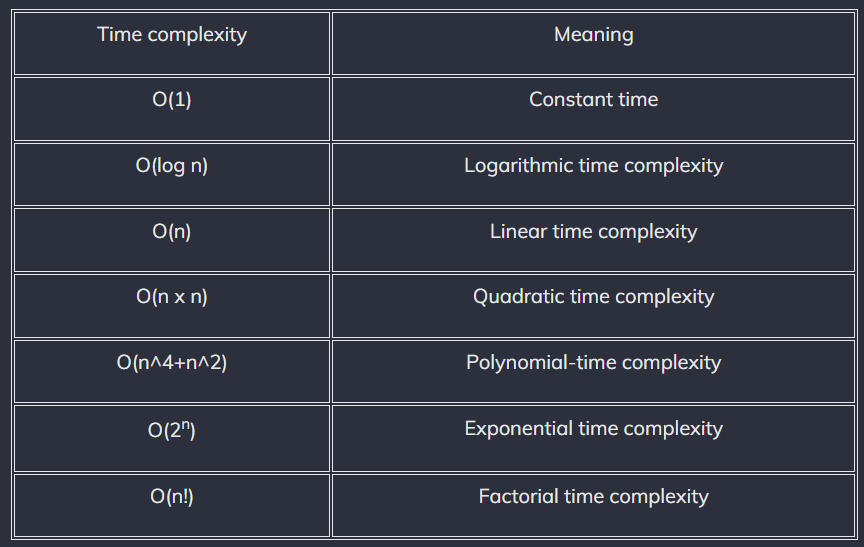
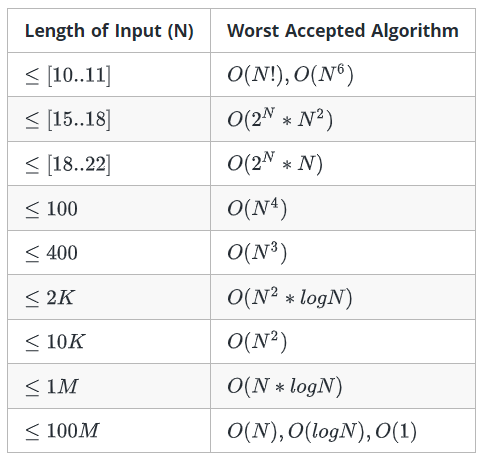
**Time and Space Complexity**





1. O(N + M) time, O(1) space

**int** a = 0, b = 0;

**for** (i = 0; i < N; i++) {

a = a + **rand**();

}

**for** (j = 0; j < M; j++) {

b = b + **rand**();

}

2. O(N\*N)

**int** a = 0;

**for** (i = 0; i < N; i++) {

**for** (j = N; j > i; j--) {

a = a + i + j;

}

}

3. O(nLogn)

**int** i, j, k = 0;

**for** (i = n / 2; i <= n; i++) {

**for** (j = 2; j <= n; j = j \* 2) {

k = k + n / 2;

}

}

4. O(log N)

**int** a = 0, i = N;

**while** (i > 0) {

a += i;

i /= 2;

}

5.O(logkn)

**for**(**var** i=0;i<n;i++)

i\*=k

6. n(n-1)

int value = 0;

for(int i=0;i<n;i++)

for(int j=0;j<i;j++)

value += 1;

7. O(n+m).

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

cout<<i<<endl;

}

for(int j=1;j<=m;j++){

cout<<j<<endl;

}

8. O(n\*logn )

for(int i=n/2;i<=n;i++){

for(int j=1;j<=n;j=j\*2){

cout<<i<<j<<endl;

}

}

9.O(1)

if(i>j){

j==0? j++ : j--;

}

10. O(n\*n)

for (int i = 0; i < n; i++)

for (int j = i; j > 0; j--)

cout << i << j;

11. O(√n)

**void** function(**int** n)

{

**int** i = 1, s = 1;

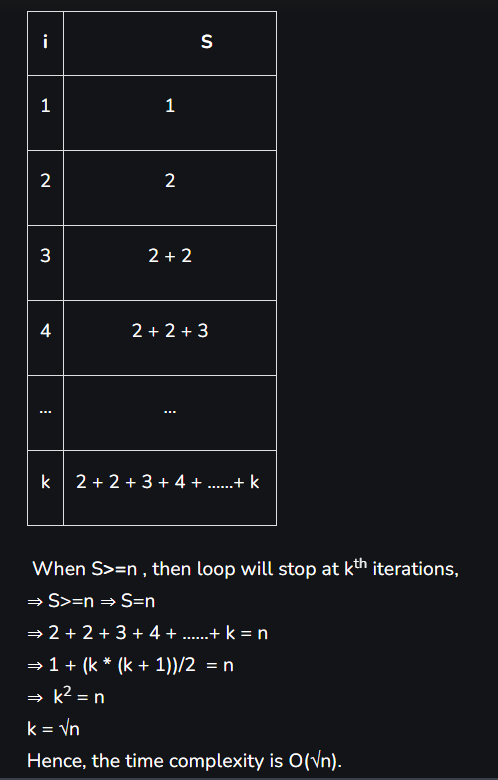
**while** (s < n) {

s = s + i;

i++;

