アルゴリズム&データ構造ゼミ #5 探索アルゴリズム (1)

高校 2 年 赤澤侑

2020/10/22

1　探索アルゴリズム

今回から本格的にアルゴリズムについて扱います．まず最初に扱うのは探索アルゴリズムです．探索アルゴリズムとは解としてあり得るものをいくつか調べることで解に至るような手法[1]のことを指します．コンピュータの性能を駆使すれば何も考えずに全ての答え，或いは状態を再現してみることで解を見つけることができる場合も多いですが，上手く調べ方を工夫する必要があることもあり，非常に奥深いアルゴリズムです．アルゴリズム&データ構造ゼミの最初のテーマにふさわしいものであることに間違いありません．このアリゴリズムについて全三回に分けて基本的な諸アルゴリズムや代表的な問題を考えていきます．第 1 回は全探索，第 2 回は二分探索，第 3 回はグラフ上の探索です．

2　全探索への招待

唐突な事実ですが，コンピュータは一秒間に約 7×10^8 ステップ程度の計算をこなすことができます．ここで次のような問題を考えてみます．

問題

$3X + 4Y + 5Z = XYZ$ を満たす 100 より小さい正整数 (X, Y, Z) の組であって X の値が最大であるものを求めよ．

この問題は数学 A の整数問題として解くことができるのですが，少し別の視点から考えてみます．X, Y, Z に関する条件から答えとしてあり得る組み合わせは 10^6 程度であることがわかります．これならコンピュータを使って全てを試すことによって答えに至ることができそうです．

このように考えられる全てを実際にやってみることによって解を見つけるような手法を一般に全探索 (Brute-force search あるいは Exhoustive search) といいます．全探索はコンピュータと非常に相性の良い手法です．今回は全探索[2]について考えていきましょう．

3　ループによるナイーヴな全探索

まずは for ループなどで素直に実装できるタイプの全探索を扱います．次の例題を確認してください．

[1] 探索アルゴリズムの厳密な定義は残念ながら見つけることができませんでした．ただし探索アルゴリズム全体に言えるような性質を論じることは非常に稀（というかほぼ見たことないです）なのでちょっと気持ち悪いかもしれませんが「そういうものなんだな」って思ってほしいです．

[2] 全探索も定義を見つけることが出来ませんでした．これもあり得るものを全部見てみるというお気持ちを理解してもらえればプログラムを書く上では問題ないです．

問題 1　倍数探し

非負整数 a, b が与えられる．a 以上 b 以下の整数で “3 の倍数または 5 の倍数であって，7 の倍数でないもの” の個数を求めよ．ただし $0 \leq a \leq b \leq 10^6$ である．また整数 x, k に対して $x \times m = k$ となる整数 m が存在するとき k は x の倍数であることとする．

入力　`a b`

これも中高の初歩的な数学の問題として計算式を書き並べれば解くことが可能です．然しながら，あなたの手元にあるコンピュータを使えば包除原理を考えるまでもなくこの問題を解くことができます．答えは a〜b にあるわけですからこれらを全て実際に 3,5,7 で割ってみて条件を満たすかどうか調べてみれば良いです．以下にソースコードを示します．

