A → B	2	AB	2	AB
C	7	AC	7	AC
D	4	ABD	3	ACEGD
E	9	ACE	9	ACE
F	6	ABDF	5	ACEGDF
G	8	ABDFG	2	ACEG

So, there are 3 wrong path calculated to D, F and G vertices.

64. (d)
Huffman codes follow prefix code property (i.e., code of one element will not be prefix for another element code). Option (d) does not follow prefix codes property.
Hence, option (d) is correct.

65. (d)



$A \rightarrow 0$
$B \rightarrow 10 = 2$
$C \rightarrow 110 = 6$
$D \rightarrow 1110 = 14$
$E \rightarrow 1111 = 15$

Expected decode time = $\sum_{i=1}^{5} q_i d_i$

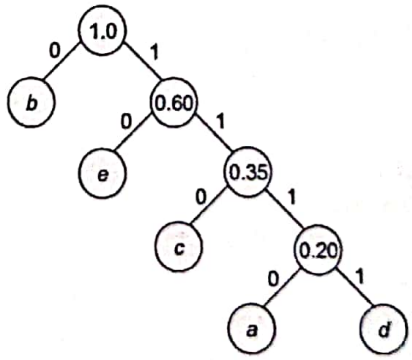
q_i = frequency of message m_i
d_i = distance from root to message m_i
The given sequence is $D\,B\,A\,C\,A\,B\,E\,E\,D$

$\sum_{i=1}^{5} q_i d_i = 2(0 + 2 + 14 + 15) + 6 = 68$ units

66. (a)
Both statements will be correct.
S_1: Bellman Ford works correctly if there are no negative weight cycles.
S_2: Floyd Warshall algorithm uses dynamic programming, therefore all the possibilities will be considered. So given that if shortest path exists (i.e., no negative weight cycles), then Floyd Warshall will surely find it.
Hence, option (a) is correct.

67. (d)
Greedy algorithm fails in both *A* and *B*.
Case A: 140 cents
Greedy: 100 + 34 + 1 + 1 + 1+ 1 + 1 + 1 = 8 coins
Optimal: 70 + 70 = 2 coins
Case B: 182 cents
Greedy: 100 + 70 + 10 + 1 + 1 = 5 coins
Optimal: 70 + 70 + 21 + 21 = 4 coins
So, option (d) is current option.

68. (a)



$a = 0.12$, $b = 0.40$, $c = 0.15$, $d = 0.08$ and $e = 0.25$.
So, number of bits for a = 4 bits
So, number of bits for b = 1 bits
So, number of bits for c = 3 bits

So, number of bits for $d = 4$ bits

So, number of bits for $e = 2$ bits

Average code length

$= 4 \times 0.12 + 1 \times 0.40 + 3 \times 0.15 + 4 \times 0.08 + 2 \times 0.25$

$= 2.15$ bits/character

69. (a)

Character 'c' has length of 3 bits.

70. (a)

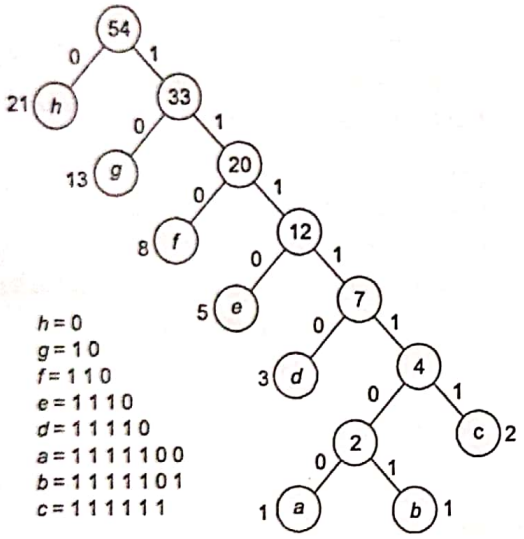
According to Huffman coding use Min heap data structure.

Step-1: Take two smallest probability.

Step-2: Add the probability taken in step-1.

Step-3: Put back previous sum in minimum heap.

Step-4: Repeat step-1 until probability becomes 1.



$h = 0$
$g = 10$
$f = 110$
$e = 1110$
$d = 11110$
$a = 1111100$
$b = 1111101$
$c = 111111$

So, 1101111001110 10

f d h e g

71. (c)

Sort A by finish time

$S = \{A_1\}$

$J = 1$

For $i = 2$ to n

Do

If $Si \geq Sj$

$S = S \cup \{Ai\}$

$J = i$

Then $\Rightarrow 0(n \log n)$

72. (a)

Since the access is sequential, greater the distance, greater will be the access time. Since all the files are referenced with equal frequency, overall access time can be reduced by arranging them as then ascending order of record as in option (a).

73. (b)

The activities are A, B, C, D, E, F, G and H. The starting and ending times are "$a_s b_s c_s a_e d_s c_e e_s f_s b_e d_e g_s e_s f_e h_s g_e h_e$.

The gap between starting and ending time of B is greater than all other activities. Consider the order $b_s c_s a_e d_s c_e e_s f_s b_e$ be the gap between b_s to b_e is 6 in which two activity a_e and c_e ended so minimum number of required rooms is 4.

74. (b)

Option (b) is correct.

75. (a)

1. is true: The Dijkstra's and Bellman Ford algorithms differs in how many times they relax each edge and the order in which they relax edges. Dijkstra's algorithm relaxes each edge exactly once. The Bellman-Ford algorithm relaxes each edge $|V-1|$ times. But both the algorithms will definitely terminate no matter what. Now, on Dijkstra's algo., it always terminates after $|E|$ relaxations and $|V|+|E|$ priority queue operations, but may produce incorrect results in case of negative weight edges.

2. is true: If shortest path is well defined then it cannot include a cycle. Thus, the shortest path contains at most $(V-1)$ edges. Running the usual $(V-1)$ iterations of Bellman-Ford will therefore find that path.

So, option (a) is correct.

76. (2)

Kruskal's algorithm:

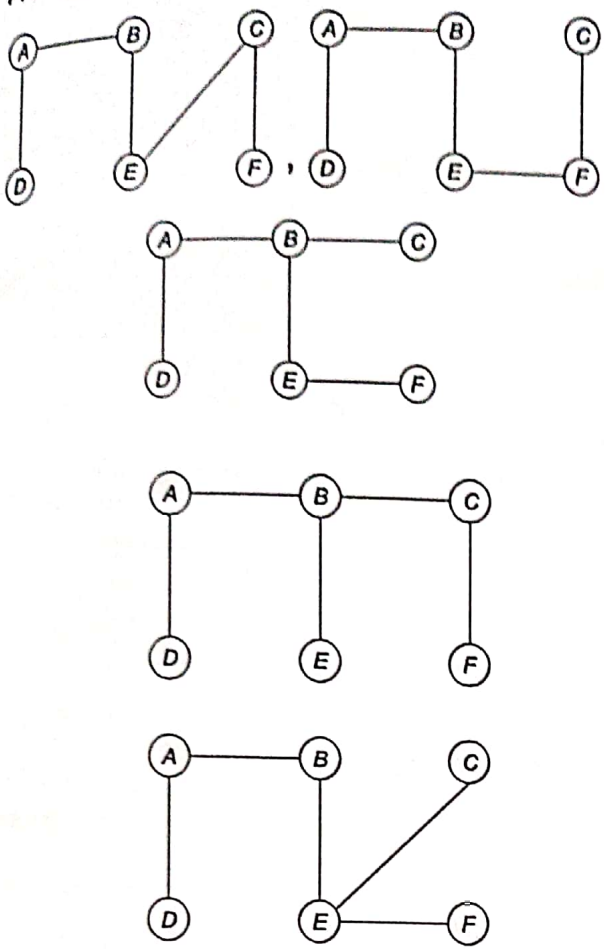
AE, AG, AB, CE, FI, FH, CD, CF

Prim's algorithm:

AE, AG, AB, CE, CD, CF, FI, FH

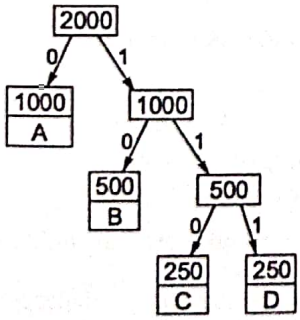
$\max\left|(e_{p_i})_{\text{Prim's}} - (e_{p_i})_{\text{Kruskal's}}\right| = |5-7| = 2$

77. (b)



All above trees MST with cost:
21 + 22 + 22 + 23 + 23 = 111

78. (b)
Scheme 1
$A - 0$
$B - 10$
$C - 110$
$D - 111$
Huffman tree is given below:



79. (b)

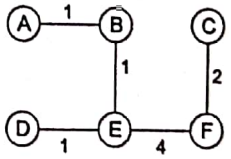
	A	B	C	D	E	F	G	H
	0 Nill	∞	∞	∞	∞	∞	∞	∞
A	–	2 A	15 A	23 A	∞	∞	∞	∞
B	–	–	15 A	23 A	17 B	7 B	∞	∞
F	–	–	13 F	23 A	11 F	–	36 F	y+7 F
Given → E	–	–	13 F	23 A	–	–	36 F	x+11 E

Compare above table with given table

$$x + 11 = 19$$
$$x = 8$$
$$y + 7 \geq x + 11$$
$$y + 7 \geq 8 + 11$$
$$\Rightarrow \quad y \geq 12$$

80. (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.

81. (a)
Cost of MST = 9
Cost of maximum spanning tree is 22.



So, option (a) is correct.

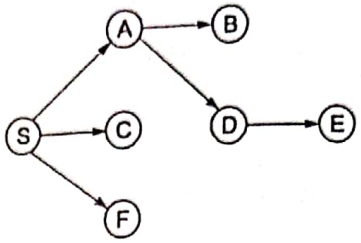
82. (a)
Dijkstra algorithm will visit the nodes in the following order: A, B, C, D, F, E.
So, option (a) is correct.

83. (51)
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 $E_{max} - E_{sp} = 58 - 7 = 51$

84. (100)
Cost of MST graph 1, $A = 13$
Number of spanning trees graph 2, $B = 16$
$4A + 3B = 4 * 13 + 3 * 16 = 100$

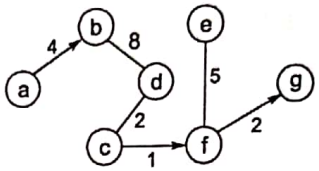
85. (c)
Here, the cost between any two tolls is represented by $f[i, j]$ and we need to find the optimal path from A to B provided there are n tolls (represented as nodes in between them). Our aim here is to optimize the toll charges so that it is minimal, this is the single source shortest path which can be achieved by applying Dijkstra's algorithm on the list of toll charges $f[i, j]$, hare we have 'n' nodes in total and f is an unsorted array. If an unsorted array is used then Dijkstra algorithm gives $O(|V|^2)$.
So, option (c) $\theta(n^2)$ is correct.

86. (a)
Dijkstra will visit the vertices in the following order: S, C, A, D, F, E, B.
Dijkstra will relax the edge from D to E before the edge from F to E, since D is closer to S than F is.
As a result, the parent of each node is:



87. (c)
Statements S_1, S_3 and S_4 are true.

88. (a)



So, option (a) is correct.

■■■■

CHAPTER 5

Graph Based Algorithms

Multiple Choice Questions & NAT Questions

Q.1 Number of undirected graphs (not necessarily connected) can be constructed given set $V = \{1, 2, 3, 4\}$ of 4 vertices, are _____.

Q.2 Let G be a graph with 'n' vertices and 'm' edges.

S_1: All its DFS forest (for traversals starting at different values) will have the same number of trees.

S_2: All its DFS forests will have the same number of tree edges and the number of back edges.

(a) S_1 - True; S_2 - True
(b) S_1 - True; S_2 - False
(c) S_1 - False; S_2 - True
(d) S_1 - False; S_2 - False

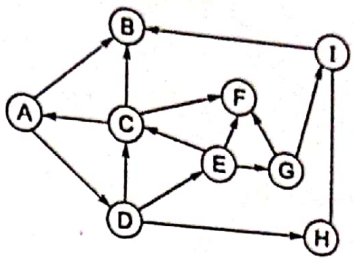
Q.3 Which one of the following statements is false?

