Ruby extensions in C

Paweł Placzyński

Ragnarson

19 marca 2014

github.com/placek
facebook.com/placzynski.pawel

Ruby vs. C

Ruby

- prostota składni, intuicyjność
- lukier składniowy
- duck typing, programowanie funkcyjne, programowanie imperatywne, metaprogramowanie

C

Ruby vs. C

Ruby

- prostota składni, intuicyjność
- lukier składniowy
- duck typing, programowanie funkcyjne, programowanie imperatywne, metaprogramowanie
- „zasobożerność”
- wydajność

C

Ruby vs. C

Ruby

- prostota składni, intuicyjność
- lukier składniowy
- duck typing, programowanie funkcyjne, programowanie imperatywne, metaprogramowanie
- „zasobożerność”
- wydajność

C

- kontrola wykonywanych poleceń (niskopoziomowość)
- wydajność !!!

Ruby vs. C

Ruby

- prostota składni, intuicyjność
- lukier składniowy
- duck typing, programowanie funkcyjne, programowanie imperatywne, metaprogramowanie
- „zasobożerność”
- wydajność

C

- kontrola wykonywanych poleceń (niskopoziomowość)
- wydajność !!!
- łatwo o pomyłkę
- nieczytelny kod
- nie intuicyjne zachowanie

Ruby vs. C

„Ruby jest prosty z wyglądu, ale bardzo skomplikowany w środku, tak jak ciało ludzkie.”

