



# 競技プログラミングの鉄則\*

やさしい理系お兄ちゃん

December 15, 2023

## Contents

0 はじめに	2
1 アルゴリズムと計算量	2
1.1 導入問題	2
1.2 全探索 1	3
1.3 全探索 2	4
1.4 2進法	5
1.5 チャレンジ問題	5
2 累積和	6
2.1 一次元の累積和	6

---

\*<https://atcoder.jp/contests/tessoku-book>

## 0 はじめに

この PDF は関数型言語 Haskell の習得のために、「競技プログラミングの鉄則」の問題に対して Haskell で記述したコード例をまとめた資料です。私のプログラミング力の向上を主目的とし、Haskell を学びたい人への手助けとなると良いと考えています。Haskell の I/O<sup>1</sup>は一筋縄でいかないので、アルゴリズム以前の問題で苦戦してしまうかもしれません。ですが、入出力に時間を取られるのは勿体無いため、そのような面でも Haskell 習得の支えになれば良いと願います。

## 1 アルゴリズムと計算量

### 1.1 導入問題

**問題.** 整数  $N$  が与えられるので、一辺の長さが  $N$  であるような正方形の面積を出力するプログラムを作成してください。

Listing 1: A01.hs

```
1 main :: IO ()
2 main = do
3     n <- readLn :: IO Int
4     putStrLn $ show (n * n)
```

**解説.** `readLn` の型は `Read a => IO a` です。ここで、`:: IO Int` と後ろに記述することで型変数 `a` は `Int` 型であると宣言することができます。もちろん、書かなくても `(*)` という演算が後ろで行われていることから、型クラス `Num` に属することを推論してくれます。(書いた方が見やすいのではないのでしょうか?) `n <- readLn :: IO Int` を実行すると、`n` は通常の整数値になります。`show` の型は `Show a => a -> String` であるから、`n * n` の結果を `show` に渡すことで、整数値を文字列に変換することができます。最後に `String -> IO ()` という型をもつ `putStrLn` に渡せば完了です。

**問題.** 整数  $A$  と  $B$  が与えられるので、 $A + B$  の値を出力するプログラムを作成してください。ただし、制約は  $1 \leq A \leq 100$ 、 $1 \leq B \leq 100$  であるとします。

Listing 2: B01.hs

```
1 main :: IO ()
2 main = do
3     [a, b] <- map read . words <$> getLine
4     putStrLn $ show (a + b)
```

**解説.** まずは2つの空白区切りの整数の入力から説明します。`<$>`について、定義は以下の通りです。

```
(<$>) :: (Functor f) => (a -> b) -> f a -> f b
f <$> x = fmap f x
```

`<$>`の型に注目して下さい。`IO` は `Functor` のインスタンスであるから、`f` を `IO` に置き換えると、`<$> :: (a -> b) -> IO a -> IO b` となります。また、

<sup>1</sup>I/O : Input(標準入力), Output(標準出力)

- `getLine :: IO String`
- `map read . words :: Read b => String -> [b]`

と定義されています。以上を合わせると

```
map read . words <$> getLine :: (Read b) => IO [b]
```

となることが分かります。IO [b] については、コンパイラが勝手に IO [Int] としてくれます。<sup>2</sup> 当然、`:: IO Int` と型を指定するのもいいことでしょう。最終的に、結果を Int 型のリストとして `[a, b]` に束縛することができます。(➡注 入力が2つであると分かっているので、`[a, b]` としています。) あとは計算結果を A01 と同様に出力すれば完了です。

## 1.2 全探索1

**問題.**  $N$  個の整数  $A_1, A_2, \dots, A_N$  の中に、整数  $X$  が含まれるかどうかを判定するプログラムを作成してください。

Listing 3: A02.hs

```
1 main :: IO ()
2 main = do
3     [n, x] <- map read . words <$> getLine :: IO [Int]
4     as <- map read . words <$> getLine :: IO [Int]
5     putStrLn $ if x `elem` as then "Yes" else "No"
```

**解説.** 3, 4 行目に関しては Listing 2 の解説で紹介しました。そろそろ気付いたかもしれませんが、Haskell では `for` を使った繰り返し処理を行わないので、リストの長さを示す `n` を受け取る必要がないことが多いです。そのため、3 行目は

```
[_, x] <- map read . words <$> getLine
```

としても良いです。そして、リスト `as` の中に `x` が含まれているかは、関数 `elem` で判定することができます。関数 `elem` は中置記法を用いることが多いです。

**問題.**  $A$  以上  $B$  以下の整数のうち、100 の約数であるものは存在しますか。答えを Yes か No で出力するプログラムを作成してください。

Listing 4: B02.hs

```
1 main :: IO ()
2 main = do
3     [a, b] <- map read . words <$> getLine :: IO [Int]
4     let x = [a..b]
5     putStrLn $ if 0 `elem` map (100 `mod`) x then "Yes" else "No"
```

