Chapter 1

Algorithm Analysis and Asymptotic Notations

One-mark Questions

1. Which one of the following is the tightest upper bound that represents the number of swaps required to sort n numbers using selection sort? **[2013]**

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

Solution: (b)
$n - 1$ passes are required for selection sort, and in each pass there is one swap so the number of swaps (passes) = $O(n)$
Hence, the correct option is (b).

2. 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? **[2013]**

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

Solution: (c)
To insert an element we compare all elements, so $O(n)$.
Hence, the correct option is (b).

3. What is the time complexity of Bellman-Ford single-source shortest path algorithm on a complete graph of n vertices? **[2013]**

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

Solution: (c)
Time complexity of Bellman-Ford algorithm on a graph with n vertices and m edges is $O(nm)$.
For a complete graph, $m - n_{c_2} = O(n^2)$. As there is an edge between all pair of vertices. Hence, Time complexity = $O(n^2.n) = O(n^3)$
Hence, the correct option is (c).

4. The worst case running time to search for an element in a balanced binary search tree with $n2^n$ elements is **[2012]**

(a) $\Theta(n \log n)$ (b) $\Theta(n2^n)$
(c) $\Theta(n)$ (d) $\Theta(\log n)$

Solution: (c)

$$\log_2 (n\, 2^n) = \log_2 n + \log_2 (2^n)$$
$$= \log_2 n + n$$
$$= \theta (n)$$

Hence, the correct option is (c).

5. Let $W(n)$ and $A(n)$ denote respectively, the worst case and average case running time of an algorithm executed on an input of size n. Which of the following is ALWAYS TRUE? **[2012]**

(a) $A(n) = \Omega (W(n))$
(b) $A(n) = O(W(n))$
(c) $A(n) = \Theta(W(n))$
(d) $A(n) = 0(W(n))$

Solution: (c)
Average case running time is always less than (or) equal to worst case time. Hence, $A(n) = O(w(n))$. According to asymptotic notations, average lies always between the best and the worst case inclusive. Hence, the correct option is (c).

6. Two alternative packages A and B are available for processing a database having 10^k records. Package A requires $0.0001n^2$ time units and package B requires $10n.\log^n_{10}$ time units to process n records. What is the smallest value of k for which package B will be preferred over A? **[2010]**
(a) 12 (b) 10 (c) 6 (d) 5

Solution: (c)
A takes $0.0001\ n^2 = 10^{-}4\ n^2$
B takes $10\ n\ \log_{10} n$ units of time for $k = 6$
A takes $10^{-}4 \times (10^6)^2 = 10^8$
B takes $10 \times 10^6\ \log_{10} 10^6 = 6 \times 10^7$
Hence, $A > B$.
Hence, the correct option is (c).

7. 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: **[2007]**
(a) $[\log_2 n] + 1$ (b) n
(c) $[\log_2 n]$ (d) $[\log_2 n] + 1$

Solution: (d)
As 'j' increases in powers of 2; i.e. 2, 4, 8… in the loop $2^k \le n$
When $2^k > n$ loop exits, so no. of comparisons made in execution of loop for any $n > 0$ is $[\log_2 n] + 1$
Hence, the correct option is (d).

8. Consider the following C-program fragment in which i, j and n are integer variables for($i = n$, $j = 0$; $I > 0$; $i/ = 2$, j+= i);
Let value (j) denote the value stored in the variable j after termination of the *for* loop. Which one of the following is true? **[2006]**
(a) val (j) = $\theta(\log n)$
(b) val (j) = $\theta(\sqrt{n})$
(c) val (j) = $\theta(n)$
(d) val (j) = $\theta(n \log n)$

Solution: (d)

$$j = \frac{n}{1} + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \ldots + 1$$

$$\Rightarrow n\left(\frac{1}{1} + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \ldots + \frac{1}{n}\right)$$

$\Rightarrow O(n)$ [$\because$ Polynomial of degree 1]
Hence, the correct option is (d).

9. The time complexity of computing the transitive closure of a binary relation on a set of n elements is known to be: **[2005]**
(a) $O(n)$ (b) $O(n \log n)$
(c) $O(n^{3/2})$ (d) $O(n^3)$

Solution: (d)
All-pairs shortest paths can be found with the help of the transitive closure.
Hence, the correct option is (d).

10. Consider an array multiplier for multiplying two n bit numbers. If each gate in the circuit has a unit delay, the total delay of the multiplier is **[2003]**
(a) $\Theta(1)$ (b) $\Theta(\log n)$
(c) $\Theta(n)$ (d) $\Theta(n^2)$

Solution: (d)
Multiplying two n-bit binary no's of size 'n' and 'm' is $O(nm)$.
Hence, the correct option is (d).

11. Consider the following three claims **[2003]**
I. $(n + k)^m = \Theta(n^m)$ where k and m are constants
II. $2^{n+1} = O(2^n)$
III. $2^{2n} = O(2^n)$
Which of these claims are correct?
(a) I and II
(b) I and III
(c) II and III
(d) I, II, and III

Solution: (a)
$(n + k)^m$ is a polynomial of power m
$\therefore O(n^m)$ is True
$2^{n+1} = O(2^n.2)$
$= O(2^n)$
Hence, it is true
$2^{2n} = O(2^n 2)$
$\neq O(2^n)$
Therefore, I and II only are correct
Hence, the correct option is (a).

12. In the worst case, the number of comparisons needed to search a singly linked list of length n for a given element is **[2002]**

(a) $\log n$ (b) $\frac{n}{2}$

(c) $\log_2 n - 1$ (d) n

Solution: (d)

Linked list can always use linear search and hence $O(n)$.

In the worst case, the comparison is done between the element to be searched and all elements of linked list.

Hence, the correct option is (d).

13. Let $f(n) = n^2 \log n$ and $g(n) = n(\log n)^{10}$ be two positive functions of n. Which of the following statements is correct? **[2001]**

(a) $f(n) = O(g(n)$ and $g(n) \neq O(f(n))$

(b) $g(n) = O(f(n)$ and $f(n) \neq O(g(n))$

(c) $f(n) \neq O(g(n))$ and $g(n) \neq O(f(n))$

(d) $f(n) \neq O(g(n))$ and $g(n) \neq O(f(n))$

Solution: (b)

The term $n \log n$ is common between $f(n)$ and $g(n)$ is; therefore the term n of $f(n)$ and $(\log n)^9$ of $g(n)$ are compared; it is true asymptotically that $(\log n)^9$ is $O(n)$; therefore $g(n)$ is $O(f(n))$

Hence, the correct option is (b).

