COMPUTER SCIENCE & IT

POSTAL
BOOK PACKAGE
2024

OBJECTIVE
PRACTICE SETS

MADE EASY
Publications

www.madeeasypublications.org

POSTAL
BOOK PACKAGE
2024

CONTENTS

COMPUTER SCIENCE & IT

Algorithms

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) $O(N * \text{Sqrt}(N))$ (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) $O(\text{Sqrt}(N))$
(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) $O(n\log n)$
(c) $O(n^2)$ (d) $O(\log n)$

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

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

Identify the functions in increasing order of growth.
(a) $\dfrac{n}{\log n}$, $\log(\log n)$, $(\log n)$, $(\log n)^2$, $\sqrt{n}$, $\sqrt{n}\cdot\log n$, n
(b) $\log(\log n)$, $\log n$, $(\log n)^2$, $\sqrt{n}$, $\sqrt{n}\log n$, $\dfrac{n}{\log n}$, n, $n \log n$
(c) $\log(\log n)$, $\log n$, $(\log n)^2$, $\sqrt{n}$, $\dfrac{n}{\log n}$, $\sqrt{n}\log n$, 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 $f, g, h, k : N \rightarrow N$.
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

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, $\underset{\text{(Big–oh)}}{O(n^2)}$ 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^o, 2^1, ... 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

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

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

$$T_n \approx S_n$$

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

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

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

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

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

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

$= N + (N-1) + (N-2) + ... + 1 + 0$

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

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

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

18. (a)

Recurrence relation of function F

$F(n) = 0$ if $n = 0$

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

Time complexity = $O(n)$

19. (b)

for ($i = 1$; $i \le n$; i++) ⇒ $O(n)$

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

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.

$n! = 0(n^n)$

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

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

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 $f(n) = O\,g(n)$
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. $\dfrac{1}{2}(n^2) = \omega(n)$

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

So true always

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

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

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)

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

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

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

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

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

$= \Omega(n)$

28. (a)

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

29. (b)

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

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

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

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

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

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 ++;
```