***Week – 4 (19.04.2021 – 25.04.2021)***

***CODES IN PDF***

1. ***Reverse Bits:***

class Solution {

public:

uint32\_t reverseBits(uint32\_t n) {

int i, t;

uint32\_t rev=0;

for(i=31; i>=0; i--)

{

rev = rev + ((1 & n) << i);

n = n>>1;

}

return rev;

}

};

1. ***Power of Two:***

class Solution {

public:

bool isPowerOfTwo(int n) {

if(n <= 0) return false;

int i;

for(i=0; i<32; i++)

if(n & (n-1)) return false;

return true;

}

};

1. ***Prime Number of Set Bits in Binary Representation:***

class Solution {

public:

bool isprime(int n)

{

if(n<2) return false;

if(n%i == 0) return false;

return true;

}

int countPrimeSetBits(int L, int R) {

int i, bit, count = 0, n;

for(i=L; i<=R; i++)

{

bit = 0;

n = i;

while(n)

{

if(n & 1 == 1) bit++;

n = n>>1;

}

if(isprime(bit)) count++;

}

return count;

}

};

1. ***Count number of bits to be flipped to convert A to B:***

int flipatob(int a, int b)

{

int c = a ^ b, count = 0;

while(n)

{

count++;

n &= (n-1);

}

return count;

}

1. ***Compute the integer absolute value (abs) without branching:***

int absolute(int a)

{

int mask = a>>31;

return ((a ^ mask) - mask);

}

1. ***Turn off the rightmost set bit:***

int rightmostsetbit(int a)

{

return (n&(n-1));

}

1. ***Position of rightmost set bit:***

int posoffirstsetbit(int a)

{

return log2 (n&-n) + 1;

}

1. ***Add two numbers without using arithmetic operators:***

int add(int a, int b)

{

int carry;

while(y)

{

carry = a & b;

a = a ^ b;

b = carry << 1;

}

return x;

}

1. ***Rotate bits of a number:***

int rotate(int a, int d)

{

a = (a << d)|(a >> (32 - d));

return (a >> d)|(a << (32 - d));

}

1. ***Swap all odd and even bits:***

int swapevenoddbits(int a)

{

int odd = a & 0xAAAAAAAA, even = a & 0x55555555;

odd = odd >> 1;

even = enen << 1;

return (even | odd);

}

1. ***Perform nibble wise swap:***

int swapNibbles(int a)

{

return ((a&0x0F)<<4 | (a&0xF0)>>4);

}

1. ***Find position of the only set bit:***

int setbitpos(int a)

{

if (!(a&&(!(a&(a-1))))) return -1;

int i = 1, pos = 1;

while (!(i & a)) {

i = i << 1;

++pos;

}

return pos;

}