3.3 節末問題 3.3 的解答

問題 3.3.1

這是測試對情況數的公式（→**3.3.4項**、**3.3.5項**）的理解的問題。答案如下：

$$ {}_{2}\mathrm{C}_{1} = \frac{2}{1} = 2 $$

$$ {}_{8}\mathrm{C}_{5} = \frac{8\times7\times6\times5\times4}{5\times4\times3\times2\times1} = 56 $$

$$ {}_{7}\mathrm{P}_{2} = 7\times6 = 42 $$

$$ {}_{10}\mathrm{P}_{3} = 10\times9\times8 = 720 $$

此外，二項係數 nCr 是 nPr 的 $1/r!$ 倍，因此可以計算下式。

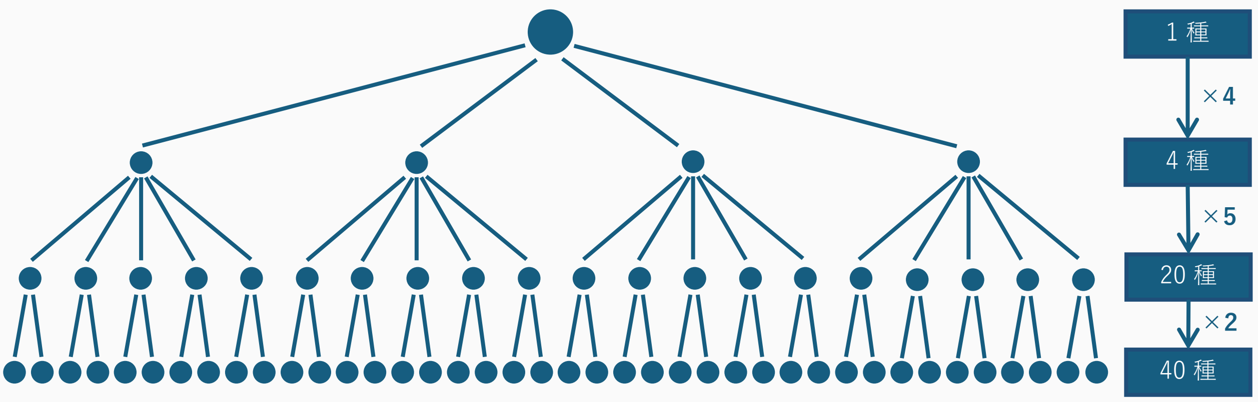
$$ nCr = \frac{nPr}{r!} = \frac{n\times(n-1)\times(n-2)\times(n-r+1)}{r\times(r-1)\times(r-2)\times\cdots\times1} $$

問題 3.3.2

這是測試對乘法原理（→**3.3.2項**）的理解的問題。

- 大小的選擇方式：4 種
- 配料的選擇方式：5 種
- 名牌的選擇方式：2 種

因此，答案是 $4\times5\times2 =$ 40 種。



問題 3.3.3

首先，計算以下的值。（計算 $n!$ 的方法：→**節末問題 2.5.3**）

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

此時，所求的 nCr 的は `Fact_n / (Fact_r * Fact_nr)`，因此，撰寫輸出該值的程式即為正解。C++ 的實作例如下。

```
#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() {
    // 輸入
    cin >> n >> r;

    // 階乘的計算
    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;

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

※ Python等原始碼請參閱 chap3-3.md。

問題 3.3.4

如書中所解釋的實作即可。以下是 C++的解答範例。本問題的限制條件為 $N \leq 200000$ 之大，答案有可能會超過 10^{10} 種。注意如 `int` 型態等的 32 位元整數會**溢出**。（在 Python 不會有這個問題）

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

long long N;
```

下一頁

```
long long A[200009];
long long a = 0, b = 0, c = 0, d = 0; // 為了避免溢出，使用 64 位元的整數

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

    // 計數 a、b、c、d 的個數
    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;
    }

    // 輸出（答案為 a * d + b * c）
    cout << a * d + b * c << endl;
    return 0;
}
```

※ Python等原始碼請參閱 chap3-3.md。

問題 3.3.5

如書中所解釋的實作即可。以下是 C++ 的解答範例。本問題的限制條件為 $N \leq 500000$ 之大，答案有可能會超過 10^{11} 種。

注意如 `int` 型態等的32 位元整數會溢出，因此建議使用 `long long` 型態等 64 位元整數。（在Python不會有這個問題）

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

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

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

    // 計數 a、b、c、d 的個數
    for (int i = 1; i <= N; i++) {
        if (A[i] == 1) x += 1;
        if (A[i] == 2) y += 1;
```

下一頁

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

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

※ Python等原始碼請參閱chap3-3.md。

問題 3.3.6

首先，令 $A_1, A_2, \ldots, A_N$ 中 i 的個數為 `cnt[i]`，則`cnt[1]`, `cnt[2]`, …, `cnt[99999]` 可以如下計數。。

```
// 將陣列 cnt[i] 初始化
for (int i = 1; i <= N; i++) cnt[i] = 0;

// 出現 A[i] 的話，於 cnt[A[i]] 加上1
for (int i = 1; i <= N; i++) cnt[A[i]] += 1;
```

因此，和為 100000 的 2 張卡片的選擇方法可以列舉如下：

- 選擇 1 和 99999 的卡片（有 `cnt[1]*cnt[99999]` 種方法）
- 選擇 2 和 99998 的卡片（有 `cnt[2]*cnt[99998]` 種方法）
- 選擇 3 和 99997 的卡片（有 `cnt[3]*cnt[99997]` 種方法）
- ：
- 選擇 49999 和 50001 的卡片（有 `cnt[49999]*cnt[50001]`種方法）
- 選擇兩張 50000 的卡片（有`cnt[50000]*(cnt[50000]-1)/2` 種方法）

注意當選擇兩張 50000 的卡片時，不是 `cnt[50000]*cnt[50000]` 種方法。

因此，撰寫將紅色標記值的總和輸出的程式，即可得到正確答案。以下是 C++ 的解答範例。。

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

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

int main() {
```

下一頁

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

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

    // 求出答案
    for (int i = 1; i <= 49999; i++) {
        Answer += cnt[i] * cnt[100000 - i];
    }
    Answer += cnt[50000] * (cnt[50000] - 1) / 2;

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

※ Python等原始碼請參閱chap3-3.md。

問題 3.3.7

首先，從起點到終點以最短距離（14 步）前進的充要條件（→**2.5.6項**）如下：

> 「向上移動 1 步」和「向右移動 1 步」各進行 7 次。除此之外，不進行其他移動。

因此，答案是從 14 步中選擇 7 步向上移動的情況數如下。

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

直接手算比較麻煩，但將節末問題 3.3.3 的程式中 $n = 14$ 和 $r = 7$ 代入，便可簡單得到答案。