5.4 節末問題 5.4 の解答

問題 5.4.1

まず、「一つ以上 6 の目が出ること」の余事象は「すべて 5 以下」であるため、

$$(\text{一つ以上 6 の目が出る確率}) = 1 - (\text{すべて 5 以下となる確率})$$

という式が成り立ちます。そこで、すべて 5 以下となる確率は、積の法則（→**3.3.2 項**）より $(5/6) \times (5/6) \times (5/6) = 125/216$ です。よって、答えは以下の通りです。

$$1 - \frac{125}{216} = \mathbf{\frac{91}{216}}$$

問題 5.4.2

5.4.4 項で解説された通りに実装すれば良いです。実装例として以下が考えられます。なお、各変数・配列は以下のような意味を持っています。

- `gyou[i]`：i 行目の総和
- `retu[j]`：j 列目の総和
- `Answer[i][j]`：i 行目・j 列目のマスに対する答え

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

int H, W, A[2009][2009];
int gyou[2009]; // 行の総和
int retu[2009]; // 列の総和
int Answer[2009][2009];

int main() {
	// 入力
	cin >> H >> W;
	for (int i = 1; i <= H; i++) {
		for (int j = 1; j <= W; j++) cin >> A[i][j];
	}

	// 行の総和を計算する
	for (int i = 1; i <= H; i++) {
```

```
            gyou[i] = 0;
            for (int j = 1; j <= W; j++) gyou[i] += A[i][j];
    }

    // 列の総和を計算する
    for (int j = 1; j <= W; j++) {
            retu[j] = 0;
            for (int i = 1; i <= H; i++) retu[j] += A[i][j];
    }

    // 各マスに対する答えを計算する
    for (int i = 1; i <= H; i++) {
            for (int j = 1; j <= W; j++) {
                    Answer[i][j] = gyou[i] + retu[j] - A[i][j];
            }
    }

