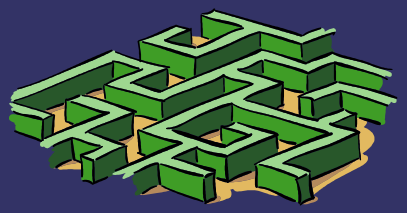
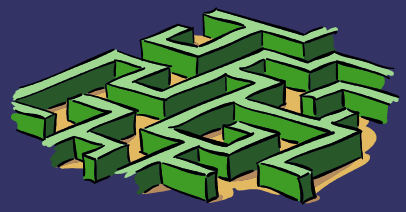
Rubyのダイエット

田中 哲
産業技術総合研究所
Japan GNU/Linux Conference 2007



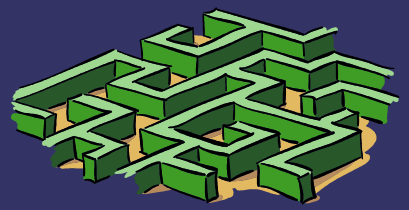
目標

Ruby のメモリ消費を削減する

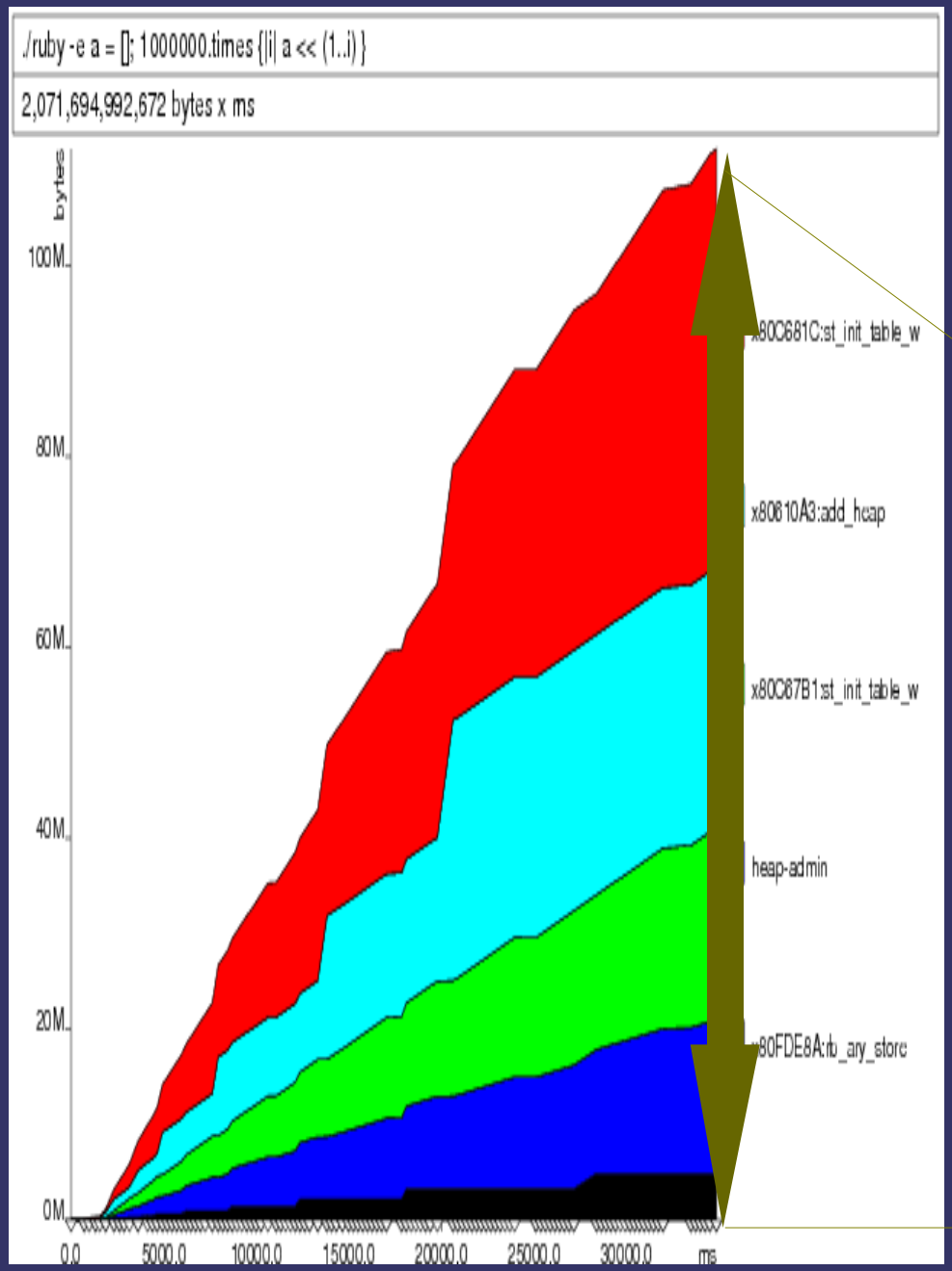


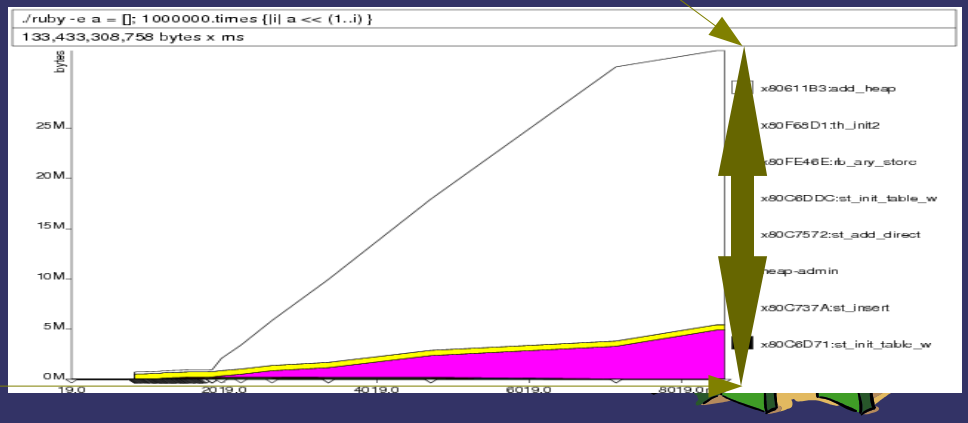
方針

- Ruby 本体を改善する
- 無駄なところを節約する
- 既存スクリプト無修正でメモリ消費削減



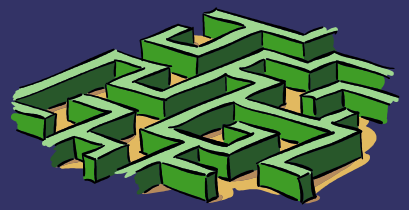
極端なケースでのメモリ削減結果





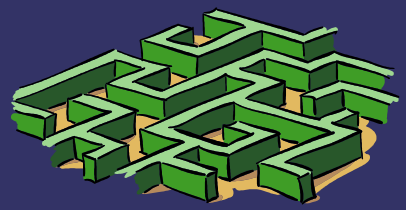
結果

- 1.9 でそれなりに削減ができた
- そのうちそれなりに幸せになる

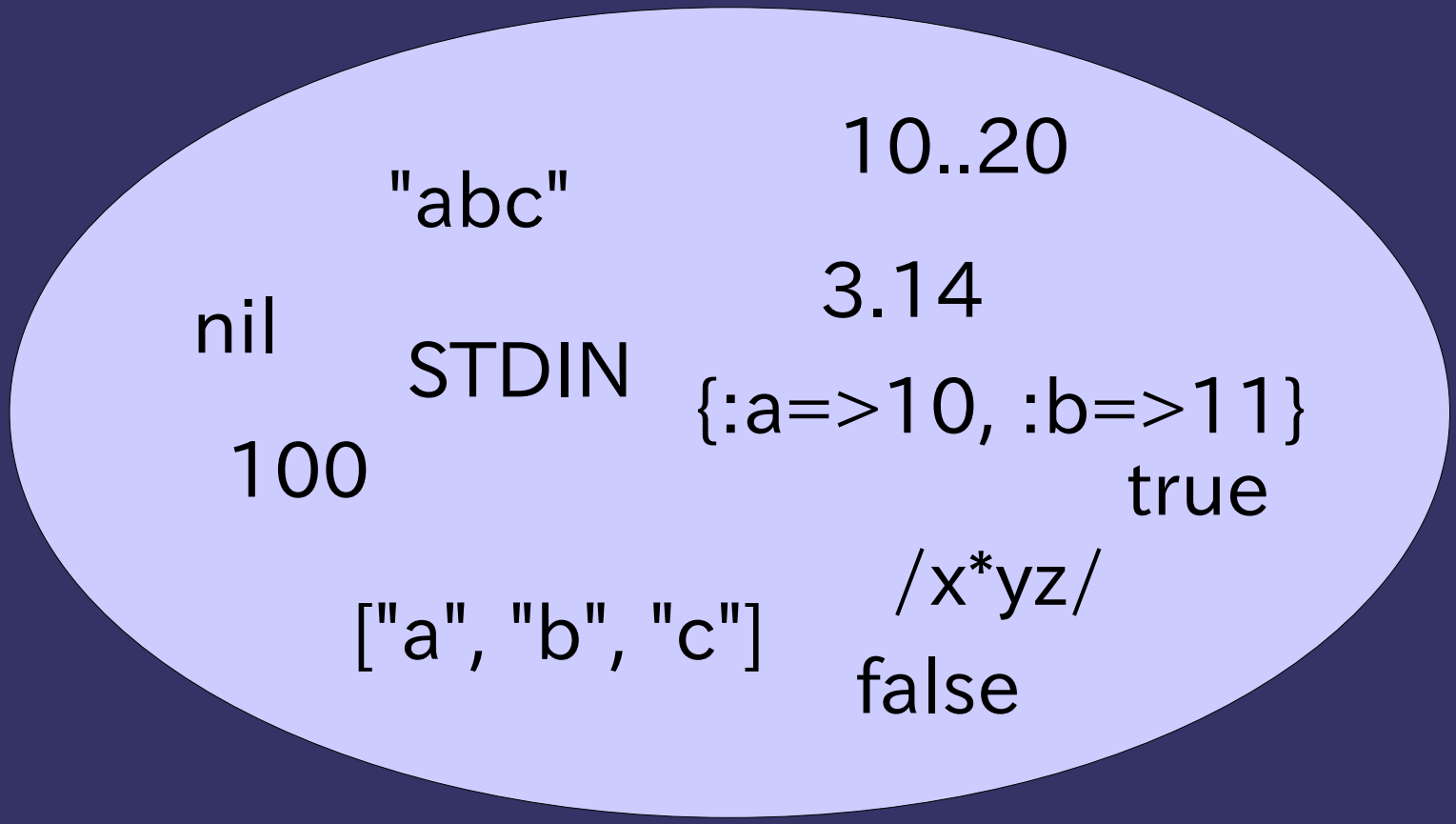


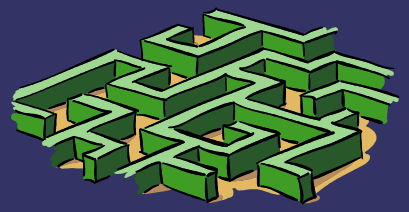