(a) Optimal binary search tree construction can be performed efficiently using dynamic programming.
(b) Breadth-first search cannot be used to find connected components of a graph.
(c) Given the prefix and postfix walks over a binary tree, the binary tree cannot be uniquely constructed.
(d) Depth-first search can be used to find connected components of a graph.

Q.4 Which of the following problem cannot be computed by DFS?

(a) Finding connected components in directed graph.
(b) Finding shortest path in unweighted graph.
(c) Finding directed cycle in directed graph.
(d) Finding path from a source vertex to all other reachable vertices in directed graph.

Q.5 If DFS algorithm is applied to the given graph starting from vertex A the maximum height of the DFS tree for DFS traversal is _____?



Q.6 In an unweighted, undirected connected graph, the shortest path from a node S to every other node is computed most efficiently, in terms of time complexity, by

(a) Dijkstra's algorithm starting from S.
(b) Warshall's algorithm
(c) Performing a DFS starting from S
(d) Performing a BFS starting from S

Q.7 Consider two vertices 'a' and 'b' that are simultaneously on the FIFO queue at same point during the execution of breadth first search from 's' in an undirected graph. Consider the following statements:

S_1: The number of edges on the shortest path between 's' and 'a' is atmost one more than the number of edges on the shortest path between 's' and 'b'.

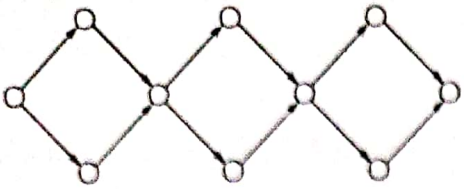
S_2: The number of edges on the shortest path between 's' and 'a' is atleast one less than the number of edges on the shortest path between 's' and 'b'.

S_3: There is a path between 'a' and 'b'.

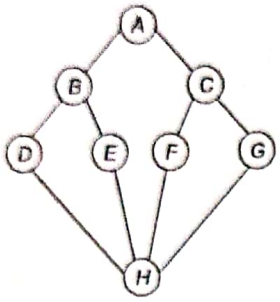
Which of the following is true?

(a) S_1 only (b) S_1 and S_2 only
(c) S_1 and S_3 only (d) S_1, S_2 and S_3

Q.8 Consider the directed Acyclic graph given below. How many topological ordering are possible for the given graph?



Q.9 Consider the following graph:



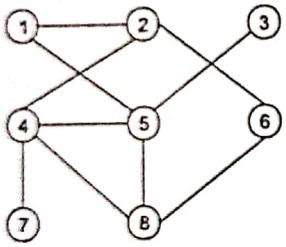
Which of the following is NOT a depth first search traversal of the above graph?

(a) ACFHEBDG (b) ACFHGEBD
(c) ABDHFCGE (d) ABDEHFCG

Q.10 Preorder is same as

(a) Depth-first order (b) Breadth-first order
(c) Topological order (d) Linear order

Q.11 Consider the following graph:



Which does not represent Depth First Search (DFS) sequence for the above graph?

(a) 12475386 (b) 12685437
(c) 12453867 (d) 12458637

Q.12 Which of the following statements are true?

S_1: Suppose we do a DFS on a directed graph G. If we remove all of the back edges found, the resulting graph is now acyclic.

S_2: Both BFS and DFS requires Big-omega (V) storage for their operation. (That is, for working storage, above and beyond the storage needed to represent the input).

S_3: A depth-first search of a directed graph always produces the same number of tree edges (i.e., independent of the order in which the vertices are provided and independent of the order the adjacency lists).

(a) S_1 and S_3 (b) S_1 only
(c) S_1 and S_2 (d) S_2 only

Q.13 Maximum number of BFS traversal possible on BST of height 3 is? (Root node is at height 0) _____?

Q.14 Which of the following are Incorrect?

S_1: While running DFS on a directed graph, if from vertex v we visit a finished vertex v, then the edge (u, v) is cross-edge.

S_2: Changing the RELAX function to update if $d[v] \geq d[u] + w(u, v)$ (instead of strictly greater then) may produce different shortest paths, but will not affect the correctness of the Bellman-ford outputs (distance d and parent π)

S_3: If a weighted directed graph G is known to behave no shortest paths longer than k edges, then it suffices to run Bellman-ford for only k passes in order to solve the single source shortest path problem on G.

S_4: If a topological sort exists for the vertices in a directed graph, then a DFS on the graph will produce no back edges.

(a) S_1 and S_4 (b) S_1 and S_2
(c) S_1, S_2 and S_3 (d) S_2 and S_3

Q.15 Consider the following statements:

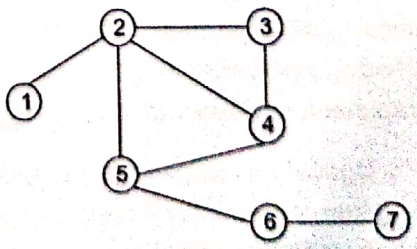
S_1: DFS of directed graph always produces the same number of edges in the traversal irrespective of starting vertex.

S_2: If we remove all of back edges found while DFS traversal on directed graph, the resultant graph is acyclic.

Find which of the following is true.

(a) Both S_1 and S_2 are valid
(b) Only S_1 valid
(c) Only S_2 Valid
(d) Neither S_1 nor S_2 is valid

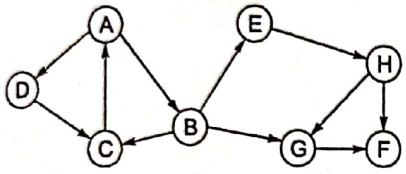
Q.16 Consider the following graph:



What is the depth first search (DFS) sequence for the above graph?

(a) 4, 5, 2, 1, 3, 6, 7
(b) 2, 3, 4, 5, 6, 1, 7
(c) 2, 1, 5, 3, 4, 6, 7
(d) 4, 2, 1, 5, 6, 3, 7

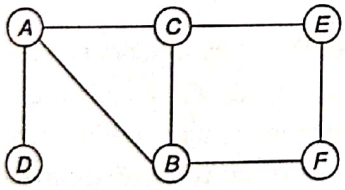
Q.17 Consider the following graph. If there is ever a decision between multiple neighbour nodes in the BFS or DFS algorithm assume we always choose the letter closest to the beginning of the alphabet first.



In what order will the nodes be visited using a breadth-first-search and depth-first-search respectively?

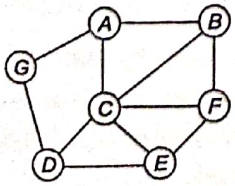
(a) BFS : ABDCEGHF; DFS : ABGFEHCG
(b) BFS : ABDECGHF; DFS : ABCEHFGD
(c) BFS : ABDCEGFH; DFS : ABCEHFGD
(d) BFS : ABDCEGHF; DFS : ABCEHFGD

Q.18 The depth first search algorithm has been implemented using the stack data structure. One possible order of visiting the node of the following graph is



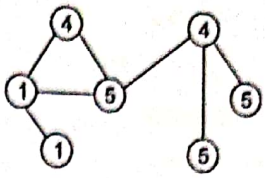
(a) *A D B C F E* (b) *A C E B F D*
(c) *B A C D F E* (d) *B C E F A D*

Q.19 Which of the following are the articulation points in the following graph?



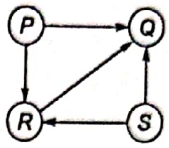
(a) Only *C* (b) *D, C, F*
(c) *B, C, F* (d) None of these

Q.20 The number of articulation points of the following graph is:



(a) 0 (b) 1
(c) 2 (d) 3

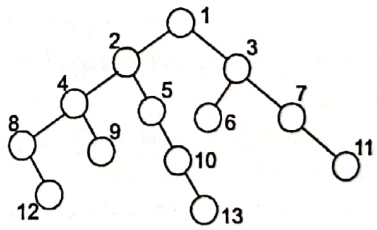
Q.21 Consider the directed graph given below:



Which one of the following is TRUE?

(a) The graph does not have any topological ordering.
(b) Both PQRS and SRQP are topological orderings.
(c) Both PSRQ and SPRQ are topological orderings.
(d) PSRQ is the only topological ordering.

Q.22 In the following graphs, assume that if there is ever a choice amongst multiple nodes, both the DFS and BFS algorithms will choose the left-most node first.



Starting from the root 1 at top, which algorithm will visit the least number of nodes before visiting the 6th node?

(a) BFS
(b) DFS
(c) Neither DFS or BFS will ever encounter the goal node in his graph.
(d) BFS and DFS encounter same number of nodes before encounter the goal node.

Q.23 Which of the following is the correct match between the algorithm and their time complexity?

Algorithm

P. Dijkstra (Priority queue and adjacency cost)
Q. Prim's algorithm (Binary heap)
R. BFS
S. Floyd Warshall

T. Kruskal Algorithm
U. Bellman Ford algorithm

Time Complexity

1. $O(V+E)$
2. $O(E \log V)$
3. $O(V^3)$
4. $O((E+V) \log V)$
5. $O(VE)$

(a) P-5, Q-2, R-3, S-4, T-5, U-3
(b) P-4, Q-2, R-1, S-3, T-2, U-5
(c) P-5, Q-2, R-2, S-3, T-2, U-5
(d) P-2, Q-4, R-3, S-3, T-5, U-5

Q.24 A mother vertex in a graph $G = (V, E)$ is a vertex V such that all other vertices in G can be reached by a path from V. What is time complexity of an efficient algorithm to find the mother vertex in a directed connected graph?

(a) $O(V(V+E))$ (b) $O(V^2)$
(c) $O(V \log V)$ (d) $O(V+E)$

Q.25 Which of the following statements are true?

S_1: For all weighted graph and all vertices s and t, Bellman-Ford starting at s will always return a shortest path to t.

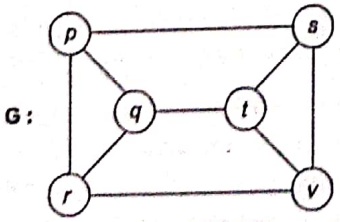
S_2: If all edges in a graph have distinct weights, then the shortest path between two vertices is unique.

S_3: In an unweighted graph where the distance between any two vertices is atmost T, any BFS tree has depth atmost T but a DFS tree might have larger depth.

S_4: There is no edge in an undirected graph that jumps more than one level of any BFS tree of the graph.

(a) S_1 and S_3 (b) S_3 only
(c) S_3 and S_4 (d) S_1, S_2 and S_4

Q.26 Consider the following graph G.



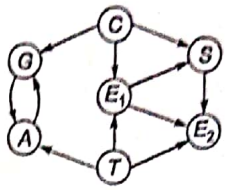
Modified DFS on G applied as follows:

- Starting vertex is 'p'.
- Vertex is visited based on alphabetic order.
- Vertices are visited in order p, q, r, s, t, v.
- It works same as DFS except the visiting order restriction

What is the number of back edges during the above DFS traversal on G?

(a) 2 (b) 3
(c) 4 (d) 5

Q.27 Find the number of strong components in the following graph. Strong component is the maximum subgraph which is strongly connected. Union of all strong components should contain all vertices of given graph.



(a) 1 (b) 2
(c) 3 (d) None of these

Q.28 Consider an undirected unweighted graph G. Let a breadth-first traversal of G be done starting from a node r. Let $d(r, u)$ and $d(r, v)$ be the lengths of the shortest paths from r to u and v respectively in G. If u is visited before v during the breadth-first traversal, which of the following statement is correct?

(a) $d(r, u) < d(r, v)$ (b) $d(r, u) > d(r, v)$
(c) $d(r, u) \leq d(r, v)$ (d) None of these

Q.29 Consider DFS over an undirected graph with 4 vertices $< P; Q; R; S >$. The discovery and finishing times of them in the order P and S is given for which of the following options is the graph connected?

(a) <(1, 6), (2, 5) (3, 4), (8, 10)>
(b) <(6, 7), (2, 5) (3, 4), (8, 9)>
(c) <(4, 5), (2, 8) (1, 7), (3, 6)>
(d) <(7, 8), (1, 2) (5, 6), (3, 4)>

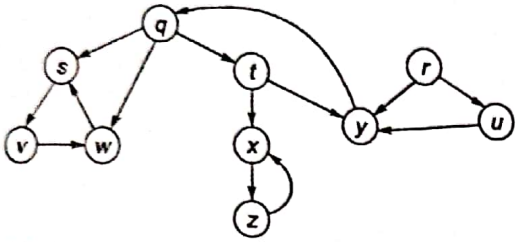
Q.30 Which of the following statements are true?

