3.3 節末問題 3.3 の解答

問題 3.3.1

この問題は、場合の数の公式（→**3.3.4項**、**3.3.5項**）の理解を問う問題です。答えは、

$$ {}_2\mathrm{C}_1 = \frac{2}{1} = 2 $$

$$ {}_8\mathrm{C}_5 = \frac{8 \times 7 \times 6 \times 5 \times 4}{5 \times 4 \times 3 \times 2 \times 1} = 56 $$

$$ {}_7\mathrm{P}_2 = 7 \times 6 = 42 $$

$$ {}_{10}\mathrm{P}_3 = 10 \times 9 \times 8 = 720 $$

となります。なお、二項係数 nCr は nPr の $1/r!$ 倍であるため、

$$ nCr = \frac{nPr}{r!} = \frac{n \times (n-1) \times (n-2) \times (n-r+1)}{r \times (r-1) \times (r-2) \times \cdots \times 1} $$

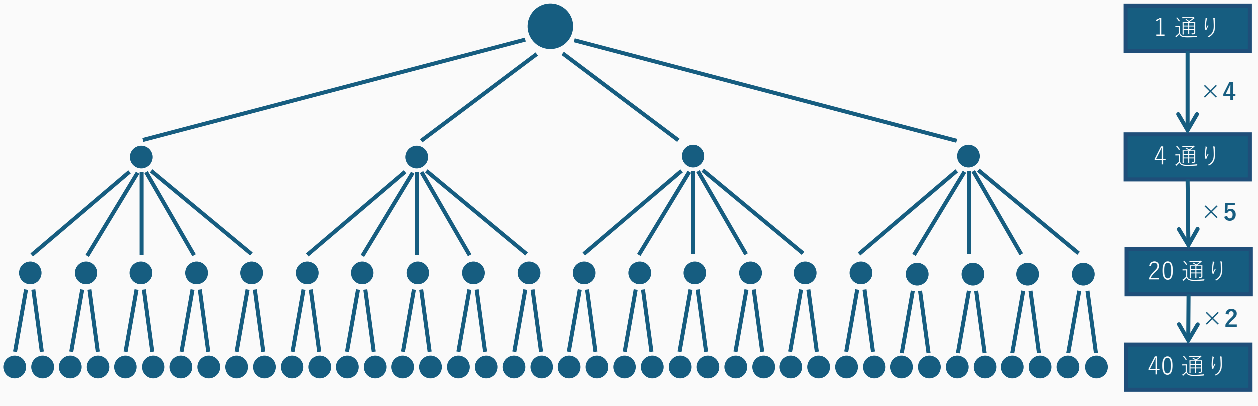
と計算することができます。

問題 3.3.2

この問題は、積の法則（→**3.3.2項**）を理解を問う問題です。

- 大きさの選び方： 4 通り
- トッピングの選び方： 5 通り
- ネームプレートの選び方： 2 通り

あるため、答えは $4 \times 5 \times 2 = 40$ 通りとなります。以下はイメージ図となります。



問題 3.3.3

まず、以下の値を計算しましょう。（$n!$ を計算する方法：→**節末問題 2.5.3**）

- `Fact_n`：$n!$ の値
- `Fact_r`：$r!$ の値
- `Fact_nr`：$(n-r)!$ の値

このとき、求める nCr の値は `Fact_n / (Fact_r * Fact_nr)` となるため、この値を出力するプログラムを書けば正解となります。C++ での実装例は次の通りです。

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

long long n, r;
long long Fact_n = 1;
long long Fact_r = 1;
long long Fact_nr = 1;

int main() {
    // 入力
    cin >> n >> r;

    // 階乗の計算
    for (int i = 1; i <= n; i++) Fact_n *= i;
    for (int i = 1; i <= r; i++) Fact_r *= i;
    for (int i = 1; i <= n - r; i++) Fact_nr *= i;