**解説.** 4 行目の `[a..b]` は、`[a, a+1, a+2, ..., b-1, b]` を表します。このリストのすべての要素について、100 から割った余りを考えます。その余りのリストの中に 0 が含まれているか否かを判定します。

<sup>2</sup>Type Defaulting というものが使われています。詳しくはいつか書きます。(December 15, 2023)

### 1.3 全探索2

**問題.** 赤いカードが  $N$  枚あり、それぞれ整数  $P_1, P_2, \dots, P_N$  が書かれています。また、青いカードが  $N$  枚あり、それぞれ整数  $Q_1, Q_2, \dots, Q_N$  が書かれています。太郎くんは、赤いカードの中から1枚、青いカードの中から1枚、合計2枚のカードを選びます。選んだ2枚のカードに書かれた整数の合計が  $K$  となるようにする方法は存在しますか。答えを出力するプログラムを書いてください。

Listing 5: A03.hs

---

```

1 main :: IO ()
2 main = do
3     [_, k] <- map read . words <$> getLine :: IO [Int]
4     ps <- map read . words <$> getLine :: IO [Int]
5     qs <- map read . words <$> getLine :: IO [Int]
6     putStrLn $ if k `elem` [p + q | p <- ps, q <- qs] then "Yes"
                    else "No"

```

---

**解説.** 6行目の

`[p + q | p <- ps, q <- qs]`

では、リスト `ps` とリスト `qs` から一つずつ要素を取り出し、それらを足し合わせて可能性のある答えをすべて計算して提示しています。これを**非決定性計算**とみなすことができます。このとき、

`(+) <$> ps <*> qs`

と書くことができます。これについては**アプリカティブ**という用語を聞くまでは保留で構いません。非決定性計算という言葉も同様です。(いずれ、後者の記法の方がきれいだと感じる日が来るはずです！私はまだ来ていませんが。)

**問題.**  $N$  個の商品があり、商品  $i$  ( $1 \leq i \leq N$ ) の価格は  $A_i$  円です。異なる3つの商品を選び、合計価格をピッタリ 1000 円にする方法は存在しますか。答えを Yes か No で出力するプログラムを作成してください。ただし、制約は  $3 \leq N \leq 1000$  であるとしします。

Listing 6: B03.hs

---

```

1 import Control.Applicative
2
3 justNum :: Int -> [Int] -> Bool
4 justNum t (x : xs) = if elem True (map (\m -> m + x == t) (xs))
                        then True else justNum t (xs)
5 justNum _ _ = False
6
7 justThousand :: [Int] -> Bool
8 justThousand (x : xs) = if justNum (1000 - x) xs then True else
                            justThousand xs
9 justThousand _ = False
10
11 main :: IO ()
12 main = do
13     n <- read <$> getLine :: IO Int
14     as <- map read . words <$> getLine :: IO [Int]
15     putStrLn if justThousand as then "Yes" else "No"

```

---

## 1.4 2進法

**問題.** 整数  $N$  が10進法表記で与えられます。 $N$  を2進法に変換した値を出力するプログラムを作成してください。

Listing 7: A04.hs

```
1 toBinary :: Int -> [Int] -> [Int]
2 toBinary 0 x = x
3 toBinary t x = toBinary (div t 2) ([mod t 2] ++ x)
4
5 makeZero :: Int -> String
6 makeZero 0 = ""
7 makeZero x = "0" ++ makeZero (x - 1)
8
9 binary2str :: [Int] -> String
10 binary2str [] = ""
11 binary2str (x : xs) = show x ++ binary2str xs
12
13 main :: IO ()
14 main = do
15     n <- readLn
16     let ns = toBinary n []
17     putStrLn $ (makeZero $ 10 - (length ns)) ++ (binary2str ns)
```

**問題.** 整数  $N$  (8桁以内) が2進法表記で与えられます。 $N$  を10進法に変換した値を出力するプログラムを作成してください。

Listing 8: B04.hs

```
1 int2list :: Int -> [Int] -> [Int]
2 int2list 0 t = t
3 int2list x t = int2list (div x 10) ([mod x 10] ++ t)
4
5 list2int :: [Int] -> Int
6 list2int [] = 0
7 list2int (x : xs) = x * (2 ^ length xs) + list2int xs
8
9 main :: IO ()
10 main = do
11     x <- readLn
12     putStrLn $ show $ list2int $ int2list x []
```

## 1.5 チャレンジ問題

**問題.** 赤・青・白の3枚のカードがあります。太郎くんは、それぞれのカードに1以上  $N$  以下の整数を書かなければなりません。3枚のカードの合計を  $K$  にするような書き方は何通りありますか。

Listing 9: A05.hs

```
1 import Control.Applicative
2
3 third :: (Int, Int, Int) -> Int
4 third (_, _, c) = c
```

```

5
6 main :: IO ()
7 main = do
8     [n, k] <- map read . words <$> getLine
9     putStrLn $ show $ length [(a, b, c) | a <- [1..n], b <- [1..n]
      ], let c = k - a - b, 1 <= c, c <= n]

```

---

## 2 累積和

### 2.1 一次元の累積和

**問題.** ある遊園地では  $N$  日間にわたるイベントが開催され、 $i$  日目には  $A_i$  人が来場しました。以下の  $Q$  個の質問に答えるプログラムを作成してください。

- 質問 1:  $L_1$  日目から  $R_1$  までの来場者数は?
- $\vdots$
- 質問  $Q$ :  $L_Q$  日目から  $R_Q$  までの来場者数は?