1. In a depth-first search of an undirected graph G, every edge of G is either a tree edge or a back edge.
2. Forward and cross edges never occur in a depth-first search of an undirected graph.
3. A directed graph is Acyclic if and only if a depth-first search yields no back edges.

(a) 1 only (b) 1 and 2
(c) 2 and 3 (d) All of these

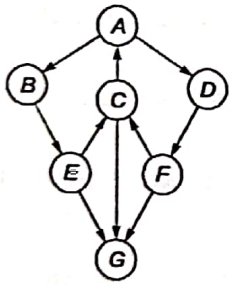
Q.31 Which of the following is application of depth-first search?
(a) Only topological sort.
(b) Only strongly connected components.
(c) Both topological sort and strongly connected components.
(d) Neither topological sort nor strongly connected components.

Q.32 What are the strongly connected components in the below figure?



(a) {s, v, w}, {x, z}, {q, t, y}, {r}, {v}
(b) {q, s, v}, {s, v, w}, {t, w, z}, {q, y}, {r, v}
(c) {r}, {u}, {s, v, w}, {t, w, z}, {r, v, y}
(d) {s, v, w}, {x, z}, {q, t, y}, {r, v}

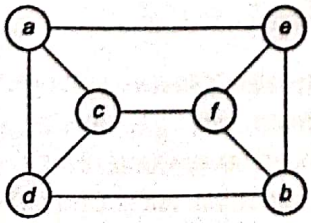
Q.33 Consider the following graph *G*.



Perform a DFS on *G* starting at vertex *A* and selection of adjacent vertex in DFS decided by the lexicographical order. In graph *G*, *B* and *D* are adjacent to *A*. First it selects B, because *B* comes first in lexicographical order. Find number of cross edges when the given DFS is performed on *G*?
(a) 1 (b) 2
(c) 3 (d) 4

Q.34 Consider the following graph G.

G:

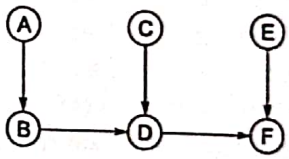


Modified BFS traversal on G applied as follows
- Assume starting vertex is '*a*'.
- At any level (breadth), vertices are visited in alphabetic order.
- In above graph G, after visiting vertex '*a*', the order of vertices visited in G are *c*, *d* and *e*.
- It works same as BFS except the restriction on order of vertices visited.

Which of the following is not a cross edge during given BFS traversal on G?
(a) {d, c} (b) {d, b}
(c) {e, b} (d) {e, f}

Q.35 Which of the following condition is sufficient to detect cycle in a directed graph?
(a) There is an edge from currently being visited node to an already visited node.
(b) There is an edge from currently being visited node to an ancestor of currently visited node in DFS forest.
(c) Every node is seen twice in DFS.
(d) None of the above

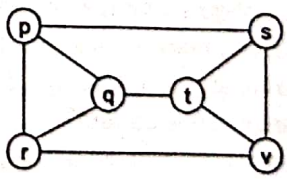
Q.36 Consider the following graph:



The number of topological orders for given graph are _____.

Q.37 Consider the following graph G.

G:



Modified DFS on *G* applied as follows:
- Starting vertex '*P*'.
- Vertex is visited based on alphabetic order.
- Vertices are visited in order *p*, *q*, *r*, *s*, *t*, *v*.
- It works same as DFS except the visiting order restriction.

What is number of backedges during the above DFS traversal on *G* _____?

Multiple Select Questions (MSQ)

Q.38 Consider the following statements:

Which of the following statement(s) is/are true?

(a) Depth – first search is used to traverse a rooted tree.

(b) Pre – order, Post-order and Inorder are used to list the vertices of an ordered rooted tree.

(c) Huffman's algorithm is used to find an optimal binary tree with given weights.

(d) Topological sorting provides a labelling such that the parents have larger labels than their children.

Which of the following statements(s) is/are correct?

Q.39 Consider a directed graph with distinct and nonnegative edge lengths and a source vertex s. Fix a destination vertex t, and assume that the graph contains at least one $s-t$ path. Which of the following statements is/are true?

(a) There is a shortest $s-t$ path with no repeated vertices (i.e. a "simple" or "loopless" such path).

(b) The shortest (i.e., minimum-length) $s-t$ path might have as many as $n-1$ edges, where n is the number of vertices.

(c) The shortest $s-t$ path must include the minimum-length edge of G.

(d) The shortest $s-t$ path must exclude the maximum-length edge of G.

Q.40 Consider a directed graph G = (V, E) and a source vertex s with the following properties: edges that leave the source vertex s have arbitrary (possibly negative) lengths; all other edge lengths are non-negative; and there are no edges from any other vertex to the source s. Then which of the following statement(s) is/are correct?

(a) Bellman Ford shortest-path algorithm correctly compute shortest-path distances (from s) in this graph?

(b) Bellman Ford shortest-path algorithm incorrectly compute shortest-path distances (from s) in this graph?

(c) Dijkstra's shortest-path algorithm correctly compute shortest-path distances (from s) in this graph?

(d) Dijkstra's shortest-path algorithm incorrectly compute shortest-path distances (from s) in this graph?

Q.41 Suppose you implement the functionality of a priority queue using a sorted array (e.g., from biggest to smallest). Then which of the following statement(s) is/are correct.

(a) The worst-case running time of Insert is $\Theta(n)$.

(b) The worst-case running time of Insert $\Theta(1)$.

(c) The worst-case running time of Extract-Min $\Theta(n)$.

(d) The worst-case running time of Extract-Min $\Theta(1)$.

Q.42 Suppose you implement the functionality of a priority queue using a unsorted array (e.g., from biggest to smallest). Then which of the following statement(s) is/are correct.

(a) The worst-case running time of Insert is $\Theta(n)$.

(b) The worst-case running time of Insert $\Theta(1)$.

(c) The worst-case running time of Extract-Min $\Theta(n)$.

(d) The worst-case running time of Extract-Min $\Theta(1)$.

Q.43 You are given a heap with n elements . Which of the following tasks can you achieve in $O(\log(n))$ time?

(a) Find the minimum element stored in the heap.

(b) Find the median of the elements stored in the heap.

(c) Find the fifth-smallest element stored in the heap.

(d) Find the largest element stored in the min heap.

Q.44 Which of the following statement(s) is/are true?

(a) The different minimum cuts are there in a tree with n nodes is $n-1$.

(b) The different minimum cuts are there in a tree with n nodes is n.

(c) On adding one extra edge to a directed graph G, the number of strongly connected components increases.

(d) On adding one extra edge to a directed graph G, the number of strongly connected components decreases.

Q.45 Which of the following statement(s) is/are correct for a heap with n objects?

(a) Insert takes $O(\log n)$ time.

(b) Extract-Min takes $O(\log n)$ time.

(c) Extract-Min takes O(1000) time in best case.
(d) Insert takes O(1000) time in best case.

Q.46 Which of the following statement(s) is/are true?
(a) The running time of depth - first search, as a function of n and m, if the input graph $G = (V, E)$ is represented by an adjacency matrix (i.e., NOT an adjacency list), where as usual $n = |V|$ and $m = |E|$ is $\theta(n^2)$.
(b) The running time of depth - first search, as a function of n and m, if the input graph $G = (V, E)$ is represented by an adjacency matrix (i.e., NOT an adjacency list), where as usual $n = |V|$ and $m = |E|$ is $\theta(n \log n)$.
(c) Let $0 < \alpha < 0.5$ be some constant. Consider running the Partition subroutine on an array with no duplicate elements and with the pivot element chosen uniformly at random (as in QuickSort). Then the probability that, after partitioning, both sub arrays (elements to the left of the pivot, and elements to the right of the pivot) have size at least α times that of the original array is $1 - 2\alpha$.
(d) Let $0 < \alpha < 0.5$ be some constant. Consider running the Partition subroutine on an array with no duplicate elements and with the pivot element chosen uniformly at random (as in QuickSort). Then the probability that, after partitioning, both sub arrays (elements to the left of the pivot, and elements to the right of the pivot) have size at least α times that of the original array is $1 - \alpha$.

Q.47 Which of the following statement(s) hold? (As usual n and m denote the number of vertices and edges, respectively, of a graph.)
(a) Breadth-first search can be used to compute the connected components of an undirected graph in $O(m + n)$ time.
(b) Depth-first search can be used to compute the strongly connected components of a directed graph in $O(m + n)$ time.
(c) Breadth-first search can be used to compute shortest paths in $O(m + n)$ time (when every edge has unit length).
(d) Depth-first search can be used to compute a topological ordering of a directed acyclic graph in $O(m + n)$ time.

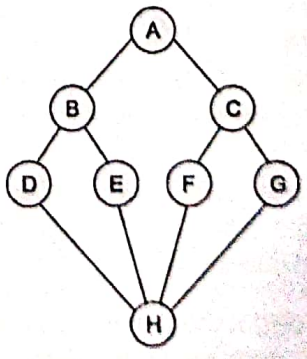
Q.48 When does a directed graph have a unique topological ordering (choose all the correct statements)?
(a) Whenever it is directed acyclic.
(b) Whenever it has a unique cycle.
(c) Whenever it is a complete directed graph.
(d) Whenever it is a line graph.

Q.49 Which of the following statements about Dijkstra's shortest-path algorithm are true for input graphs that might have some negative edge lengths?
(a) It is guaranteed to terminate.
(b) It may or may not correctly compute shortest-path distances (from a given source vertex to all other vertices), depending on the graph.
(c) It may or may not terminate (depending on the graph).
(d) It is guaranteed to correctly compute shortest-path distances (from a given source vertex to all other vertices).

Q.50 Which of the following statement is/are false for Bellman Ford algorithm.
(a) It will detect all the negative weight cycles if it exists.
(b) Every edge is relaxed once.
(c) Bellman ford algorithm runs in O(E log V)
(d) It will detect all the negative weight cycles iff it is reachable from the source.

Q.51 In Prim's algorithm, we use decrease key operation. Which of the following is/are correct with respect to the time complexity of this decrease key operation in case of Prim's Algorithm:
(a) $O(n \log n)$ (b) $O(n)$
(c) $O(1)$ (d) $O(\log n)$

Q.52 Consider the following graph:



Which of the following is/are can be the depth first search traversal of the above graph?

(a) ACFHEBDG (b) ACFHGEBD
(c) ABDHFCGE (d) ABDEHFGH

Q.53 Consider the already built heap (Max-Heap).
24, 20, 20, 7, 8, 9, 19, 1, 2
After root deletion operation, which of the following Max Heaps are valid.
(a) 20, 8, 20, 7, 2, 9, 19, 1
(b) 20, 20, 19, 7, 8, 9, 2, 1
(c) 20, 20, 9, 7, 8, 2, 19, 1
(d) 20, 7, 20, 2, 8, 9, 19, 1

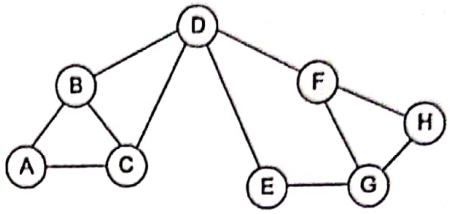
Q.54 Consider the following statement regarding Kruskal and Prim's Algorithm for generating minimum spanning tree:
(a) While generating MST, it is possible to obtain forest during execution of the algorithm in Kruskal's algorithm.
(b) The time complexity for generating MST is O(E log E) for Kruskal's algorithm.
(c) The time complexity for generating MST is O(E) for Prim's algorithm.
(d) While generating MST, it is possible to obtain forest during execution of the algorithm in Prim's algorithm.

Q.55 Consider the following statements regarding Dijkstra's shortest path algorithm:
(a) Dijkstra's Algorithm can be implemented on both directed and undirected graphs.
(b) Dijkstra Algorithm provides shortest distance from every node to every other node.
(c) Dijsktra Algorithm just provide the shortest distance, it doesn't provide the path information.
(d) Dijkstra Algorithm provides shortest distance from one node to every other node.
Which of the above statement(s) is/are correct?

