**Nama : Fajar Satria**

**NRP : 05111940000083**

**Kelas : B**

**Pengganti Kuis 02**

a. **TRIMOD2 – The Triangle of Pascal Modulo 2**

**PERMASALAHAN**

Consider Pascal's triangle modulo 2. The first nine rows are given below:

```
        1
       1 1
      1 0 1
     1 1 1 1
    1 0 0 0 1
   1 1 0 0 1 1
  1 0 1 0 1 0 1
 1 1 1 1 1 1 1 1
1 0 0 0 0 0 0 0 1
```

Let $F(n)$ be the number of 1 in the first $n$ rows. So that $F(0) = 0$, $F(1) = 1$, $F(2) = 3$, etc. Given $a$ between 0 and $10^{18}$, find the smallest integer $n$ such that $F(n) \geq a$. Let $N(a)$ denote this integer.

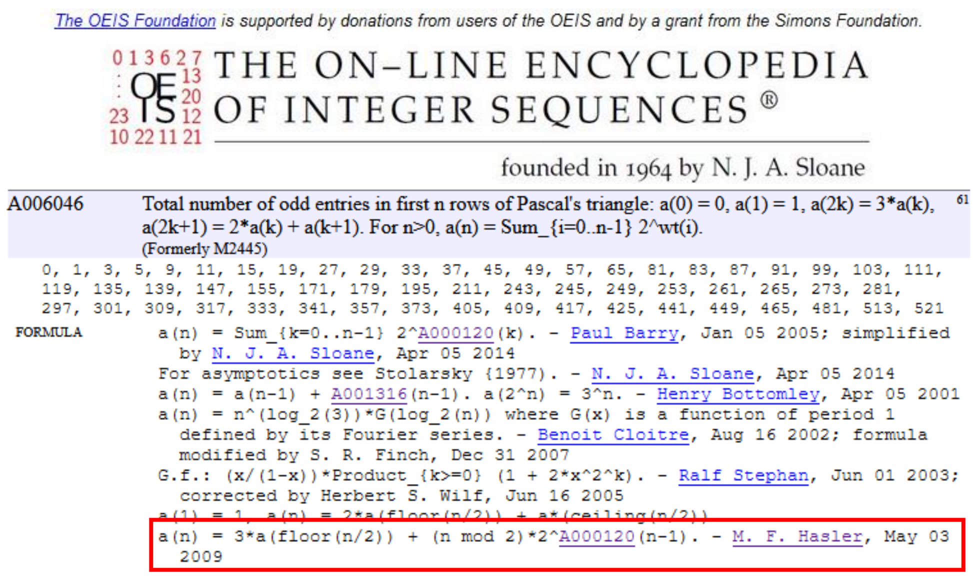
**SOLUSI**

Persoalan ini tampak mudah, seperti hanya rekursi biasa. Namun kompleksitas waktunya akan sangat bermasalah karena *constraint* yang besar yaitu 10^18. Dan juga dengan jumlah testcase yang tidak diketahui.

Solusi pertama yang saya pikirkan adalah dengan menggunakan DP. Yakni menghitung jumlah total angka 1 dari baris pertama hingga ke n dari segitiga pascal mod 2 yang dihasilkan. Namun, solusi ini masih memiliki kompleksitas waktu yang cukup besar yaitu **O(2n log n).**

Solusi kedua yang saya pikirkan adalah dengan menggunakan DP, namun penggunaannya berbeda. Jika sebelumnya menghitung jumlah total, pada solusi ini saya menghitung jumlah angka 1 perbaris dari segitiga pascal mod 2 tersebut. Namun, lagi lagi solusi ini masih memiliki kompleksitas waktu yang cukup besar yaitu **O(nlogn).**

Solusi ketiga saya adalah dengan prinsip binary search. Namun binary search tersebut harus dibarengi dengan sebuah fungsi hitung. Dengan berbekal deret yang saya dan Referensi OEIS. Pada situs ini, saya mendapatkan sequence dengan kode A006046 yang identik dengan deret yang saya dapatkan pada solusi sebelumnya.



Dengan bantuan situs tersebut didapatkan Formula

$$a(n) = 3 * a\left(floor\left(\frac{n}{2}\right)\right) + (n \: mod \: 2) * 2\text{^}A000120(n-1)$$

dimana A000120 merupakan sequence lain dari situs tersebut.