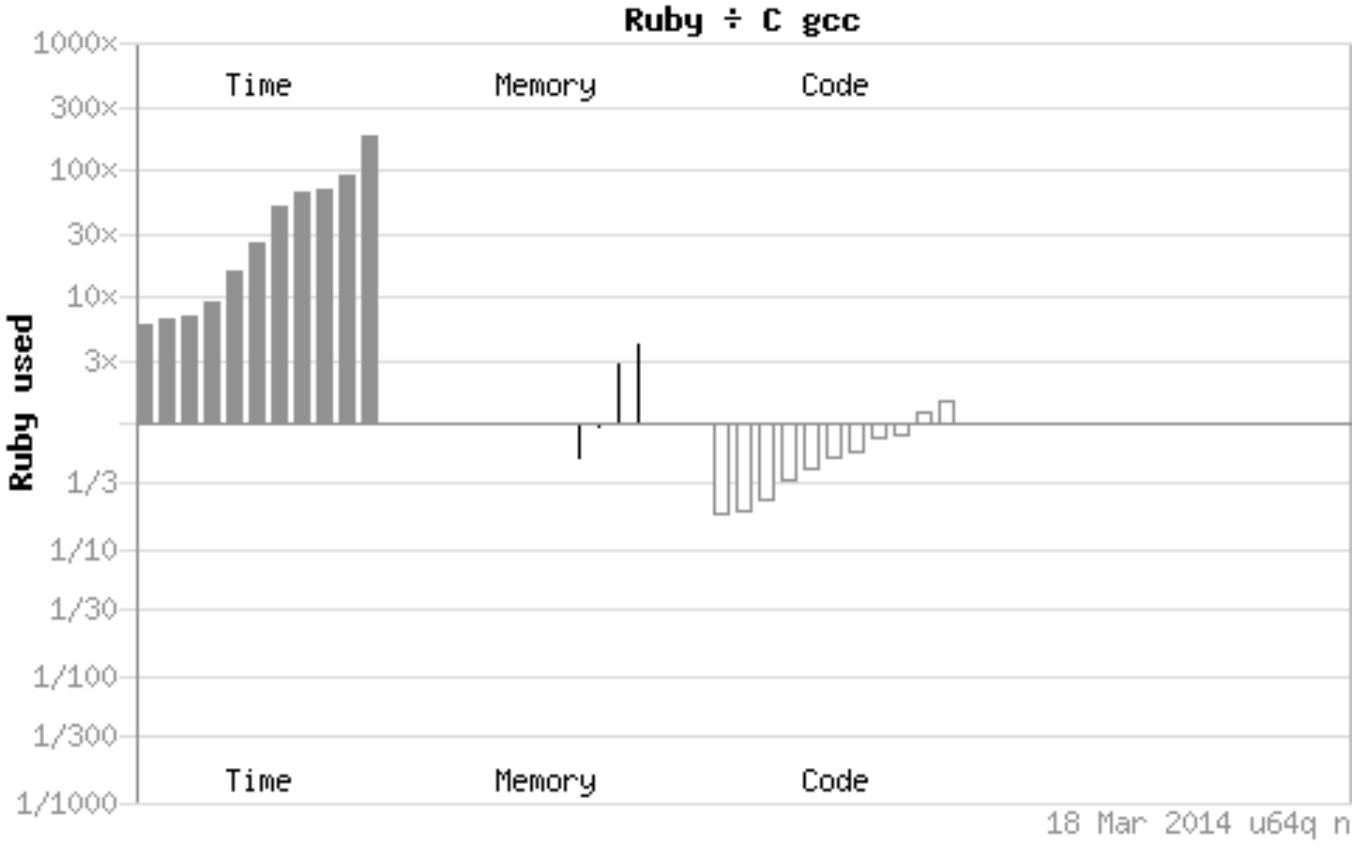
- - Matz

Ruby vs. C

benchmarksgame.alioth.debian.org

Ruby vs. C

Rysunek : Porównanie języka Ruby z językiem C.



Algorytm Levenshtein'a – definicja

Odległością pomiędzy dwoma napisami jest najmniejsza liczba działań prostych, przeprowadzających jeden napis na drugi.

Działaniem prostym na napisie nazwiemy:

- wstawienie nowego znaku do napisu,
- usunięcie znaku z napisu,
- zamianę znaku w napisie na inny znak.

Algorytm Levenshtein'a – przykład

foka

Algorytm Levenshtein'a – przykład

foka

koka

Algorytm Levenshtein'a – przykład

foka

koka

kotka

Algorytm Levenshtein'a – przykład

foka

koka

kotka

dwa proste działania – odległość równa 2

Algorytm Levenshtein'a – więcej informacji

pl.wikipedia.org/wiki/Odległość_Levenshteina

www.algorytm.org/przetwarzanie-tekstu/odleglosc-levenshteina-odleglosc-edycyjna.html

Benchmark

- przykładowy tekst z lipsum.com
- próbki: 100, 200, 300, . . . , 700 słów
- kombinacja (bez powtórzeń) dwóch słów z próbki
- obliczenie odległości Levenshtein'a dla każdej pary słów

Benchmark – czysty Ruby

```
$ ruby benchmark.rb --ruby
Levenstein distance benchmark
  avg word length: 5.62142857142857
  attempt: 1, words: 100, combinations: 4950
          0.580000   0.380000   0.960000 (  0.949640)
  attempt: 2, words: 200, combinations: 19900
          2.590000   1.370000   3.960000 (  3.969015)
  attempt: 3, words: 300, combinations: 44850
          5.430000   3.300000   8.730000 (  8.735403)
  attempt: 4, words: 400, combinations: 79800
         10.960000   5.810000  16.770000 ( 16.772477)
  attempt: 5, words: 500, combinations: 124750
         16.470000   9.380000  25.850000 ( 25.859487)
  attempt: 6, words: 600, combinations: 179700
         25.990000  13.840000  39.830000 ( 39.842691)
  attempt: 7, words: 700, combinations: 244650
         30.260000  18.730000  48.990000 ( 49.011860)
```

