4.6 節末問題 4.6 的解答

問題 4.6.1 (1)

由於乘法運算中，無論在哪個時間點取餘數，答案都不會改變，因此下列兩者相等。

- $21 \times 41 \times 61 \times 81 \times 101 \times 121$ 除以 20 的餘數
- $1 \times 1 \times 1 \times 1 \times 1 \times 1$ を除以 20 的餘數

後者顯然為 1，所以本題的答案是 1 。

問題 4.6.1 (2)

即使在計算前取所有數字除以 100 的餘數，答案也不會改變，因此以下會一致。

- 202112^5 除以 100 的餘數
- 12^5 除以 100 的餘數

由於 $12^5 = 248832$，答案是 32，但手動計算有點麻煩。因此，可以在計算過程中不斷取餘數，如下：

- $12 \times 12 = 144 \equiv 44 \pmod{100}$
- $44 \times 12 = 528 \equiv 28 \pmod{100}$
- $28 \times 12 = 336 \equiv 36 \pmod{100}$
- $36 \times 12 = 432 \equiv 32 \pmod{100}$

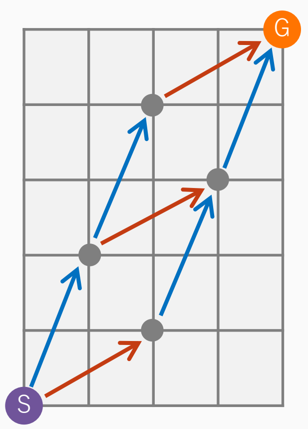
這樣，只需進行最多三位數的計算即可求出答案。

問題 4.6.2

首先從具體例來考慮。要將棋子移動到格子$(4, 5)$，需要進行：

- $(i, j) \rightarrow (i + 1, j + 2)$ 的移動 2 次
- $(i, j) \rightarrow (i + 2, j + 1)$ 的移動 1 次

反過來，只要滿足這個條件，就一定可以到達目的位置。因為在 3 次移動中有 2 次是朝 $(i + 1, j + 2)$ 的移動，所以移動方法的數量是 ${}_3C_2 = 3$種。



接下來，考慮一般的情況。

- $(i,j) \to (i+1, j+2)$ 的移動 a 次（設為移動 A）
- $(i,j) \to (i+2, j+1)$ 的移動 b 次（設為移動 B）

進行以上行動時，為了移動到格子 (X,Y)，需要滿足以下三個條件：。

- a 和 b 是非負整數。
- x 座標的限制： $a + 2b = X$
- y 座標的限制： $2a + b = Y$

在此，同時滿足第 2 及第 3 個條件的只有$(a,b) = \left(\frac{2Y-X}{3}, \frac{2X-Y}{3}\right)$ 這一組解。

然而，依據第一個條件，也有可能成為答案為 0 種的狀況。

- 因為a、b為整數，若 $2Y - X$ 及 $2X - Y$ が不是 3 的倍數，則為 0 種
- 因為a、b為0以上，若 $2Y - X < 0$ 或 $2X - Y < 0$，則為 0 種