解説-環境の想定

- Ruby のメモリ管理の概要から
- 想定
  - 32bit マシン
  - double が 4byte alignment



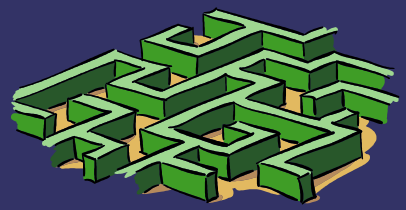
Ruby の世界はオブジェクトの集合





Ruby のメモリ管理

- C レベルでは VALUE 型でオブジェクトを表現
- VALUE は 32bit 符号無整数型

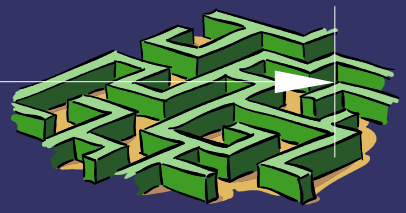


即値オブジェクト

- nil, true, false, Fixnum, Symbol は即値

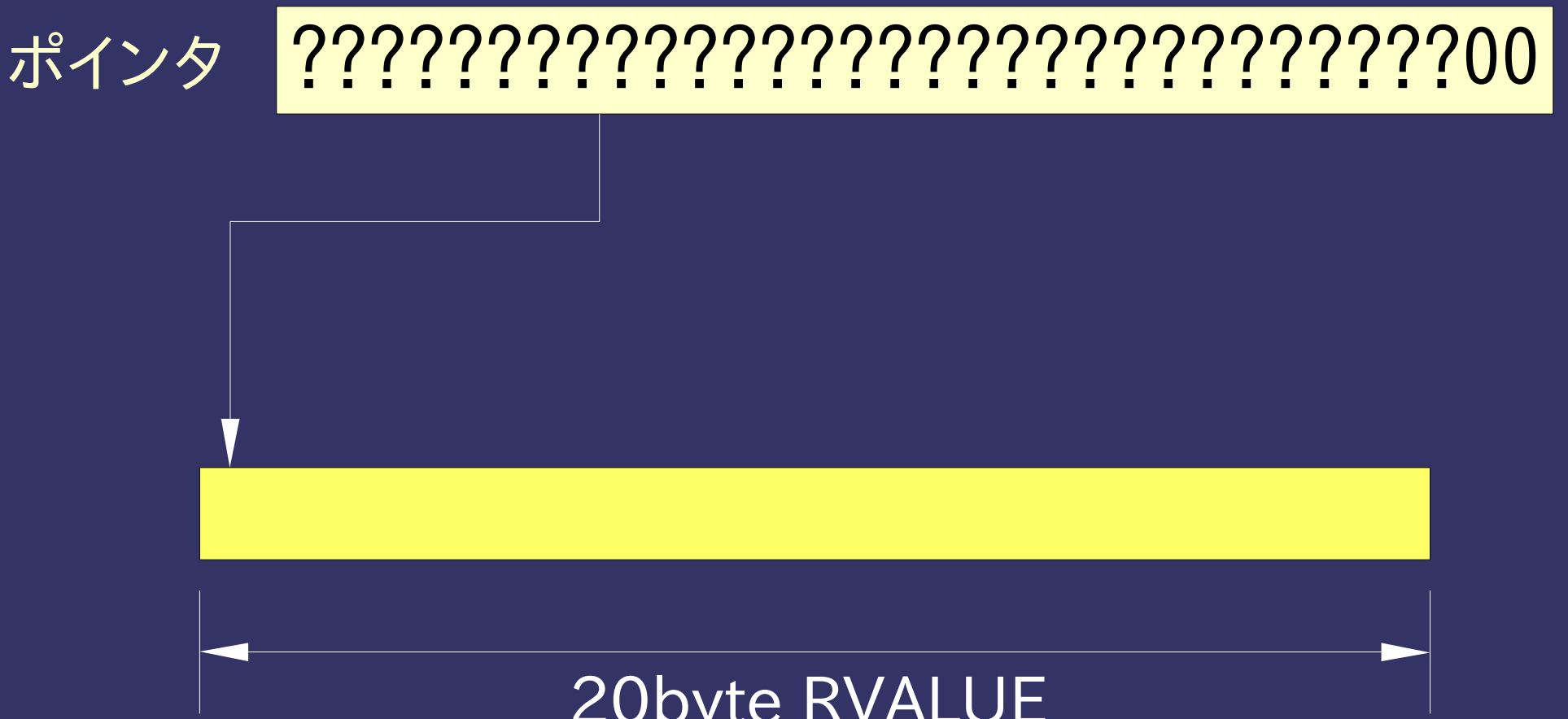
false	00000000000000000000000000000000
true	00000000000000000000000000000010
nil	00000000000000000000000000000100
Fixnum	????????????????????????????????1
Symbol	????????????????????????????1110

