ad

M. Youth Finale

- chenjiuri_guilin2020
- 先算出初始排列的一个逆序对情况。
- 每一次操作对于前面的变化，一直被保持维护，
- 每一次操作检查后面的情况，都会有一定的变化规律。

补个鬼，我说怎么一直wa。原来是看错题了。

生长思考:

- 第一点，deque的普通操作常数比较小。常数体现大的地方应该是删除。释放

E. Draw a triangle

```
在网格上，有两个坐标点，找到第三个点。
构成一个三角形并且让该三角形的面积最小。
```

$thinking$

- 怎么寻找一个离直线最近的点?
- 这一个并不是简单的找规律问题。别老想着在非签到题上找规律。
  - 设第三个点为u, v
  - 那么由三角形面积计算相关的叉积公式$S = (a \times b) = \frac{1}{2}|xv - yu|$;
  - 然后联想到$exgcd$式子。
-

C. Array Concatenation

```
对于一个数组选择两种操作的一种进行:
第一种:   复制一份，拼接在当前的数组之后。
第二种，   复制一份，做一次翻转，然后拼接在之后。
求取使得数组的前缀和的和ans%mod后最大的值。
```

- 一共有m此操作，次数10^5,且取模运算之后的结果具有一定的随机性。
  - 可以猜测最终有一定的关于下面几点的规律
    - 前缀和之和的计算方法。
    - 最终的结果和取mod运算。
    - 最终的本质不同的解非常的少。
- 发现一旦做出了一个第二种计算之后，操作的后面的第一中操作，和第二种操作相同。
- 归纳出各种情况下的前缀和的计算公式
  - 归纳技巧，将结果分成若干整体的贡献。

- 每一次操作之后，新生成的段的总体贡献。
- 每一个元素贡献。

$$只操作1：f1 = 2^m * hs(前缀和之和) + \frac{(2^m - 1) \times 2^m}{2} n * s;$$

$$两种操作：f2 = 2^{m-1}hs + 2^{m-1}ts + \frac{(2^m - 1) \times 2^m}{2} n * s;$$

code

```
#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 = 1e5 + 10;
//------code-----٩(ˊωˋ*)و ------靓仔代码-----٩(ˊωˋ*)و ----talk is cheap ,
show me the code--------

ll a[maxn];
ll s[maxn];
ll ss[maxn];
ll mod = 1e9 + 7;
ll p2[maxn];

void MAIN()
{
    ll n, m;
    cin >> n >> m;
    for (int i = 1; i <= n; i++)
        cin >> a[i];
    ll hs = 0, ts = 0;
    for (int i = 1; i <= n; i++)
    {
        s[i] = (a[i] + s[i - 1]) % mod;
        hs = (s[i] + hs) % mod;
        ss[n - i + 1] = (a[n - i + 1] + ss[n - i + 2]) % mod;
        ts = (ss[n - i + 1] + ts) % mod;
    }
    ll ans = 0;
    p2[0] = 1;
    for (int i = 1; i <= m; i++)
        p2[i] = (p2[i - 1] * 2) % mod;
```

```
    ll ch1 = (p2[m] * hs % mod + (p2[m] - 1) * p2[m - 1] % mod * n % mod
* s[n]) % mod;
    ll ch2 = (p2[m - 1] * hs % mod + p2[m - 1] * ts + (p2[m] - 1) % mod
* p2[m - 1] % mod * n % mod * s[n] % mod) % mod;
    ans = max(ch1, ch2);
    cout << ans << '\n';
}
```

生长思考

- 可以手写快速幂函数。推广到其它底数的快速幂函数。
- 将一些模板手写。