

Ques 1. Asymptotic notations are languages that allow us to analyze an algorithm running time by identifying its behaviour as the input size of algorithm.

Types  $\rightarrow$

① Big O: It is commonly used for worst case, and gives us upper bound for the growth rate of runtime of algorithm.

② Example: Big O notation for linear search is  $O(n)$ .

② ~~Small o~~ Big Omega: It is notation used for best case complexity, it provides us with an asymptotic lower bound.

Ex: Big Omega of linear search is  $\Omega(1)$ .

③ Theta: It is used for tight bound on the growth rate of runtime of algo.

Ex: Theta of linear search is  $\Theta(n)$ .

④ Small o  $\rightarrow$  It is used to denote the upper bound (i.e. not asymptotically tight).

$$f(n) = o(g(n)) \quad \forall \quad f(n) < c(g(n)) \quad c > 0$$

⑤ Small omega: To denote lower bound (that is not asymptotically tight)

Ans 2. for ( $i=1$  to  $n$ )

{  $i = i+2$ ; }

$\Rightarrow O(\log n)$

Ans 3.  $T(n) = 3T(n-1)$

$$T(1) = 1$$

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

$$T(3) = 3T(2) = 9$$

$$T(4) = 3T(3) = 27$$

$$T(n) = (n-1)^3$$

Time complexity  $\rightarrow O(3^n)$

Ans 4.  $T(n) = 2(T(n-1) - 1)$

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

$$T(n) = 4T(n-2) - 2 - 1$$

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

$$T(n) = 8T(n-3) - 4 - 2 - 1$$

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

$$T(n) = 16T(n-4) - 8 - 4 - 2 - 1$$

$$T(n) = 2^k - 2 - 2^2 - 2^3 - 2^4 - \dots$$

$\approx$

$\approx$

$O(2^n)$

$\approx$

$O(1)$

Ans 5.

S

i

1

1

3

2

6

3

10

4

$O(\sqrt{n})$



Ans 6.

$$i * i = n$$

$$i^2 = n$$

$$i = \sqrt{n}$$

$$O(\sqrt{n})$$

Ans 7.

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

Ans 8.

Ans 9. Total  $T = O(n \log n)$

Ans 10.  $n^k$  is  $O(c^n)$  as for example

Of use take  $n=2$

$$k=2, c=2$$

Then  $2^2 \leq 2^2$  so  $c^n$  is upper limit of  $n^k$ .

Ans 11.

$$j=1 \quad i=0$$

$$1 \quad 1$$

$$2 \quad 3$$

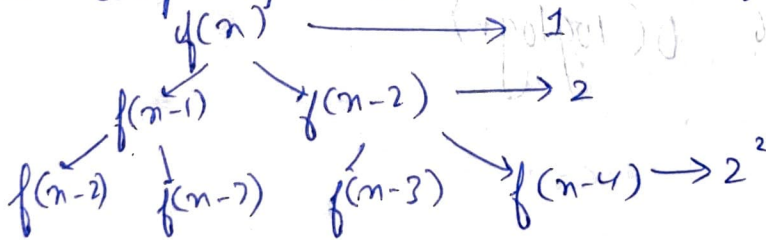
$$3 \quad 6$$

$$4 \quad 10$$

the series of  $i$  is nearly dependent on  $i$  as  $2^i$   
so  $O(2^n)$

Ans 12.

space complexity =  $O(n)$  as it calls  $f(n-1)$



time complexity =  $O(2^n)$

$$2^n$$

Ans 13  $n \log n$

for (i=0; i<n; i++)  
for (j=0; j<n; j=j+2)  
c++;

$n^3$

for (i=0; i<n; i++)  
for (j=0; j<n; j++)  
for (k=0; k<n; k++)  
c++;

$\log(\log n)$

int funt (int n) {  
if (n==1)  
return n;

else  
return funt( $\sqrt{n}$ ) + funt( $\sqrt{n}$ );  
}

Ans 14

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

using master  $\rightarrow$   
a=2, b=2

$$C=1$$

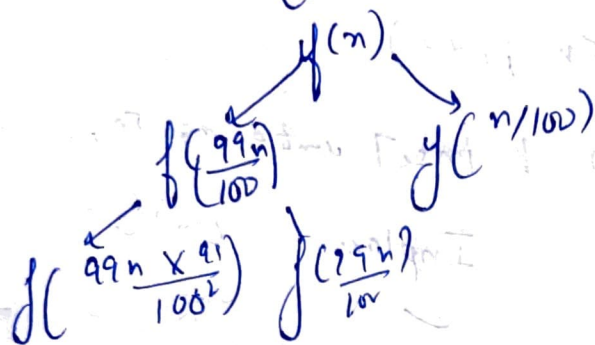
$$f(n) > n^a \quad \text{if } n^2 > 1$$

$$O(n^2)$$

Ans 15  $O(n\sqrt{n})$

Ans 16  $O(\log \log n)$

Ans 17  $T(n) = T\left(\frac{99n}{100}\right) + T\left(\frac{n}{100}\right)$



$O(\log n)$

Ans 18 a)  $100 < \log \log n < \log n < \sqrt{n} < n \log(n!) < n \log^n n < n^2 < 2^n < 2^{2^n} < 4^n < n!$

b)  $1 < \log \log n < \sqrt{\log n} < \log^2 n < \log n < 2 \log n < n < n^2 < 2^n < 4^n < n^n < n! < 2(2^n) < n!$

c)  $96 < \log_2 n < \log_2^2 n < \log_2 5n < \log n! < n \log n < n \log_2 n < 8n^2 < 7n^3 < 8^{2^n} < n!$

Ans 19 - linear (arr, Key)

```

for (int i=0; i<n; i++)
    if (arr[i] == Key)
        return i;
  
```

return -1;

Ans 20 3 Insert(arr, n)

if (n <= 1) return;

recursively sort n-1 element

sort(arr, n-1);

Pick the element arr[n] & insert it into sorted sequence

~

## Iterative

Insert(arr, n) {

for (i = 1; i < n; i++)

{ Pick arr[i]

Insert into arr[0, ..., i-1]

}

stable

Inplace

Online

Bubble sort

✓

✓

✗

Select

✗

✓

✓

Insert

✓

✓

✗

Ans

Best

Avg

Worst

Space complex

Bubble

$O(n^2)$

$O(n^2)$

$O(n^2)$

1

Select

$O(n^2)$

$O(n^2)$

$O(n^2)$

1

Insert

$O(n)$

$O(n^2)$

$O(n^2)$

1

Ques 23 - Recursion

Binary(arr, l, r, key) {

if (l > r) {

mid = l + (r - l) / 2; if (arr[mid] == key) return 1;

if (arr[mid] < key) Binary(l, mid + 1, key);

else Binary(l, mid - 1, key);

}

Binary(arr, l, r, key)

}

Iterative

while (l <= r)

{ mid = l + (r - l) / 2

if (arr[mid] == key) return 1;

if (key < arr[mid])

l = mid - 1;

else l = mid + 1;

}



Ans 24  $T(n) \neq T\left(\frac{n}{2}\right) + 1$

Ans 1  $k =$