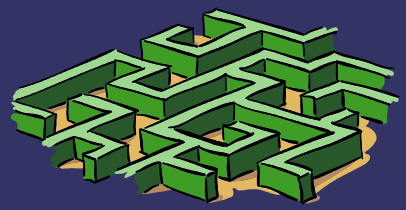
32bit VALUE



非即値オブジェクト

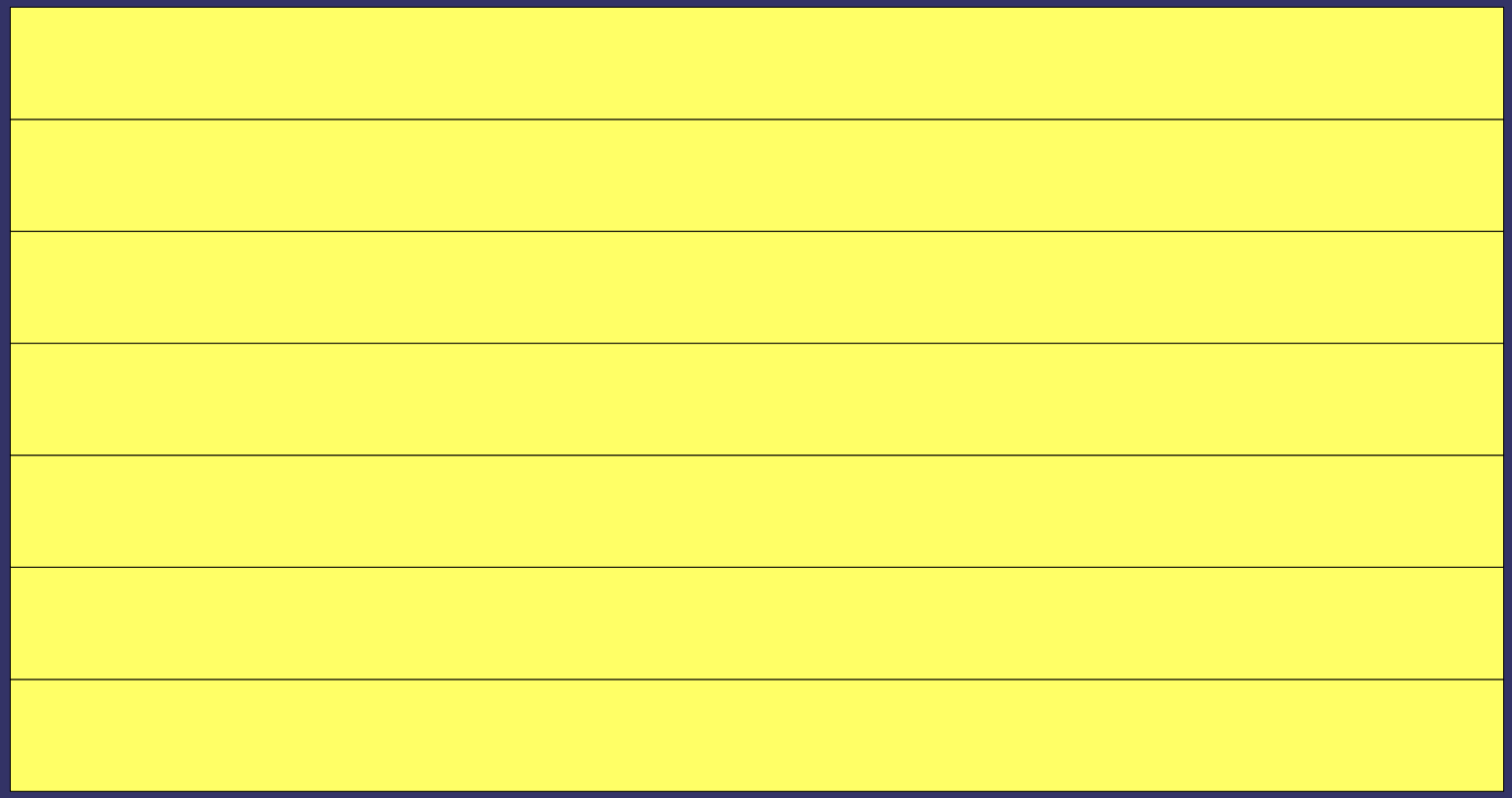
- 5word (20byte) の RVALUE 型へのポインタ



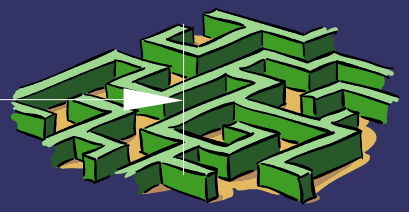


ヒープ(1)

- RVALUE の配列。非即値オブジェクトは
  ここから確保される

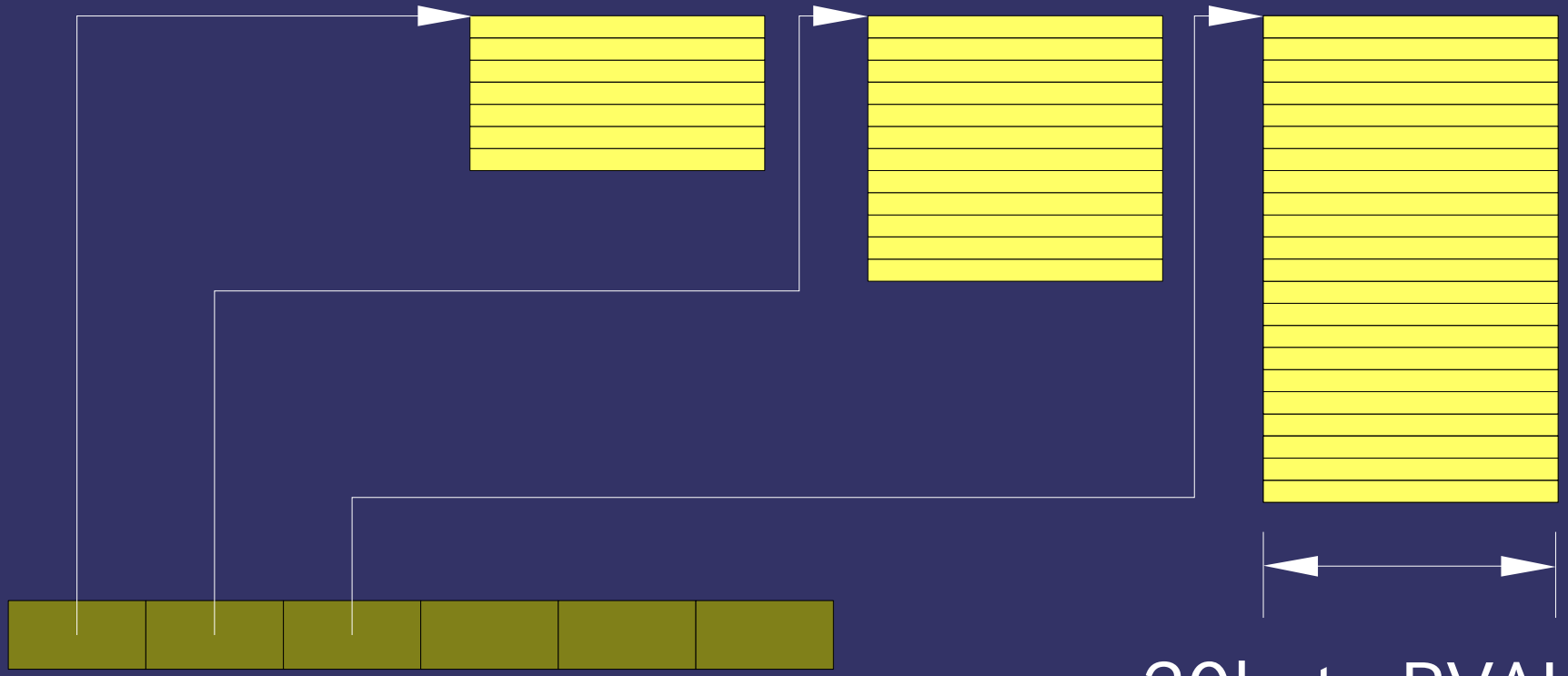


20byte RVALUE

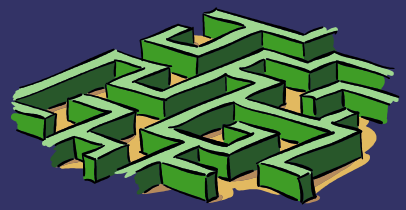


ヒープ (2)

