第 1 章

アルゴリズムと計算量

1.1 問題 B01：A+B Problem (難易度：★1相当)

この問題では、2 つの整数 A と B を入力し、$A + B$ の値を出力しなければなりません。

C++ の場合は `cin` を使って入力を行い、`cout` を使って出力を行うことができるので、以下のようなプログラムを書くと正解が得られます。

解答例（C++）

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

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

    // 出力
    cout << A + B << endl;
    return 0;
}
```

※Python のコードはサポートページをご覧ください

1.2 問題 B02：Divisor Check (難易度：★1相当)

この問題では、以下のように **A 以上 B 以下の整数を全探索する**ことで、正しい答えを求めることができます。

- A は 100 の約数か？
- $A+1$ は 100 の約数か？
- $A+2$ は 100 の約数か？
  ：
- B は 100 の約数か？

} どれか 1 つでも Yes なら答えは Yes

全探索のアルゴリズムを C++ で実装すると、以下のようになります。ここで整数 x が 100 の約数であるかどうかは、`100%x==0` かどうかで判定できることに注意してください。

解答例（C++）

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

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

    // 答えを求める
    bool Answer = false;
    for (int i = A; i <= B; i++) {
        if (100 % i == 0) Answer = true;
    }

    // 出力
    if (Answer == true) cout << "Yes" << endl;
    else cout << "No" << endl;
    return 0;
}
```

※Python のコードはサポートページをご覧ください

1.3 問題 B03：Supermarket 1 (難易度：★2相当)

この問題を全探索で解くプログラムは、三重の for 文を使って以下のように実装することができます。各変数は次のようになっています。

変数名	説明
i	選んだ商品のうち 1 個目の番号（つまり値段は A_i 円）
j	選んだ商品のうち 2 個目の番号（つまり値段は A_j 円）
k	選んだ商品のうち 3 個目の番号（つまり値段は A_k 円）

そして、各 (i,j,k) に対しては合計価格が 1000 円になっているか、つまり $A_i + A_j + A_k = 1000$ を満たすかどうかを調べています。なお、選ぶ商品はすべて異なる必要があるので、i,j,k が相異なるかどうかも条件分岐でチェックしています。

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

int main() {
    // 入力
    int N, A[109];
    cin >> N;
    for (int i = 1; i <= N; i++) cin >> A[i];

    // 答えを求める
    bool Answer = false;
    for (int i = 1; i <= N; i++) {
        for (int j = 1; j <= N; j++) {
                for (int k = 1; k <= N; k++) {
                        if (A[i] + A[j] + A[k] == 1000) { // 合計価格は 1000 円か？
                                if (i!=j && j!=k && i!=k) { // 商品はすべて異なるか？
                                        Answer = true;
                                }
                        }
                }
        }
    }

    // 出力
    if (Answer == true) cout << "Yes" << endl;
    else cout << "No" << endl;
    return 0;
}
```

しかし、**商品番号の小さい順に変数 i,j,k を割り当てるようにする**※1と、3つの値 i,j,k がすべて異なるかどうかの判定をする必要がなくなり、実装が簡潔になります。具体的には次のようなループを行えば良いです。

- j は $i+1$ 以上 N 以下の範囲でループ
- k は $j+1$ 以上 N 以下の範囲でループ

全探索のアルゴリズムを C++ で実装すると以下のようになり、計算量は $O(N^3)$ です。制約は $N \leq 100$ ですので、実行時間制限の 1 秒には余裕を持って間に合います。

解答例（C++）

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

int main() {
    // 入力
    int N, A[109];
    cin >> N;
    for (int i = 1; i <= N; i++) cin >> A[i];

    // 答えを求める
    bool Answer = false;
    for (int i = 1; i <= N; i++) {
        for (int j = i + 1; j <= N; j++) {
                for (int k = j + 1; k <= N; k++) {
                        if (A[i] + A[j] + A[k] == 1000) Answer = true;
                }
        }
    }

    // 出力
    if (Answer == true) cout << "Yes" << endl;
    else cout << "No" << endl;
    return 0;
}
```

※Python のコードはサポートページをご覧ください

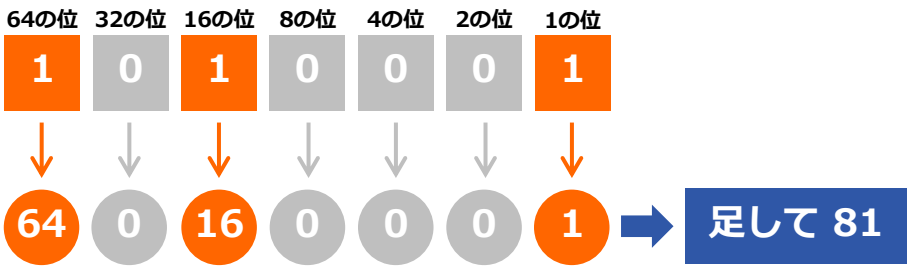
※1　つまり、たとえば $(i,j,k)=(3,2,1)$ などは探索しないということです

1.4 問題 B04：Binary Representation 2 (難易度：★2相当)

本の 36 ページ（1.4 節）にも記した通り、以下のような方法で 2 進法を 10 進法に変換することができます。

> 2 進法の下の位から順に「1 の位」「2 の位」「4 の位」「8 の位」と倍々になるように付けていく。このとき、「数字×位」の総和が 10 進法に変換した値である。

たとえば、2 進法の「1010001」を 10 進法に変換すると、64+16+1=81 となります。イメージ図を以下に示します。



このアルゴリズムを実装すると、以下の解答例のようになります。このプログラムでは、入力を文字列 N として受け取っており、C++ の文字列は 0 文字目から始まるため、「数字×位」としては

```
N[i] * (1 << (N.size() - 1 - i))
```

を足しています。ここで `N[i]` は文字列の `i` 文字目を表し、`(1 << x)` は 2 の `x` 乗を意味します。

解答例（C++）

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

```cpp
int main() {
    // 入力
    string N;
    cin >> N;

    // 答えを求める
    int Answer = 0;
    for (int i = 0; i < N.size(); i++) {
        int keta;
        int kurai = (1 << (N.size() - 1 - i));
        if (N[i] == '0') keta = 0;
        if (N[i] == '1') keta = 1;
        Answer += keta * kurai;
    }

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

※Python のコードはサポートページをご覧ください