14. The maximum gate delay for any output to appear in an array multiplier for multiplying two n bit number is **[1999]**

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

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

Solution: (a)

Multiplying two n-bit binary no's of size 'n' and 'm' is $O(nm)$.

Hence, the correct option is (a).

15. The concatenation of two lists is to be performed on 0(1) time. Which of the following implementations of a list should be used? **[1997]**

(a) Singly linked list

(b) Doubly linked list

(c) Circular doubly linked list

(d) Array implementation of list

Solution: (c)

While all other types of linked lists need to reach till end to attach the second list but the traversal for concatenation is not done by circular linked lists.

Hence, the correct option is (c).

16. Which of the following is false? **[1996]**

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

(b) $\sqrt{\log n} = O(\log n)$

(c) If $0 < x < y$ then $n^x = O(n^y)$

(d) $2^n \neq O(n^k)$

Solution: (b)

(a) $100\, n \log n = O(n \log n)$

$$= O\left[\frac{n \log n}{100}\right] \quad (\because 100 \text{ is constant})$$

$$\sqrt{\log n} = \log^{1/2} n \rightarrow (\text{True})$$

$$= \frac{1}{2} \log n$$

$$= O(\log n)$$

Which is not $O(\log n)$

(b) $0 <' x < y$

Let $x = 3$, $y = 5$

$n^3 = 0\ (n^5)$

lower = O (higher)

(c) $2^n \neq 0\ (n^k)$

Here, 2^n is higher growth rate function there n^k

Hence True.

Hence, the correct option is (b).

Two-marks Questions

1. The number of elements that can be sorted in $\Theta(\log n)$ time using heap sort is **[2013]**

(a) $\Theta(1)$ (b) $\Theta\left(\sqrt{\log n}\right)$

(c) $\Theta\left(\frac{\log n}{\log \log n}\right)$ (d) $\Theta(\log n)$

Solution: (c)

Suppose the number of element is 'K'. They can be sorted in $\Theta(k \log k)$ time.

By trying the options in decreasing order of complexity as we need a tight band i.e. Θ

$$\text{i.e } \Theta(\log n), \Theta\left(\frac{\log n}{\log \log n}\right), \Theta\left(\sqrt{\log n}\right), \Theta \quad (1)$$

So if $K \in \Theta(\log n)$ time required for heap sort is. $\Theta\ (k \log k)$ i.e. $\Theta\ (\log n \times \log \log n)$, But this is not in $\Theta\ (\log n)$.

Time required for heap sort

$$\text{if } k \in \Theta\left[\frac{\log n}{\log\log n}\right]$$

$$\Theta\left[\frac{\log n}{\log\log n} \times \log\left[\frac{\log n}{\log\log n}\right]\right]$$

$$\text{i.e}\,\Theta\left(\log n \times \left[\frac{\log\left(\frac{\log n}{\log\log n}\right)}{\log\log n}\right]\right),$$

$$\left[\frac{\log\left[\frac{\log n}{\log\log n}\right]}{\log\log n}\right] \le 1$$

So, this is in $\Theta(\log n)$
Hence, the correct option is (c).

2. Which of the given options provides the increasing order of asymptotic complexity of functions f_1, f_2, f_3 and f_4?

$f_1(n) = 2^n$
$f_2(n) = n^{3/2}$
$f_3(n) = n\log_2^n$
$f_4(n) = n^{\log 2n}$ **[2011]**

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

Solution: (a)
With $n = 1024$ the values of given functions are $f_1(1024) = 2^{1024}, f_2(1024) = 1024^{3/2}, f_3(1024) = 1024^{\log_2^{1024}}, f_4(1024)$

$= 1024^{\log_2^{1024}} = (1024)^{10}$
$n\log_2 n < n^{3/2} < n^{\log_2 n} < 2^n$
$f_3(n) < f_2(n) < f_4(n) < f_1(n)$
Hence, the correct option is (a).

3. 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? **[2008]**

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

Solution: (d)
$f(n) = 2^n$
$f(n) = O(2^n)$
$g(n) = n! - g(n) = O(n!)$
$h(n) = n^{\log n} - h(n) = O(n^{\log n})$
$n^{\log n} < c^n < n!$
$n^{\log n} < c2^n < n!$
$h(n) = O(f(n)), g(n) = \Omega f(n)$
Hence, the correct option is (d).

4. The minimum number of comparisons required to determine if an integer appears more than $n/2$ times in a sorted array of n integers is **[2008]**

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

Solution: (a)
By initializing the counter to zero and checking linearly in a loop.
Hence, the correct option is (a).

5. You are given the post order traversal, P, of a binary search tree on the n element, 1, 2,…,n. You have to determine the unique binary search tree that has P as its post order traversal. What is the time complexity of the most efficient algorithm for doing this? **[2008]**

(a) $\Theta(\log n)$
(b) $\Theta(n)$
(c) $\Theta(n \log n)$
(d) None of the above, as the tree cannot be uniquely determined

Solution: (c)
Unique tree can be constructed with in order along with either pre-order or post-order traversal. Since we know in order and post order of BST, so tree can be constructed using $O(n\log n)$ time.
Hence, the correct option is (c).

6. Consider the following C functions:

```
int fl (int n)
{
if (n = = 0|| n = = 1)
return n;
```

```
else
return (2 * fl( n – 1)+ 3 * fl(n – 2));
}
int f2 (int n)
{
int i;
int X[N], Y[N], Z[N];
X [0] = Y[0] = Z[0] = 0;
X[1] = 1; Y[1] = 2; Z[1] = 3;
for(I =2; I <= n; i++){
X[i] = Y[I – l] + Z[I – 2];
Y[i] = 2 * X [i];
Z[i] = 3*X[i];
}
return X[n];
}
```

The running time of $f1\ (n)$ and $f2\ (n)$ are **[2008]**

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

Solution: (b)

The function $f2\ ()$ repeats for n-times, therefore $O(n)$; whereas the recurrence for $f1\ ()$ is derived as

$T(n) = l,\ n = 1$

$T(n) = 2\mathrm{T}\ (n-1) + 3T(n-2) + \mathrm{c},\ n > l$

Hence, the correct option is (b).