- RVALUE の配列がいっぱいになって、GC でも十分に空かなければ別に確保する

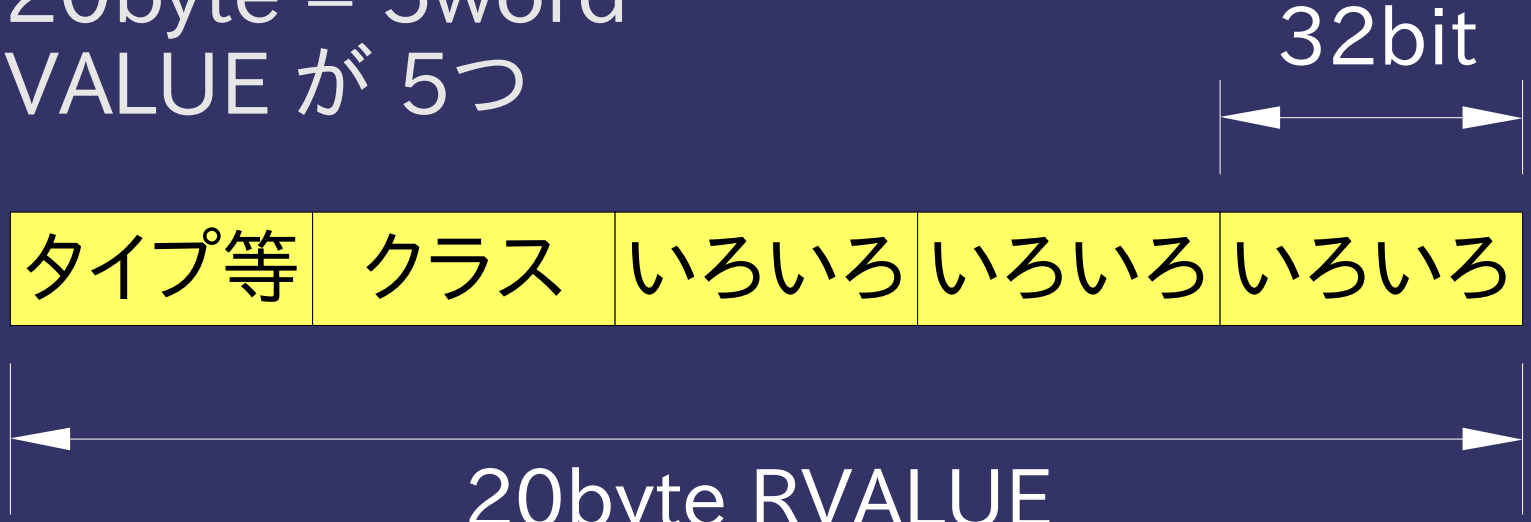


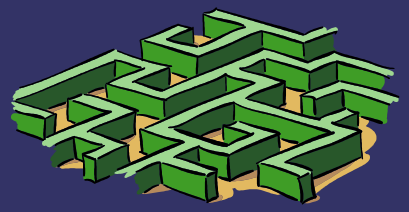
20byte RVALUE



非即値オブジェクト

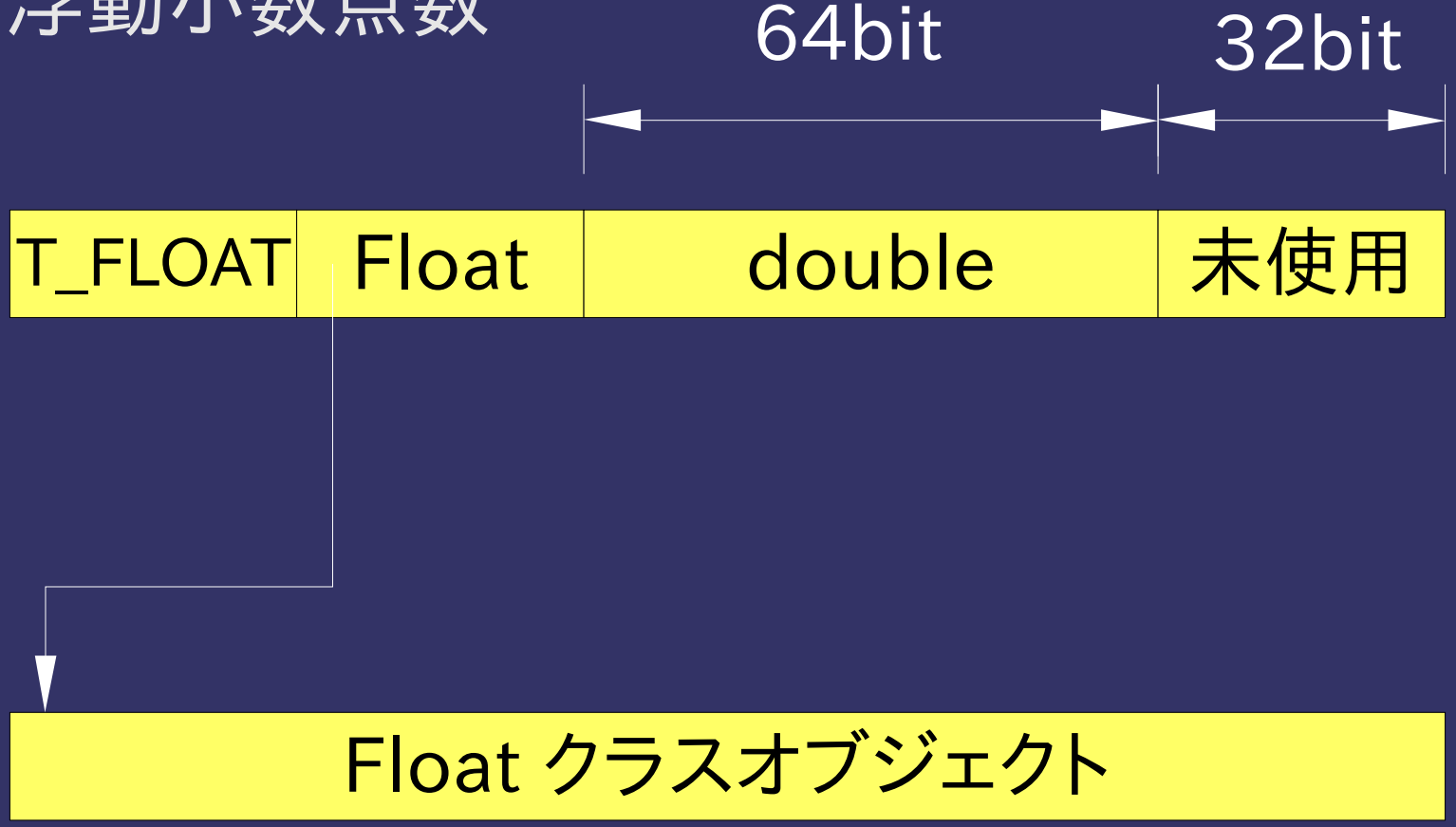
- 20byte = 5word
- VALUE が 5つ

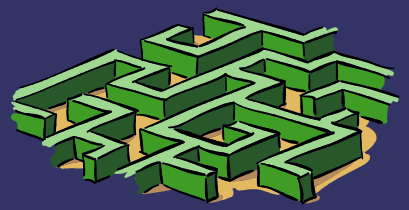




Float