Q.56 Which of the following problem(s) can be solved through BFS?
(a) Verifying given graph connected or not.
(b) Finding number of connected components in a graph.
(c) Checking given graph contain cycle or not.
(d) Single Source Shortest Path in given unweighted graph.

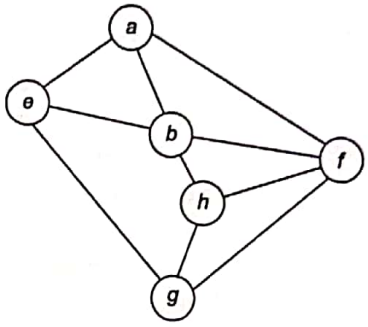
Q.57 You are given a binary tree (via a pointer to its root) with n nodes, which may or may not be a binary search tree. Then which of the following is/are correct to check whether or not the tree satisfies the search tree property?
(a) $O(n)$ time is necessary.
(b) $O(\log n)$ time is necessary.
(c) $O(n)$ time is sufficient.
(d) $O(\log n)$ time is sufficient.

Q.58 What will be the path from A-H if BFS is used in the following graph?



(a) ABCDEFGH (b) ACBDEFHG
(c) ACBDFEGH (d) ABCDFEHG

Q.59 Consider the following graph:



Which of the following are the depth-first traversals of the above graph?
(a) *abfehg* (b) *abfhge*
(c) *afghbe* (d) *abeghf*

Q.60 Single source shortest path problems can be implemented by greedy algorithms using
(a) Singly linked list (b) Min heap
(c) AVL tree (d) Doubly linked list

Q.61 Suppose we run Prim's algorithm and Kruskal's algorithm on a graph G and the two algorithms produce minimum cost spanning trees T_p and T_k respectively. Then, which of the following is/are true?
(a) T_p must be identical to T_k.
(b) If e is a minimum cost edge in G, e belongs to both T_p and T_k.
(c) If T_p is different from T_k, some pair of edges in G have the same weight.
(d) If e is a maximum cost edge in G, e belongs to neither T_p nor T_k.

Q.62 Which of the following is/are true about the Kruskal's algorithm?
(a) It is a greedy algorithm.
(b) It constructs MST by selecting edges in increasing order of their weights.
(c) It can accept cycles in the MST.
(d) It uses union-find data structure.

Q.63 Consider the following statements:
Which of the following statement(s) is/are true?
(a) Kruskal's algorithm might produce a non-minimal spanning tree.
(b) Kruskal's algorithm can efficiently implemented using the disjoint-set data structure.
(c) Min priority queue is the most commonly used data structure for implementing Dijkstra's Algorithm.
(d) The maximum number of times the decrease key operation performed in Dijkstra's algorithm will be equal to total number of edges.

Q.64 Given pseudo code of Dijkstra's Algorithm:

```
1. //Initialise single source(G, s)
2. S = 0
3. Q = V[G]
4. While Q! = 0
5.     Do u = extract-min(Q)
6.             S = S union {u}
7.             For each vertex v in adj[u]
8.             Do relax(u, v, w)
```

Then which of the following statement(s) is/are true when while loop in line 4 is changed to while Q > 1?
(a) While loop gets executed for V – 1.
(b) While loop gets executed for V times.
(c) Above algorithm will not give shortest path to every vertex in the graph.
(d) Above algorithm still work because that was just a more stronger condition.

Q.65 Which of the following is/are the application of Catalan Numbers?
(a) Counting the number of Dyck words.
(b) Counting the number of expressions containing n pairs of parenthesis.
(c) Counting the number of ways in which a convex polygon can be cut into triangles by connecting vertices with straight lines
(d) Creation of head and tail for a given number of tosses.

Q.66 Which of the following is/are true?
(a) Prim's algorithm can also be used for disconnected graphs.
(b) Kruskal's algorithm can also run on the disconnected graphs.
(c) Prim's algorithm is simpler than Kruskal's algorithm.
(d) In Kruskal's algorithm sort edges are added to MST in decreasing order of their weights.

Q.67 Consider the following statements:
Which of the following statement(s) is/are true?
(a) Kruskal's algorithm will never produce non-minimal spanning tree.
(b) Kruskal's algorithm can efficiently implemented using the disjoint-set data structure.
(c) Kruskal's algorithm is best suited for the dense graphs than the Prim's algorithm.
(d) Prim's algorithm is best suited for the dense graphs than the Kruskal's algorithm.

Q.68 When will the Depth First Search of a graph is/are unique?
(a) When the graph is a binary tree.
(b) When the graph is a linked list.
(c) When the graph is a n-ary tree.
(d) When the graph is a ternary tree.

Q.69 Which of the following is/are can be the application of Breadth First Search?
(a) Finding shortest path between two nodes
(b) Finding bipartiteness of a graph
(c) GPS navigation system
(d) Path Finding

Q.70 Which of the following statement(s) is/are true?
(a) For a connected, weighted graph with n vertices and exactly n edges, it is possible to find a minimum spanning tree in $O(n)$ time.
(b) In a simple, undirected, connected, weighted graph with at least three vertices and unique edge weights, the heaviest edge in the graph is in no minimum spanning tree.
(c) Given a hash table with more slots than keys, and collision resolution by chaining, the worst case running time of a lookup is constant time.
(d) If the DFS finishing time $f[u] > f[v]$ for two vertices u and v in a directed graph G, and u and v are in the same DFS tree in the DFS forest, then u is an ancestor of v in the depth first tree.

■■■■

Answers Graph Based Algorithms

1.	(64)	2.	(d)	3.	(b)	4.	(b)	5.	(6)	6.	(d)	7.	(c)	8.	(8)	9.	(d)
10.	(a)	11.	(b)	12.	(c)	13.	(128)	14.	(b)	15.	(c)	16.	(a)	17.	(d)	18.	(d)
19.	(d)	20.	(d)	21.	(c)	22.	(a)	23.	(b)	24.	(d)	25.	(c)	26.	(c)	27.	(c)
28.	(c)	29.	(c)	30.	(d)	31.	(c)	32.	(a)	33.	(b)	34.	(b)	35.	(b)	36.	(15)
37.	(4)	38.	(a, b, c, d)	39.	(a, b, c, d)	40.	(a, c)	41.	(a, d)	42.	(b, c)	43.	(a, c)	44.	(a)		
45.	(a, b, d)	46.	(a, c)	47.	(a, b, c, d)	48.	(d)	49.	(a, b)	50.	(a, b, c)	51.	(a, b, d)				
52.	(a, b, c)	53.	(a, b)	54.	(a, b)	55.	(a, d)	56.	(a, b, c, d)	57.	(a, c)	58.	(a, b, c, d)				
59.	(b, c, d)	60.	(a, b, c, d)	61.	(c)	62.	(a, b, d)	63.	(b, c, d)	64.	(a, c)	65.	(a, b, c)				
66.	(b)	67.	(a, b, d)	68.	(b)	69.	(a, b, c)	70.	(a)								

Explanations Graph Based Algorithms

1. (64)
As number of undirected graph is = $2^{(n(n-1)/2)}$

2. (d)
Both statement are false because forest is large version of trees. So, there is no bound about number of edges and vertices.

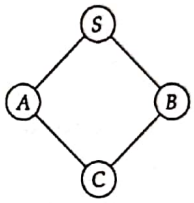
3. (b)
Connected components of a graph can be computed in linear time by using either breadth-first search or depth-first search.

4. (b)
We cannot compute the shortest path in an undirected graph with the help of DFS, BFS is used for it. So, option (b) is the answer.

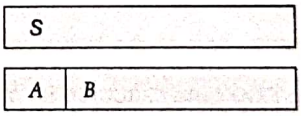
5. (6)
(A, D); (D, E), (D, H);
(E, C), (E, G); (C, F);
(F, I); (I, B)
The maximum height of tree is 6.

6. (d)
In an unweighted graph performing a BFS starting from S takes $O(n)$ time if graph contains n vertices. Dijkstra's algorithm only find the single source shortest path and takes $O(n^2)$ time but it is good for weighted graph. Warshall's algorithm for all pair shortest path takes $O(n^3)$ time but it is also applicable for weighted graph.

7. (c)
Consider a graph:



During the breadth first traversal of the graph. The status of the queue will be as follows:



$\therefore$ $S - A \rightarrow 1$ edge
$S - B \rightarrow 1$ edge
Difference = 0

$\therefore$ $S - B \rightarrow 1$ edge
$S - C \rightarrow 2$ edge
Difference = 1

Hence, statements S_1 and S_3 are correct.

8. (8)
In the above graph, there are three stages with 2 vertices. Topological sort picks the element with zero in degree at any point of time. At each of two

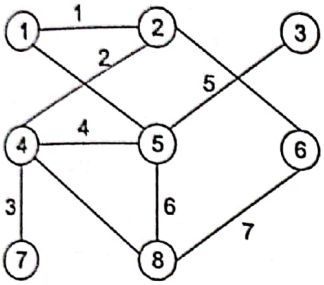
vertices stages, either the top vertex or the bottom vertex can be processed. So at each of these stages there will be two possibilities. Total number of possibilities = 2 × 2 × 2 = 8.

9. (d)
ABDE H FGH is not DFS traversal

10. (a)
Preorder is same as depth first order.

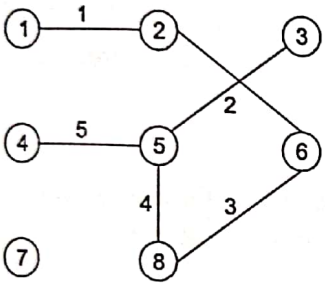
11. (b)

(a)



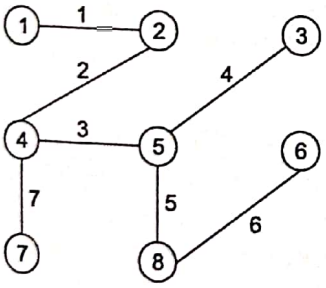
DFS safe sequence

(b)



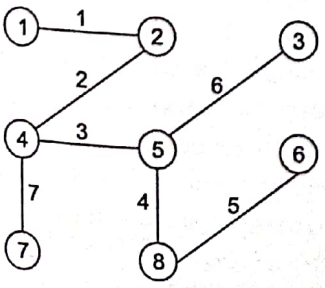
4 to 3 not follow DFS so not DFS sequence because have to go 4 to 7 first then 7 to 3.

(c)



DFS safe sequence

(d)

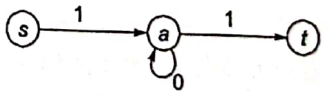


DFS safe sequence

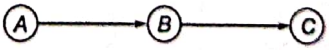
12. (c)
Statement 1 → True. If DFS finds no back edges, then the graph is acyclic. Removing any back edges found does not change the traversal order of DFS, so running DFS again on modified graph would produce no back edges.
Statement 2 → True. Each needs to keep track of the vertices that have already been visited.
Statement 3 → False. The DFS forest may contain different numbers of trees (and tree edges) depending on the starting vertex and upon the order in which vertices are searched.
Hence, option (c) is current.

13. (128)
Since, maximum number of BFS traversal is asked the traversal should start from the root node. Every internal node have two choices to visit the node.
The total number of internal nodes = 7.
So, there are 2^7 ways = 128.

14. (b)
Statement 1: False: The edge could be either a cross-edge or a forward edge. The edge cannot be a back edge (a back edge goes to a vertex that has started but not finished).
Statement 2: False: The parent pointers may not lead back to the source node if a zero-length cycle exists. In example given 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 path of s, so the algorithm is incorrect.



Statement 3: True: The i^{th} iteration finds shortest paths in G of i or fewer edges, by the path relaxation property.
Statement 4: True: Both parts of the statement hold if and only if the graph is acyclic.

15. (c)
S_1: 

Starting with A will get 2 edges
Starting with B will get only 1 edge
Starting with C will get no edge
∴ DFS on directed graph may not give same number of edges.

S_2: Removing all back edges makes the graph as acyclic.

So S_2 is valid statement.

Option (c) is correct.