    // 空白区切りで出力
    for (int i = 1; i <= H; i++) {
            for (int j = 1; j <= W; j++) {
                    if (j >= 2) cout << " ";
                    cout << Answer[i][j];
            }
            cout << endl;
    }
    return 0;
}
```

※ Python などのソースコードは chap5-4.md をご覧ください。

問題 5.4.3 (1)

一般に、N 以下の整数のうち M の倍数であるものの個数は $\lfloor N/M \rfloor$ 個であるため、

- 3 の倍数は $A_1 = \lfloor 1000 \div 3 \rfloor =$ **333 個**
- 5 の倍数は $A_2 = \lfloor 1000 \div 5 \rfloor =$ **200 個**
- 7 の倍数は $A_3 = \lfloor 1000 \div 7 \rfloor =$ **142 個**
- 15 の倍数は $A_4 = \lfloor 1000 \div 15 \rfloor =$ **66 個**
- 21 の倍数は $A_5 = \lfloor 1000 \div 21 \rfloor =$ **47 個**
- 35 の倍数は $A_6 = \lfloor 1000 \div 35 \rfloor =$ **28 個**
- 105 の倍数は $A_7 = \lfloor 1000 \div 105 \rfloor =$ **9 個**

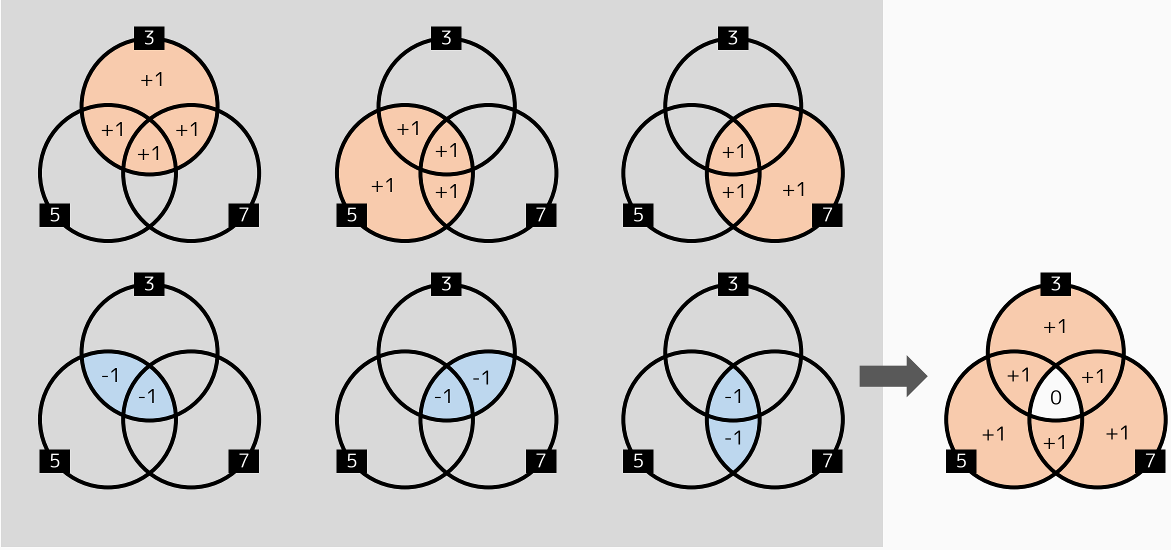
となります。

問題 5.4.3 (2), (3), (4), (5)

(2) $A_1 + A_2 + A_3$ と数えた場合、たとえば **15 という数**が 2 回数えられています。
（A_1 と A_2 両方にカウントされています）

(3) $A_1 + A_2 + A_3 - A_4 - A_5 - A_6$ と数えた場合、**105 という数**はA_1, A_2, A_3 で 3 回分足されていますが、A_4, A_5, A_6 で 3 回分引かれているため、全く数えられていないのと同然です。

なお、下図に示すように、正しく数えられていない数は 105 の倍数のみです。



(4) (3) でカウントされなかった 105 の倍数を足せば良いです。答えは $\boldsymbol{A_1 + A_2 + A_3 - A_4 - A_5 - A_6 + A_7}$ となります。

(5) (1) の答えより、$333 + 200 + 142 - 66 - 47 - 28 + 9 =$ **543 個**です。

問題 5.4.4

集合 $S_1, S_2, S_3, \ldots, S_N$ の和集合（どれか一つでも含まれる部分）の要素数は、以下の式で表されます。

> N 個の集合の中から 1 つ以上を選ぶ方法は $2^N - 1$ 通りあるが、それらすべてに対して以下の値を加算したもの。
>
> - 奇数個選んだとき：選んだ集合の共通部分の要素数
> - 偶数個選んだとき：選んだ集合の共通部分の要素数 × (-1)

たとえば、集合 S_1, S_2, S_3 の和集合の要素数は、以下をすべて足した値です。

- S_1 の要素数
- S_2 の要素数
- S_3 の要素数
- S_1 と S_2 の共通部分 × (-1)
- S_1 と S_3 の共通部分 × (-1)
- S_2 と S_3 の共通部分 × (-1)
- S_1 と S_2 と S_3 の共通部分 × (-1)

S_1 に「1000 以下の 3 の倍数」、S_2 に「1000 以下の 5 の倍数」、S_3 に「1000 以下の 7 の倍数」を当てはめてみてください。問題 5.4.3 と同じ結果になると思います。

問題 5.4.5

集合 S_1 を「N 以下の V_1 の倍数」、集合 S_2 を「N 以下の V_2 の倍数」、…、集合 S_K を「N 以下の V_K の倍数」とするとき、求めるべき答えは $S_1, S_2, \ldots, S_K$ の共通部分となります。

したがって、N 個の集合の選び方（どの倍数を選ぶか）を $2^N - 1$ 通り全探索する以下のようなプログラムを書くと、正解が得られます。ビット全探索（→**コラム1**）という実装方法を使っています。

なお、$P_1, P_2, \ldots, P_M$ すべての倍数である N 以下の整数の個数は、

$$\frac{N}{P_1, P_2, P_3, \ldots, P_M \text{ の最小公倍数}}$$

という式で表されます。（3 個以上の最小公倍数の求め方は →**節末問題3.2.3**）

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

long long N, K;
long long V[20];
long long Answer = 0;

// 最大公約数を返す関数
long long GCD(long long A, long long B) {
    if (B == 0) return A;
    return GCD(B, A % B);
}
```

```cpp
// 最小公倍数を返す関数
long long LCM(long long A, long long B) {
	return (A / GCD(A, B)) * B;
}

int main() {
	// 入力
	cin >> N >> K;
	for (int i = 1; i <= K; i++) cin >> V[i];

	// ビット全探索
	for (int i = 1; i < (1 << K); i++) {
		long long cnt = 0; // 選んだ数の個数
		long long lcm = 1; // 最小公倍数
		for (int j = 0; j < K; j++) {
			if ((i & (1 << j)) != 0) {
				cnt += 1;
				lcm = LCM(lcm, V[j + 1]);
			}
		}
		long long num = N / lcm; // 選ばれた数すべての倍数であるものの個数
		if (cnt % 2 == 1) Answer += num;
		if (cnt % 2 == 0) Answer -= num;
	}

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

※ Python などのソースコードは chap5-4.md をご覧ください。