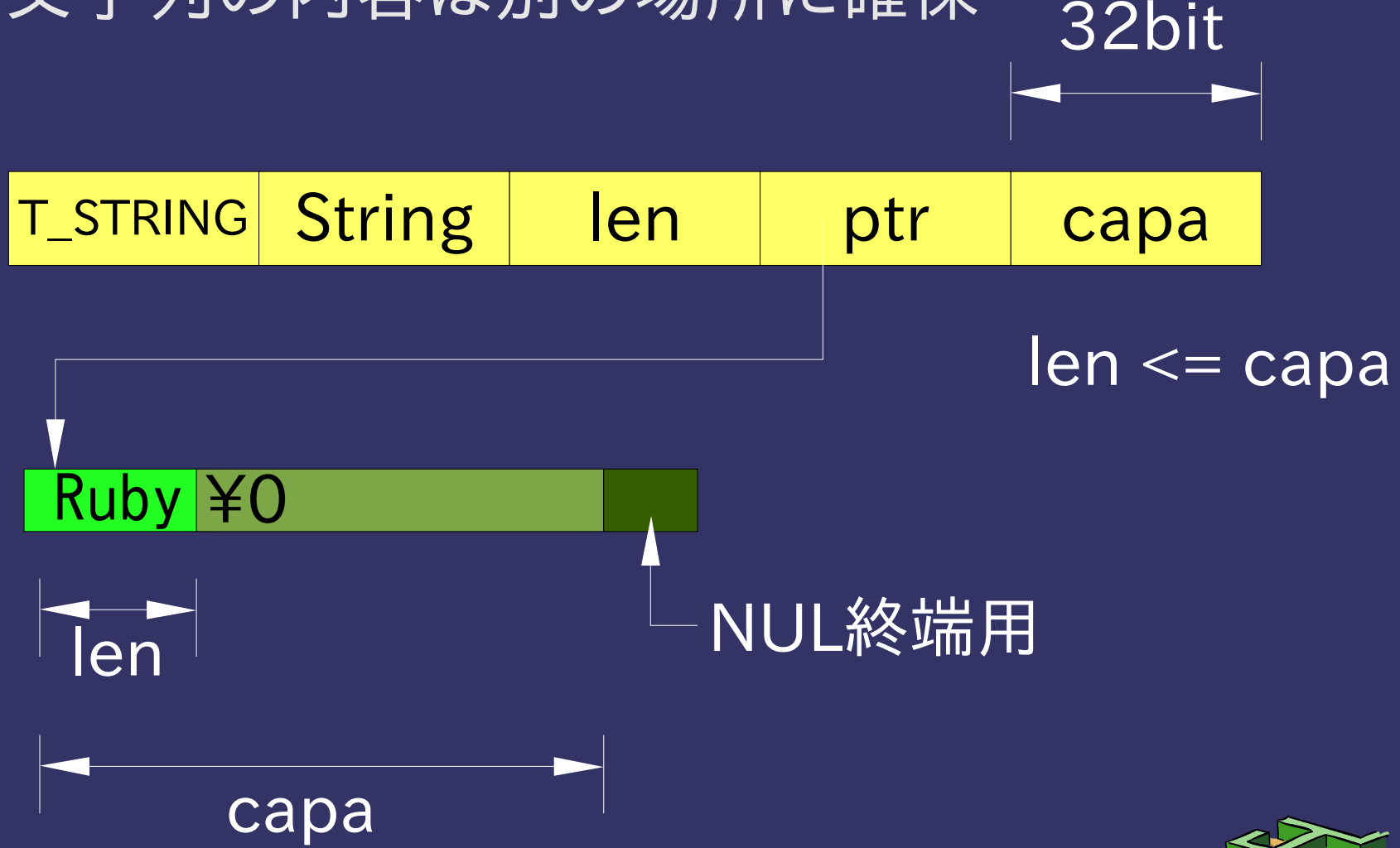
- 浮動小数点数

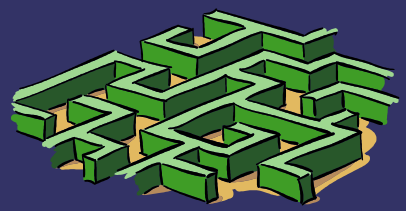




String

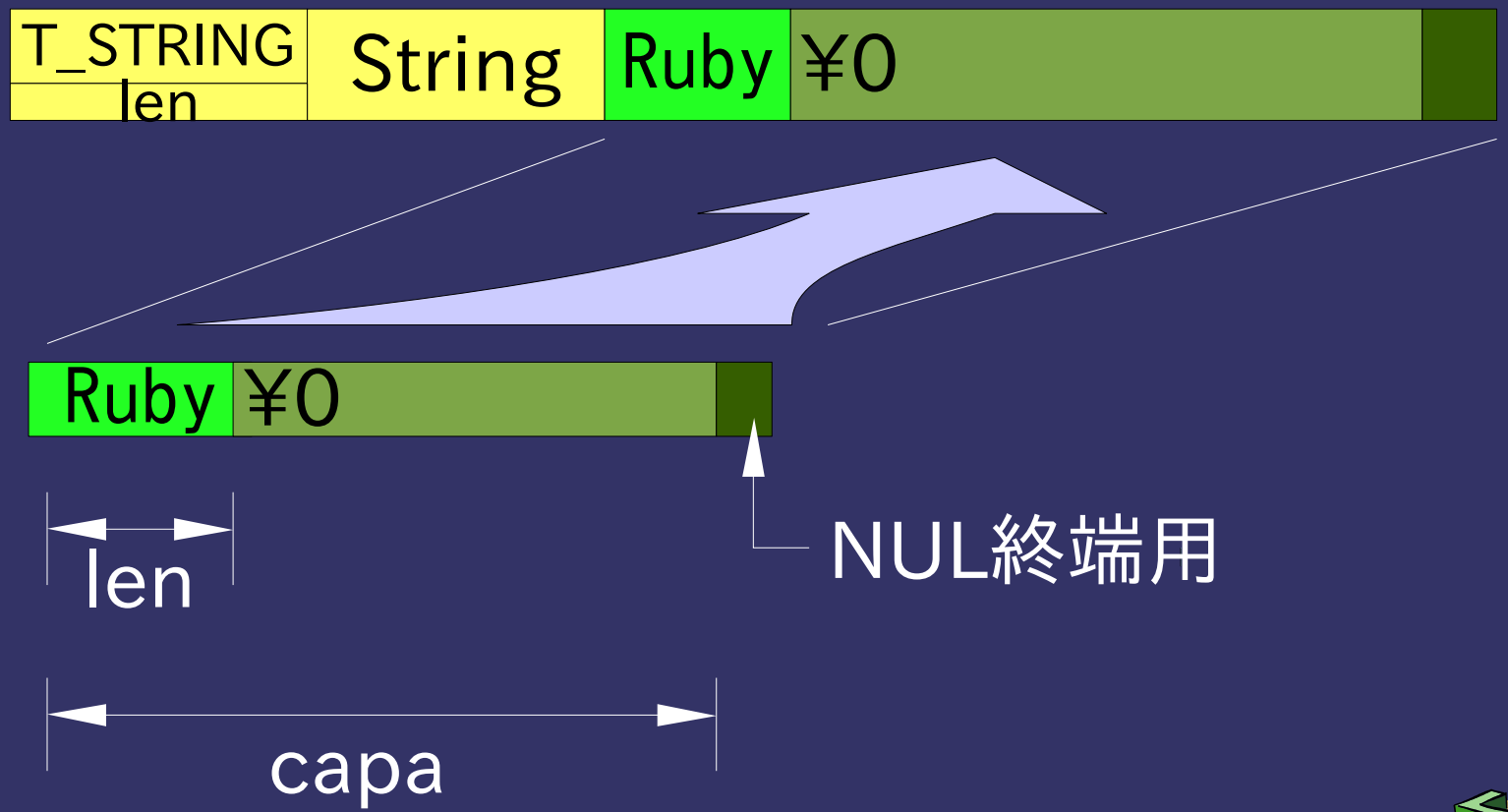
- 文字列の内容は別の場所に確保

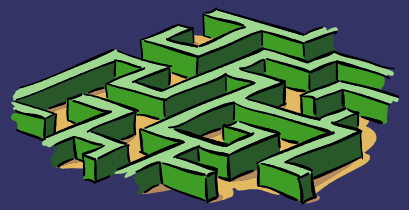




Stringのメモリ節約

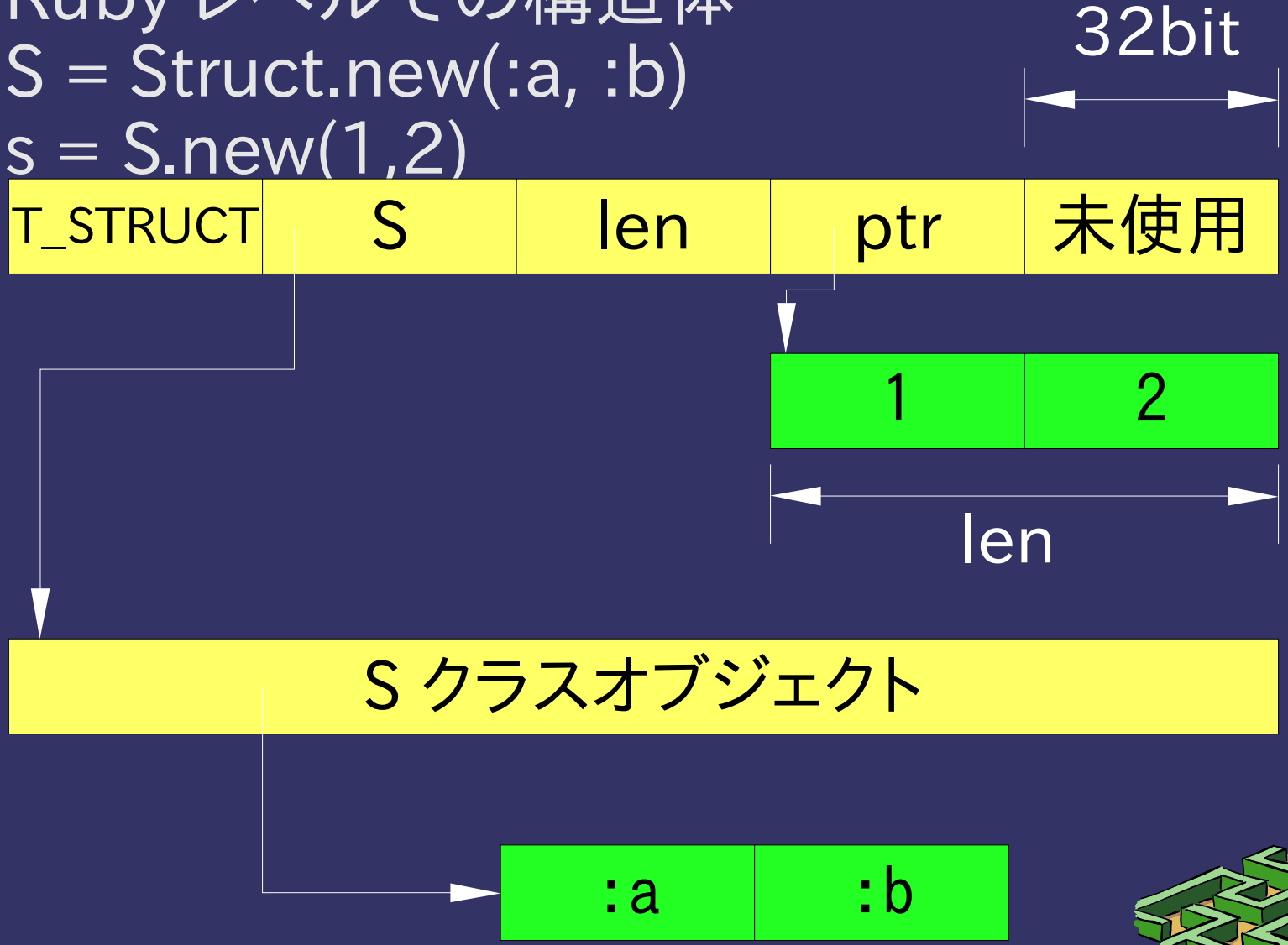
- len <= 11 なら、埋め込める
- 長さは先頭ワード内の 5bit で記録する