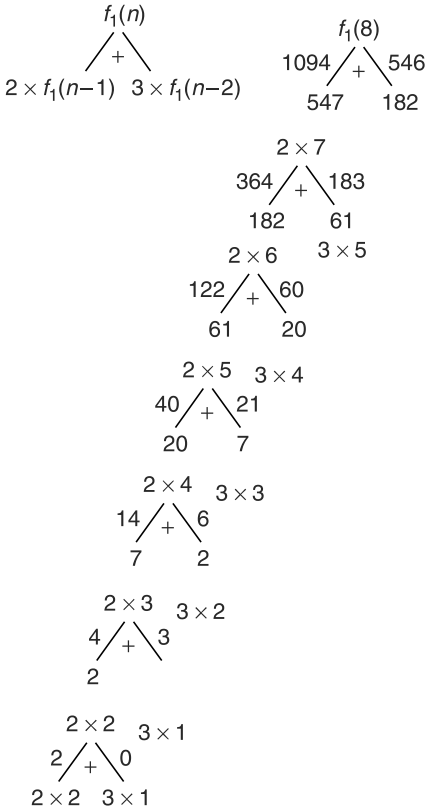
7. Consider the following C functions:

```
int fl (int n)
{
if (n = = 0|| n = = 1)
return n;
else
return (2 * fl( n - 1)+ 3 * fl(n - 2));
}
int f2 (int n)
{
int i;
int X[N], Y[N], Z[N];
X [0] = Y[0] = Z[0] = 0;
X[1] = 1; Y[1] = 2; Z[1] = 3;
for(I = 2; I <= n; i++){
X[i] = Y[I - l] + Z[I - 2];
Y[i] = 2 * X [i];
Z[i] = 3*X[i];
}
return X[n];
}
```

$f_1(8)$ and $f_2(8)$ return the values

(a) 1661 and 1640
(b) 59 and 59
(c) 1640 and 1640
(d) 1640 and 1661

Solution: (c)



$1094 + 546 = 1640$

1. $X[i]$	$Y[i]$	$Z[i]$
2. $X[2] + Z[0]$	$Y[2] = 2 \times [2] = 4$	$Z[2] = 3 \times 2 = 6$
3. $X[3] = Y[2] + 2[1] = 4 + 3 = 7$	$Y[3] = 2 \times [3] = 2 \times (7) = 14$	$Z[3] = 3 \times 7 = 21$
4. $X[4] = Y[3] + Z[2] = 14 + 6 = 20$	$Y[4] = 2 \times [4] = 2 \times [20] = 40$	$Z[4] = 3 \times 20 = 60$
5. $X[5] = Y[4] + Z[3] = 40 + 21 = 61$	$Y[5] = 2 \times [61] = 122$	$Z[5] = 3 \times 61 = 183$
6. $X[6] = Y[5] + 2[4] = 122 + 60$	$Y[6] = 2 \times 182 = 364$	$Z[6] = 3 \times 182 = 546$
7. $X[7] - Y[6] + Z[5] = 364 + 183 = 547$	$Y[7] = 2 \times 547 = 1094$	$Z[7] = 3 \times 547 = 1641$
8. $X[8] = Y[7] + Z[6] = 1094 + 546 = 1640$	$Y[8] = 2 \times 1640 = 3280$	$Z[8] = 3 \times 1640 = 4920$

Hence, the correct option is (b).

8. What is the time complexity of the following recursive function? **[2007]**

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

(a) $\Theta 6\ (n^2)$ (b) $\Theta(n \log_2 n)$
(c) $\Theta(\log_2 n)$ (d) $\Theta(\log_2 \log_2 n)$

Solution: (d)

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

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

By continuing this process up to 'K^{th}' iteration, then

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

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

$$\frac{1}{2^k} = \log_n 2$$

$$2^k = \log_2 n$$

$$K = \log_2 \log_2 n$$

Hence, the correct option is (d).

9. Consider the following C code segment:; int Is Prime(n)

```
{
int i, n;
for(i=2; i<=sqrt (n); i++)
if(n%i = = 0)
printf("Not Prime\n"); return 0
}
return i:
}
```

Let $T(n)$ denote the number of times the for loop is executed by the program on input n. Which of the following is TRUE? **[2007]**

(a) $T(n) = O(\sqrt{n})$ and $T(n) = \Omega(\sqrt{n})$
(b) $T(n) = O(\sqrt{n})$ and $T(n) = \Omega(1)$
(c) $T(n) = O(n)$ and $T(n) = \Omega(\sqrt{n})$
(d) None of the above

Solution: (b)

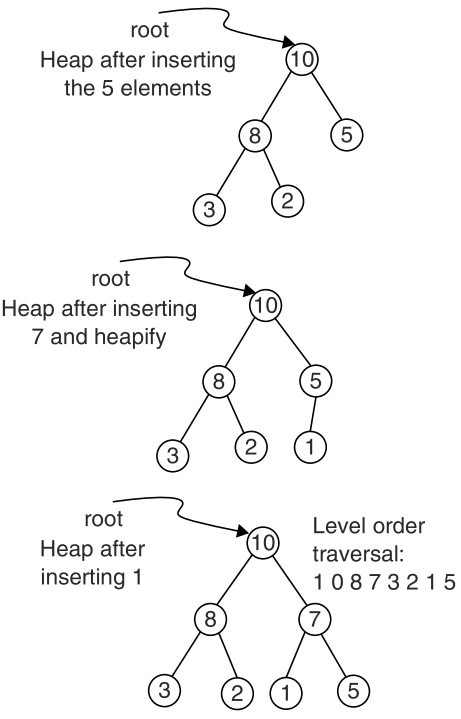
The upper bound cannot be more than $\sqrt{n}$ and lower bound will be $\Omega(1)$ when the loop [terminates with the condition $n \% i = 0$ for the first time.

Big O notation describes the tight upper bound and Big Omega notation describes the tight lower bound for a algorithm. The for loop in the question is run maximum $\sqrt{n}$ times and minimum 1 time. Therefore, $T(n) = O(\sqrt{n})$ and $T(n) = \Omega(l)$.

10. A Priority-Queue is implemented as a Max-Heap. Initially, it has 5 elements. The level-order traversal of the heap is given below: 10, 8, 5, 3, 2
Two new elements '1' and '7' are inserted in the heap in that order. The level order traversal of the heap after the insertion of the elements is: **[2005]**

(a) 10, 8, 7, 5, 3, 2, 1
(b) 10, 8, 7, 2, 3, 1, 5
(c) 10, 8, 7, 1, 2, 3, 5
(d) 10, 8, 7, 3, 2, 1, 5

Solution: (d)



Hence, the correct option is (d).

