E Round Numbers

题意：求区间[start, finish]里 Round Number 的个数(二进制中 0 的个数大于等于 1 的个数的数称为 Round Number).

思路：记 f(start, finish)为[start, finish]里 Round Number 的个数,那么要求 f(start,finish),只需用 f(0,finish)-f(0,start-1)即可,则问题转化为给定 x,求出 f(0,x).

假定 x=(10101101),其长度为 8 位.而[0,x]中的数可分为二进制长度小于 8 位的和二进制长度等于 8 位的.

首先看二进制长度小于 8 位的,即求出长度在[0,7]区间内的 Round Number 个数.对于长度为 len 的二进制（最高位一定是 1）,记其 Round Number 个数为 R(len),可按照 len 的奇偶性分两种情况.

1. len=2k+1 时,去掉最高位的 1,剩下 2k 位里至少要有 k+1 个 0,用 C(n,m)表示 n 个位置选 m 个位置的方法.

有 R(len)=C(2k,k+1)+C(2k,k+2)+...+C(2k,2k)=1/2*(2^(2k)-C(2k,k))

2. 同理可得,len=2k 时,R(len)=1/2*(2^(2k-1))

接下来看二进制长度为 8 位的.

首先判断 x 本身,然后对于 x 的二进制,若将其中除了最高位的 1 以外的任意一个或多个 1 变为 0,得到的数一定小于 x,那么就能通过这种方法得到二进制位数和 x 相等且比 x 小的 Round Number 个数了.

最后将两部分结果求和即可.

精度方面,2000000000<2*1024*1024*1024=2^31,故用 31 位表示数组,又第一位总为 1,所以组合数只用求到 30,C(30,k) (0<=k<=30) <= 2^30,故用 int 即可.

Problem: 3252 **User:** zhyu
Memory: 392K **Time:** 0MS
Language: G++ **Result:** Accepted

```
#include <cstdio>

const int N = 31;

int c[N][N],pow2[N];
bool bin[N];

inline void init(void)
{
    for(int i=0;i<N;i++)   c[i][0]=c[i][i]=1,pow2[i]=(1<<i);
    for(int i=2;i<N;i++)
        for(int j=1;j<i;j++)
            c[i][j]=c[i-1][j-1]+c[i-1][j];
}
```

```
int gao(int x)
{
    if(x<=1)        return 0;
    int i,j,k,n1,n0,len,res=0;
    for(i=0;i<N;i++)        bin[i]=((pow2[i]&x)!=0);
    for(i=N-1;i>=0 && bin[i]==0;i--);
    for(len=i;len>=1;len--)
        if(len&1)       res+=((pow2[len-1]-c[len-1][(len-1)/2])>>1);
        else    res+=(pow2[len-1]>>1);
    for(j=i,n0=n1=0;j>=0;j--)
        if(bin[j])      n1++;
        else    n0++;
    if(n1<=n0)      res++;
    for(j=i-1,n0=0,n1=1;j>=0;j--)
        if(bin[j])
        {
            for(k=j;k>=0 && k+n0+1>=j-k+n1;k--)     res+=c[j][k];
            n1++;
        }
        else    n0++;
    return res;
}
int main(void)
{
    init();
    int st,ed;
    scanf("%d%d",&st,&ed);
    printf("%d\n",gao(ed)-gao(st-1));
    return 0;
}
```