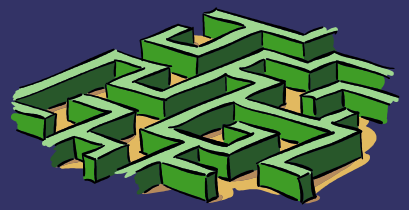




Struct

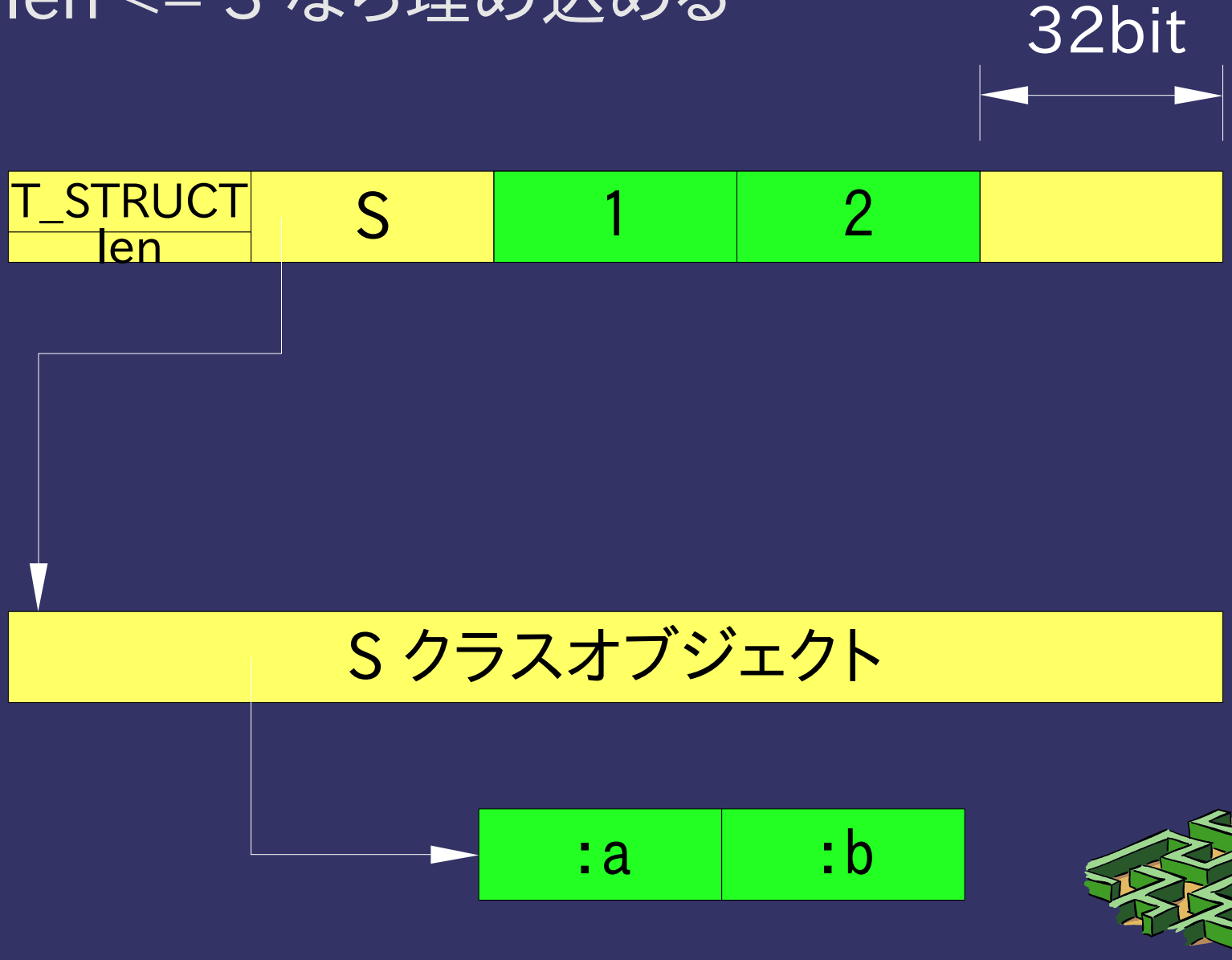
- Ruby レベルでの構造体
- S = Struct.new(:a, :b)
  s = S.new(1,2)





