Problema K - ETH Zurich Competitive Programming Contest Spring 2024

Igor Borja

1 Análise

Note que nenhuma moeda é paga caso ele seja sorteado.

Para que Oli gaste exatamente $q > 0$ moedas, é necessário que

- $(q-1) \cdot n$ pessoas não sejam sorteadas, ($(q-1)$ ciclos completos), para que ele gaste $q-1$ moedas
- As $k-1$ pessoas na sua frente, e ele próprio, não sejam sorteadas (portanto, k pessoas)
- Alguma das $n-k$ pessoas atrás dele seja sorteada, ou ninguém seja sorteado nesse ciclo e uma das $k-1$ pessoas na frente dele seja sorteada (ou ele próprio) no ciclo seguinte. Ou seja, (pensando em posições **zero-indexadas** - sendo que Oli está na $(k-1)$-ésima posição) existe um $0 \leq j \leq n-1$ tal que $k, \ldots, (k+j-1) \mod n$ não são sorteados e $k+j \mod n$ é sorteado, depois das duas condições anteriores.

Ou seja, pondo $p = \dfrac{a}{b}$ temos que a probabilidade de ele gastar exatamente q moedas é:

$$P(X=q) = (1-p)^{(q-1)\cdot n+k} \sum_{j=0}^{n-1} (1-p)^j \, p$$
$$= p\,(1-p)^{(q-1)\cdot n+k} \frac{1-(1-p)^n}{1-(1-p)} = (1-p)^{(q-1)\cdot n+k} \left(1-(1-p)^n\right)$$

Portanto, o valor esperado é dado por:

$$\mathbb{E}[X] = \sum_{q>0} qP(q) = (1-p)^k \left(1-(1-p)^n\right) \sum_{q>0} q\,(1-p)^{(q-1)n}$$

Porém, é fácil mostrar que, para toda P.G de razão x com $|x| < 1$:

$$\sum_{k\geq 0} kx^{k-1} = \frac{d}{dx} \sum_{k\geq 0} x^k = \frac{d}{dx} \frac{1}{1-x} = \frac{1}{(1-x)^2}$$

e portanto:

$$\mathbb{E}[X] = (1-p)^k(1-(1-p)^n)\frac{1}{(1-(1-p)^n)^2} = \frac{(1-p)^k}{(1-(1-p)^n)}$$

$$= \frac{\left(\frac{b-a}{b}\right)^k}{1-\left(\frac{b-a}{b}\right)^n} = \frac{(b-a)^k b^{n-k}}{b^n-(b-a)^n}$$

2 Implementação

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

#define i64 int64_t
const i64 MOD = (i64)1e9 + 7;

i64 bexp(i64 a, i64 ex){
    if (ex == 0) {
        return 1;
    } else {
        i64 b = bexp(a, ex / 2);
        if (ex % 2 == 0){
            return (b * b) % MOD;
        } else {
            return (a * ((b * b) % MOD)) % MOD;
        }
    }
}


int main(){
    i64 n, k, a, b;
    cin >> n >> k >> a >> b;

    i64 num = (bexp(b - a, k) * bexp(b, n - k)) % MOD;
    i64 denum = (bexp(b, n) + MOD - bexp(b - a, n)) % MOD;
    cout << (num * bexp(denum, MOD - 2)) % MOD << endl;
}
```