11. Consider the following C-function:

```
double foo (int n) {
int i;
double sum;
```

```
if (n = = 0) return 1.0;
else {
sum = 0.0;
for (i = 0; I < n; i++)
sum + = foo(i);
return sum;
}
}
```

The space complexity of the above function is:

[2005]

(a) $O(1)$ (b) $O(n)$ (c) $O(n!)$ (d) $O(n^n)$

Solution: (b)

```
double foo (int n)
{
int i;
double sum;
if(n=0)
return;
else
sum = 0;
for (i = 0; i < n; i++)
{
sum = sum + foo (i);
return(sum);
}
}
```

The maximum size of recursion stack space is 'n' Hence, the correct option is (b).

12. Consider the following C-function:

```
double foo (int n) {
int i;
double sum;
if (n = = 0) return 1.0;
else {
sum = 0.0;
for (i = 0; I < n; i++)
sum + = foo(i);
return sum;
}
}
```

Suppose we modify the above function foo( ) and store the values of foo(i), $0 \le i \le n$, as and when they are computed. With this modification, the time complexity for function foo( ) is significantly reduced. The space complexity of the modified function would be: **[2005]**

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

Solution: (b)

An array of size 'n' is required to store temporary results. So solution is $O(n)$.

Hence, the correct option is (b).

13. What does the following algorithm approximate?

```
(Assume m > 1, ∈ > 0)
x = m;
y = 1;
while (x ∈ y > ∈)
{
x = (x + y)/2;
y = m/x;
}
print (x);
```

[2004]

(a) $\log m$ (b) m^2
(c) $m^{1/2}$ (d) $m^{1/3}$

Solution: (c)

The loop will fail for $x - y \le y$

So $x \le y$

$$x \le \frac{m}{x}\left(\because y = \frac{m}{x}\right)$$

$x^2 \le m$

$x \le m^{1/2}$

Hence, the correct option is (c).

14. Let $A[1,...,n]$ be an array storing a bit (1 or 0) at each location, and $f(m)$ is a function whose time complexity is $0(m)$. Consider the following program fragment written in a C like language: counter = 0;

```
for(i=l; I < = n; i++){
if(A[i] = = l)counter++;
else {
f(counter); counter = 0;
}}
```

The complexity of this program fragment is

[2004]

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

Solution: (c)

In all cases, when $A[i] = l$ for all $i = l, n$

$A[i] = 0$; for all $i = l, n$

$A[i] = l$; for $i = l, n/2$

$= 0$; for $i = \frac{n}{2} + 1, \ldots n$

The order of magnitude is $O(n)$

Hence, the correct option is (c).

15. The time complexity of the following C-function is (assume $n > 0$)

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

[2004]

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

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

Solution: (d)

The recurrence for the given problem is

$T(n) = l, n = l$

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

According to back-substitution, it is $O(2^n)$

Hence, the correct option is (d).

16. The recurrence equation

$T(l) = l$

$T(n) = 2T(n-l) + n, n > 2$

Evaluates to **[2004]**

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

(b) $2^n - n$

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

(d) $2^n + n$

Solution: (c)

According to back-substitution.

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

$A_1 = \frac{2}{3},\ 3A_0 = l,\ A_0 = 13$

$n = 1\ A_0 + A_1 = 1 \Rightarrow A_0 + A_1 = 1$

$n = 0 - [2A_0\ (-l) + A_1] = 0$

$2A_0 - A_1 = 0$

$a_n - 2a_n - l = n$

$(A_0 + A_1\ n) - 2\ [A_0 + A_1\ (n-1)] = n$

$n = l \Rightarrow A_0 + A_1 - 2A_0 = l$

$n = 0 \Rightarrow A_0 - 2A_0 - 2A_1\ (-l) = 0$

$A_1 - A_0 = l - A_1 + A_0 = -l$

$-A_0 + 2 + 1 = 0$

$2A_1 - A_0 = 0$

$A_1 = -l$

$A_0 = A_1 - l = \boldsymbol{l - 1 = -2}$

$a_n = 2^{n+1} - n - 2$

$c \cdot 2 - 2 - 1 = 1 \Rightarrow 2c = 4\ \therefore\ c = 2$

Hence, the correct option is (c).

17. The cube root of a natural number n is defined as the largest natural number m such that $m^3 \le n$. The complexity of computing the cube root of n (n is represented in binary notation) is **[2003]**

(a) $O(n)$ but not $O(n^{0.5})$

(b) $O(n^{0.5})$ but not $O((\log n)^k)$ for any constant $k > 0$

(c) $O((\log n)^k)$ for some constant $k > 0$, but not $O((\log n)^m)$ for any constant $m > 0$

(d) $O((\log n)^k)$ for some constant $k > 0$ 5, but not $O((\log n)^{0.5})$

Solution: (c)

18. Consider the following algorithm for searching for a given number x in an unsorted array $A[1 \ldots n]$. having n distinct values:

1. Choose an i uniformly at random from $l \ldots n$;
2. If $A[i] = x$ then Stop else Goto 1;

Assuming that x is present A, what is the expected number of comparisons made by the algorithm before it terminates? **[2002]**

(a) n (b) $n - 1$

(c) $2n$ (d) $\frac{n}{2}$

Solution: (a)

The maximum number of comparisons can be 'n'.

Hence, the correct option is (a).

19. The running time of the following algorithm

Procedure $A(n)$

If $n <= 2$ return (1) else return

$\left(A\left(\sqrt{n}\right)\right)$;

Is best described by **[2002]**

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

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

Solution: (c)

$T(n) = l,\ n \le 2$

$T(n) = T\left(\sqrt{n}\right) + K,\ n > 2$

$n^{2^k} = 2$

$2^{-K} = \log_n 2$

$2^K = \log_2 n \Rightarrow K = \log_2 n$

Hence, the correct option is (c).