}

}



12. Time complexity = O(1) in best case and O(n) in worst case.

**void** fun(**int** n)

{

**if** (n < 5)

cout << "Phitron";

**else** {

**for** (**int** i = 0; i < n; i++) {

cout << i;

}

}

}

13. Time complexity = O(1) in best case and O(max(a, b)) worst case.

**void** fun(**int** a, **int** b)

{

// Consider a and b both are positive integers

**while** (a != b) {

**if** (a > b)

a = a - b;

**else**

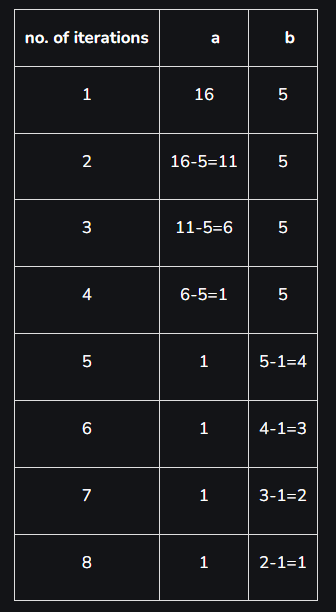
b = b - a;

}

}

If the value of a and b are the same, then while loop will not be executed. Hence, time complexity will be O(1).

But if a!=b, then the while loop will be executed. Let a=16 and b=5;



For this case, while loop executed 8 times (a/2⇒16/2⇒8).

If a=5 and b=16, then also the loop will be executed 8 times. So we can say that time complexity is O(max(a/2,b/2))⇒O(max(a, b)), it is considered as worst case because it takes more time.

14. Time complexity = O(√n).

**void** fun(**int** n)

{

**for**(**int** i=0;i\*i<n;i++)

cout<<"Phitron";

}

15. Time complexity = O(logxn).

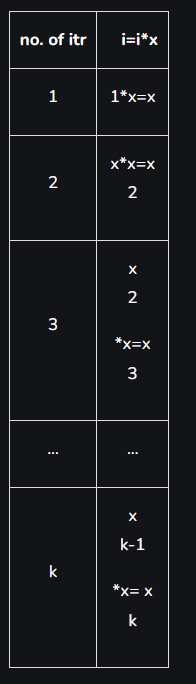
void fun(int n, int x)

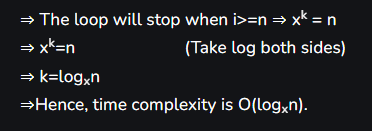
{

for (int i = 1; i < n; i = i \* x) //or for(int i = n; i >=1; i = i / x)

cout << "Phitron";

}





16.



#include <iostream>

**using** **namespace** std;

**void** fun(**int** n)

{

**for** (**int** i = 0; i < n / 2; i++)

**for** (**int** j = 1; j + n / 2 <= n; j++)

**for** (**int** k = 1; k <= n; k = k \* 2)

cout << "Phitron";

}

**int** main()

{

**int** n=8;

fun(3);

}

17.Time complexity = O(nlogn).

**void** fun(**int** n)

{

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

**for** (**int** j = 1; j <= n; j = j + i)

cout << "Phitron";

}

18.



void fun(int n)

{

for (int i = 0; i <= n / 3; i++)

for (int j = 1; j <= n; j = j + 4)

cout << "Phitron";

}

19.



**void** fun(**int** n)

{

**int** i = 1;

**while** (i < n) {

**int** j = n;

**while** (j > 0) {

j = j / 2;

}

i = i \* 2;

}

}

20.

Time Complexity: O(2^n) without memoization, O(n) with memoization

Space Complexity: O(n) with memoization

int fib(int n) {

if (n <= 1) return n;

return fib(n - 1) + fib(n - 2);

}

21.

Time Complexity: O(n^2)

Space Complexity: O(1)

void selectionSort(int arr[], int n) {

for (int i = 0; i < n - 1; ++i) {

int min\_index = i;

for (int j = i + 1; j < n; ++j) {

if (arr[j] < arr[min\_index]) {

min\_index = j;

}

}

swap(arr[i], arr[min\_index]);

}

}