**Logarithms**

34=3X3X3X3=81

So,

34=81

Here,

**Log381=4** where 3 is the base and power of 81 and answer is 4

Log5125=X

5x=125

Here x=3

Log232=X

Then

2X=32

So X=5

Another

Log41=Y

4y=1

So Y=0

(Any Non-Zero number)0=1

So Y=0

**Code analysis clues**

1. To check whether a number is power of 2 we can use logarithms. We know about log2 which we can use for power of 2

if the unknown number is assumed n then

ceil(log2(n))==floor(log2(n))

As we know

Log2N=2N Here ceil and floor is to take the number to real integer number

1. We can also check from by using a loop the loop will continue as long as n%2==0 while n/=2

The code will be this

While(n%2==0) n/=2

When the code will be keep being divided by 2 if it is power of 2 the the final value will be 1. Here the time complexity will be O(n)

1. We can also check whether a number is power of 2 using bit manipulation

Let’s consider the binary presentation of any number that is a power of 2.

We’ll find only one bit is set in their binary representation.

2=00000010

4=00000100

8=00001000

16=0001000

And number which is one less has following binary representation

3=00000011

7=00000111

If we do the bitwise AND of a number which is a power of 2 and a number which is one less than the number (one less than the number which is a power of 2) the result will be 0.

0 0 0 0 0 1 0 0 (Binary representation of 4)

& 0 0 0 0 0 0 1 1 (Binary representation of 3)

0 0 0 0 0 0 0 0

So if a number is power of 2 which is consider n then the number of n AND n-1 will be 0 and if not then n AND n-1 will not be 0.

The code will be

return n>0 &&((n&(n-1))==0);

1. Check whether x is power of y

It can easily be solved by log If we take log(y)/log(x) and store it into integer variable and then store it into double variable, if the x is power of y then both variable will be same or not.