20. Consider the following functions

$f(n) = 3n^{\sqrt{n}}$

$g(n) = 2^{\sqrt{n}\log_2 n}$

$h(n) = n!$

Which of the following is true? **[2000]**

(a) $h(n)$ is $O(f(n))$
(b) $h(n)$ is $O(g(n))$
(c) $g(n)$ is not $O(f(n))$
(d) $f(n)$ is $O(g(n))$

Solution: (d)

$n = 256, f(n) = 3 \times 2^{128}$

$g(n) = 2^{16} \times 2^3 = 2^{19}$ $h(n) = 256!$

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

Hence, the correct option is (d).

21. Consider the following two functions:

$$g_1(n) = \begin{cases} n^3 \text{ for } 0 \le n < 10,000 \\ n^2 \text{ for } n \ge 10,000 \end{cases},$$

$$g_2(n) = \begin{cases} n \text{ for } 0 \le n < 100 \\ n^3 \text{ for } n \ge 100 \end{cases}$$

Which of the following is true: **[1994]**

(a) $g_1(n)$ is $O(g_2(n))$
(b) $g_1(n)$ is $O(n^3)$
(c) $g_2(n)$ is $O(g_1(n))$
(d) $g_2(n)$ is $O(n)$

Solution: (a)

$g_1(n)$ is $O(g_2(n))$ whenever $n > 100$

Hence, the correct option is (a).

22. $\sum_{1 \le k \le n} O(n)$, where $O(n)$ stands for order n is: **[1993]**

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

Solution: (b)

It is like a nesting of loop as given

for $k \leftarrow 1$ to n

for $i \leftarrow 1$ to n

S;

For each value of 'K', 'S' executed 'n' times, so total time is $O(n^2)$

Hence, S is $O(n^2)$

(or)

$$\sum_{1 \le k \le n} o(n) = O(n) \sum_{1 \le k \le n} i = O(n^2)$$

Hence, the correct option is (b).

23. Express $T(n)$ in terms of harmonic number $H_n = \sum_{t=1}^{n} 1/i, n \ge 1$, where $T(n)$ satisfies the recurrence relation,

$$T(n) = \frac{n+1}{2} T(n-1) + 1, \text{ for } n \ge 2 \text{ and } T(l) = 1$$

What is the asymptotic behaviour of $T(n)$ as a function of n? **[1990]**

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

Back-substitution method can be used to find the values of $T(n-1)$, $T(n-2)$ using equation (1) and eliminate the recurrence using $T(l) = 1$. The R.H.S of the recurrence will be in the harmonic series form, $\sum_{i=1}^{n} \frac{1}{i}, (i+1) + n$

24. Solve the recurrence equations **[1987]**

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

$T(l) = 1$

$T(n) = O(n^2)$

$T(n) = n + (n-1) + (n-2) + \ldots + T(1)$

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

(or)

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

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

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

25. What is the generating function $G(z)$ for the sequence of Fibonacci numbers? **[1987]**

The recurrence of Fibonacci numbers is

$F_n = F_{n-1} + F_{n-2}$

$$F_n = \left(\frac{\sqrt{5}+1}{2\sqrt{5}}\right)\left(\frac{1+\sqrt{5}}{2}\right) + \left(\frac{\sqrt{5}-1}{2\sqrt{5}}\right)\left(\frac{1-\sqrt{5}}{2}\right)$$

FIVE-MARKS QUESTION

1. Consider the following algorithms. Assume, procedure A and procedure B take $O(1)$ and $O\frac{1}{n}$ unit of time respectively. Derive the time complexity of the algorithm in O - notation. **[1999]**

```
algorithm what (n)
begin
if n = 1 then call A
else begin
what(n – 1);
call B(n)
end
end
```

Solution:

Let $T(n)$ denote the timing complexity of procedure what (n); $T(n) = 1$ $n = 1$

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

Expanding it using back substitution

$T(n) = O(n)$

Chapter 2

Divide and Conquer

One-mark Questions

1. Let P be quicksort programme to sort numbers in ascending order using the first element as the pivot. Let t_1 and t_2 be the number of comparisons made by P for the inputs [1 2 3 4 5] and [4 1 5 3 2] respectively. Which one of the following holds? **[2014]**

(a) $t_1 = 5$
(b) $t_1 < t_2$
(c) $t_1 > t_2$
(d) $t_1 = t_2$

Solution: (c)

Case (i): If the array elements are in the increasing order then the quick sort takes number of comparisons are

$$t_1 = \frac{n(n-1)}{2}$$

Case (ii): If the array elements are randomly distributed then the quick sort takes number of comparison are

$$t_2 = 2(n+1)(\log_e^n + 0.577) - 4n$$

$$\therefore\ t_1 > t_2$$

Hence, the correct option is (c).

2. Which one of the following correctly determines the solution of the recurrence relation with $T(1) = 1$? **[2014]**

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

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

Solution: (a)

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

By using Master Theorem, $a = 2$, $b = 2$, $k = 0$, p = l
As $a > b^k$, so it is case (i) of Master Theorem

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

Hence, the correct option is (a).

3. The minimum number of arithmetic operations required to evaluate the polynomial $P(X) = X^5 + 4X^3 + 6X + 5$ for a given value of X, using only one temporary variable is _____. **[2014]**

Solution: (7)
By using Divide and Conquer approach
$a^n = a$ if $n = 1$
$= a \,.\, a^{n-1}$ if n is odd
$= (a^{n/2})^2$ if n is even
$\therefore\ P(x) = x^5 + 4x^3 + 6x + 5$
$= x(x^4) + 4(x)(x^2) + 6(x) + 5$
$= x(x^2)^2 + 4\ (x)\ (x)\ (x) + 6(x) + 5$
$= x(x((x)))^2 + 4(x(x(x))) + 6(x) + 5$
Hence, total 7 multiplications are required.

4. You have an array of n elements. Suppose you implement quicksort by always choosing the

central element of the array as the pivot. Then the tightest upper bound for the worst case performance is **[2014]**

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

Solution: (a)

In the worst case the selected pivot element will be placed in either first (or) last position. Then the required recurrence equation is

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

By using substitution method

$$T(n) = O(n^2)$$

Hence, the correct option is (a).

5. 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. **[2009]**

(a) *P* only
(b) *Q* only
(c) both *P* and *Q*
(d) Neither *P* nor *Q*

Solution: (d)

Bellman-Ford works only if edges have negative weights but not having negative weight cycles.

Hence, the correct option is (d).

6. 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 any. If, instead, we use binary search to identify the position, the worst case running time will **[2003]**

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

Solution: (c)

As binary search takes time of $O(\log n)$ for a set of n-elements. So, total time for n-elements is $O(n \log n)$

Hence, the correct option is (c).