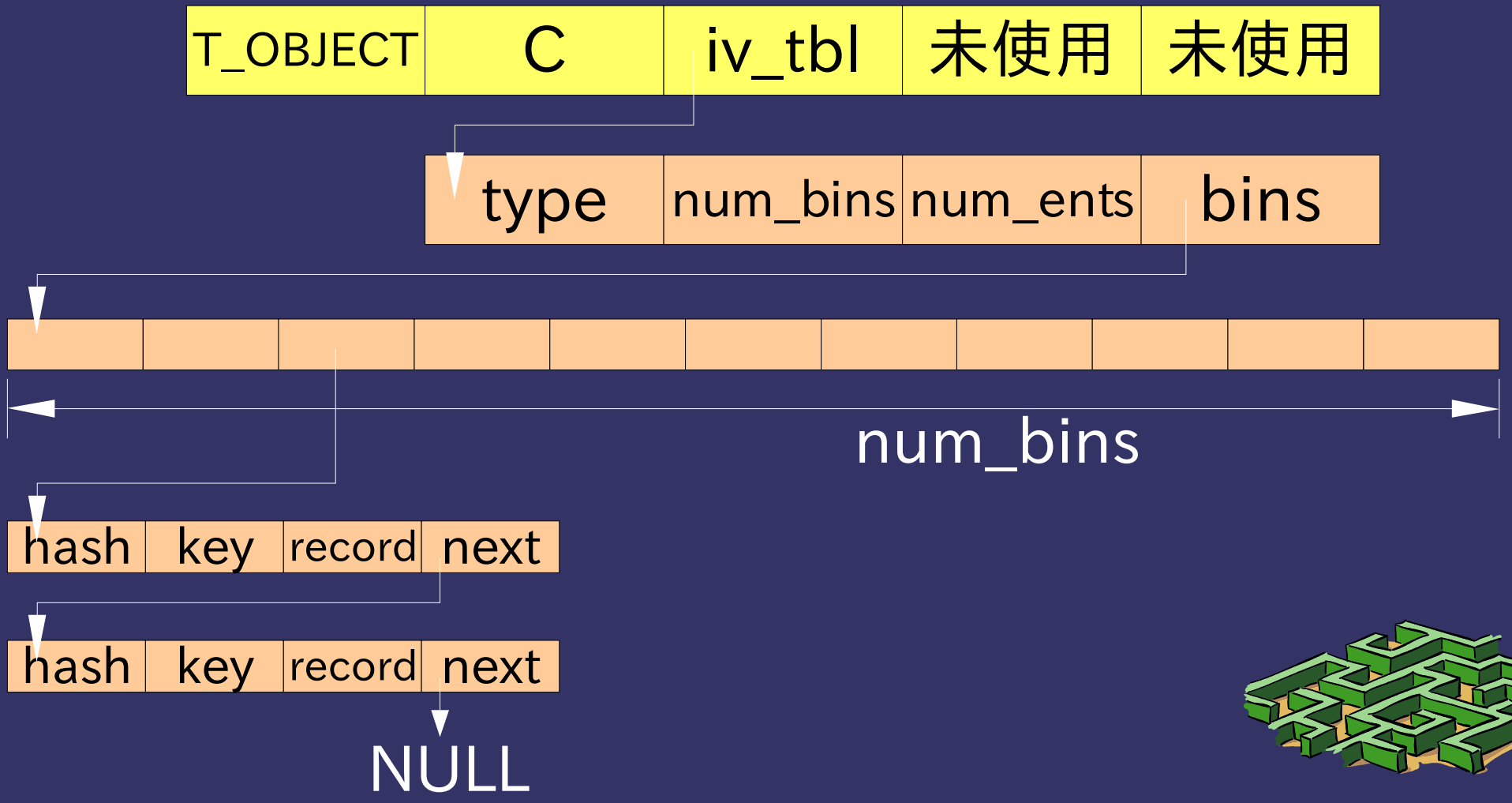
Structのメモリ節約

- len <= 3 なら埋め込める



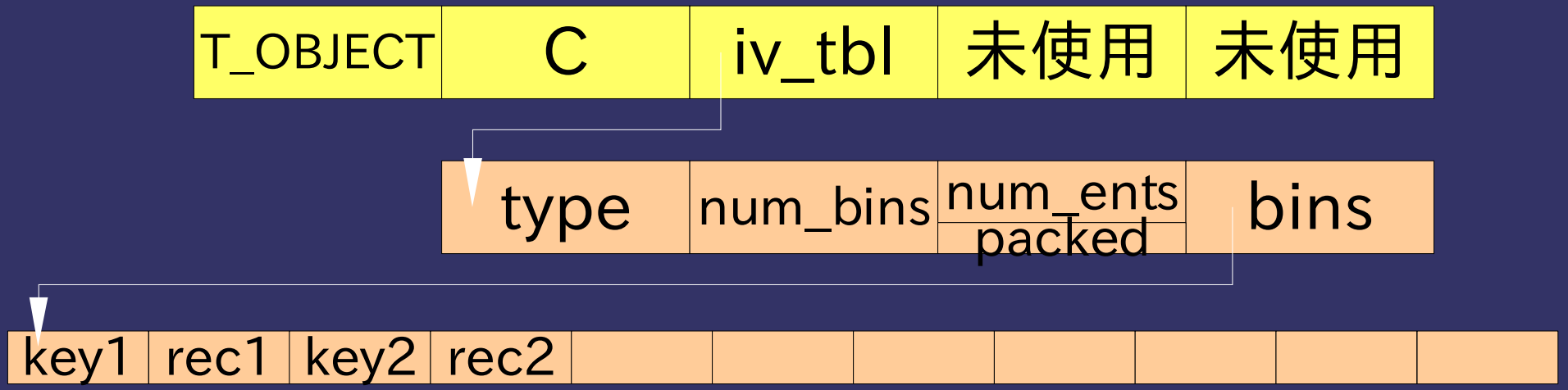
ユーザ定義クラス

- class C < Object
- ハッシュ表でインスタンス変数を管理

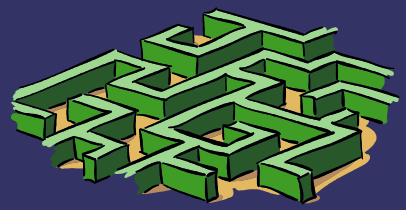


ユーザ定義クラスのメモリ節約

- bins に key, record を詰める
- iv_tbl の場合、key == hash なので hash 不要

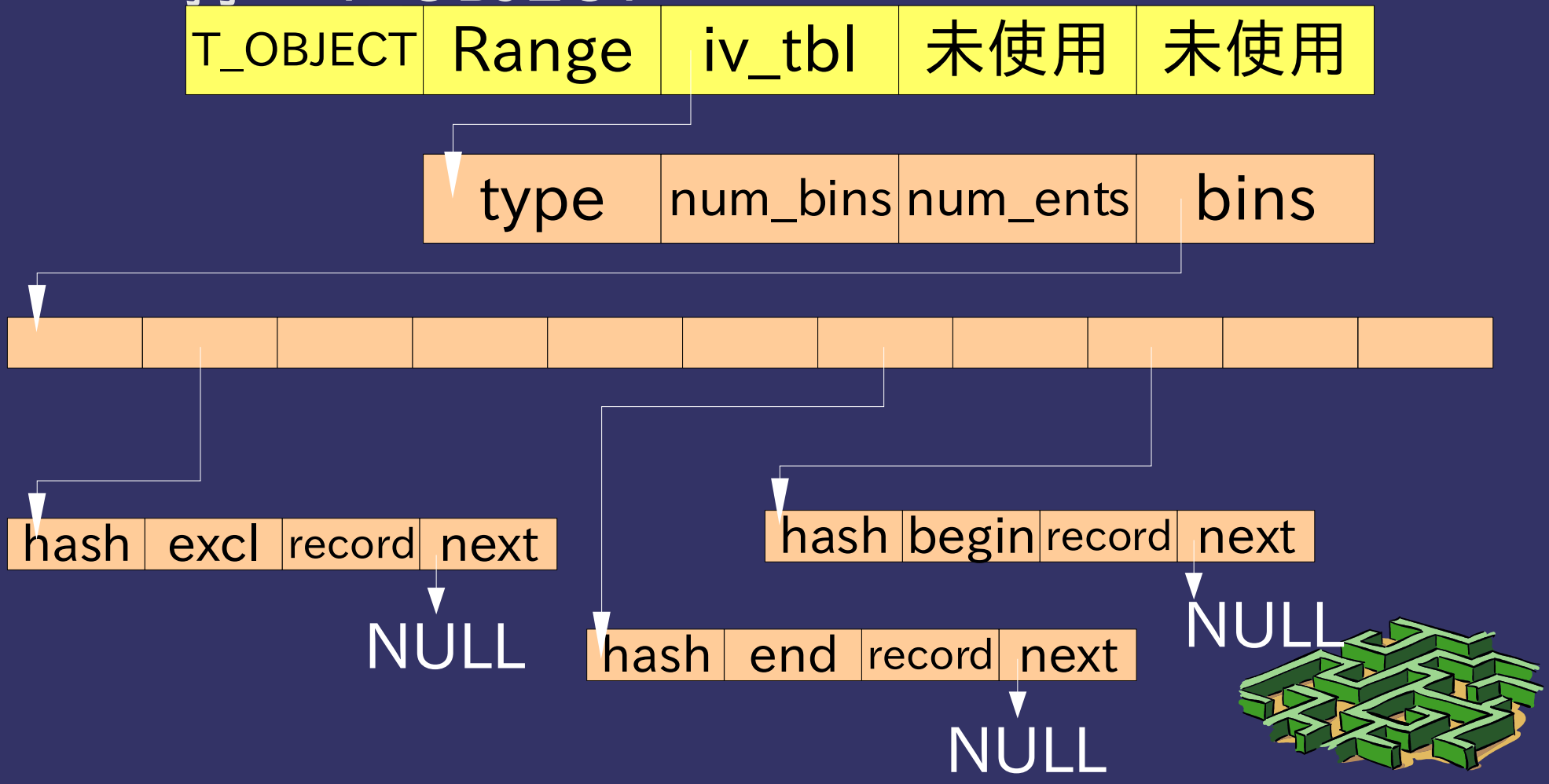


num_bins はデフォルトで 11なので、
インスタンス変数が最大 5つ入る
収まらなくなったら linked list にする



Range

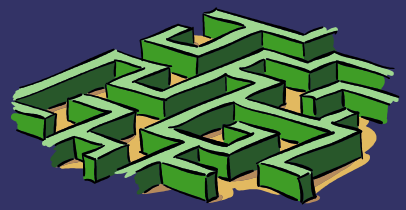
- 1..10 とか
- begin, end, excl の 3つのインスタンス変数を持つ T_OBJECT



Rangeのメモリ節約

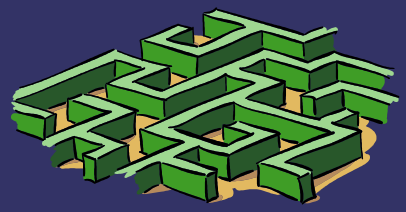
- len=3 の T_STRUCT に変更

T_STRUCT 3	Range	begin	end	excl



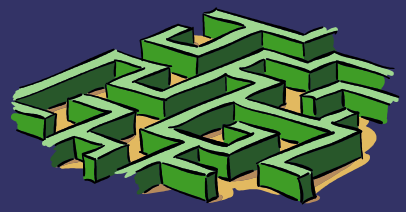
いままでのメモリ節約

- 2006-02 akr: Struct (3要素まで)
- 2006-08 matz: String (11byte まで)
- 2007-08 akr: Object (インスタンス変数 5つまで)
- 2007-08 akr: Hash (0要素)
- 2007-09 akr: Bignum ($\pm 2^{96}$-1 まで)
- 2007-09 akr: Range
- 2007-09 nobu: Hash (3要素まで)

