## B. Reverse Game

- chenjiuri20221026

[B. Reverse Game]

```
在s串中，选择相关的子串，然后将它们反转。
形如10 100 0101 110
```

- 分析大概题意
  - 最终的结果必然是 000001111这种形式的。
- 10min

```
问题转换：
把0前移，或者把1向后移动。
发现不同的操作中，对1前后移动的贡献为1或者为2，。
同时注意到总可以进行为2的操作。
除非到达最后的临界状态。
考虑每一个元素移到该到达的位置上。
计算最终的总贡献。
如果恰好是3的倍数。bob一直保持让剩余贡献是3的倍数，必然胜利。
如果不是，alice，同上操作。
```

## $E. Divisible by 3$

```
对于个数组找子段，满足两两之间相乘的和为3的倍数，计算满足该条件的序列的个数。
```

## 20MIN

- 向下遍历过程中，关注一些有意义的前缀。

$$\text{假设当前算到某一步：}$$
$$sum_i;$$
$$\text{前面维护了} sum_{i-1}, time_{i-1};$$
$$\text{可以通过两个前缀来计算，前面的任何一段} time_{(l,r)}$$
$$\text{关注前面一段} k; (time_{k+1,i})$$
$$time_{(k+1,i)} = time_{(1,i)} - time_{(1,k)} - (sum_{(1,i)} - sum_{(1,k)}) * sum_{(1,k)}$$
$$time(k+1,i)\%3 = 0;$$
$$\text{则}(time_{1,k} - sum_k^2 + sum_i * sum_k)\%3 = time_{1,i}\%3; //$$

## 题解

- $f_{i,j,k}$分别表示，以$i$为右边界，$sum\%3=j$，$time\%3=k$的子区间的个数。
- 一六种区间接上，更新构造出来的类型。

```
#include <bits/stdc++.h>
using namespace std;

void MAIN();
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(nullptr), cout.tie(nullptr);
    MAIN();
}
typedef long long ll;
const int maxn = 2e5 + 10;
//------code-----٩(´ω`*)و ------靓仔代码-----٩(´ω`*)و ----talk is cheap , show me
the code--------

ll f[2][4][4];
int now = 0;

void MAIN()
{
    int n;
    cin >> n;
    ll ans = 0;
    for (int i = 1; i <= n; i++)
    {
        ll t;
        cin >> t;
        t %= 3;
        for (int j = 0; j < 3; j++)
            for (int k = 0; k < 3; k++)
                f[now][j][k] = 0;
        for (int j = 0; j < 3; j++)
            for (int k = 0; k < 3; k++)
                f[now][(j + t) % 3][(k + j * t) % 3] += f[now ^ 1][j][k]; //怎么
进行初始化？
        f[now][t % 3][0]++;
        ans += f[now][0][0] + f[now][1][0] + f[now][2][0];
        now ^= 1;
    }
    cout << ans << '\n';
}
```

## 生长思考

- 关于连续区间，连续子串的一个计数问题，考虑设计动态规划。屡试不爽。
- 动态规划的设计角度为：
    - 以当前i为结尾的满足某一些性质独立，连续区间。
    - 向下更低一位的区间迁移。
- 我用一个前缀和的角度是否出错？

**前缀记录题解；**

```
#include <bits/stdc++.h>
using namespace std;

void MAIN();
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(nullptr), cout.tie(nullptr);
    MAIN();
}
typedef long long ll;
const int maxn = 5e5 + 10;
//------code-----٩(´ω`*)و ------靓仔代码-----٩(´ω`*)و ----talk is cheap , show me
the code--------
ll cu[10][10];
ll a[maxn];
ll sum[maxn];
ll ttime[maxn];
ll ans = 0;
void MAIN()
{
    int n;
    cin >> n;
    for (int i = 1; i <= n; i++)
    {
        cin >> a[i];
        sum[i] = (sum[i - 1] + a[i]) % 3;
        ttime[i] = (ttime[i - 1] % 3 + sum[i - 1] * a[i] % 3) % 3;
        if (ttime[i] % 3 == 0)//一整串的情况下。
            ans++;
        ans += cu[sum[i] % 3][ttime[i] % 3];
        //自身也是一个贡献。
        for (int j = 0; j < 3; j++) //记录当满足某些性质的子段。
            cu[j][((ttime[i] % 3 - sum[i] * sum[i] % 3 + 3) % 3 + j * sum[i] %
3) % 3]++;
    }
    cout << ans << '\n';
}
```