7. The solution to the recurrence equation $T(2^k) = 3T(2^{k-1}) + l$, $T(l) - 1$ is **[2002]**

(a) 2^k
(b) $(3^{k+1} - 1)/2$
(c) $3^{\log_2 k}$
(d) $2^{\log_3 k}$

Solution: (b)

Using back substitution,

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

Let $n = 2^K$

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

(or)

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

$$a_K = 3a_{K-1} + l$$

$$(E-3)a_K = l$$

$$a_K = C_1 3^K$$

$$ak = c \cdot 3^k \quad ak = \frac{1}{1-3} = \frac{-1}{2}$$

$$ak = c \cdot 3^k - \frac{1}{2}a_o = c - \frac{1}{2} = 1 \Rightarrow c = \frac{3}{2}$$

$$ak = \frac{3}{2} \cdot 3^k - \frac{1}{2} = \left(\frac{3^{k+1} - 1}{2}\right)$$

8. Randomised quick sort is an extension of quick sort where the pivot is chosen randomly. What is the worst case complexity of sorting n numbers using randomised quick sort? **[2001]**

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

Solution: (b)

9. Let s be sorted array of n integers. Let $t(n)$ denote the time taken for the most efficient algorithm to determine if there are two elements with sum less than 1000 in s. Which of the following statements is true? **[2000]**

(a) $t(n)$ is $O(1)$
(b) $n < t(n) < n \log_2^n$
(c) $n \log_2^n < t(n) < \binom{n}{2}$
(d) $t(n) = \binom{n}{2}$

Solution: (a)

It is enough to check the first two elements of the array as the given array is already sorted.

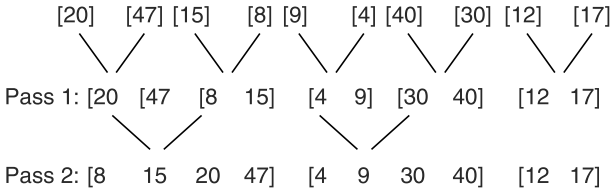
Hence, the correct option is (a).

10. A sorting technique is called stable if **[1999]**
(a) it takes $O(n \log n)$ time
(b) it maintains the relative order of occurrence of non-distinct elements
(c) it uses divide and conquer paradigm
(d) it takes $O(n)$ space

Solution: (b)

11. 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 second pass of the algorithm is: **[1999]**
(a) 8, 9, 15, 20, 47, 4, 12, 17, 30, 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

Solution: (b)



Hence, the correct option is (b).

12. Merge sort uses **[1995]**
(a) Divide and conquer strategy
(b) Backtracking approach
(c) Heuristic search
(d) Greedy approach

Solution: (a)
In merge sort algorithm, it uses divide and conquer strategy because for every iteration, array is divided into two equal parts.
Hence, the correct option is (a).

13. For merging two sorted lists of sizes m and n into a sorted list of size $m + n$, we require comparisons of **[1995]**
(a) $O(m)$
(b) $O(n)$
(c) $O(m + n)$
(d) $O(\log m + \log n)$

Solution: (c)
The complexity depends on the number of comparisons involved in the merging process, which is not more than $(m + n)$.
Hence, the correct option is (c).

14. Which of the following statements is true?
I. As the number of entries in a hash table increases, the number of collisions increases.
II. Recursive programmes are efficient
III. The worst case complexity for Quicksort is $O(n^2)$
IV. Binary search using a linear linked list is efficient **[1995]**
(a) I and II
(b) II and III
(c) I and IV
(d) I and III

Solution: (d)

Two-marks Questions

1. The minimum number of comparisons required to find the minimum and the maximum of 100 number is **[2014]**

Solution: 148
$1.5\ (100) - 2 = 148$

2. Consider the following pseudo code. What is the total number of multiplications to be performed? **[2014]**

$D = 2$
for $i = 1$ to n do
for $j = i$ to n do
for $k = j + 1$ to n do
$D = D * 3$
(a) Half of the product of the 3 consecutive integers.
(b) One-third of the product of the 3 consecutive integers.
(c) One-sixth of the product of the 3 consecutive integers.
(d) None of the above.

Solution:
If we take $n = 3$ then the total number of multiplications performed are 4 which is equivalent to One-sixth of the product of 3 consecutive integers, i.e. (2*3*4)/6.

3. 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 **[2013]**

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

Solution: (b)

Outer loop: $\frac{n}{2}$

Inner loop: $\log n$

K will be incremented by $\left(\frac{n}{2}\right)$ for $\frac{n}{2}$ $\log n$ times. So, k will be of order

$0(n) * \frac{n}{2} \log n = 0(n^2 \log n)$

4. Consider the following operation along with Enqueue and Dequeue operations on queues, where k is a global parameter.

```
MultiDequeue (Q) {
m = k
While (Q is not empty) and (m > 0)
{
Dequeue (Q)
m = m - 1
}
}
```

What is the worst case time complexity of a sequence of n queue operations on an initially empty queue? **[2013]**

(a) $\Theta(n)$ (b) $\Theta(n + k)$
(c) $\Theta(nk)$ (d) $\Theta(n^2)$

Solution: (a)

The complexity of a sequence of 'n' operations = Total complexity of enqueue operations (α) + Total complexity of dequeue operations (β). As, Total complexity of dequeue operation (β) ≤ Total complexity of enqueue operation (α).

i.e. $(\beta) \le \alpha$ (1)

We know, Total complexity of enqueue operation

$(\alpha) \le n$ (2)

$\therefore$ Total complexity of n operations = $\alpha + \beta$
$\le \alpha + \alpha$ (From (1))
$\le n + n$ (From(2))
$\le 2n$.

Hence the worst case time complexity of a sequence of n queue operations on an initially empty queue is $\Theta(n)$

Hence, the correct option is (a).

5. 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 **[2012]**

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

Solution: (b)

In merge sort algorithm number of splits are proportional to height and in each level work done is n^2.

Hence, Total time complexity $O(n^2 \log n)$. The complexity of merge sort for elements is $O(n \log n)$

Hence, the correct option is (b).

6. The running time of an algorithm is represented by the following recurrence relation:

$$T(n) \begin{cases} n & n \le 3 \\ T\left(\frac{n}{3}\right) + cn & \text{otherwise} \end{cases}$$

Which one of the following represents the time complexity of the algorithm? **[2009]**

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

Solution: (a)

Applying master-theorem Case-III holds.

