Name → Nishul Patni Uni Roll no → 1961118
Course → B.Tech (CS) Sem → V
Sec → A

Q1

Ans Asymtotic notations are the mathmatical notations used to describe the running time of an algo when the input tends towards a particular value or a limiting value.

There are mainly 3 types of asmtotic notation.

1. Big O notation

- It represents the upper bond of running time of an algo.

- $O(g(n)) = \{ f(n)$: there exist positive constants c & n_0 such that $0 \leq f(n) \leq gn$ for all $n \geq n_0$, where $C > 0$ & $n \geq n_0$.

2. Big Onega (Ω) notation

- It represents the lower bond of the

running time of an algo.

- $\Omega(g(n)) = \{ f(n)$: there exist positive constant c & n_0 such that $0 \leq g(n) \leq f(n)$ $\forall\ n,\ n \geq n_0$

3 Theta (Θ) notation

- It represents the upper & lower bound of running time of an algo.

- $\Theta(g(n)) = \{ f(n)$: their exist positive constant c_1, c_2 & n_0 such that $0 \leq c_1 * g(n) \leq f(n) \leq c_2 * g(n)$ $\forall\ n > n_0$

Θ_2

Ans $i = 2, 4, 8, 16, \ldots, k^{th}$ term, $\ldots, n$

$y_n = a^{n-1}$

$y_n = (2)^{k-1}$

$n = 2^{k-1}$

$\log(n) = (k-1)\log 2$

$k = \lg n + 1$

$O(an) = \log n$

Θ_3

Ans $T(n) = \begin{cases} 3T(n-1) & \text{if } n > 0, \\ 1 & \text{otherwise} \end{cases}$

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

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

$T(n) = 3 \times 3T(n-2)$

$\hookrightarrow T(n-2) = 3T(n-3)$

$T(n) = 3 \times 3 \times T(n-3)$

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

$\hookleftarrow T(n-3) = 3T(n-4)$

$T(n) = 3^3 \times 3T(n-4)$

⋮

general form :-

$T(n) = 3^i T(n-i) \quad \cdots (i)$

$T(n-i) = T(0)$

$n - i = 0$

$n = i$

putting $n = i$ in (i)

$T(n) = 3^n T(n-n) = 3^n T(0)$

$T(n) = 3^n$

$\boxed{T(n) = O(3^n)}$

Q4

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

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

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

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

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

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

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

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

classmate
Date
Page

:
:

gen form

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

$T(n-i) = T(0)$

$n - i = 0$

$\boxed{n = i}$

$T(n) = 2^n T(0) - (1 + 2 + 2^2 + 2^3 + \cdots + 2^{n-1})$

$\boxed{T(0) = 1}$

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

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

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

$\boxed{T(n) = O(2^n)}$

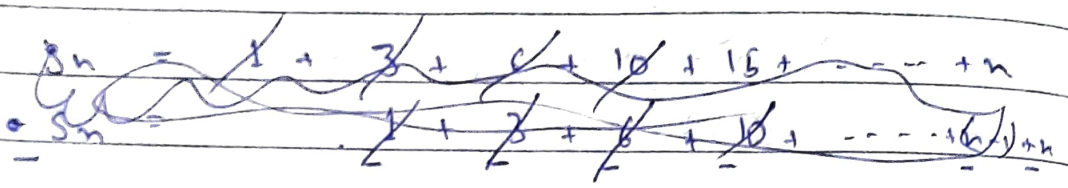
Q5

Ans

no of steps	S	i
0	0	1
1	1	2
2	3	3
3	6	4
4	10	5
⋮	⋮	⋮
k step	n	

$T(n) = O(K)$

$S = 0, 1, 3, 6, 10, \ldots\ldots n$



$$S_n = 1 + 3 + 6 + 10 + 15 + \ldots\ldots + n$$

$$S_n = \quad 1 + 3 + 6 + 10 + \ldots\ldots (n-1) + n$$

$$1 + 2 + 3 + 4 + 5 + \ldots\ldots n$$

$n = 1 + 2 + 3 + 4 + \ldots\ldots$ K step

$$n = \frac{K}{2}[2(1) + (K-1)]$$

$$2n = K[2 + K - 1]$$

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

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

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

$$T(n) = T(K)$$

$$\boxed{T(n) = \sqrt{n}}$$

Q1 8

Ans Since i is moving from 1 to $\sqrt{n}$ with

linear growth

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

Q7

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

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

Q8

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

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

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

gen term

$T(n) = T(n-i) + n^2 + (n-1)^2 + (n-2)^2 + \cdots + (n-i)^2$

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

$n = i+1 \Rightarrow \boxed{n-1 = i}$

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

$T(n) = T(1) = n^2 + (n-1)^2 + (n-2)^2 + \cdots + 1^2$

$T(n) = \frac{n(n+1)(2n+1)}{6}$

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

Q9

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

Q.10

Ans If $c > 1$ then the exponential c^n for outgrows any term, so that ans is:

n^k is $O(c^n)$

Q.11

Ans $j = 0, 1, 3, 6, 10, 15, \ldots$

$j = 1, 2, 3, 4, 5, 6, \ldots$

So, i will go on till n & gen formula

for k^{th} term is $n = \frac{k(k+1)}{2}$

$\boxed{T.C = O(\sqrt{n})}$

Q.12

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

$T(n-2) \approx T(n-1)$

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

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

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

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

$\hookrightarrow T(n-2) = 2T(n-3) + C$

$T(n) = 2^3 (2T(n-3) + C) + 2C + C$

⋮ ⋮ ⋮

gen Term

$$T(n) = 2^i\, T(n-i) + (2^0 + 2^1 + 2^2 \cdots 2^{i-1})\, c$$

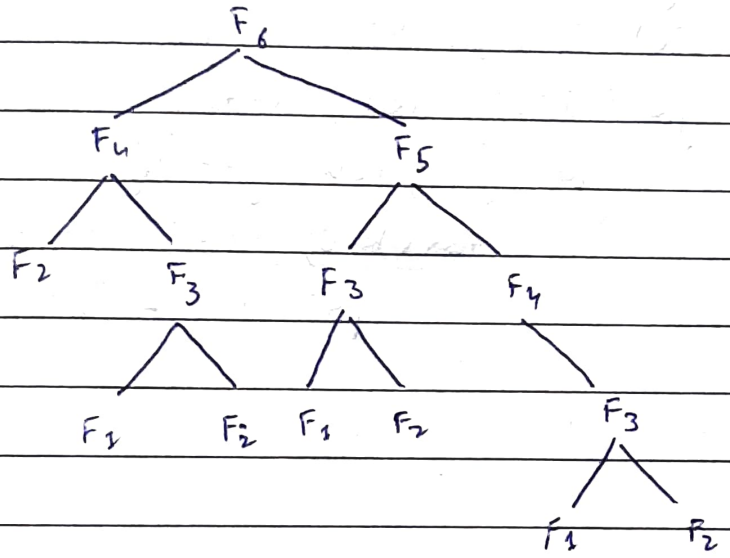
$n - i = 0$

$\boxed{n = i}$

$$T(n) = 2^n (1) + 2^0 \frac{(2^{n-1} - 1)}{2-1} c$$

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

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



The max depth is proportional to N, hence the space complexity of fabonacci is $O(N)$.

Q.3

(i) $n \log n$

```
void fun(){
    int i, j;
    for(i=1; i<=n; i++){
```

```
        for (j=0; j<=n; j=j*2){
            printf("#");
        }
        printf("\n");
    }
}
```

(ii) n^3

```
void fun (n){
    int j, i, k;
    for (i=0; i<n; i++){
        for (j=0; j<n; j++){
            for (k=0; k<n; k++){
                printf("#");
            }
        }
    }
}
```

Ans

(iii) $\log(\log(n))$

```
void fun (int n){
    bool prime[n+1];
    memset (prime, true, sizeof(prime));
    for (int p=2; p*p<=n; p++){
        if (prime[p] == true){
```