Benchmark – rozszerzenie Ruby w języku C

```
$ ruby benchmark.rb --c
Levenstein distance benchmark
  avg word length: 5.62142857142857
  attempt: 1, words: 100, combinations: 4950
          0.050000   0.010000   0.060000 (  0.059438)
  attempt: 2, words: 200, combinations: 19900
          0.220000   0.050000   0.270000 (  0.269328)
  attempt: 3, words: 300, combinations: 44850
          0.430000   0.050000   0.480000 (  0.478186)
  attempt: 4, words: 400, combinations: 79800
          1.000000   0.110000   1.110000 (  1.121147)
  attempt: 5, words: 500, combinations: 124750
          1.190000   0.190000   1.380000 (  1.369337)
  attempt: 6, words: 600, combinations: 179700
          2.820000   0.220000   3.040000 (  3.033939)
  attempt: 7, words: 700, combinations: 244650
          2.440000   0.320000   2.760000 (  2.757448)
```

Benchmark – kod

github.com/placek/lrug

Rozszerzenia Ruby w języku C

ext/extconf.rb

```
require "mkmf"

def error msg
  message msg + "\n"
  abort
end

create_makefile("levenshtein_distance")
```

Rozszerzenia Ruby w języku C

ext/levenshtein_distance.c

```c
#include "ruby.h"

...

static VALUE levenshtein_distance(int argc, VALUE* argv)
{

...

}

VALUE LD;
void Init_levenshtein_distance()
{
  LD = rb_define_module("LevenshteinDistance");
  rb_define_private_method(LD, "leven",
                           levenshtein_distance, -1);
}
```

Rozszerzenia Ruby w języku C

ext/levenshtein_distance.c

```
static VALUE levenshtein_distance(int argc, VALUE* argv)
{
  /* declarations */
  VALUE v_left, v_right;
  long left_length, right_length, i, j, cost, result;
  long ** distance;

  /* parameters validation */
  rb_scan_args(argc, argv, "20", &v_left, &v_right);
  Check_Type(v_left, T_STRING);
  Check_Type(v_right, T_STRING);

  /* initialization of calculation array */
  left_length = RSTRING_LEN(v_left);
  right_length = RSTRING_LEN(v_right);
  distance = (long**) malloc((left_length + 1) * sizeof(long*));

...
```

Rozszerzenia Ruby w języku C

Przykład – funkcje i makra dla klasy String:

```
rb_str_new(c_str, length)
rb_str_new2(c_str)
rb_str_dup(ruby_string_object)
rb_str_plus(string_object_1, string_object_2)
rb_str_times(string_object_1, fixnum_object)
rb_str_substr(string_object, begin, length)
rb_str_cat(string_object, c_str, length)
rb_str_cat2(string_object, c_str)
rb_str_append(string_object_1, string_object_2)
rb_str_concat(string_object, ruby_object)
StringValueCStr(ruby_object)
StringValue(ruby_object)
RSTRING_PTR(return_value)
RSTRING_LEN(return_value)
```

Rozszerzenia Ruby w języku C

www.ruby-doc.org/core-2.0/
clalance.blogspot.com
stackoverflow.com/tags/ruby-c-api/info

Rozszerzenia Ruby w języku C

```
$ ls
extconf.rb levenshtein_distance.c
$ ruby extconf.rb
creating Makefile
$ make
compiling levenshtein_distance.c
linking shared-object levenshtein_distance.so
$ ls
extconf.rb levenshtein_distance.c levenshtein_distance.o
levenshtein_distance.so Makefile
```

Rozszerzenia Ruby w języku C

```
$ ls
extconf.rb levenshtein_distance.c levenshtein_distance.o
levenshtein_distance.so Makefile
$ irb -r `pwd`/levenshtein_distance
>> include LevenshteinDistance
=> Object
>> leven("test", "tost")
=> 1
```

github.com/seattlerb/rubyinline
1.8.2 only :(

```
require "inline"
class MyTest
  inline do |builder|
    builder.c "
      long factorial(int max) {
        int i = max, result = 1;
        while (i >= 2) { result *= i--; }
        return result;
      }"
  end
end
t = MyTest.new()
factorial_5 = t.factorial(5)
```

pytania?