17. (d)

$A \rightarrow B \rightarrow D$

$B \rightarrow C \rightarrow E \rightarrow G$

$D \rightarrow C$

$C \rightarrow A$

$E \rightarrow H$

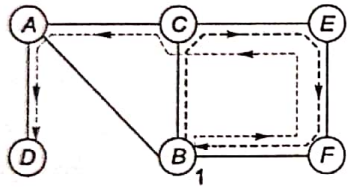
$G \rightarrow F$

$H \rightarrow F \rightarrow G$

So, for BFS, the answer is ABDCEGHF AND FOR DFS, the answer is ABCEHFGD.

So, option (d) is correct.

18. (d)



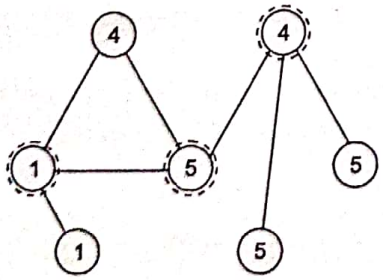
$\because$ $BCEFAD$ is correct depth first search order.

19. (d)

In a connected graph, a vertex v is said be an articulation point if by removing that vertex together with its edges the graph become disconnected. In the given graph there is no articulation point.

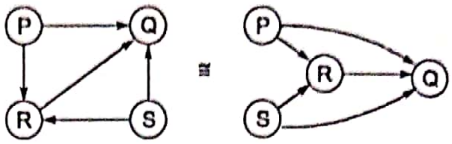
20. (d)

An articulation point (or cut vertex) is that vertex removing which (along with its edges) disconnects the graph.



Therefore number of articulation points is 3.

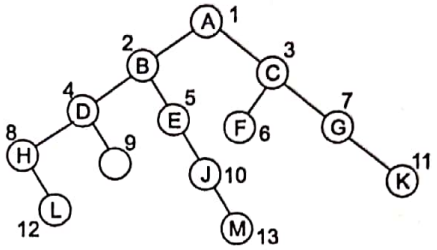
21. (c)



$\therefore$ Topological orderings: PSRQ and SPRQ

22. (a)

For BFS algorithm, visiting a node's siblings before its children, while in DFS algorithm, visiting a node's children before its siblings.



Before countering goal node F:

BFS algorithm encounters nodes: ABCDE

DFS algorithm encounters nodes : ABDHLIEJMC

So, option (a) is correct.

23. (b)

Dijkstra algorithm $\rightarrow O((E+V)\log V)$

Prim algorithm $\rightarrow O(E \log V)$

BFS $\rightarrow O(V+E)$

Floyd-Warshall $\rightarrow O(V^3)$

Kruskal algorithm $\rightarrow O(E \log V)$

Bellman Ford $\rightarrow O(VE)$

So, option (b) is correct.

24. (d)

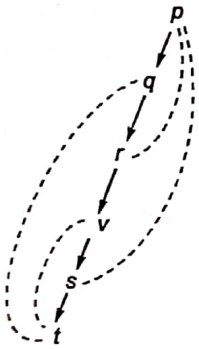
In a graph of strongly connected components, mother vertices are always vertices of source component in component graph. If these exist mother vertex (or vertices), then one of the mother vertices is the last finished vertex in DFS.

1. Do DFS traversal of the given graph. While doing traversal keep track of last finished vertex 'V'. This step takes $O(V+E)$ time.
2. If there exist mother vertex (or vertices), then V must be one (or one of them). Check if V is a mother vertex by doing DFS/BFS from V. This step also takes $O(V+E)$ time.

So, option (d) is correct.

25. (c)
Statement 1: False: If the graph contains a negative weight cycle then no shortest path exists.
Statement 2: False: Even if no two edges have some weight there could be two paths with the same weight.
Statement 3: True: Since all vertices are connected by a path with at most T edges, and since BFS always finds the path with the fewest edges, the BFS tree will have depth atmost T. ADFS tree may have depth up to V-1 (for example in a complete graph).
Statement 4: True: if such an edge existed, if would provide a shorter path to some node than the path found by BFS (interms in number of edges). This cannot happen, as BFS always finds the path with the fewest edges.
Hence, statement 3 and 4 are true.

26. (c)
Apply the given DFS



Tree edges are: $\{p, q\}, \{q, r\}, \{r, v\}, \{v, s\}, \{s, t\}$.
Back edges are: $\{t, q\}, \{t, v\}, \{s, p\}, \{r, p\}$.

27. (c)
There are three strong components
1. G, A
2. E_1, T, S, E_2
3. C

∴ Option (c) is correct.

28. (c)
If u is visited before v, $d(r, u) \le d(r, v)$ i.e., r to u is having shortest or equal length compared from r to v.

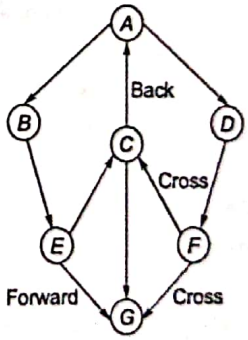
29. (c)
(a) 'S' is not part of PQR component.
(b) $Q \rightarrow R$, but then P and S are separate component.
(d) For Q its (1, 2) it means it is not connected to any other vertex.
(c) Start with R then Q (start time 2), S(start time 3), then P(start time P). So graph is connected. As its undirected graph we don't need to worry about direction of edges.
So, option (c) is correct.

30. (d)
In directed graph, no vertex visited twice means acyclic graph then no back edges in directed graph means acyclic graph.
Hence, all statements are true.

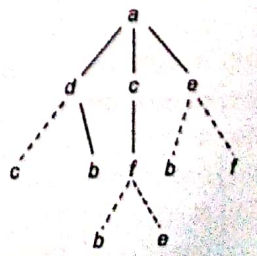
31. (c)
DFS can be used to find strongly connected components and topological sort.
Hence, option (c) is correct.

32. (a)
5 strongly component.
$\{s, v, w\}, \{x, z\}, \{q, t, y\}, \{r\}, \{v\}$

33. (b)
⇒ 2 Cross edges in G,
1 Forward edge in G,
1 Back edge in G
∴ Number of cross edges = 2



34. (b)
Given BFS traversal on G is:



Tree edges are: {a, d}, {a, c}, {a, e}, {d, b}, {c, f}
Cross edges are: {d, c}, {e, b}, {e, f}, {f, b}, {f, e}
⇒ {d, b} is not a cross edge.

35. (b)

Depth first traversal can be used to detect a cycle in a graph. DFS for a connected graph produces a tree. There is a cycle in a graph only if there is back edge present in the graph. A back edge is an edge that is from a node to itself (self-loop) or one of its ancestor in the tree produced by DFS. For a disconnected graph, we get the DFS forest as output. To detect cycle, we can check for a cycle in individual trees by checking back edges. To detect a backedge, we can keep track of vertices currently in recursion stack of function for DFS traversal. If we search a vertex that is already in the recursion stack, then there is a cycle in the tree. The edge that connects current vertex in the recursion stack is a backedge.

Time complexity of this method is the same as time complexity of DFS traversal which is $O(V+E)$.

So, option (b) is correct.

Option (a) is not correct because it may be guaranteed. So, option (b) is correct.

36. (15)

There are 8 possibilities starting with A.

We are missing : : (Starting with A)

$A-C-E-B-D-F$

$A-B-E-C-D-F$ + money more

Starting with $A = 8$, with $C = 4$, with $E = 3$

⇒ Total = 15

and $E-F-A-B-D-C$ is not a topological sort.

So, the number of topological order = 15.

37. (4)

Definition of backedge → if v is an ancestor of u, then edge (u, v) is a backedge.

- Now, starting from p next unvisited node $\{q, r, s\}$ choosing based on alphabetical order i.e. q.
- Standing at node q next unvisited nodes $\{r, t\}$ choosing based on alphabetical order i.e., r.]
- Standing at node r next unvisited nodes $\{v\}$ choosing based on alphabetical order i.e. v.
- Standing at node v next unvisited node $\{s, t\}$ choosing based on alphabetical order i.e. s.
- Standing at node s next unvisited node $\{t\}$ choosing based on alphabetical order i.e., t.

All nodes are traversed.

Back edges are as follows: (t, v), (s, p), (t, q), (r, p)

Number of backedges = 4

Hence, 4 is the answer.

38. (a, b, c, d)

Depth first search is used to traverse a rooted tree.

Pre-order, Post-order and Inorder are used to list the vertices of an ordered rooted tree.

Huffman's algorithm is used to find an optimal binary tree with given weights.

Topological sorting provides a labelling such that the parents have larger labels than their children.

40. (a, c)

Dijkstra's does correctly compute the shortest-path distance. To claim that, let's go back to the major idea in the proof of correctness.

We argue by the time Dijkstra's algorithm set to shortest path, that path has to be shortest, and we argue that by splitting an arbitrary path from s to t into three parts.

$s \Rightarrow v$ (where we know $s \rightarrow v$ is an already known shortest path)

$v \rightarrow x$ (where x is some node that we do not know the shortest path yet), and finally

$x \rightarrow t$ it may or may not be getting back to some nodes with known shortest path.

The key thing that we need from the non-negativity of edge length is that we need to be sure that the path $x \rightarrow t$ has non-negative length, and yes, this is satisfied in our modified problem because x is the node with unknown shortest path and therefore cannot be s, but all paths with potentially negative edge weight must start with s, so we can still claim $x \rightarrow t$ has non-negative edge weight.

Bellman Ford obviously gives shortest path.

41. (a, d)

In the worst case, the element being inserted is the biggest one and therefore we have to shift the whole array, therefore $\Theta(n)$.

For delete min, we simply take the last one in the array (presumably we maintained a variable for the length of the array) and reduce the variable by 1. All of these work are constant time, so $\Theta(1)$.

42. (b, c)

The worst case running time for insertion is easy, one can simply append it in the end of the unsorted array, so $\Theta(1)$.

The worst case running time for extract min is not so easy to claim because the algorithm is not specified. Apparently since the array is unsorted there is naturally no better algorithm than just linear search. Worst case performance for linear search is $\Theta(n)$.

The question is, it possible to have an algorithm that perform better than $\Theta(n)$? The answer is no and it is proved by the adversary argument. The algorithm can choose what element to read and the adversary is going to give any element that is not minimum until $n-1$ element are read and therefore the adversary forced the algorithm to read through all elements.

43. (a, c)

Yes, minimum element can be found.

No, median can't be find within logn time.

Yes, just do delete min 5 times and insert all the other elements back (or alternatively just to a tree traversal of the first 5 levels = 32 nodes, a constant).

No, the maximum is among the leaf nodes, there are $O(n)$ leaf nodes and we do not know which one it is.

44. (a)

Any edge is a minimum cut, so there are $n-1$ minimum cuts.

Anything could happen!

Suppose the graph is a direct chain, adding an edge to make it a cycle can make it reduce a lot of strongly connected components.

$1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ (4 strongly connected components)

Adding $4 \rightarrow 1$ will result in just 1 strongly connected components.

On the other hand adding one more edge, say $1 \rightarrow 3$, does not decrease it any more.

So the answer is this choice: ...might or might not remain the same (depending on the graph).

45. (a, b, d)

This is just memorizing the standard results.

46. (a, c)

- Each node need to be gone through. And for each of them it need to look at all other nodes for a possible edge, so the time is $\theta(n^2)$.
- It is easiest to look at problem of this sort in term of concrete numbers, it is very easy that way, let say $\alpha = 0.2$ and $n = 10$. Then the requirement is that we have at least 2 elements in both sub arrays. Further suppose the numbers are simply 1,2,..,10, then as long as we are not picking 1, 2, 9, 10, we are good to go.

Generalizing, it is easy to see that if we are not picking the $n\alpha$ largest numbers and $n\alpha$ smallest numbers, then we satisfy the condition, the probability of that is simply $1 - 2\alpha$.

47. (a, b, c, d)

All of these options are correct and all of them are just standard results.

48. (d)

This is wrong $1 \rightarrow 2, 3$ has two topological order

This is wrong, any graph with a cycle do not have a topological order.

A complete directed graph of size > 1 will necessarily have cycle, so no topological order at all.

Option (d) is correct trivially.

49. (a, b)

The algorithm, regardless of correctness, also make progress towards completion by removing a vertex from the unknown set, so it must terminate.

