Name : Ranjana Singh.
Section : E
Roll no : 46
Uni. Roll no : 2014801

Assignment - I

Ans 1:- Asymptotic notations are languages that allow us to analyze an algorithm running timidly identifying its behaviour as the input size of algorithm.

Types:-

(a) Big O: It is commonly used for worst case, and gives upper bound for the growth rate of runtime of algorithm,
Ex:- Big O notation for linear search is O(n)

(b) Big Omega: It is notation used for best can complexity, it provides as with an asymptotic lower bound.
Ex:- Big Omega of linear search is $\Omega(1)$

(c) Theta: It is used for right bound on the growth rate of runtime of algo.
Ex:- Theta of linear search is $\Theta(n)$.

(d) Small Omega: To denote lower bound (that is not asymptotic right).

Ans 2:- for (i = 1 to n)
{ i = i+2; }

$\Rightarrow$ O(log n)

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

$T(1) = 1$

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

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

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

$\vdots$

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

Time complexity → $O(3^n)$

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

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

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

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

$T(n) \quad 8T(n-3) - 4 - 2 - 1$

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

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

$T(n) = 2^k - \cdots 2^3 - 2^2 - 2^1 - 2^0$

$= O(1)$

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 9:- Total T $= O(n \log n)$

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

Of:- when we take $n=2, k=2, c=2$

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

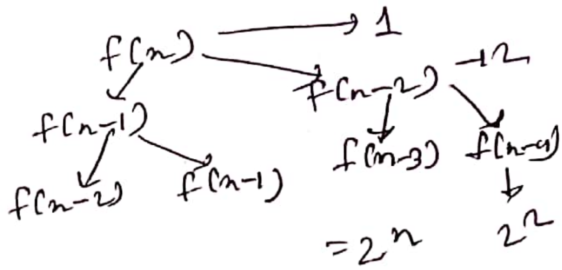
Ans 11:

j=1	i=0
1	1
2	3
3	6
4	10

The series is nearly dependent on i as 2^i

So $O(2^n)$

Ans 12: Space complexity $= O(n)$ as clear call of $(n-1)$



time complexity $= O(2^n)$

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 funct (int n) {
    if (n==1)
        return n;
    else
return func(√n) +
    func(√n); }
```

Ans 14:- $T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + Cn^2$

Using mast $\Rightarrow$

$a = 2, \; b = 2$

$c = 1$

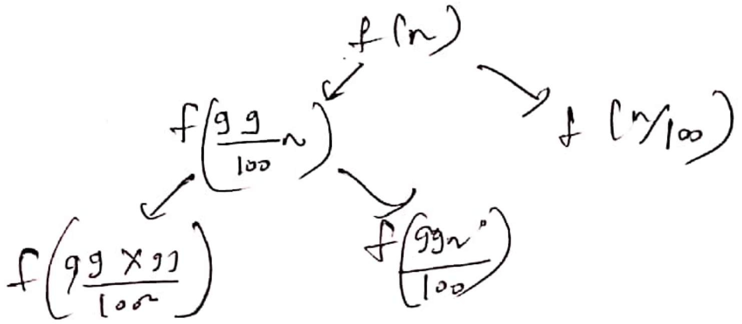
$f(n) > n^k$ & $n^2 > 1$

$O(n^2)$

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

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

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



$= O(\log n)$

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

b) $1 < \log\log n < \sqrt{\log n} < \log^2 n < \log n <$
$2\log_2 n < n < 2n < 4n < n^2 < n! < 2(2)^{2n} < n!$

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

Ans 19: linear (arr, key) {
for (int i=0; i<n; i++)
if (arr[i] == key)
return i;
return -1;
}

Ans 20: Ins (arr, n) {
if (n <= 1) return;
recursively for n-1 element
Insert Sor (arr, n-1)

$\Rightarrow$ Pick last element arr [i] & Ins [i into sorted sequence]

Iterations:-

```
Insert (arr, n) {
    for (i = 1, i<n; i++)
    {
     Pick arr(i) & insert into arr [0, ---- 1-1]
    }
```

	Stable	Inplace	Online
Bubble Sort	✓	✓	✗
Selection "	✗	✓	✗
Insertion "	✓	✓	✓

Ans 22:

	Best	Avg	Worst
Bubble	$O(n^2)$	$O(n^2)$	$O(n^2)$
Selection	$O(n^2)$	$O(n^2)$	$O(n^2)$
Insertion	$O(n)$	$O(n)$	$O(n^2)$

Ans 23:- Recursive:-

```
Binary (arr, l, r, key) {
 if (l<r) {
     mid = l+ (r-l)/2;
  if (arr[mid] = = key) ret-1;
  if (key < arr mid])
  Binary (l, mid - 1, key);
 else
   Binary (mid +1, r, key)
 }
```

Iterative:-

```
    while (l<r)
    {
      mid = l+ (r-l)/2
  if (arr[mid] = = key) return 1;
   if (key < arr[mid])
       r = mid-1;
     else  l= mid +1;
         }
```

Ans 24:

$T(n) \neq T(n/2)$.