    // 出力
    cout << Fact_n / (Fact_r * Fact_nr) << endl;
    return 0;
}
```

※ Python などのソースコードは chap3-3.md をご覧ください。

問題 3.3.4

書籍で解説された通りに実装すれば良いです。以下は C++ での解答例となります。なお、この問題は制約が $N \leq 200000$ と大きく、答えが 10^{10} 通りになる可能性があります。`int` 型などの 32 ビット整数では**オーバーフロー**を起こすことに注意してください。（Python の場合は関係ありません）

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

long long N;
```

```
long long A[200009];
long long a = 0, b = 0, c = 0, d = 0; // オーバーフロー回避のため 64 ビット整数を使う

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

    // a, b, c, d の個数を数える
    for (int i = 1; i <= N; i++) {
        if (A[i] == 100) a += 1;
        if (A[i] == 200) b += 1;
        if (A[i] == 300) c += 1;
        if (A[i] == 400) d += 1;
    }

    // 出力（答えは a * d + b * c）
    cout << a * d + b * c << endl;
    return 0;
}
```

※ Python などのソースコードは chap3-3.md をご覧ください。

問題 3.3.5

書籍で解説された通りに実装すれば良いです。以下は C++ での解答例となります。なお、この問題は制約が $N \leq 500000$ と大きく、答えが 10^{11} 通り以上になる可能性があります。

`int` 型などの 32 ビット整数ではオーバーフローを起こすため、`long long` 型などの 64 ビット整数を使うことが推奨されます。（Python の場合は関係ありません）

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

long long N;
long long A[500009];
long long x = 0, y = 0, z = 0;

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

    // a, b, c, d の個数を数える
    for (int i = 1; i <= N; i++) {
        if (A[i] == 1) x += 1;
        if (A[i] == 2) y += 1;
```

```
        if (A[i] == 3) z += 1;
    }

    // 出力
    cout << x * (x - 1) / 2 + y * (y - 1) / 2 + z * (z - 1) / 2 << endl;
    return 0;
}
```

※ Python などのソースコードは chap3-3.md をご覧ください。

問題 3.3.6

まず、$A_1, A_2, \ldots, A_N$ の中に i が何個存在するかを cnt[i] とするとき、cnt[1], cnt[2], …, cnt[99999] は以下のようにして数えることができます。

```
// 配列 cnt[i] を 0 に初期化する
for (int i = 1; i <= N; i++) cnt[i] = 0;

// A[i] が現れたら cnt[A[i]] に 1 を加算する
for (int i = 1; i <= N; i++) cnt[A[i]] += 1;
```

そこで、和が 100000 となる 2 枚のカードの選び方をリストアップすると、

- 1 と 99999 のカードを選ぶ（cnt[1]*cnt[99999] 通り）
- 2 と 99998 のカードを選ぶ（cnt[2]*cnt[99998] 通り）
- 3 と 99997 のカードを選ぶ（cnt[3]*cnt[99997] 通り）
- ：
- 49999 と 50001 のカードを選ぶ（cnt[49999]*cnt[50001] 通り）
- 50000 のカードを 2 枚選ぶ（cnt[50000]*(cnt[50000]-1)/2 通り）

となります。50000 のカードを 2 枚選んだとき、cnt[50000]*cnt[50000] 通りにならないことに注意してください。

したがって、赤く記された値の合計を出力するプログラムを書くと、正解が得られます。以下のプログラムは、C++ での解答例です。

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

long long N, A[200009];
long long cnt[100009];
long long Answer = 0;

int main() {
```

```
    // 入力
    cin >> N;
    for (int i = 1; i <= N; i++) cin >> A[i];

    // cnt[1], cnt[2], ..., cnt[99999] を数える
    for (int i = 1; i <= 99999; i++) cnt[i] = 0;
    for (int i = 1; i <= N; i++) cnt[A[i]] += 1;

    // 答えを求める
    for (int i = 1; i <= 49999; i++) {
        Answer += cnt[i] * cnt[100000 - i];
    }
    Answer += cnt[50000] * (cnt[50000] - 1) / 2;

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

※ Python などのソースコードは chap3-3.md をご覧ください。

問題 3.3.7

まず、スタートからゴールまで最短距離（14 手）で行くための必要十分条件（→**2.5.6項**）は以下のようになります。

> 「上方向に 1 つ移動すること」「右方向に 1 つ移動すること」を 7 回ずつ行う。それ以外の移動は行わない。

したがって、答えは 14 回のうち 7 回上方向を選ぶ場合の数、すなわち

$$_{14}C_7 = \frac{14!}{7! \times 7!} = 3432 \text{ 通り}$$

となります。直接手計算で求めるのは面倒ですが、節末問題 3.3.3 のプログラムに $n = 14, r = 7$ を代入すると、簡単に答えが分かります。