A000120 1's-counting sequence: number of 1's in binary expansion of n (or the binary weight of n). 1461
(Formerly M0105 N0041)

0, 1, 1, 2, 1, 2, 2, 3, 1, 2, 2, 3, 2, 3, 3, 4, 1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3,
4, 4, 5, 1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5, 2, 3, 3, 4, 3, 4, 4, 5, 3, 4,
4, 5, 4, 5, 5, 6, 1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5, 2, 3, 3, 4, 3, 4, 4,
5, 3, 4, 4, 5, 4, 5, 5, 6, 2, 3, 3, 4, 3, 4, 4, 5, 3 (list; graph; refs; listen; history; text; internal format)

Dari penjelasan pada situs tersebut terlihat bahwa deret tersebut adalah sebuah deret dari jumlah angka 1 pada representasi biner sebuah bilangan. Misal dari angka integer **10** maka representasi binernya adalah **1010**, jadi total angka 1 nya ada **2**. Dikarenakan formula yang disediakan terlalu panjang, saya memikirkan solusi lain untuk membuat ulang deret tersebut. Yakni dengan membuat fungsi **countSetBits** dengan memanfaatkan **bitwise and** dan **bitwise shift right**.

countSetBits(n)

```
cnt = 0
while n>0
    cnt = cnt + (n & 1)
    n = n >> 1
return cnt
```

Kembali ke Formula sebelumnya,

$$a(n) = 3 * a\left(floor\left(\frac{n}{2}\right)\right) + (n \: mod \: 2) * 2\text{^}A000120(n-1)$$

karena fungsi A000120 sama dengan fungsi countSetBits, maka

$$a(n) = 3 * a\left(floor\left(\frac{n}{2}\right)\right) + (n \: mod \: 2) * 2\text{^}\mathrm{countSetBits}(n-1)$$

karena formula tersebut merupakan formulai rekursi, maka dapat dibuat sebuah fungsi baru dengan terminate case pada **n=0** dan **n=1**.

pencari(n)

```
if n==0
    return 0
else if n==1
    return 1
else
    return 3 * pencari( n / 2 ) + (n mod 2)* pow(2 , countSetBits( n – 1 ))
```

Setelah selesai dengan kedua fungsi tersebut. Binary search dapat digunakan dengan batas atas high = *(sebuah n yang memenuhi **pencari(n)>=10^18**)*, maka didapatkan **high = 250672682987** dan **low = 0** dan juga **mid = (low + high)/2**. Untuk mengatur **low/high**, diperlukan pemanggilan terhadap fungsi **pencari(mid)** jika hasilnya kurang dari **a** (bilangan yang dicari), maka **low=mid**, selain itu **high=mid**. Hal tersebut dilakukan secara terus menerus sampai **high-low<=1**. Sehingga **high** akan menjadi hasil dari binary search tersebut.

Karena penggunaan binary search tersebut, dan juga penggunaan rekursi dengan formula yang tepat maka solusi tersebut menghaisilkan kompleksitas **O(3logn)**. Mungkin saja ada solusi yang lebih optimal, namun saya belum menemukannya.

**PSEUDOCODE**

countSetBits (n)

```
cnt = 0
while n>0
    cnt = cnt + (n & 1)
```

```
        n = n >> 1
    return cnt
```

pencari(n)

```
    if n==0
        return 0
    else if n==1
        return 1
    else
        return 3 * pencari( n / 2 ) + (n mod 2)* pow(2 , countSetBits( n – 1 ))
```

solve(a)

```
    if a<=0
        return 0
    else
        high = 250672682987
        low = 0
        while high – low > 1
            mid = (low + high) / 2
            if pencari(mid) < a
                low = mid
            else
                high = mid
    return high
```

## SOURCECODE

```cpp
#include <iostream>
#include <cmath>
using namespace std;
typedef long long ll;
typedef unsigned long long ull;
ll countSetBits(ll n){
    ll count = 0;
    while(n){
        count += n & 1;
        n >>= 1;
    }
    return count;
}
ull pencari(ull i){
    if(i==0) return 0;
    else if(i==1) return 1;
    return 3*pencari(i/2)+(i%2)*(ull)pow(2,countSetBits(i-1));
}
ll cari,high,low,mid;
int main(){
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);cout.tie(NULL);
    while(cin>>cari){
        if(cari<=0) cout << "0\n";
        else{
            high=250672682987,low=0,mid;
            while(high-low>1){
                mid = (low+high)/2;
                (pencari(mid)<cari?low:high)=mid;
            }
            cout << high << '\n';
        }
    }
    return 0;
}
```

## SUBMISSIONS

| ID | | DATE | PROBLEM | RESULT | TIME | MEM | LANG |
|---|---|---|---|---|---|---|---|
| 28095050 | | 2021-06-22 17:01:24 | The triangle of Pascal modulo 2 | accepted<br>edit ideone it | 0.04 | 5.2M | CPP |