問題

整数 N, M, P が与えられる．このとき，N^P を M で割った余りを求めよ．

制約

- $1 \leq N, M \leq 10^9$
- $1 \leq P \leq 10^{14}$

出典

AtCoder Typical Contest 002 B

コード解説

normal1.cpp

for文を用いてansにNをP回かける．計算量は$O(P)$．コードはリスト1に示す．

リスト 1　normal1.cpp

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

int main() {
    long long n, m, p;
    cin >> n >> m >> p;
    long long ans = 1;

    // 計算量はO(p)
    for (int _ = 0; _ < p; _++) {
        ans *= n;
        ans %= m;
    }

    cout << ans << endl;
    return 0;
}
```

normal2.cpp

再帰関数を用いた累乗のナイーブな実装．

$$N^X = N^{X-1} \cdot N$$

を用いて再帰的に計算する．計算量は$O(P)$．コードはリスト2に示す．

リスト 2　normal2.cpp

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

// return n^p % m
// 計算量はO(p)
long long pow(long long n, long long p, long long m) {
    if (p == 0) return 1;
    return n * pow(n, p - 1, m) % m;
}

int main() {
    long long n, m, p;
    cin >> n >> m >> p;
    cout << pow(n, p, m) << endl;
    return 0;
}
```

kuhuu1.cpp

繰り返し2乗法の再帰帰数を用いた実装．

$$N^X = \begin{cases} N^{X/2} \cdot N^{X/2} & (X\text{が偶数}) \\ N^{(X-1)/2} \cdot N^{(X-1)/2} \cdot N & (X\text{が奇数}) \end{cases}$$

を用いて再帰的に計算する．計算量は$O(\log P)$．コードはリスト3に示す．

リスト 3　kuhuu1.cpp

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

// return n^p % m
// 計算量はO(log(p))
long long power(long long n, long long p, long long m) {
    n %= m;
    if (n < 0) n += m;
    if (p == 0) return 1;
    long long res = power(n, p / 2, m);
    if (p % 2 == 0) return (res * res) % m;
    else return (res * res) % m * n % m;
}

int main() {
    long long n, m, p;
    cin >> n >> m >> p;
    cout << power(n, p, m) << endl;
    return 0;
}
```

kuhuu2.cpp

繰り返し2乗法の非再帰的な実装．

$$X = \sum_{i=0}^{h} a_i \cdot 2^i,$$

のように書けたとき，

$$N^X = \left(N^{2^0}\right)^{a_0} \cdot \left(N^{2^1}\right)^{a_1} \cdot \cdots \cdot \left(N^{2^h}\right)^{a_h}$$

という性質 (分配法則です．) を使い，N^{2^i}を計算しながら，$a_i = 1$の時に，ansにN^{2^i}をかける．

$$N^{2^{i+1}} = N^{2^i} \cdot N^{2^i}$$

であるから，N^{2^i}は簡単に計算できる．$h = \lceil \log_2(X) \rceil$だから，計算量は$O(\log P)$．コードはリスト$4$に示す．

リスト 4　kuhuu2.cpp

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

int main() {
    long long n, m, p;
    cin >> n >> m >> p;
    long long tmp = n % m, ans = 1, exp = p;
    // 再帰を使わない方法
    // 計算量はO(log(p))
    while (exp) {
        if (exp & 1) {
            ans *= tmp;
            ans %= m;
        }
        tmp *= tmp;
        tmp %= m;
        exp >>= 1;
    }
    cout << ans << endl;
    return 0;
}
```

計測

実行時間の測定は，AtCoderのコードテストで行った．ローカル環境だと，PCの個体差が出てしまうと考えたためである．言語はC++ 23 (gcc 12.2)を用いている．また，見やすくするために，normal2とkuhuu1のコードを比較し (どちらも再帰．)，計算量に強く依存しているP以外のN, Mは固定する．ここでは，$N = 3, M = 100$とした．計測時間の比較を表1に示す．

表1　実行時間の比較

P	normal (ms)	kuhuu (ms)	normal/kuhuu
10^1	1	1	1
10^2	1	1	1
10^3	1	1	1
10^4	1	1	1
10^5	2	1	2
10^6	11	1	11
10^7	108	1	108
$8 \cdot 10^7$	851	1	851

normal は $P = 9 \cdot 10^7$ でスタックオーバフローでエラーを吐いた．$P = 10^5$ 程度までならば，normal と kuhuu の実行時間はほぼ変わらない．しかし，$P = 10^7$ 程度までになると，その比は 100 倍以上になっている．kuhuu は $P = 10^{18}$ まで計測したが，ずっと $1\mathrm{ms}$ であった．

まとめ

繰り返し2乗法の実装は，再帰関数を用いたものと，用いなかったもののどちらも，Github Copilot に大まかに実装してもらい，少し修正を加えている (余りを取るところ．)．

参考記事

- 「998244353 で割ったあまり」の求め方を総特集！ 〜 逆元から離散対数まで 〜 4章 累乗