**LC#231: POWER OF TWO**

**APP: if it is a power of two then the n & n-1 should be 0**

**Example = 16 ->(10000)**

**15 ->(01111)**

* **If(n&(n-1)) == 0**

class Solution {

    public boolean isPowerOfTwo(int n) {

        if(n<=0)

            return false;

        return (n&n-1)==0;

      }

}

**APP2: (/2 AND CHECK IF N==1 AT LAST)**

**TC: O(LOG N) SC:O(1)**

boolean isPowerOfTwoBrute(int n) {

if (n < 1) return false;

while (n % 2 == 0) n /= 2;

return n == 1;

}

**LC#326: POWER OF THREE**

**APP1:**

boolean isPowerOfThreeBrute(int n) {

if (n < 1) return false;

while (n % 3 == 0) n /= 3;

return n == 1;

}

**APP2:**

class Solution {

    public boolean isPowerOfThree(int n) {

        return (int)(Math.log10(n)/Math.log10(3))==(double)(Math.log10(n)/Math.log10(3));

    }

}

**APP3:**

boolean isPowerOfThreeOptimal(int n) {

**int maxPowerOf3 = 1162261467; // 3^19 < 2^31-1**

return n > 0 && maxPowerOf3 % n == 0;

}

**LC#342:POWER OF FOUR:**

**APP1:**

boolean isPowerOfFourBrute(int n) {

if (n < 1) return false;

while (n % 4 == 0) n /= 4;

return n == 1;

}

**APP2:**

class Solution {

    public boolean isPowerOfFour(int n) {

       return (int)(Math.log10(n)/Math.log10(4))==(double)(Math.log10(n)/Math.log10(4));

    }

}