6.4 最終確認問題 16-20 的解答

問題 16

將可能的所有模式逐一調查的方法稱為「全搜尋」（→**2.4.5 項**）。

在本問題中，對於滿足$1 \leq a < b < c \leq N$,且 $a + b + c = X$ 的整數 (a, b, c)的組合，要完全一點不漏地列舉出來絕非易事。然而，將滿足 $1 \leq a < b < c \leq N$ 的 (a, b, c) 進行列舉是簡單的，將其全體進行測試並計算滿足 $a + b + c = X$ 的個數，這是最簡單的方法。。

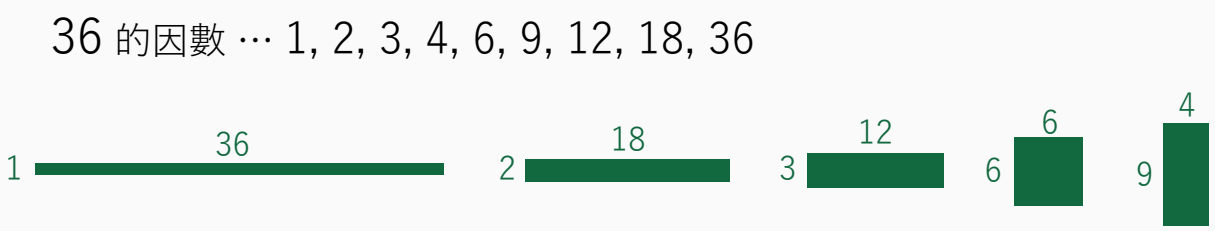
將此解法以 C++ 實作如下。 (a, b, c) 的迴圈處理與**程式碼3.3.1**相似

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

int main() {
    // 輸入
    int N, X;
    cin >> N >> X;

    // 嘗試所有 (a, b, c) 的組合
    int answer = 0;
    for (int a = 1; a <= N; a++) {
        for (int b = a + 1; b <= N; b++) {
            for (int c = b + 1; c <= N; c++) {
                if (a + b + c == X) {
                    answer += 1;
                }
            }
        }
    }

    // 輸出答案
    cout << answer << endl;
    return 0;
}
```

※ Python等原始碼請參閱 chap6-16_20.md。

問題 17

矩形的面積是以（縱長）×（橫長）計算的。由於這兩者都是整數，所以為了使面積為 N，其縱長和橫長必須是「N 的因數」。



因此，若用 **3.1.5 項** 的方法列舉約數，可以調查所有可能的矩形。從其中找出周長最小的。

另外，也有更簡單的解法。利用假設（縱長）≦（橫長）也不會失去一般性（→ **5.10.4 項**）的性質，則（縱長）≦ $\sqrt{N}$ 。因此，對縱長 x 在 1 以上到$\sqrt{N}$以下的範圍進行全搜尋，找出以下條件：

- 橫長 $N \div x$ 是整數的情況
- 在其中找出周長 $2x + 2(N \div x)$ 為最小的情況

可以用計算複雜度 $O(\sqrt{N})$ 以，C++ 實作如下。

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

int main() {
    // 輸入
    long long N;
    cin >> N;

    // 將垂直長度從 1 到 √N 為止進行全搜尋
    long long answer = (1LL << 60);
    for (long long x = 1; x * x <= N; x++) {
        if (N % x == 0) {
            answer = min(answer, 2 * x + 2 * (N / x));
        }
    }

    // 輸出答案
    cout << answer << endl;
    return 0;
}
```

※ Python等原始碼請參閱chap6-16_20.md。

問題 18

整數 A、 B的最大公因數為 $\mathrm{GCD}(A,B)$， 最小公倍數為 $\mathrm{LCM}(A,B)$ 。因此， $A\times B=\mathrm{GCD}(A,B)\times\mathrm{LCM}(A,B)$ 的關係（→**2.5.2 項**）會成立。因此， 若可以用輾轉相除法（→**3.2 節**）計 $\mathrm{GCD}(A,B)$， 則最小公倍數可以用下式求出。

$$\mathrm{LCM}(A,B)=A\times B\div\mathrm{GCD}(A,B)$$

計算複雜度是 $O(\log(A+B))$ 。

然而， 直接計算可能會在 C++ 等語言中發生溢出。因為即使使用 long long 型態， 也只能處理大約 19 位的數字， 而在計算 $A\times B$ 時可能超過這個範圍。這個問題可以透過以下兩種方法來應對：

1. 將計算順序從「$A\times B\div\mathrm{GCD}(A,B)$」變為「$A\div\mathrm{GCD}(A,B)\times B$」
2. 用巧妙的方法判斷最小公倍數是否超過 10^{18}