ソースコード 1: “倍数探し” の解答例

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      int a, b;
6      cin >> a >> b;
7      int ans = 0;
8      for(int i = a; i <= b; ++i){
9          if(((i % 3 == 0) || (i % 5 == 0)) && (i % 7 != 0))++ans;
10     }
11     cout << ans << endl;
12     return 0;
13 }
```

入力の受け取りなどは良いとして，8 行目から 10 行目がプログラムの本質的な部分です．for 文の中身を見てみると a から b まで全てチェックするようなループになっていることがわかります．次の if 文が問題文中に示された条件をチェックしています．“ある整数 k を整数 x で割った余りが 0 であれば k は x の倍数” と考えると剰余を取ってみれば判定できることがわかりますね．C++においては (多くの言語でもそうですが) 剰余は `A % B` とすることで求められます．また OR 条件を表す論理演算子は `||` で，AND 条件は `&&` です．この 2 つの演算の優先順位は同じなので左から右へと評価されますが，条件を明確にするために `((i % 3 == 0) || (i % 5 == 0))` としています[3]．また条件の否定，〜でないを表現する論理演算子は `!=` です．

課題 1　“倍数探し” を解いてみよ．

さて，問題 1 が解けたら次の問題に移りましょう．

[3] 本来なら `(i % 3 == 0)` の括弧もなくて大丈夫ですが，わかりやすさのために付けています．

問題 2 ハイパー鶴亀算

マッドサイエンティストであるあなたはロボツルーとロボカメーの製造に成功した．ロボツルーの足は a 本，ロボカメーの足の本数は b 本である．また 2 種類のロボットの足は同じ鉄パイプからできている．あなたの手元に N 本の鉄パイプがあるとき，それらを全て使ってロボツルーとロボカメーは何体ずつ作ることができるか．その数を空白区切りで出力せよ．ただし答えが複数ある場合にはどれを出力してもよく，答えが存在しない場合はただ一つ -1 を出力せよ．

- $0 \leq N \leq 5 \times 10^3$
- $1 \leq a, b \leq 10^3$

入力　N a b

一般的な鶴亀算では a, b の値が定まっていますが今回はそうではありません．しかし，このようなときにもコンピュータを使えば答えを求めることができます．s, t を全探索してみれば良いです．実際にソースコードを書いてみましょう．

ソースコード 2: “ハイパー鶴亀算”の解答例

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      int N, a, b;
6      cin >> N >> a >> b;
7      for(int s = 0; s*a <= N; ++s){
8          for(int t = 0; s*a + t*b <= N; ++t){
9              if(s*a + t*b == N){
10                 cout << s << ' ' << t << endl;
11                 return 0;
12             }
13         }
14     }
15     cout << -1 << endl;
16     return 0;
17 }
```

それではプログラムの解説です．7〜15 行目が本質なのでここの説明を行います．今回は s, t を全探索するという方針なのでループの先頭で `s, t` を宣言しています．ループの続行条件はそのときの s, t で必要な足の本数が N を超えないことです．9 行目では条件を満たす場合に分岐し，答えを出力し `return 0;` で終了します．最後まで調べても答えが見つからない場合には `-1` を出力して終了です．

課題 2 “ハイパー鶴亀算”を解いてみよ．

最後の問題は AtCoderBeginnersSelection からの出題です．

問題 3 Otoshidama

日本でよく使われる紙幣には，10000 円札，5000 円札，1000 円札の 3 種類がある．N 枚のお札の合計金額が Y 円であることはあり得るか．ありうる場合はお札の枚数を 10000 円札，5000 円札，1000 円札の順番で空白区切りで出力せよ．あり得ない場合は `-1 -1 -1` を出力せよ．

- $1 \leq N \leq 2 \times 10^3$
- $10^3 \leq Y \leq 2 \times 10^7$
- $N \in \mathbb{Z}$
- Y は 1000 の倍数

入力　N Y

さて，この問題も今までの問題どおり全探索してみましょう！次のようなコードが思い浮かぶことでしょう．

ソースコード 3: “Otoshidama” の愚直解

```c
#include<stdio.h>

int main(){
    int N, Y;
    scanf("%d␣%d", &N, &Y);
    for(int a = 0; 10000*a <= Y; ++a){
        for(int b = 0; 10000*a + 5000*b <= Y; ++b){
            for(int c = 0; 10000*a + 5000*b + 1000*c <= Y; ++c){
                if(10000*a + 5000*b + 1000*c == Y && a + b + c == N){
                    printf("%d␣%d␣%d\n", a, b, c);
                    return 0;
                }
            }
        }
    }
    printf("%d␣%d␣%d\n", -1, -1, -1);
    return 0;
}
```

10000 円札，5000 円札，1000 円札がそれぞれ何枚ずつあるかをそれぞれ `a, b, c` と表現すると，その金額の総和が Y を超えない間ループを回し，もし Y と等しくなればそれを出力，そうでなければ `-1, -1, -1` を出力するプログラムです．今回は入出力に `scanf` と `printf` を使っています．これらは C 言語から存在する関数です．詳しい使い方は各々調べてほしいのですが，最初に入力・出力のフォーマットを文字列で指定し，その後に入力先あるいは出力する変数を書きます．`%d` は「整数 (`int` 型) の変数を入出力しますよ〜」という意味で，`\n` は改行を意味します．