This is true because the graph may not have any negative edge at all.

This is not true given option (a).

This is not true, or else we don't really need the restriction.

50. (a, b, c)

Bellman Ford algorithm has a time complexity of $O(VE)$, as every edge in graph is relaxed $V-1$.

Times, to find all lengths (1, 2, ...$V-1$) paths to find single source shortest path.

So option (a), (b), (c) are false.

51. (a, b, d)

In Prim's algorithm, heap of vertices are created and decrease key operation is performed on the

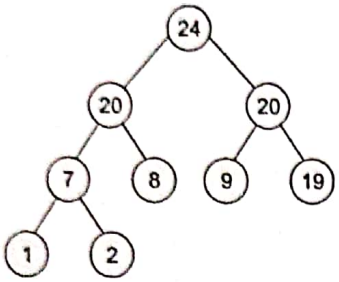
heap elements which takes O(log$n$) time. Decrease key operation is performed to change the value of distance in every iteration.

52. (a, b, c)

ABDEHFGH is not DFS traversal.

53. (a, b)

The given heap is



When we delete root element in max heap we replace root with last node in the tree. Now, while Max-Heapify down, we select the maximum of the two children and swap it with root.

Here the two children are both 20 and 20 and we can swap with either of them and the only two possible heaps after deletion and heapifying are

1. 20, 8, 20, 7, 2, 9, 19, 1
2. 20, 20, 19, 7, 8, 9, 2, 1

54. (a, b)

It is possible to obtain the forest when MSTs are generated using Kruskal algorithm

It is not possible to obtain the forest when MSTs are generated using Prim's algorithm.

Time complexity for Kruskal Algorithm = O(E log E)

Time Complexity for Prim's Algorithm = O(E log V) = O(E log E)

55. (a, d)

- It is correct. Dijsktra can be implemented on both directed and undirected graphs.
- It is incorrect because Dijsktra provides shortest distance from source to all nodes.
- It is incorrect because we can get path information form the table which provides the order of nodes visited.
- It is correct because Dijsktra provides shortest distance from source to all nodes.

57. (a, c)

A top down solution would look like this:

```
function check(Node node, int minimum, int maximum)
    if (node == null)
        return true;
    if (node.value > maximum)
        return false;
    if (node.value < minimum)
        return false;
return check(node.left) && check(node.right)
```

The algorithm inspect every element exactly once, so it is O(n).

Suppose there exist an algorithm that can run asymptotically faster than O(n), it cannot inspect all nodes. If it cannot inspect all nodes, then the adversary is free to choose what value to put in that node. In particular, for a case when the algorithm return true, the adversary can always put a value in that not inspected node to invalid the validity of the tree as follow.

Suppose the node being not inspected in the rightmost node, then the algorithm can put a value smaller than any value in the tree, that would sabotage the algorithm.

Otherwise, the algorithm can put a value that is larger than any value in the tree, the adversary can also sabotage the algorithm.

Therefore, in order to not get sabotaged, the algorithm must inspect all node, that prove the problem will take at least $\Omega(n)$ time.

58. (a, b, c, d)

All the statements are correct.

59. (b, c, d)

For GATE purpose, without actually applying DFS, we can answer by just seeing options.

In DFS, we go in depth first i.e., one node to another in depth first order.

Here, abfehg is not possible as we can not go from f to e directly.

In all the other options we can reach directly from the node to the next node.

60. (a, b, c, d)

We can use all of the above for single source shortest path but efficiently it will be done by min-heap.

62. (a, b, d)

Kruskal's algorithm is a greedy algorithm to construct the MST of the given graph. It constructs the MST by selecting edges in increasing order of their weights and rejects an

edge if it may form the cycle. So, using Kruskal's algorithm cycle is never formed.

63. (b, c, d)

In Kruskal's algorithm, the disjoint-set data structure efficiently identifies the components containing a vertex and adds the new edges. And Kruskal's algorithm always finds the MST for the connected graph.

Minimum priority queue is the most commonly used data structure for implementing Dijkstra's Algorithm because the required operations to be performed in Dijkstra's Algorithm match with specialty of a minimum priority queue.

If the total number of edges in all adjacency list is E, then there will be a total of E number of iterations, hence there will be a total of at most E decrease key operations.

64. (a, c)

In the normal execution of Dijkstra's Algorithm, the while loop gets executed V times. The change in the while loop statement causes it to execute only V – 1 times. For similar reasons it will not give correct result.

65. (a, b, c)

Counting the number of Dyck words, Counting the number of expressions containing n pairs of parenthesis, Counting the number of ways in which a convex polygon can be cut into triangles by connecting vertices with straight lines are the applications of Catalan numbers where as creation of head and tails for a given number of tosses is an application of Pascal's triangle.

66. (b)

Prim's algorithm iterates from one node to another, so it can not be applied for disconnected graph. Kruskal's algorithm can be applied to the disconnected graphs to construct the minimum cost forest. Kruskal's algorithm is comparatively easier and simpler than Prim's algorithm.

67. (a, b, d)

In Kruskal's algorithm, the disjoint-set data structure efficiently identifies the components containing a vertex and adds the new edges. And Kruskal's algorithm always finds the MST for the connected graph.

Prim's algorithm outperforms the Kruskal's algorithm in case of the dense graphs. It is significantly faster if graph has more edges than the Kruskal's algorithm.

68. (b)

When Every node will have one successor then the Depth First Search is unique. In all other cases, when it will have more than one successor, it can choose any of them in arbitrary order.

69. (a, b, c)

Breadth First Search can be applied to Bipartite a graph, to find the shortest path between two nodes, in GPS Navigation. In Path finding, Depth First Search is used.

70. (a)

True. This graph only contains one cycle, which can be found by a DFS. Just remove the heaviest edge in that cycle.

False. If the heaviest edge in the graph is the only edge connecting some vertex to the rest of the graph, then it must be in every minimum spanning tree.

False. In the worst case we get unlucky and all the keys hash to the same slot, for $\theta(n)$ time.

False. In a graph with three nodes, r, u and v, with edges (r, u) and (r, v) and r is the starting point for the DFS, u and v are siblings in the DFS tree, neither as the ancestor of the other.

■■■■

6 CHAPTER

Dynamic Programming

Multiple Choice Questions & NAT Questions

Q.1 Consider the following problems:

1. Longest common subsequence.
2. Optimal Binary search tree.
3. 0/1 Knapsack problem.
4. Matrix chain multiplication.

Which of the above problems can be solved using dynamic programming?

(a) 1 and 2 only (b) 2 and 3 only
(c) 1, 3 and 4 only (d) 1, 2, 3 and 4

Q.2 Which of the following standard algorithms is not a greedy algorithm?

(a) Selection sort
(b) Kruskal's algorithm
(c) Bellman Ford shortest path algorithm
(d) Dijkstra's shortest path algorithm

Q.3 Consider the following steps:

S_1: Characterize the structure of an optimal solution.

S_2: Compute the value of an optimal solution in bottom-up fashion.

Which of the following step(s) is/are common to both dynamic programming and greedy algorithms?

(a) Only S_1
(b) Only S_2
(c) Both S_1 and S_2
(d) Neither S_1 nor S_2

Q.4 Match the following groups:

Group-1

A. Divide and conquer
B. Greedy approach
C. Dynamic programming
D. Back tracking

Group-2

1. Longest common subsequence
2. Huffman codes
3. Quick sort
4. Chess game

Codes:

	A	B	C	D
(a)	1	2	3	4
(b)	3	1	2	4
(c)	3	2	1	4

(d) None of these

Q.5 Consider two strings A = "anandarmy" and B = "algorithms". Let 'y' be the length of the longest common subsequences between A and B and let 'x' be the number of such longest common subsequences between A and B.

Then $2x + 3y =$ _____?

Q.6 Given an text array $T[1.....n]$ and a pattern array $P[1....m]$ such that T and P are character taken from alphabet Σ, $\Sigma = a, b, c ..... z$. String matching problem is to find all the occurrence of P in T. A pattern occur with shift S in T if $P[1 ....m] = T[S+1, ...S + m]$.

Consider $T = bacacbaacacac$

$P = cac$

The sum of the value of all S is _____.

Q.7 Consider the following matrices with dimensions.

A_1 is 4×6
A_2 is 6×8
A_3 is 8×4
A_4 is 4×5

Which of the following multiplication order gives optimal solution?

(a) $((A_1 A_2) A_3) A_4$ (b) $(A_1 (A_2 A_3))A_4$
(c) $A_1 ((A_2 A_3) A_4)$ (d) $(A_1 A_2) (A_3 A_4)$

Q.8 Consider the following steps:

S_1: Characterize the structure of an optimal solution.

S_2: Compute the value of an optimal solution in bottom-up fashion.

Which of the following steps(s) is/are common to both Greedy algorithms and dynamic programming?

(a) Only S_1
(b) Both S_1 and S_2
(c) Only S_2
(d) Neither S_1 and S_2

Q.9 Consider the following strings x and y:
x = "csemadeeasy"
y = "gateexam"
How many longest common subsequences are possible from x and y?

Q.10 In the following table, the left column contains the names of standard graph algorithms and the right column contains the time complexities of the algorithms. Match each algorithm with its time complexity.

List-I
1. Bellman-Ford algorithm
2. Kruskal's algorithm
3. Floyd-Warshall algorithm
4. Topological sorting

List-II
A. $O(m \log n)$
B. $O(n^3)$
C. $O(nm)$
D. $O(n + m)$

(a) 1-C, 2-A, 3-B, 4-D
(b) 1-B, 2-D, 3-C, 4-A
(c) 1-C, 2-D, 3-A, 4-B
(d) 1-B, 2-A, 3-C, 4-D

Q.11 Which of the following is the recurrence relation for the matrix chain multiplication problem where $p[i-1] * p[i]$ gives the dimension of i^{th} matrix?
(a) $dp[i, j] = 1$ if $i = j$
$dp[i, j] = \min\{dp[i, k] + dp[k+1, j]\}$
(b) $dp[i, j] = 1$ if $i = j$
$dp[i, j] = \min\{dp[i, k] + dp[k+1, j]\} + p[i-1] * p[k] * p[j]$
(c) $dp[i, j] = 0$ if $i = j$
$dp[i, j] = \min\{dp[i, k] + dp[k+1, j]\}$
(d) $dp[i, j] = 0$ if $i = j$
$dp[i, j] = \min\{dp[i, k] + dp[k+1, j]\} + p[i-1] * p[k] * p[j]$

Q.12 How many number of longest common subsequences of X = gate123 and Y = aget 321 can be formed?
(a) 2 (b) 8
(c) 12 (d) 14

Q.13 Let $X = abcbdab$ and $Y = bdcaba$.
Find the length of longest common subsequence (LCS) of X and Y.
(a) 3 (b) 4
(c) 5 (d) 6

Q.14 Consider the following knapsack problem with capacity $W = 8$

Item	Profit	Weight
I_1	13	1
I_2	8	5
I_3	7	3
I_4	3	4

Which of the following item is not selected in the optimal solution of 0/1 Knapsack problem?
(a) I_1 only (b) I_2 only
(c) I_3 only (d) None of these

Q.15 Total number of ways to perform matrix multiplication having 7 matrices is _____?

