**Question 1**

Consider the following code snippet

for(int j=0; j<n; j\*=2){

System.out. println(i);

}

Find the time complexity of the above code snippet

Solution

Here the loop will run (log n)-1 times

1st iteration, j = 1

2nd iteration, j = 2

3rd iteration, j = 4

4th iteration, j = 8

….

kth iteration, j = 2 ^ (k-1)

So, 2 ^ (k-1) < n

now taking log on both sides

log (2 ^ (k-1)) < log n

k-1 = log n

Time Complexity: O(log n)

----------------------------------------------------------------------------------------------------------------

**Question 2**

Find the time complexity of the following equation

T(n) = T(√n ) + 1

Solution

Let's assume T(2) = 1

T(2^2)=T(2)+1=2

T(2^(2^2))=T(4)+1=3

T(2^(2^3))=T(16)+1=4

T(2^(2^4))=T(256)+1=5

So, we are getting T(n)=lglgn+1⟹T(n)=O(loglogn)

----------------------------------------------------------------------------------------------------------------

**Question 3**

T(n) = 2T(√n) + log n.

Let n = 2 m ⇒ m = log n

Then T(2m) = 2T(2m/2 ) + m.

Now let S(m) = T(2m).

Then S(m) = 2S(m/2) + m.

This is case-2 of master theorem and has the solution

S(m) = O(m log m).

So T(n) = T(2m)

S(m) = O(m log m) = O(log n log log n).

----------------------------------------------------------------------------------------------------------------

**Question 4**

int a = 0;

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

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

a = a + i + j;

}

}

The above code runs total no of times

= N + (N – 1) + (N – 2) + … 1 + 0

= N \* (N + 1) / 2

= 1/2 \* N^2 + 1/2 \* N

O(N^2) times.

----------------------------------------------------------------------------------------------------------------

**Question 5**

int i, j, k = 0;

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

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

k = k + n / 2;

}

}

j keeps doubling till it is less than or equal to n. Several times, we can double a number till it is less than n would be log(n).

Let’s take the examples here.

for n = 16, j = 2, 4, 8, 16

for n = 32, j = 2, 4, 8, 16, 32

So, j would run for O(log n) steps.

i runs for n/2 steps.

So, total steps = O(n/ 2 \* log (n)) = O(n\*logn)

----------------------------------------------------------------------------------------------------------------