それではこれを AtCoder で提出してみましょう．

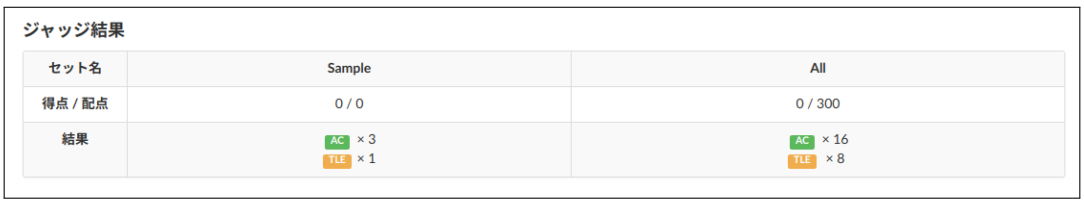

図 1: ソースコード 3 の提出結果

おぉっと〜〜これは困ってしまいましたね．`TLE` というのは実行時間制限超過 (Time Limit Exceeded) を意味します．つまり時間制限に間に合ってませんよということです．実際に問題文の上部を見てみると `実行時間制限:2 sec` とあります．

図 2: 実行時間制限についての記述

上の画像はアドオンを入れているため見た目が多少違っていますが実行時間制限がどこに書いてあるかわかると思います．さあ，どうして実行時間を過ぎてしまったのでしょうか？提出したプログラムの計算量を考えてみます．提出したコードには 3 重のループがあります．結論から言えばこれが原因です．`a b c` はそれぞれ最大で N 程度になりますから，計算量は $O(N^3)$ です．$N \leq 2 \times 10^3$ という制約に於いて $O(N^3)$ はとても間に合いません．これはコンピュータは 1 秒間に約 10^8 ステップ程度の計算しかできないことからわかります．ではどうすればよいでしょうか？実はこの問題は $O(N^2)$ で解くことができます．最後のループについて考えてみましょう．題意より等式 $a+b+c=N$ を満たす必要がありますから，`a b` が確定すれば答えとなりうる `c` は一意に定まります．つまり 3 重ループのうち，最後のループはやる必要がありません．$c = N-a-b$ としてしまえば良いのです．すると次のようなコードになります．

ソースコード 4: “Otoshidama” の解答例

```
#include<stdio.h>

int main(){
    int N, Y;
    scanf("%d␣%d", &N, &Y);
    for(int a = 0; 10000*a <= Y; ++a){
        for(int b = 0; 10000*a + 5000*b <= Y; ++b){
            int c = N - a - b;
            if(10000*a + 5000*b + 1000*c == Y){
                printf("%d␣%d␣%d\n", a, b, c);
                return 0;
            }
        }
    }
    printf("%d␣%d␣%d\n", -1, -1, -1);
    return 0;
}
```

これで無事 AC することができました[4].

[4] 実はこの問題には $O(1)$ 解法があって，詳しくはこのツイート (https://twitter.com/chokudai/status/976418956050223108) に詳しいです．何らかの因果でツイートが消える可能性もあるので説明すると，$5000 \times 9 = 10000 \times 4 + 1000 \times 5$ が成立するのでもし 5000 円札が 9 枚以上ある時，使用する枚数は同じで 9 枚未満の 5000 円札といくつかの 10000 円札，1000 円札でその金額を表現可能，故に 5000 円札を 0〜8 で全探索してあとは鶴亀算をすることで $O(1)$ になります．

類題　練習問題-全探索

A. AtCoder Beginner Contest 136 B - Uneven Numbers
B. AtCoder Beginner Contest 175 B - Making Triangle
C. AtCoder Beginner Contest 051 B - Sum of Three Integers
D. AtCoder Beginner Contest 157 C - Guess The Number
E. AtCoder Regular Contest 106 A - 106
F. AtCoder Beginner Contest 179 C - A × B + C やや難
G. AtCoder Beginner Contest 177 B - Substring
H. AtCoder Beginner Contest 112 C - Pyramid 難

4　順列全探索

5　bit 全探索

6　再帰関数を用いる全探索

7　枝刈りによる効率化

8　半分全列挙