Q.16 Consider the problem of chain $< A_1, A_2, A_3, A_4 >$ of four matrices. Suppose that the dimensions of the matrices A_1, A_2, A_3 and A_4 are 30×35, 35×15, 15×5 and 5×10 respectively. The minimum number of scalar multiplication needed to compute the product $A_1 A_2 A_3 A_4$ is _____.
(a) 14875 (b) 21000
(c) 9375 (d) 11875

Q.17 Which of the following statement is/are correct:
S_1: For a dynamic programming algorithm, computing all values in a bottom-up fashion is asymptotically faster than using recursion.
S_2: The running time of dynamic programing algorithm is always $\theta(P)$ where P is the number of subproblems.
(a) S_1 – True; S_2 – True
(b) S_1 – True; S_2 – False
(c) S_1 – False; S_2 – True
(d) S_1 – False; S_2 – False

Q.18 Running time of 0/1 Knapsack problem using dynamic programming is _____. Assume n is the number of items and w is the capacity of Knapsack.
(a) $O(n^2)$ (b) $O(n^w)$
(c) $O(w^n)$ (d) $O(nw)$

Q.19 Given a set of n points: $S = \{P_1, P_2, P_3, P_4 \ldots P_n\}$, where $P_i = (x_i, y_i)$. Finding the pair of points that has the smallest distance among all pairs that can be solved in $O(n \log n)$ time using
(a) Divide and Conquer
(b) Greedy Technique
(c) Dynamic programming
(d) All of these

Q.20 Match List-I with List-II and select the correct answer using the codes given below the lists:

List-I
A. $O(n^2)$
B. $O(n)$
C. $O(n \log n)$
D. $O(2^n)$

List-II
1. Merge sort
2. The Towers of Hanoi problem
3. Shortest path between two nodes in graph
4. Finding an element in the unsorted array

Codes:

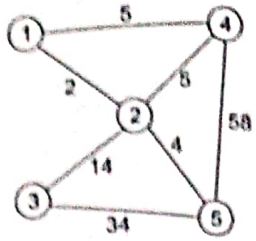
	A	B	C	D
(a)	1	2	4	3
(b)	3	4	1	1
(c)	4	2	1	3
(d)	4	2	3	1

Q.21 Consider the following chain of matrices A_1 to A_4 having dimensions given below:
$A_1 \rightarrow 2 \times 3$, $A_2 \rightarrow 3 \times 5$, $A_3 \rightarrow 5 \times 4$, $A_4 \rightarrow 4 \times 2$
The following table is filled up using dynamic programming in order to find minimum number of scalar multiplications:

j \ i	1	2	3	4
1	0	30	70	Q
2	0	0	P	70
3	0	0	0	40
4	0	0	0	0

where are the values of P and Q?
(a) 60, 140 (b) 60, 82
(c) 60, 40 (d) 60, 92

Q.22 Consider the following graph which represents 5 locations access a city and a person has to travel all of them starting from vertex 1 and returning back. What is cost of the optimal tour.



(a) 1 - 2 - 3 - 4 - 5 - 1 (b) 1 - 5 - 3 - 2 - 4 - 1
(c) 1 - 4 - 3 - 5 - 2 - 1 (d) 1 - 4 - 5 - 3 - 2 - 1

Q.23 Given an array of n numbers, give an algorithm for finding a contiguous subsequence $A(i) \ldots A(j)$ for which the sum of elements is maximum
Eg. $[-2, 11, -4, 13, -5, 2] \rightarrow 20$

If dynamic programming approach is used then what is time complexity and space complexity?
(a) $O(n^3), O(1)$ (b) $O(n), O(n)$
(c) $O(n^3), O(n)$ (d) $O(n^2), O(1)$

Q.24 The simplified SOP(Sum of Product) from the Boolean expression $(P+\bar{Q}+\bar{R}) \cdot (P+Q+R) \cdot (P+Q+\bar{R})$ is
(a) $(\bar{P}.Q+\bar{R})$ (b) $(P+Q.\bar{R})$
(c) $(P.\bar{Q}+R)$ (d) $(P.Q+R)$

Q.25 In the Floyd-Warshall Algorithm for all pairs shortest path. The matrix D represents
(a) The shortest path of length i in between all the vertices.
(b) The shortest path of the first i vertices among themselves.
(c) Intermediate paths of length i in the shortest path between all pairs of vertices.
(d) None of the above

Q.26 The length of the largest common subsequences for the given following strings is _____.
"hippopotamus"
"spontaneous"

Q.27 Consider the following adjacency matrix 'D' for represented the directed graph with distances (costs) between every pair of vertices.

$$D = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{bmatrix} 0 & 3 & 2 \\ 1 & 0 & 2 \\ 4 & 1 & 0 \end{bmatrix} \end{array}$$

If warshall's algorithm is applied on 'D' to compute the shortest distances then following resultant matrix is obtained.

$$D^* = \begin{array}{c} \\ A \\ B \\ C \end{array}\begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{bmatrix} 0 & 3 & 2 \\ 1 & 0 & 2 \\ 2 & 1 & 0 \end{bmatrix} \end{array}$$

If the edge from B to C in the given directed graph is removed then how many shortest distances will be changed in D^*?

Q.28 What is the running time of efficient algorithm for the finding the shortest path between two vertices in a directed graph? Assume that all edges are having equal weights. (V is set of vertices and E is set of edges)

(a) $|V|$ (b) $|V| \log |E|$
(c) $O(|V| + |E|)$ (d) None of these

Q.29 If the tightest upper bound for the time complexity required to solve the travelling salesman problem by using dynamic programming is represented by $O(f(n))$ and that of 0/1 Knapsack problem is by using dynamic programming $O(g(n))$ (for small capacity of Knapsack when compared to the number of items) and that of Floyd Warshall's algorithm is $O(h(n))$ then which of the following captures the relation in between them.

(a) $f(n) = O(g(n))$; and $g(n) = O(n(n))$
(b) $f(n) = \Omega(g(n))$; and $g(n) = O(n(n))$
(c) $f(n) = O(g(n))$; and $g(n) = \Omega(n(n))$
(d) $f(n) = \Omega(g(n))$; and $g(n) = \Omega(n(n))$

Q.30 Given 4 matrices M_1, M_2, M_3, M_4 which are having the orders of the following form $2 * 3$, $3 * 4$, $4 * 7$, $7 * 2$, this matrix multiplication was done by using the optimal parenthesization then number of multiplication operations done are _____?

Q.31 The optimal number of multiplication required to multiply four matrices with the following dimensions are _____.
$M_1 : (100 \times 1)$ $M_2 : (1 \times 100)$ $M_3 : (100 \times 1)$;
$M_4 : (1 \times 100)$

Q.32 Given that $X = < A, B, C, B, D, A, B >$ and $Y = < B, D, C, A, B, A >$ Which of the following is true?

(a) $<B, C, A>$ is the longest common subsequence (LCS) of X and Y.
(b) $<B, C, B, A>$ is one of the LCS of X and Y.
(c) LCS of length 5 or more exists.
(d) None of these

Q.33 Let $X = < x_1, x_2, \ldots x_m >$ and $Y = < y_1, y_2, \ldots y_n >$ be sequences. What is the running time to find the longest common subsequence of X and Y using dynamic programming.

(a) $O(m + n)$ (b) $O(mn)$
(c) $O(n^m)$ (d) $O(n^m)$

Q.34 Let $P_0 = 5$, $P_1 = 6$, $P_2 = 7$, $P_3 = 1$, $P_4 = 10$, $P_5 = 2$ and for $1 \le i \le 5$. Let X_i be a matrix with P_{i-1} rows and P_i columns, and $X = X_1.X_2.X_3.X_4.X_5$. Which of the following is optimum parenthesization for computing X?

(a) $((X_1 (X_2 \cdot X_3)) X_4) X_5$
(b) $(X_1 (X_2 \cdot X_3)) (X_4 \cdot X_5)$
(c) $(X_1 X_2) ((X_3 X_4) \cdot X_5)$
(d) $((X_1 \cdot X_2) (X_3 \cdot X_4)) X_5$

Q.35 Consider the following statements:

S_1: Greedy algorithm exists.
S_2: Dynamic programming algorithm exists.
S_3: Exhibit optimal substructure property.

Which of the above statements are true for 0-1 Knapsack problem?

(a) Only S_1 and S_2
(b) Only S_2 and S_3
(c) Only S_1 and S_3
(d) S_1, S_2 and S_3

Q.36 Let $G = (V, E)$ be a directed graph. Each edge of G is represented as (i, j) with length $l[i, j]$. If there is no edge from i to j then $l[i, j] = \infty$. Assume n vertices in V and $d^k_{i,j}$ is the length of shortest path from i to j that does not pass through any vertex in $\{k + 1, k + 2, \ldots, n\}$.

$$d^k_{i,j} = \begin{cases} l[i, j] & \text{if } k = 0 \\ \min\{A, B\} & \text{if } 1 \le k \le n \end{cases}$$

If the above $d^k_{i,j}$ computed recursively to find all pairs shortest path, identify A and B respectively?

(a) $d^{k-1}_{i,j}$ and $d^{k-1}_{i,j} + d^{k-1}_{k,j}$
(b) $d^{k-1}_{i,j}$ and $d^{k-1}_{i,k} + d^{k-1}_{k,j}$
(c) $d^k_{i,j}$ and $d^k_{i,k} + d^k_{k,j}$
(d) $d^k_{i,j}$ and $d^k_{i,k} + d^k_{j,k}$

Q.37 Consider the weighted undirected graph given with 4 vertices where the weight of edge (i, j) is the every w_{ij} in the matrix w.

$$w = \begin{bmatrix} 0 & 2 & 8 & 5 \\ 2 & 0 & 5 & 8 \\ 8 & 5 & 0 & x \\ 5 & 8 & x & 0 \end{bmatrix}$$

The largest possible integer value of x, for which atleast one shortest path between some pair of vertices will contain the edge with weight x is _____?

Q.38 Given two sequences A and B. Let $X(A, B)$ denote the number of times that A appears as subsequence of B i.e. sequence ab appears 4 times as a subsequence of *aebabdb*. Let A_i denotes the first i characters of string A and $A[i]$ denote the i^{th} character.

Which of the following will computes the recurrence relation $C(A_i, B_j)$?

$$C(A_i, B_j) = \begin{cases} 1; & \text{if } (i = 0) \\ 0; & \text{if } (i > 0 \text{ and } j = 0) \\ \text{----} & \text{----------} \\ \text{----} & \text{----------} \end{cases}$$

(a) $C(A_i, B_{j-1})$; if $A[i] \neq B[j]$
$C(A_i, B_{j-1}) + C(A_{i-1}, B_{j-1})$; if $A[i] = B[j]$

(b) $C(A_{i-1}, B_j)$; if $A[i] = B[j]$
$C(A_i, B_{j-1}) + C(A_{i-1}, B_{j-1})$; if $A[i] = B[j]$

(c) $C(A_i, B_{j-1})$; if $A[i] = B[j]$
$C(A_i, B_{j-1}) + C(A_{i-1}, B_{j-1})$; if $A[i] \neq B[j]$

(d) None of the above

Multiple Select Questions (MSQ)

Q.39 Consider the following statements:

(a) Greedy algorithm exists
(b) Dynamic programming algorithm exists
(c) Exhibit optimal substructure property.
(d) Inhibit optimal substructure property.

Which of the above statements is/are true for 0-1 Knapsack problem?

Q.40 Which of the following statement(s) is/are correct?

(a) The number of moves required to solve Tower of Hanoi problem for k disks $2^k - 1$.
(b) The number of moves required to solve Tower of Hanoi problem for k disks $2^k + 1$
(c) The time complexity of balancing parentheses algorithm O(N log N).
(d) The time complexity of balancing parentheses algorithm O(N).

■■■■

Answers Dynamic Programming

1.	(d)	2.	(c)	3.	(a)	4.	(c)	5.	(11)	6.	(20)	7.	(b)	8.	(a)	9.	(1)
10.	(a)	11.	(d)	12.	(c)	13.	(b)	14.	(b)	15.	(132)	16.	(c)	17.	(d)	18.	(d)
19.	(a)	20.	(b)	21.	(b)	22.	(d)	23.	(b)	24.	(b)	25.	(d)	26.	(b)	27.	(1)
28.	(c)	29.	(b)	30.	(92)	31.	(10200)	32.	(b)	33.	(b)	34.	(b)	35.	(b)	36.	(b)
37.	(13)	38.	(a)	39.	(b, c)	40.	(a, d)										

Explanations Dynamic Programming

1. **(d)**
Longest common subsequence, longest increasing subsequence, sum of subsets, optimal BST, 0/1 Knapsack, matrix chain multiplication, Travelling salesperson, Balanced partition, Fibonacci sequence, Multistage graph problems are solved using dynamic programming.

2. **(c)**
Bellman Ford shortest path algorithm is example of dynamic programming.

3. **(a)**
Because both Greedy algo and dynamic algo finds the optimal solution for their problems but only dynamic programming uses bottom-up approach. Whereas, greedy algo uses top-bottom approach.
Hence, option (a) is correct.

