

Competitive Programming in Ruby (101)

RubyKaigi 2016 Kyoto After Party LT (2016-09-10)

NaCl

@yhara (Yutaka Hara)

2016-08-13

■ [scheme] Fomentを読む(5)

[前回](#)からだいぶ間が空いた。ちょっと別にやりたいことができたので、実を言うとこっちは休れない。一回分の更新にはそれほど時間がかからないし、考えたことは全て文字に落とし
今回はExpandSyntaxRulesの他にCompileSyntaxRulesという関数がエクスポート
なのでsynrules.cppのたくさんの関数は、もしかしたら2種類に分類できるかもしれない

■ 関数一覧

一覧を再掲する。引数のうち、型名が省略されているものはFObject。

```
static FObject LiteralFind(se, list, obj)
static FObject CopyLiterals(se, obj, ellip)
static FObject PatternVariableFind(se, list, var)
static FObject CompilePatternVariables(se, form, lits, pat, ellip, pva
static void AssignVariableIndexes(pvars, int_t idx)
static int_t CountPatternsAfterRepeat(pat)
static FObject RepeatPatternVariables(se, pvars, pat, rvars)
static FObject CompilePattern(se, lits, pvars, ellip, pat)
static int_t ListFind(list, obj)
static int_t AddVarToTemplateRepeat(var, trs)
static FObject CompileTemplate(se, form, pvars, ellip, tpl, trs, int_t
static FObject CompileRule(se, form, lits, rule, ellip)
FObject CompileSyntaxRules(se, obj)
```

RubyKaigi 2016 Kyoto LT (2016-09-10)

Rubyで作る
奇妙なプログラミング言語
ヘンな言語のつくりかた

原 浩 (著)

Esoteric Language programming in Ruby

あなたは「+」「>」「<」「,」「[]」の8つの記号しかない言語や、
空白だけで構成される言語など、奇妙な言語
(Esoteric Language) があるのを知っていますか？

本書ではこのような言語を題材に
プログラミング言語の作り方を詳しく解説しています。

Recently I'm trying Competitive
Programming with my colleagues

Competitive Programming

Programming contest

Write a program fast and correctly

Need C(++ , #) or Java?

You can use Ruby too!

Sites which supports Ruby

- Sphere Online Judge
- Aizu Online Judge
- yukicoder
- AtCoder ← Today's topic

atcoder.jp

- AtCoder Grand Contest
- AtCoder Regular Contest
- AtCoder Beginner Contest
- AtCoder Typical Contest

Don't join the contest

Try previous contests first 🙇



- Anytime
- No time limit

- AtCoder Grand Contest
- AtCoder Regular Contest
- **AtCoder Beginner Contest**
- AtCoder Typical Contest

AtCoder Beginner Contest

- A: Very easy
- B: Easy
- C: Easy?
- D: **Not really easy**

But Ruby is slow...? 😐

No problem!

	A	B	C	D
abc035	✓	✓	✓	✓
abc036	✓	✓	✓	✓
abc037	✓	✓	✓	✗
abc038	✓	✓	✓	✓
abc039	✓	✓	✓	✓
abc040	✓	✓	✓	✓
abc041	✓	✓	✓	✓

Algorithm Matters 🧐

Example: one-line Reversi

one-line Reversi

- Given N pieces, flip Q ranges
- Calculate the final state

●●●●●●●●

1..4

2..6

8..9



○●●●○○●○○

Just flipping each pieces?

```
ranges.each do |r|  
  r.each do |i|  
    line[i] = !line[i]  
  end  
end
```

入力

入力は以下の形式で標準入力から与えられる。

```
 $N$    $Q$   
 $l_1$   $r_1$   
.  
.  
.  
 $l_Q$   $r_Q$ 
```

- 1 行目に駒の数と操作回数を表す 2 つの整数 N, Q ($1 \leq N, Q \leq 200,000$) が空白区切りで与えられる。
- 2 行目から続く Q 行のうち i 行目においては、 i 回目の操作の範囲を表す 2 つの整数 l_i, r_i ($1 \leq l_i < r_i \leq N$) が空白区切りで与えられる。

200,000 pieces 🤯

200,000 ranges 🤯

Billions of flips 🤔

```
ranges.each do |r|    # <- 200,000 times
  r.each do |i|       # <- 50,000 times or so
    line[i] = !line[i]
  end
end
```

 Time Limit = 2 sec.

Better algorithm

Memorize where color changes

10001 23005



● ● ● . . . ● ○ . . . ○ ● ● ●

Passed! 😊

AC	701 ms	3580 KB	詳細を確認
----	--------	---------	-----------------------

You can try Competitive
Programing in Ruby 😊
(at least for beginner contest)



	A	B	C	D
abc035	✓	✓	✓	✓
abc036	✓	✓	✓	✓
abc037	✓	✓	✓	✗
abc038	✓	✓	✓	✓
abc039	✓	✓	✓	✓
abc040	✓	✓	✓	✓
abc041	✓	✓	✓	✓

ABC037-D

- I couldn't pass with Ruby 😓
- Traversing 1000x1000 maze
- Maybe with Ruby3...?

Crystal passes

Crystal (0.12.0)	100	460 Byte	AC	1642 ms
------------------	-----	----------	----	---------

※Crystal : Ruby-like statically typed language

Pypy2 passes

PyPy2 (4.0.1)	100	630 Byte	AC	1581 ms
PyPy2 (4.0.1)	100	537 Byte	AC	1188 ms
PyPy2 (4.0.1)	100	694 Byte	AC	1806 ms
PyPy2 (4.0.1)	100	489 Byte	AC	1290 ms
PyPy2 (4.0.1)	100	577 Byte	AC	1304 ms
PyPy2 (4.0.1)	100	1149 Byte	AC	1934 ms

※PyPy2 : Faster Python implemantation

Python3 passes!? 😲

ソースコード

```
1. import subprocess as sp
2. import sys
3. code = r'''
4. #include <algorithm>
5. #include <cstdio>
6. using namespace std;
7. const int M = 1000000007;
8. int h, w;
9. int a[1000][1000], cache[1000][1000];
10. unsigned f(int x, int y) {
11.     static int dx[] = {1, 0, -1, 0};
```

```
46. }  
47. '''  
48. filename = 'sense_of_guilt.cpp'  
49. with open(filename, 'w') as f:  
50.     print(code, file=f)  
51. sp.Popen(['g++', '-std=c++11', '-02', filename]).com  
52. sp.Popen(['./a.out'], stdin=sys.stdin, stdout=sys.s
```

	Python3 (3.4.3)	100	1248 Byte	AC	1226 ms	
--	-----------------	-----	-----------	----	---------	--

(DON'T TRY THIS AT HOME 🙅)