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

7. 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? **[2009]**

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

Solution: (b)

After the $\left(\frac{n}{4}\right)$ th smallest element is selected as pivot the list gets partitioned into one with $\left(\frac{n}{4}\right)$

and the other with $\left(\frac{3n}{4}\right)$ elements. Therefore divide and conquer recurrence in this case would be

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

By using variation of Mater-Theorem $T(n) = T(\alpha n) + T(\beta n) + f(n)$

The recursion expression becomes:

$T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{3n}{4}\right) + n$. Solving the recursion using variant of master theorem, we get $\Theta(n \log n)$. Hence, the correct option is (b).

8. Consider the quick sort algorithm. Suppose there is a procedure for finding a pivot element which splits the list into two sub-lists each of which contains at least one-fifth of the elements. Let $T(n)$ be the number of comparisons required to sort n elements. Then **[2008]**

(a) $T(n) \le 2T\left(\frac{n}{5}\right) + n$

(b) $T(n) \le T\left(\frac{n}{5}\right) + T\left(\frac{4n}{5}\right) + n$

(c) $T(n) \le 2T\left(\frac{4n}{5}\right) + n$

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

Solution: (b)

By applying divide and conquer concept one list would have $\frac{1}{5}$ elements and the other would have $\frac{4}{5}$ of the no. of elements 'n' comparisons are required for fixing up the pivot.

Hence, the correct option is (b).

9. An array of n numbers is given, where n is an even number. The maximum as well as the minimum of these n numbers needs to be determined. Which of the following is TRUE about the number of comparisons needed? **[2007]**

(a) At least $2n - c$ comparisons for some constant c, are needed.
(b) At most $1.5n - 2$ comparisons are needed.
(c) At least $n \log_2 n$ comparisons are needed.
(d) None of the above.

Solution: (b)

Using Divide and conquer method not more than $\left(\frac{3n}{2} - 2\right)$ comparisons are required in all cases of input.

10. Consider the following recurrence:

$$T(n) = 2T(|\sqrt{n}|) + 1T(1) = 1,$$

Which one of the following is true? **[2006]**

(a) $T(n) = \Theta(\log \log n)$
(b) $T(n) = \Theta(\log n)$
(c) $T(n) = \Theta.(n)$
(d) $T(n) = \Theta(n)$

Solution: (a)

Applying master theorem, case-1 applies and hence it is $\Theta(\log \log n)$

(or)

$T(n) = 2T(\sqrt{n}) + l$

$T(l) = l$

By continuing, till K^{th} iteration, we get

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

$$\Rightarrow 2^{-k} = \log_n 2$$

$$\Rightarrow K = \log \log_2 n$$

11. The median of n elements can be found in $O(n)$ time. Which one of the following is correct about the complexity of quick sort, which median is selected as pivot? **[2006]**

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

Solution: (b)

Altering the pivot that is fixed at the correct place, we get two sub-lists.

The number of steps required is $\log n$ and hence the complexity becomes $\Theta(n \log n)$. Pivot gets fixed at the middle which would split the list into two parts each having $\frac{n}{2}$ elements.

Hence, the correct option is (b).

12. Suppose $T(n) = 2T\left(\frac{n}{2}\right) + n$,

$T(0) = T(1)$

Which one of the following is FALSE? **[2005]**

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

Solution: (c)
It is θ (n log n) hence is also $O(n^2)$ and $O(n \log n)$ is not $\Omega(n^2)$
If we use binary search then there will be $\log_2^{n!}$ comparisons in the worst case, which is ($n$log$n$). But the algorithm as a whole will still have a running time of $\Theta(n^2)$ on average because of the series of swaps required for each insertion.
(or)

$$T(n) = 2T\left(\frac{n}{2}\right) + n,\ T(0) = T(l) = 1$$

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

$$ak - 2^k - ^l = 2k$$

$$(\Sigma - 2)ak = 2 \cdot 2^k$$

$$ak = c \cdot 2^k$$

$$ak = c \cdot 2^k$$

$$ak = \frac{2.2^k}{\sum - 2} = 2 \cdot c(k,1)2^{k-1}$$

$$= 2k \cdot 2^{k-1}$$

$$al = c \pm l$$

$$ak = 2^k + k \cdot 2^k$$

$$T(2^k) = 9k = n + n\log_2^n \Rightarrow O(n\log n)$$

Hence, the correct option is (c).

13. Let $T(n)$ be the function defined by $T(l) = 1$, $T(n) - 2 \cdot T\left(\frac{n}{2}\right) + \sqrt{n}$ for $n \geq 2$

Which of the following statements is true? **[1997]**

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

Solution: (b)

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

$$T(1) = 1$$

Master Theorem:

$$a = 2,\ b = 2,\ k = \frac{1}{2},\ p = 0$$

$2 > 2^{1/2}\ O(n\log_2^2) = O(n)$
Hence, the correct option is (b).

14. The recurrence relation

$$T(l) = 2$$

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

Has the solution $T(n)$ equal to **[1996]**

(a) $O(n)$
(b) $O(\log n)$
(c) $O(n^{3/4})$
(d) None of the above

Solution: (a)

Applying Master Theorem, $T(n)$ is $O(n)$;
$a = 3,\ b = 4,\ K = l$
Since $a < b^K$ so it is
Master Theorem:
$a = 3,\ b = 4,\ k = 0,\ p = 0$
$3 > 4°.\ O\left(n\log_4^3\right)$
Hence, the correct option is (a).

15. Quick-sort is run on two inputs shown below to sort in ascending order

(a) 1, 2, 3…n
(b) $n, n - l, n - 2, \ldots 2, 1$

Let $C1$ and $C2$ be the number of comparisons made for the inputs (a) and (b) respectively. Then, **[1996]**

(a) $C1 < C2$
(b) $Cl > C2$
(c) $Cl = C2$
(d) We cannot say anything for arbitrary n.

Solution: (c)
It takes same number of comparisons for both the case because quick sort behaves in worst case when the elements are already in sorted order
Hence, the correct option is (c).

16. The recurrence relation that arises in relation with the complexity of binary search is: **[1994]**

(a) $T(n) = T\left(\frac{n}{2}\right) + k$, k a constant

(b) $T(n) = 2\ T\left(\frac{n}{2}\right) + k$, k a constant

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

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

Solution: (a)

After comparing the key with the middle element, the search is made either in the left or right sublist with $\frac{n}{2}$ elements.