4. **(c)**
Divide and conquer: Merge sort, Quick sort, Binary search, etc.
Greedy approach: Huffman codes, Dijkstra's, Kruskal's Algorithm, etc.
Dynamic programming: Longest common subsequence, All pairs shortest Paths, etc.
Backtracking: Chess game, 8-Queens, Checkers, etc.

5. **(11)**
LCS = *arm*
Only one subsequence
The answer will be 2 * 1 + 3 * 3 = 11

6. **(20)**
The 3 occurrences of "*cac*" are as shown:
bacacbaacacac
bacacbaacacac
bacacbaacacac
for the first occurrence,
$P[1....3] = cac = T[3....5]$
Now, $S + 1 = 3$
Thus, $S = 2$
Calculating S for the other 2 occurrences, we get $s = 8$ and $s = 10$.
So, sum of all values of S is 20.

7. **(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.

8. **(a)**
Because both Greedy algo and dynamic programming finds the optimal solution for their problem but, only dynamic programming uses bottom-up approach. Whereas, Greedy algorithm uses top-bottom approach.

9. **(1)**
x = csemadeeasy
y = gateexam
LCS (x, y) = aeea
Only "aeea" is longest common subsequence.

10. **(a)**
Bellman-Ford algorithm : $O(nm)$
Kruskal's algorithm : $O(m \log n)$
Floyd-Warshall algorithm : $O(n^3)$
Topological sorting : $O(n + m)$

11. (d)

Matrix chain multiplication relation,

for $1 \le i \le j \le n$, let $dp[i, j]$ denote minimum number of multiplication needed to compute Ai, j matrix.

$$dp[i, j] = \begin{cases} 0 & \text{if } i = j \\ \min\limits_{i \le k < j} (dp[i,k] + dp[k+1, j] + p_{i-1} * p_k * P_j & i < j \end{cases}$$

So, option (*d*) is correct.

12. (c)

$X = g\ a\ t\ e\ 1\ 2\ 3$

$Y = a\ g\ e\ t\ 3\ 2\ 1$

*gt*1, *gt*2, *gt*3, *at*1, *at*2, *at*3, *ge*1, *ge*2, *ge*3, *ae*1, *ae*2, *ae*3

Length of LCS = 3

Number of LCS's with length 3 are 12

13. (b)

bcba is one possible LCS of *X* and *Y*

LCS can be computed using dynamic programming.

14. (b)

W = 8 (capacity)

Feasible solutions:

(i) $\{I_1, I_3, I_4\}$, (ii) $\{I_2, I_3\}$

Profit of $\{I_1, I_3, I_4\}$ is 23

profit of $\{I_2, I_3\}$ is 15

Optimal solution is $\{I_1, I_3, I_4\}$ with capacity of 8 and maximum profit 23 produced.

$\therefore I_2$ is not selected in the solution.

15. (132)

Total number of ways to multiply n matrix = $(n-1)^{th}$ catalan

$$= \binom{2(n-1)}{(n-1)} \frac{}{n}$$

Put $n = 7$, Total ways = 132

16. (c)

	1	2	3	4
1	0	15750 [1]	7875 [1]	9375 [3]
2		0	2625 [2]	4375 [3]
3			0	750 [3]
4				0

$(A_1(A_2 A_3))(A_4) = 9375$.

17. (d)

In case of DP memorization can be done at the same time complexity as bottom-up method. It's first looking at the same problem in two ways.

For a Dynamic programming the complexity of solving the problem is given as (#subproblems * cost of solving one subproblem)

So, option (d) is current.

18. (d)

0/1 Knapsack problem uses dynamic programming and maintains 20 table for memorization.

Thus, O(*nw*)

Since, number of computations are $n \times w$.

Hence, option (d) is correct.

19. (a)

Finding closest pairs can be solved in O(n log$n$) time using divide and conquer approach.

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

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

Algorithm:

Divide and Conquer:

Step 1: Divide the set into two equal sized parts by the line l, and recursively compute the distance in each part. [d_1 = closest pair (left half); d_2 = closest pair (right half);] and returning the points in each set in sorted order by y-coordinate.

Merge Procedure:

[O(1)] Step 2: Let 'd' be the minimal of two minimal distances.

$$d = \min(d_1, d_2);$$

[O(n)] Step 3: Eliminate points that lie farther than 'd' apart from l.

[O(n)] Step 4: Merge the two sorted lists into one sorted list.

[O(n)] Step 5: Scan the remaining points in the y-order and compute the distances of each point to its 5 neighbors.

[O(1)] Step 6: If any of these distances is less than 'd' then update 'd'.

21. (b)

$p = 60, q = 82$

$p = (3 * 5 * 4) = 60$

$q = 70 + 2 \times 3 \times 2 = 82$

So, correct option is (*b*).

22. (d)
For the above graph, only one of the options is the Hamiltonian path exists which is only route for the optimal Hamiltonian path which is 1 - 4 - 5 - 3 - 2 - 1 or 1 - 2 - 3 - 4 - 5 - 4 - 1.

23. (b)
Initialize:
Till_max = 0;
Upto_max = 0;
Loop for each element of the array
(a) Upto_max = Upto_max + a[i];
(b) If(upto_max < 0)
Upto_max = 0;
(c) If(Till_max < Upto_max)
Till_max = Upto_max; return Till_max;
It takes O(n) time complexity and O(n) space complexity in the form of table.

24. (b)
$(P+\bar{Q}+\bar{R})\cdot(P+Q+R)\cdot(P+Q+\bar{R})$

R \ PQ	00	01	11	10
0	0	1	1	1
1	0	0	1	1

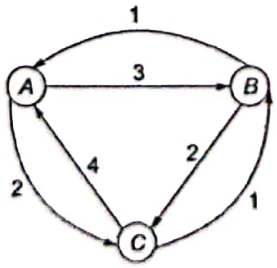
Hence, the expression is $(P+Q\bar{R})$.

25. (d)
D^i in the Floyd Warshall algorithm represents the shortest paths between all the vertices including the vertices labelled < = i.

26. (b)
The length of the longest common subsequence is 6 and it is p, o, t, a, u, s.

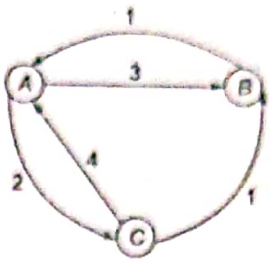
27. (1)

$$D = \begin{array}{c} \\ A \\ B \\ C \end{array}\begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{bmatrix} 0 & 3 & 2 \\ 1 & 0 & 2 \\ 4 & 1 & 0 \end{bmatrix} \end{array} \Rightarrow$$



$$D^* = \begin{array}{c} \\ A \\ B \\ C \end{array}\begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{bmatrix} 0 & 3 & 2 \\ 1 & 0 & 2 \\ 2 & 1 & 0 \end{bmatrix} \end{array}$$

After removal of (B, C) edge, the graph is:



$$\Rightarrow D_1 = \begin{array}{c} \\ A \\ B \\ C \end{array}\begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{bmatrix} 0 & 3 & 2 \\ 1 & 0 & \infty \\ 2 & 1 & 0 \end{bmatrix} \end{array} \Rightarrow D^*_1 = \begin{array}{c} \\ A \\ B \\ C \end{array}\begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{bmatrix} 0 & 3 & 1 \\ 1 & 0 & 3 \\ 2 & 1 & 0 \end{bmatrix} \end{array}$$

$\Rightarrow$ B to C shortest path distances are changed.

28. (c)
Using BFS by treating all edges as unweighted, it takes O($|V| + |E|$) time.

29. (b)
Since time complexity for solving TS using DP is approximately O($n^2 2^n$) and 0/1 Knapsack using DP is O(nW) and Floyd Warshall is O(n^3), therefore f grows faster than g and g grows faster than h, the last option is most appropriate.
Hence, option (b) is correct.

30. (92)
The recurrence relation for getting the minimum number of matrix multiplication is:

$$M[i, j] = \begin{cases} 0 & \text{if } i = j \\ \min_{i \le k < j}\{M[i, j] = M[i, k] + M[k+1, j] + p_{i-1} p_k p_j\} \end{cases}$$

$M[1, 1] = 0$
$M[2, 2] = 0$
$M[3, 3] = 0$
$M[4, 4] = 0$
$M[1, 2] = 2 * 3 * 4 = 24$
$M[2, 3] = 3 * 4 * 7 = 84$
$M[3, 4] = 4 * 7 * 2 = 56$
$M[1, 3] = \text{Min}((M[1, 1] + M[2, 3] + (2 * 3 * 7)), (M[1, 2] + M[2, 2] + (2 * 4 * 7)))$
$= \text{Min}(84 + 42, 80) = 80$
$M[2, 4] = \text{Min}((M[2, 2] + M[3, 4] + (3 * 4 * 2)), (M[2, 3] + M[4, 4] + (3 * 7 * 2)))$
$= \text{Min}(80, 84 + 42) = 80$
$M[1, 4] = \text{Min}(M[1, 1] + M[2, 4] + (2 * 3 * 2)) (M[1, 2] + M[3, 4] + (2 * 4 * 2)), (M[1, 3] + M[4, 4] + (2 * 7 * 2)))$

$= \text{Min}(80 + 12, 24 + 56 + 16\,80 + 28$
$= 92$

31. (10200)

Multiply is at $(M_1(M_2\,M_3)M_4)$

$M_2M_3 \cos t = 100$

$M_1(M_2\,M_3) = 100 + 100 = 200$

$(M_1(M_2\,M_3)M_4) = 200 + 10000 = 10200$

32. (b)

X and Y have LCS's of length 4. The following are longest common subsequences of X and Y.

$<B, C, B, A>$

$<B, D, A, B>$

Here (b) is correct option.

33. (b)

Time complexity of LCS = $O(mn)$

LCS computed as following and trace back later.

$$c[i,j] = \begin{cases} 0 & , \text{ if } i=0 \text{ or } j=0 \\ c[i-1,j-1]+1 & , \text{ if } x_i = y_j, i, j>0 \\ \max c[i,j-1], c[i-1,j] & , \text{ if } x_i \neq y_j, i, j>0 \end{cases}$$

34. (b)

$(X_1(X_2.X_3))(X_4.X_5)$

Number of multiplications = 102, which is minimum compared to other options.

$X_2.X_3 \Rightarrow$ # Multiplications = 42
$X_4.X_5 \Rightarrow$ # Multiplications = 20
$X_1.(X_2.X_3) \Rightarrow$ # Multiplications = 30
$(X_1.(X_2.X_3))_{5\times1}(X_4.X_5)_{1\times2} \Rightarrow$ # Multiplications = 10
$\Rightarrow$ 102 Multiplications

35. (b)

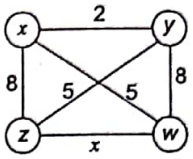
No greedy algorithm exists for 0-1 Knapsack problem. 0-1 Knapsack problem exhibit optimal substructure property and it has only dynamic programming algorithm.

36. (b)

$$\begin{cases} l[i,j], & \text{if } k=0 \\ \min\left\{d_{i,j}^{k-1}, d_{i,k}^{k-1} + d_{k,j}^{k-1}\right\}, & \text{if } 1 \le k \le n \end{cases}$$

$\therefore$ Option (b) is correct.

37. (13)



On applying Floyd Warshall,
as we known $z - w$ in $D_0 = x$.
The only way to keep $x \forall$ is 13.
So, 13 is the answer.

38. (a)

$$C(A_i, B_j) = \begin{cases} 1; & \text{if } (i=0) \\ 0; & \text{if } (i>0 \text{ and } j=0) \\ C(A_i, B_{j-1}); & \text{if } (A[i] \neq B[i]) \text{ when elements are not matched.} \\ C(A_i, B_{j-1}) + C(A_{i-1}, B_{j-1}); & \text{if } (A[i] = B[i]) \text{ when elements are matched.} \end{cases}$$

39. (b, c)

No greedy algorithm exists for 0-1 Knapsack problem. 0-1 Knapsack problem exhibit optimal substructure property and it has only dynamic programming algorithm.

40. (a, d)

Tracing of the moves in the above ToH problem will prove this result, instead you can simply add a count for each recursive call to check the number of moves.

The time complexity of balancing parentheses algorithm is mathematically found to be O(N).

■■■■