並非如此的時候，在整體移動次數 $(X+Y)/3$ 次之中，進行 $(2Y-X)/3$ 次移動 A 的話，一定會到達目的格子，因此所求答案為 ${}_{(X+Y)/3}\mathrm{C}_{(2Y-X)/3}$ 種。將程式碼4.6.5稍微變更如下實作，可得到正解。

```
#include <iostream>
using namespace std;

const long long mod = 1000000007;
int X, Y;

long long modpow(long long a, long long b, long long m) {
    // 重複平方法（p 取如 a^1, a^2, a^4, a^8, ... 的值）
    long long p = a, Answer = 1;
    for (int i = 0; i < 30; i++) {
        if ((b & (1 << i)) != 0) { Answer *= p; Answer %= m; }
        p *= p; p %= m;
    }
    return Answer;
}

// Division(a, b, m) 為傳回 a÷b mod m 的函式
long long Division(long long a, long long b, long long m) {
    return (a * modpow(b, m - 2, m)) % m;
}
```

```
int main() {
    // 輸入
    cin >> X >> Y;

    // 狀況區分（a, b 會變成負的時候）
    if (2 * Y - X < 0 || 2 * X - Y < 0) {
        cout << "0" << endl;
        return 0;
    }

    // 狀況區分（a, b 不會成為整數的時候）
    if ((2 * Y - X) % 3 != 0 || (2 * X - Y) % 3 != 0) {
        cout << "0" << endl;
        return 0;
    }

    // 求出二項係數的分子與分母（步驟 1.／步驟 2.）
    long long bunshi = 1, bunbo = 1;
    long long a = (2 * Y - X) / 3, b = (2 * X - Y) / 3;
    for (int i = 1; i <= a + b; i++) { bunshi *= i; bunshi %= mod; }
    for (int i = 1; i <= a; i++) { bunbo *= i; bunbo %= mod; }
    for (int i = 1; i <= b; i++) { bunbo *= i; bunbo %= mod; }

    // 求出答案（步驟 3.）
    cout << Division(bunshi, bunbo, mod) << endl;
    return 0;
}
```

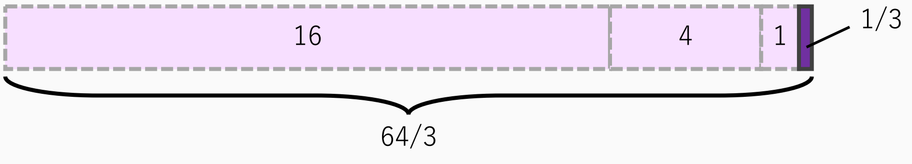
※ Python等原始碼請參閱chap4-6.md。

問題 4.6.3

首先，下式會成立。例如當 $N = 2$ 時， $4^0 + 4^1 + 4^2 = 1 + 4 + 16 = 21$， 同時 $(4^{N+1} - 1)/3 = 63 \div 3 = 21$， 兩個值一致。

$$4^0 + 4^1 + 4^2 + \cdots + 4^N = \frac{4^{N+1} - 1}{3}$$

證明雖然較難，但可以如下思考：若將長度為 $4^{N+1}/3$ 的長條每次切去 $3/4$， 且將此操作重複 $N+1$次，從第 1 次開始依序取出長度 $4^N, 4^{N-1}, \ldots, 4^0$ 的部分，則最終只會剩下長度 $1/3$。（參考：和的公式 →**2.5.10項**）。



したがって、以下のような方法で答えを で割った余りを求めることができます因此，可以用以下方法來求得答案，即除以 $M = 1000000007$ 的餘數：。

- 求出 $4^{N+1} - 1$ 除以 M 的餘數 V（→ **4.6.7項**）
- 求出 $V \div 3$ 除以 M 的餘數（→ **4.6.8項**項）

以下是實作例。其中，函數 `Division(a, b, m)` 是回傳 $a \div b$ 除以 m 的餘數。此外，由於限制是 $N \leq 10^{18}$，注意 modpow 函數的迴圈次數大約需要 $\log_2(10^{18}) \risingdotseq 60$次。

```
#include <iostream>
using namespace std;

const long long mod = 1000000007;
long long N;

long long modpow(long long a, long long b, long long m) {
    // 重複平方法（p 取如 a^1, a^2, a^4, a^8, ... 的值）
    long long p = a, Answer = 1;
    for (int i = 0; i < 60; i++) {
        if ((b & (1LL << i)) != 0) { Answer *= p; Answer %= m; }
        p *= p; p %= m;
    }
    return Answer;
}

// Division(a, b, m) 為傳回 a÷b mod m 的函式
long long Division(long long a, long long b, long long m) {
    return (a * modpow(b, m - 2, m)) % m;
}

int main() {
    // 輸入
    cin >> N;

    // 計算答案
    long long V = modpow(4, N + 1, mod) - 1;
    long long Answer = Division(V, 3, mod);

    // 輸出
    cout << Answer << endl;
    return 0;
}
```

※ Python等原始碼請參閱 chap4-6.md。