**Examples Of Time Complexity**

**EXAMPLE 1:**

void f1(int n)

{

int a=9;

int b=3;

int sum=a+b+n;

int prod=a\*b+n;

int quot=a\*n/b;

printf("%d %d %d",sum,prod,quot);

}

**Complexity analysis:** We don't care about actual number of operations. Complexity is always calculated w.r.t size of input. So , here the complexity is O(1) , because all the operations will take same time irrespective of the value of n.

**==============================================================================================================================**

**EXAMPLE 2:**

int f2(int n)

{

int sum=0;

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

sum=sum+i;

return sum;

}

**Complexity analysis:**

C(n)=1+1+(n+1)+n+n+1

C(n)=4+3n

C(n)=O(n)

Since we do not consider lower order terms and constants while expressing the complexity , rather only higher power of n is considered , so the complexity is O(n) i.e. operations change linearly with the size of input

**==============================================================================================================================**

**EXAMPLE 3:**

void f3(int n)

{

if(n%2==0)

printf("No is even");

else

{

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

printf("\n%d",i);

}

}

**Complexity analysis:** We always consider the worst case scenario which will be when n is odd.

C(n)=1+1+(n+1)+n+n

C(n)=3+3n

C(n)=O(n)

**==============================================================================================================================**

**EXAMPLE 4:**

void f4(int n)

{

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

{

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

{

printf("\n%d %d",i,j);

}

}

}

**Complexity analysis:** For every iteration of outer loop , the inner loop will run i times. So,

When i=1 , j will run ‘1’ times

When i=2, j will run ‘2’ times

When i=3, j will run ‘3’ times

.

.

When i=n , j will run ‘n’ times

So C(n)=1+2+3+…+n times

C(n)=n(n+1)/2

C(n)=(n2+n)/2

Since , only highest power of n is considered , so the complexity is O(n2)

i.e. C(n)=O(n2)

**==============================================================================================================================**

**EXAMPLE 5:**

void f5(int n)

{

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

{

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

{

printf("\n%d %d",i,j);

}

}

}

**Complexity analysis:** For every iteration of outer loop , the inner loop will run n times. So,

When i=1 , j will run ‘n’ times

When i=2, j will run ‘n’ times

When i=3, j will run ‘n times

.

.

When i=n , j will run ‘n’ times

So C(n)=n+n+…..+n times

C(n)=O(n2)

Since , only highest power of n is considered , so the complexity is O(n2)

**==============================================================================================================================**

**EXAMPLE 6:**

void f6(int n)

{

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

{

printf("\n%d",i);

}

for(i=0;i<100;i++)

{

printf("\n%d",i);

}

**Complexity analysis:**

C(n)=(1+(n+1)+n+n) +(1+100+100+101)

C(n)=(3n+2)+302

C(n)=O(n)

The complexity will be O(n) because the constant associated with the second loop can be easily ignored.

**==============================================================================================================================**

**EXAMPLE 7:**

void f7(int n)

{

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

{

printf("\n%d",i);

i=i\*2;

}

}

**Complexity analysis:**

To find out the complexity of the above code we must calculate that when will the condition i<n become false.

For this let's find out the value of i on each iteration:

1. i=1----------20

2. i=2----------21

3. i=4----------22

4. i=8----------23

5. i=16--------24

.

k. ------------2k-1

k+1------------2k

Approx at kth iteration the condition will become false. So,

2k=n

k=log2n

C(n)=log2n

**=============================================================================================================================**

**EXAMPLE 8:**

void f8(int n)

{

for(int i=n;i>1;)

{

printf("\n%d",i);

i=i/2;

}

}

**Complexity analysis:**

To find out the complexity of the above code we must calculate that when will the condition i>0 become false.

For this let's find out the value of i on each iteration:

1. i=n

2. i=n/2----------n/21

3. i=n/4----------n/22

4. i=n/8----------n/23

5. i=n/16--------n/24

.

k. ------------n/2k-1

k+1------------n/2k

Approx at kth iteration the condition will become false. So,

n/2k=1

n=2k

k=log2n

C(n)=log2n