關於2.， 判斷 $A\times B\div\mathrm{GCD}(A,B)>10^{18}$， 即 $A\div\mathrm{GCD}(A,B)>10^{18}\div B$ 即可。因為左邊必定是整數， 所以右邊取整數部分也能順利進行。因此， 撰寫如下程式即可得到正解。

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

long long GCD(long long A, long long B) {
    if (B == 0) return A;
    return GCD(B, A % B);
}

int main() {
    // 輸入
    long long A, B;
    cin >> A >> B;

    // 判斷最小公倍數是否超過 10^18
    if (A / GCD(A, B) > 1000000000000000000 / B) {
        cout << "Large" << endl;
    }
    else {
        cout << A / GCD(A, B) * B << endl;
    }
    return 0;
}
```

※ Python等原始碼請參閱 chap6-16_20.md。

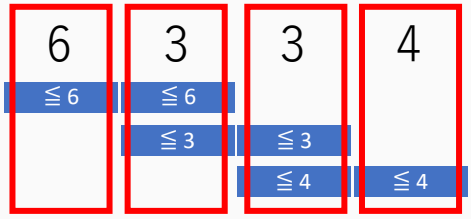
問題 19

這個問題可以使用考慮上限（→ **5.8 節**）的技巧來解決。

考慮一般的情況並不容易，因此首先以具體例 $N = 4, B_1 = 6, B_2 = 3, B_3 = 4$ 來思考。此時，可以換句話說如下。

- $\max(A_1, A_2) \leq 6$ … 「A_1 和 A_2 都為 6 以下」
- $\max(A_2, A_3) \leq 3$ … 「A_2 和 A_3 都為 3 以下」
- $\max(A_3, A_4) \leq 4$ … 「A_3 和 A_4 都為 4 以下」

因此，可知 A_1 為 6以下，A_2 為「6 以下且 3以下」，所以是 3以下，A_3 為「3 以下且 4以下」，所以是 3以下，A_4 為 4以下。



一般情況也一樣進行，得到 $A_1 \leq B_1$、$A_i \leq \min(B_{i-1}, B_i)$ $(2 \leq i \leq N-1)$, $A_N \leq B_{N-1}$ 的上限。實際上，符合這個上限的數列 A 滿足條件，撰寫求其總和的程式即為正解。以下是 C++ 的實作例。

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

int N, B[109];
int main() {
    // 輸入
    cin >> N;
    for (int i = 1; i <= N - 1; i++) {
        cin >> B[i];
    }

    // 求出數列 A 的元素總和 → 輸出答案
    int answer = B[1] + B[N - 1];
    for (int i = 2; i <= N - 1; i++) {
        answer += min(B[i - 1], B[i]);
    }
    cout << answer << endl;
    return 0;
}
```

※ Python等原始碼請參閱 chap6-16_20.md。

問題 20

對於各問題，若如以下程式求出總和，則計算複雜度是 $O(NQ)$，會超過執行時間限制（TLE）。

```
int answer1 = 0, answer2 = 0;
for (int i = L; i <= R; i++) {
    if (C[i] == 1) answer1 += P[i];
    if (C[i] == 2) answer2 += P[i];
}
```

為了高速化，可以使用累積和（→ **4.2 節**）。首先，只計算一班的總得分。假設學號 i 的學生屬於一班時為 $A_i = P_i$，屬於二班時為 $A_i = 0$，則一班的總得分是 $A_L + A_{L+1} + \cdots + A_R$。因此，使用累積和的話可以用 $O(1)$來回答各問題。

學號、班級	1	2	3	4	5
得分 P_i	50	80	100	30	40
一班的得分 A_i	50	80	0	30	0
累積和	50	130	130	160	160

… 一班

… 二班

對二班也可以進行同樣操作。如此，以整體計算複雜度 $O(N + Q)$ 來解決此問題。將此解法以C++實作如下。

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

int N, C[100009], P[100009], L[100009], R[100009], S1[100009], S2[100009];
int main() {
    // 輸入 → 求出累積和
    cin >> N;
    for (int i = 1; i <= N; i++) cin >> C[i] >> P[i];
    for (int i = 1; i <= N; i++) S1[i] = S1[i - 1] + (C[i] == 1 ? P[i] : 0);
    for (int i = 1; i <= N; i++) S2[i] = S2[i - 1] + (C[i] == 2 ? P[i] : 0);

    // 回答問題
    cin >> Q;
    for (int i = 1; i <= Q; i++) {
        cin >> L[i] >> R[i];
        cout << S1[R[i]] - S1[L[i] - 1] << " " << S2[R[i]] - S2[L[i] - 1] << endl;
    }
    return 0;
}
```

※ Python等原始碼請參閱 chap6-16_20.md。