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

where 'K' is constant.

Hence, the correct option is (a).

17. Which one of the following statements is false? **[1994]**

(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.

Solution: (b)

Breadth First Search [BFS]

Breadth-first search (BFS) is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root and explores the neighbour nodes first, before moving to the next level neighbours. Initially, BFS starts at a given vertex, which is at level 0. In the first stage it visits all vertices at level 1. In the second stage, it visits all vertices at second level. These new vertices are the one which are adjacent to level 1 vertices. BFS continues this process until all the levels of the graph are completed.

Hence, the correct option is (b).

18. Assume that the last element of the set is used as partition element in quick sort. If n distinct elements from the set $[1 \ldots n]$ are to be sorted, give an input for which quick sort takes maximum time. **[1992]**

Solution:

Quick sort takes maximum time when the elements are already in sorted order. So, when the partition element is at the end, then in the worst case it has to be at the first position of the list (in case the elements are in decreasing order).

19. Following algorithm (s) can be used to sort n integers in the range $[1 \ldots n^3]$ in $O(n)$ time? **[1992]**

(a) Heap sort

(b) Quick sort

(c) Merge sort

(d) Radix sort

Solution:

If we subtract each number by 1 then we get the range $[0, n^3 - 1]$. Considering all number as 3-digit base n: each digit ranges from 0 to $n^3 - 1$. Sorting this using radix sort, it uses only three calls to counting sort. Finally, add 1 to all the numbers. Since there are 3 calls, the complexity is $O(3n) \approx O(n)$.

20. Let P be a quicksort programme to sort numbers in ascending order. Let t_1 and t_2 be the time taken by the program for the inputs [1 2 3 4] and [5 4 3 2 1], respectively. Which of the following holds? **[1987]**

(a) $t_1 = t_2$

(b) $t_1 > t_2$

(c) $t_1 < t_2$

(d) $t_1 = t_2 + 5 \log 5$

Solution: (a)

Quicksort behaves in worst case, for the sorted list. Since both the list are already in order, it takes $O(n^2)$ as worst case.

Hence, the correct option is (a).

21. Find a solution to the following recurrence equation **[1987]**

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

$$T(l) = l$$

Solution:

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

Master Theorem

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

Since $a < b^K$, so it is $1 < 2^1$ of master theorem

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

Five-marks Questions

1. Give an optimal algorithm is pseudo-code for sorting a sequence of n numbers which has only k distinct numbers (k is not known a Priori. Give a brief analysis for the time-complexity of your algorithm). **[1991]**

Solution: Counting Sort:
Counting sort is algorithm gives $O(n)$ complexity for sorting. To achieve $O(n)$ complexity, it assumes that each of the elements is an integer in the range 1 to k, for some integer k. When $k = O(n)$, it runs in $O(n)$ time. The basic idea is to determine, for each input elements X, the number of elements less than X. This information can be used to place directly into its correct position. For example, if there 10 elements less than X, then X belongs to position 11 in output.
In the below code, $A[0 \ldots n-1]$ is the input array with length n. In counting sort we need two more arrays: let us assume array $B[0 \ldots n-1]$ contains the sorted output and the array $C[0 \ldots k-1]$ provides temporary storage.

```
void Counting Sort (int A[]. int n.
int B[] int k){
int C[K], i, j;
//Complexity: O (k)
for (i = 0; i < k; i++)
C[i] = 0;
//Complexity: O(n)
For (j = 0; j < n; j++)
C [A[j]] = C[A[j]] + 1;
//C[i] now contains the number of
elements equal to i
//Complexity: O(k)
for (i = 1; i < k; i++)
C [i] = C[i] + C [i - l];
//C[i] now contains the number of
elements
< i
//Complexity: O(n)
For (j = n - l; j >= 0; j--){
B [C[A[j]]] = A[j];
C [A[j]] = C[A[j]] - l;
}
}
```

Total Complexity: $O(k) + O(n) + O(k) + O(n) = O(n)$ if $K = O(n)$. Space Complexity: $O(n)$ if $K = O(n)$.

Note: Counting works well if $K = O(n)$. Otherwise, the complexity will be more.

2. An input file has 10 records with keys as given below:

25 7 342 70 9 61 16 49 19

This is to be sorted in non-decreasing order.

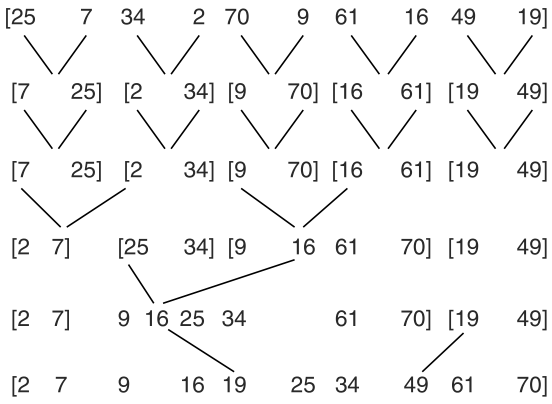
(a) Sort the input file using QUICKSORT by correctly positioning the first element of the file/sub file. Show the sub files obtained at all intermediate steps. Use square brackets to demarcate sub files.

(b) Sort the input file using 2-way MERGESORT showing all major intermediate steps. Use square brackets to demarcate sub files. **[1989]**

Solution: (a)

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
25	7	34	2	70	9	61	16	49	19
25	7	19	2	16	9	61	70	49	34
[9	7	19	2	16]	25	[61	70	49	34]
[9	7	2	19	16]	25	[61	70	4	34]
[2	7	9	19	16]	25	[61	70	49	34]
[2	7]	9	[19	16]	25	[61	70	49	34
[2	7]	9	[16	19]	25	[61	34	49	70]
[2	7]	9	[16	10]	25	[49	34]	61	[70]
[2	7	9	16	19]	25	[34	49]	61	[70]
[2	7	9	16	19	25	34	49	61	[70]

(b) 2-Way merge sort



3. Find a solution to the following recurrence equation:

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

$T(l) = 1$ **[1989]**

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

$T(l) = 1$

Using master theorem for divide and conquer recurrences and applying the cases; by Case (3) of the master theorem.

$T(n)$ is $8(\sqrt{n})$