Listing 10: A06.hs

```

1 import Control.Applicative
2 import Control.Monad
3
4 cumulative_sum :: [Int] -> [Int]
5 cumulative_sum [] = []
6 cumulative_sum [x] = [x]
7 cumulative_sum (x : xs) = [x] ++ (cumulative_sum $ [x + head xs]
  ++ tail xs)
8
9 calc :: [Int] -> [Int] -> Int
10 calc s [l, r] = if l == 1 then s !! r else (s !! (r - 1)) - (s
  !! (l - 2))
11
12 make_output :: [Int] -> String
13 make_output [x] = show x
14 make_output (x : xs) = show x ++ "\n" ++ make_output xs
15
16 main :: IO ()
17 main = do
18     [n, q] <- map read . words <$> getLine
19     as <- map read . words <$> getLine :: IO [Int]
20     lr <- replicateM q getLine
21     let x = map words lr
22     let y = map (\m -> map read m :: [Int]) x
23     putStrLn $ make_output $ map (calc (cumulative_sum as)) y

```

---

**問題.** 太郎君はくじを  $N$  回引き、 $i$  回目の結果は  $A_i$  でした。 $A_i = 1$  のとき当たり、 $A_i = 0$  のときハズレを意味します。「 $L$  回目から  $R$  回目までの中では、当たりとハズレどちらが多いか?」という形式の質問が  $Q$  個与えられるので、それぞれの質問に答えるプログラムを作成してください。計算量は  $O(N + Q)$  であることが望ましいです。

## Listing 11: B06.hs

---

```

1 import Control.Monad ( replicateM )
2
3 cumulativeSum :: Int -> [Int] -> [Int] -> [Int]
4 cumulativeSum t i [] = if length i > 1 then tail i else []
5 cumulativeSum t i [x] = if t == x then tail $ i ++ [1 + last i]
   else tail $ i ++ [last i]
6 cumulativeSum t i (x : xs) = if t == x then cumulativeSum t (i ++
   [1 + last i]) xs else cumulativeSum t (i ++ [last i]) xs
7
8 cumulativeSumOne :: [Int] -> [Int]
9 cumulativeSumOne [] = []
10 cumulativeSumOne (x : xs) = scanl (+) x xs
11
12 cumulativeSumZero :: [Int] -> [Int]
13 cumulativeSumZero [] = []
14 cumulativeSumZero x = cumulativeSumOne $ map (\m -> if m == 0 then
   1 else 0) x
15
16 calc :: [Int] -> [Int] -> Int
17 calc s [l, r] = if l == 1 then s !! (r - 1) else (s !! (r - 1))
   - (s !! (l - 2))
18
19 minus :: [Int] -> [Int] -> [Int]
20 minus [] [y] = [-y]
21 minus [x] [] = [x]
22 minus [x] [y] = [x - y]
23 minus (x : xs) (y : ys) = (x - y) : minus xs ys
24
25 makeOutput :: [Int] -> String
26 makeOutput x = unlines $ map (\x -> if x > 0 then "win" else if x
   == 0 then "draw" else "lose") x
27
28 main :: IO ()
29 main = do
30     n <- readLn :: IO Int
31     as <- map read . words <$> getLine :: IO [Int]
32     q <- readLn :: IO Int
33     lr <- replicateM q getLine :: IO [String]
34     let x = map words lr :: [[String]]
35     let y = map (\m -> map read m :: [Int]) x :: [[Int]]
36     let win = cumulativeSumOne as :: [Int]
37     let lose = cumulativeSumZero as :: [Int]
38     putStrLn $ makeOutput (minus (map (calc win) y) (map (calc lose
   ) y)) :: IO ()

```

---

**解説.** 実行時間超過です。累積和を計算する関数 `cumulativeSumZero` のオーダーに問題があります。プログラムとしては正しい出力が得られます。この関数は  $O(N^2)$  であり、 $0 \leq N \leq 10^5$  から、最悪計算量が  $10^{10}$  になってしまうため制限時間超過となります。問題文では、 $O(N + Q)$  であることが望ましいと書かれていますので、それを目標にプログラムを変更しましょう。

**問題.** ある会社では  $D$  日間にわたってイベントが開催され、 $N$  人が出席します。参加者  $i$  は  $L_i$  日目から  $R_i$  日目まで出席する予定です。各日の出席者数を出力するプログラムを作ってください。