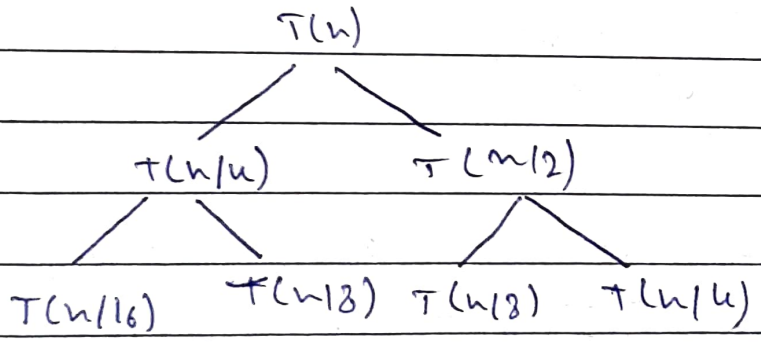
```
for (int i=p*p; i<=n; i+=p){
        prime[i] = false;
    }
}

for (int p=2; p<=n; p++){
    if (prime[p])
        cout << p << endl;
}
```

Q14

Ans $T(1) = c$

$T(n/2) = T(n/8) + T(n/4) + c(n^2/4)$

$T(n) = T(n/4) + 2T(n/16) + c(n^2/16 + n^2/4 + n^2)$



$$T(n) = c\left[n^2 + \frac{5n^2}{16} + \frac{25n^2}{256} + \cdots\right]$$

$$T(n) = n^2 c\left[1 + \frac{5}{16} + \frac{52}{16^2} + \cdots\right]$$

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

Q15

Ans

for $i = 1$, inner loop execute n times
for $i = 2$, " " " $n/2$ "
for $i = 3$, " " " $n/3$ "
.
.
.
for $i = n$, " " " n/n "

Total time $= n + n/2 + n/3 + \cdots + n/n$
$= n(1 + 1/2 + 1/3 + \cdots + 1/n)$
$= n \log(n)$

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

Q16

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

Q18

(a) 100, $\log(\log n)$, $\log n$, root n, n, $n \log n$, n^2, 2^n, 2^{2n}, 4^n, $n!$

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

(c) 96, $\log_6 n$, $\log_2 n$, $\log(n!)$, $5n$, $n\log_6 n$, $n\log_2 n$, $8n^2$, $7n^3$, 8^{2n}, $n!$

Q 19

Ans

```
LinearSearch(A, Key)
    comp ← 0, f ← 0
    for j = 1 , to A.length
    comp ← comp + 1
    if A[i] == Key
    print "Element found"
    f = 1
    if f == 0
    print "Element not found"
     print comp
```

Q 20

Ans

```
Insertion (A)          // Iterative

for j = 2. to A.length
Key = A[j]
i = j-1
while i > 0 & A[i] > Key
        A[i+1] = A[i]
        i = i-1
A[i+1] = Key

Insertion (A, n)       // recursive
if n ≤ 1
    return
Insertion(A, n-1)
```

```
Key = [n-1];
j = n-2
while j ≥ 0 and A[j] > Key
A[j+1] = A[j]
j = j - 1
A[j+1] = Key
```

→ Insertion sort considers one input element per iteration & produces a partial soln without considering future elements that's why it is called online sorting

Q27

Ans

	In place	Stable	Online
Bubble sort	Yes	Yes	Yes
Insertion "	"	"	"
Selection "	"	~~No~~	"
Merge "	No	Yes	"
Quick "	~~No~~ Yes	No	"
Heap "	"	No	"
Count "	No	Yes	"

Q23

Ans Linear Search :-

```
LinearSearch (A, Key)
        found ← 0
        for i = 1 to N
        if A[i] == Key
                found ← 1
        print "Element found"
        break
        if found == 0
                print " Element not found"
```

T.C. → O(n)
S.C. → O(1)

Binary Search :-

```
Binary Seach (A , beg , end, key)

        while beg ≤ end
                mid = beg+(end - beg)/2
        if mid == key
                return mid;
        if A[mid] < key
                beg = mid + 1
        if A[mid] > key
                end = mid - 1
```

return -1

Binary Search (Recursive)

```
BinarySeach (A, beg, end, key)
    if (end > beg)
        mid = (beg + end) / 2
    if A[mid] == item
        return mid + 1
    else if A[mid] < item
        return BinarySearch (A, mid + 1, end, key)
    else
        return BinarySearch(A, beg, mid - 1, end)

    return -1
```

T.C = O(log n)

S.C = O(1)

Q24

Ans $T(n) = T(n/2) + C$.