[2019/1/10]

一. basic Alg

56-01: 数组中只出现一次的两个数，其余均出现两次.

思路: <1> 偶次异或为0.

step1: 全部异或结果r —— 说明=者异或至少一个bit为1，不妨依最后一个bit将数组分两part

step2: r从右第1个1的位置 <2> 各part 1&有一个目标数.

<3> 各part 内部异或结果即各自目标数.

```
void FindNumsAppearOnce(int a[], int len, int *n1, int *n2)
{
    判输入....
    int resultOR=0;   // O(1) 空间开销
    for(     <len     )
        resultOR ^=a[i];          → 整int操作(充分运用了内部bit规律)
    unsigned int loc = FindFirstBit1(resultOR);   → bit操作
    *n1=0, *n2=0;
    for(int j=0; j<len; j++)
    {
        if( part1(a[j], loc))     → bit操作
            *n1 ^= a[j];
        else                      → 整int操作
            *n2 ^= a[j];
    }
}
```

DATE / /

如何找出一个int数的最右位bit是1的index.

```
unsigned int FindFistBit1(int result)
{
    int index = 0;   index = result
    while((result & 1) == 0) && (indexBit < 8 * sizeof(int)))
    {
        result = result >> 1;
        ++index;  // 控制不超过整数int的总bit数
    }
    return index;
}

bool partI(int num, unsigned int index)
{
    num = num >> I;
    return (num & 1);  // 规则同 FindFirstBit1.
}
```

只有当result最后一位为1时循环才不为1，即第index位为1.

引: 如何按bit生成一个int数.

关键: 按bit取出int中各位: &

DATE / /

56_02: 数组中只出现一次的数，其余出现三次。

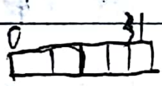
思路: 仿照01 应从bit角度考虑. 难点: 取出a[i]各位并对应累加

<1> 数组中所有元素各位之和S → 位运算中和

<2> S各bit位若能被3整除, target该位为0

否则, target该位为1

```
int FindAppearOnce(int a[], int len)
{
    int bitSum[32] = {0};
    for (int i=0; i<len; i++)    // 如何取各bit! 类比FindFirstBit1 中81.
    {
        int bitMask = 1;
        for(int j=31; j>=0; j--)
        {
            int bit = a[i] & bitMask;
            if (bit != 0)
                bitSum[j] += 1;
            bitMask = bitMask << 1;
        }
    }   // 完成了第一步. 各bit元素和 → bitSum数组.
    int result = 0;
    for(i=0; i<32; i++)
    {
        result = result << 1;
        result += bitSum[i] % 3;   // 从高到低位依次生成.
    }
    return result;
}
```



DATE / /

57-01: 从按增序排列的数组中找出和为S的两数.

思路: 增序便于移动指针 找合适大小的数 [对向型]

二分思路, 双指针得用起来

(Note: 可能找不到哟!)

```
bool FindNumWithSum(int a[], int len, int sum, int *n1, int *n2)
{
    bool found = false; // 布尔值初始化
    if(len<=0 || n1==NULL || n2==NULL)
        return found;

    int ps = 0;
    int pe = len-1;
    while(ps<pe)
    {
        long long curSum = a[ps]+a[pe];
        if(curSum == sum)
        {
            *n1 = a[ps];
            *n2 = a[pe];
            found = true;
            return found;
        }
        else if(curSum<sum)
            ps++;
        else
            pe--;
    }
    return found;
}
```

57-02: 和为S的连续正数序列

思路：仿照以两个指针，依据curSum和S的关系选择移动两指针。

区别：curSum求法不像以前简单，改变针后内部自动更新
02的curSum可能用small更新，可能用big更新
不是通过指针这种隐晦方式，故别落加减?!

```
void FindContinuousSequence(int sum)
{
    if(sum<3)
        return;
    int small=1;
    int big=2;
    int mid=(1+sum)/2;   // 依题意至少两个数推得
    int curSum= big+small;
    while(small<mid)
    {
        if(curSum==sum)
            Print(small, big);
        while((curSum>sum)&&(small<mid))
        {
            curSum-=small;   // 去掉更小的small
            small+=1;
            if(curSum==sum)
                print(small,big);
        }
        big++; curSum+=big;
    }
}   // 从开始的2个元素扩大，small和big加到curSum微妙关系，另找解法
```

通过while一步到位。
>sum和<sum应是交替出现。

理解！