TUTORIAL - 1

ANSWER

1 -> O(N+M) Time, O(1) Space

2 -> Time = O(n)

3 -> Time = O(log n)

4 -> Time = $O(\sqrt{n})$

5 -> Time = O(n)

6 -> Time = $O(2^n)$

7 -> Time = $O(\log_2 n)$

8 -> 1 - O(n)

 2 – $O(n^2)$

 3 – $O(\log_2 n)$

 4 – O(n)

 5 – $O(2^n)$

 6 – $O(3^n)$

 7 – O(log log n)

 8 – O(n)

9 -> Time = O(n)

10 -> Time = O(N*N)

11 -> O(n log n)

12 -> X will always be a better choice for large inputs

13 -> O(log N)

14 -> O(n^2.8), O(n^3), Θ(n^2.8)

15 -> f(2)>f(4)>f(3)>f(1)

16 -> $\Omega(2^n)$

17 -> $O(n^2)$

18 -> Time = O(m)

19 -> Time = O(N*N + N)

Q. What is the complexity of the following piece of code:-

1. What is the time, space complexity of following code:

```
int a = 0, b = 0;
for (i = 0; i < N; i++) {
    a = a + rand();
}
for (j = 0; j < M; j++) {
    b = b + rand();
}
```

1. O(N * M) time, O(1) space
2. O(N + M) time, O(N + M) space
3. O(N + M) time, O(1) space
4. O(N * M) time, O(N + M) space

2.

```
int sum = 0, i;
for(i=0;i<n;i = i+2)
{
   sum += i;
}
```

3

```
int sum = 0, i;
for(i=0;i<n; i = i*2){
    sum += i;
}
// even if it is i*10 or i*100
answer is same asymptotically
```

4.

```
int sum = 0, i;
for(i=0;i*i<n;i++)
{
    sum += i;
}
```

5.

```
int j = 1, i = 0;
while(i<=n){
    i = i+j;
    j++;
}
```

6.

```
void recursion(int n)
{
    if(n == 1) return;
    recursion(n-1);
    print(n);
    recursion(n-1);
}
```

7.

```
int recursion(int what[], int thisone, int
thatone, int x)
{
    if (thatone >= thisone)
    {
        int something = thisone + (thatone -
thisone)/2;
        if (what[something] == x)
            return something;
        else if (what[something] > x)
            return recursion(what, thisone,
something-1, x);
        return recursion(what, something+1,
thatone, x);
    }
    return -1;
}
```

8. Solve the following recurrence relation:- T(1) = 1

1. $T(n) = T(n-1) + 1$
2. $T(n) = T(n-1) + n$
3. $T(n) = T(n/2) + 1$
4. $T(n) = 2T(n/2) + 1$
5. $T(n) = 2T(n-1) + 1$
6. $T(n) = 3T(n-1)$, $T(0) = 1$
7. $T(n) = T(\sqrt{n})+1$
8. $T(n) = T(\sqrt{n})+n$

9

```
int sum = 0, i;
for(i=0;i<n;i++)
{
    sum += i;
}
```

10. What is the time complexity of following code:

```
int a = 0;
for (i = 0; i < N; i++) {
    for (j = N; j > i; j--) {
        a = a + i + j;
    }
}
```

Options:

1. O(N)
2. O(N*log(N))
3. O(N * Sqrt(N))
4. O(N*N)

11. What is the time complexity of following code:

```
int i, j, k = 0;
for (i = n / 2; i <= n; i++) {
    for (j = 2; j <= n; j = j * 2)
{
        k = k + n / 2;
    }
}
```

Options:

1. O(n)
2. O(nLogn)
3. O(n^2)
4. O(n^2Logn)

12. What does it mean when we say that an algorithm X is asymptotically more efficient than Y?
Options:

1. X will always be a better choice for small inputs
2. X will always be a better choice for large inputs
3. Y will always be a better choice for small inputs
4. X will always be a better choice for all inputs

13. What is the time complexity of following code:

```
int a = 0, i = N;
while (i > 0) {
    a += i;
    i /= 2;
}
```

Options:

1. O(N)
2. O(Sqrt(N))
3. O(N / 2)
4. O(log N)

14. Solve the following recurrence relation?
T(n) = 7T(n/2) + 3n^2 + 2
(a) O(n^2.8)
(b) O(n^3)
(c) θ(n^2.8)
(d) θ(n^3)

15. Sort the following functions in the decreasing order of their asymptotic (big-O) complexity:
f1(n) = n^√n , f2(n) = 2^n, f3(n) = (1.000001)^n , f4(n) = n^(10)*2^(n/2)
(a) f2> f4> f1> f3
(b) f2> f4> f3> f1
(c) f1> f2> f3> f4
(d) f2> f1> f4> f3

16. f(n) = 2^(2n)
Which of the following correctly represents the above function?
(a) O(2^n)
(b) Ω(2^n)
(c) Θ(2^n)
(d) None of these

17. T(n) = 2T(n/2) + n^2. T(n) will be
(a) O(n^2)
(b) O(n^(3/2))
(c) O(n log n)
(d) None of these

18.

```
int gcd(int n, int m){
    if (n%m ==0) return m;
    if (n < m) swap(n, m);
    while (m > 0){
        n = n%m;
        swap(n, m);
    }
    return n;
}
```

19.

```
int a = 0, b = 0;
for (i = 0; i < N; i++){
    for (j = 0; j < N; j++){
        a = a + j;}
}
for (k = 0; k < N; k++){
    b = b + k;}
```