1. Find first set bit

unsigned int getFirstSetBit(int n)

{

int pos = 1;

while(n>0)

{

if(n&1) return pos;

pos++;

n>>=1;

}

return 0;

}

1. Rightmost different bit

int posOfRightMostDiffBit(int m, int n)

{

int pos = 1;

while(m>=0&&n>=0&&!(m==0&&n==0))

{

if((m&1)^(n&1)) return pos;

m>>=1;

n>>=1;

pos++;

}

return -1;

}

1. Check whether kth bit is set or not

bool checkKthBit(int n, int k)

{

return (n>>k)&1;

}

1. Count total set bits

int countSetBits(int n)

{

if(n<1) return 0;

int k = floor(log2(n+1));

return k\*pow(2,k-1) + n+1-pow(2,k)+countSetBits(n-pow(2,k));

}

1. Bit Difference

int countBitsFlip(int a, int b){

int count = 0, c = a^b;

while(c>0)

{

count+=(c&1);

c>>=1;

}

return count;

}

1. Number is sparse or not

bool isSparse(int n)

{

if(n==1) return true;

int s1 = n&1, s2=(n>>1)&1;

n>>=1;

while(n>0)

{

if (s1&s2) return false;

s1 = s2;

n>>=1;

s2 = n&1;

}

return true;

}

1. Longest consecutive c1’s

int maxConsecutiveOnes(int N)

{

int count=0, maxcount = 0;

while(N>0)

{

if(N&1) count++;

else

{

maxcount = max(count, maxcount);

count = 0;

}

N>>=1;

}

return max(maxcount,count);

}

1. Binary to gray code equivalent

int greyConverter(int n)

{

return n^(n>>1);

}

1. Gray to binary equivalent

int grayToBinary(int n)

{

int res = 0;

while(n)

{

res^=n;

n>>=1;

}

return res;

}

1. Power of two

bool isPowerofTwo(long long n){

if(n==0) return false;

return !(n&(n-1));

}

1. SWAP ALL EVEN AND ODD BITS

unsigned int swapBits(unsigned int n)

{

unsigned int maxi = 1;

while(maxi<1000000000)

{

maxi=(maxi<<2)+1;

}

unsigned int mini = maxi>>1;

return ((n&maxi)<<1)+((n&mini)>>1);

}

1. Maximum and value

int maxAND (int arr[], int N)

{

int k = log2(\*max\_element(arr,arr+N)), count=0;

if(k<0) return 0;

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

{

if(arr[i]>=pow(2,k))

count++;

}

if(count==1)

{

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

arr[i]=arr[i]&int(pow(2,k)-1);

return maxAND(arr, N);

}

int cpy[count], j=0;

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

{

if(arr[i]>=pow(2,k))

{

cpy[j++]=arr[i]&int(pow(2,k)-1);

}

}

return pow(2,k)+maxAND(cpy, count);

}