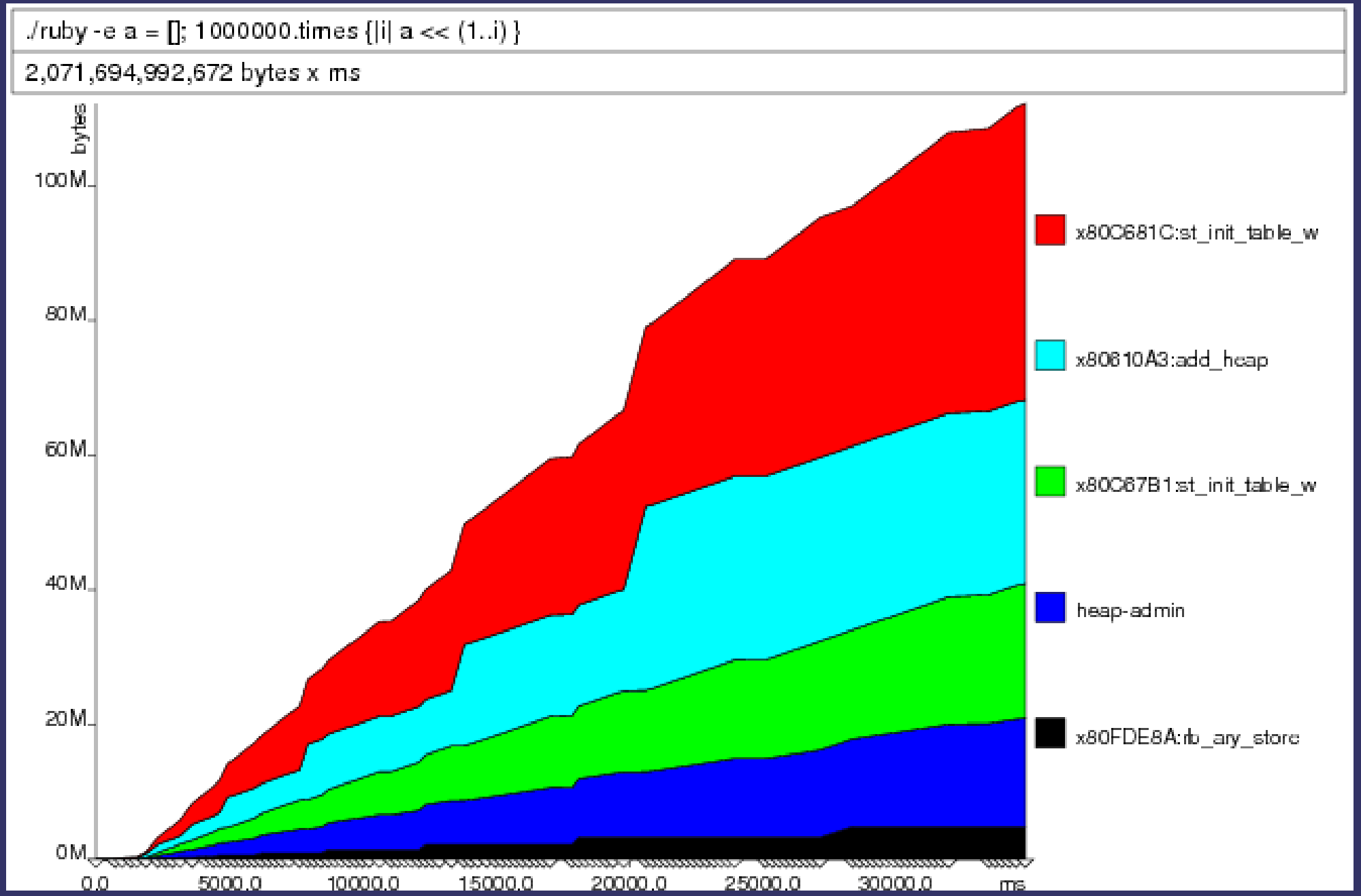


測定 (1)

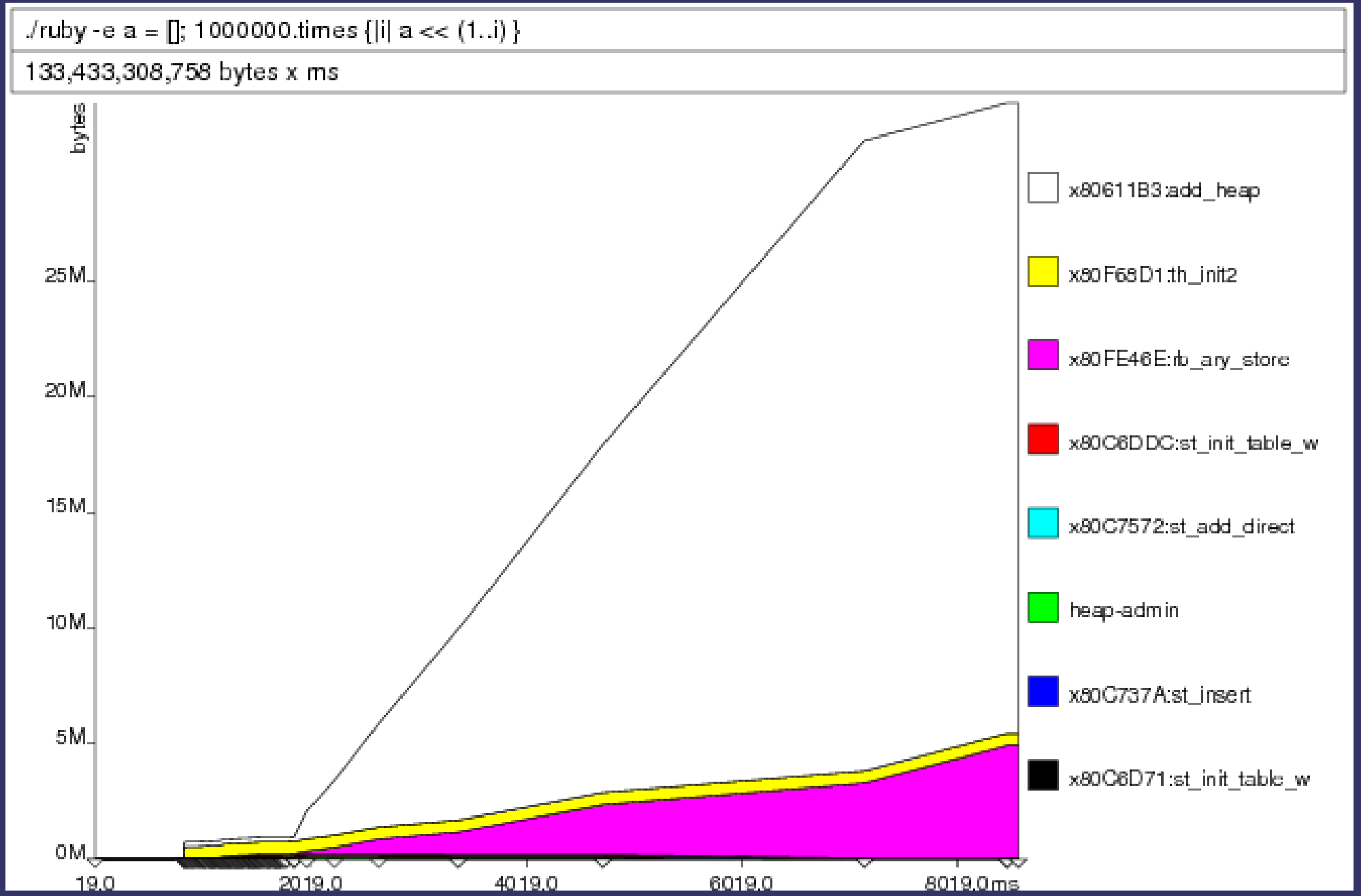
- 極端にたくさん Range を生成する
- a = []; 1000000.times {|i| a << (1..i) }
- r13409 とそれで Range のメモリ削減だけしたものの比較
- valgrind の massif でメモリ消費の時系列変化を測定する



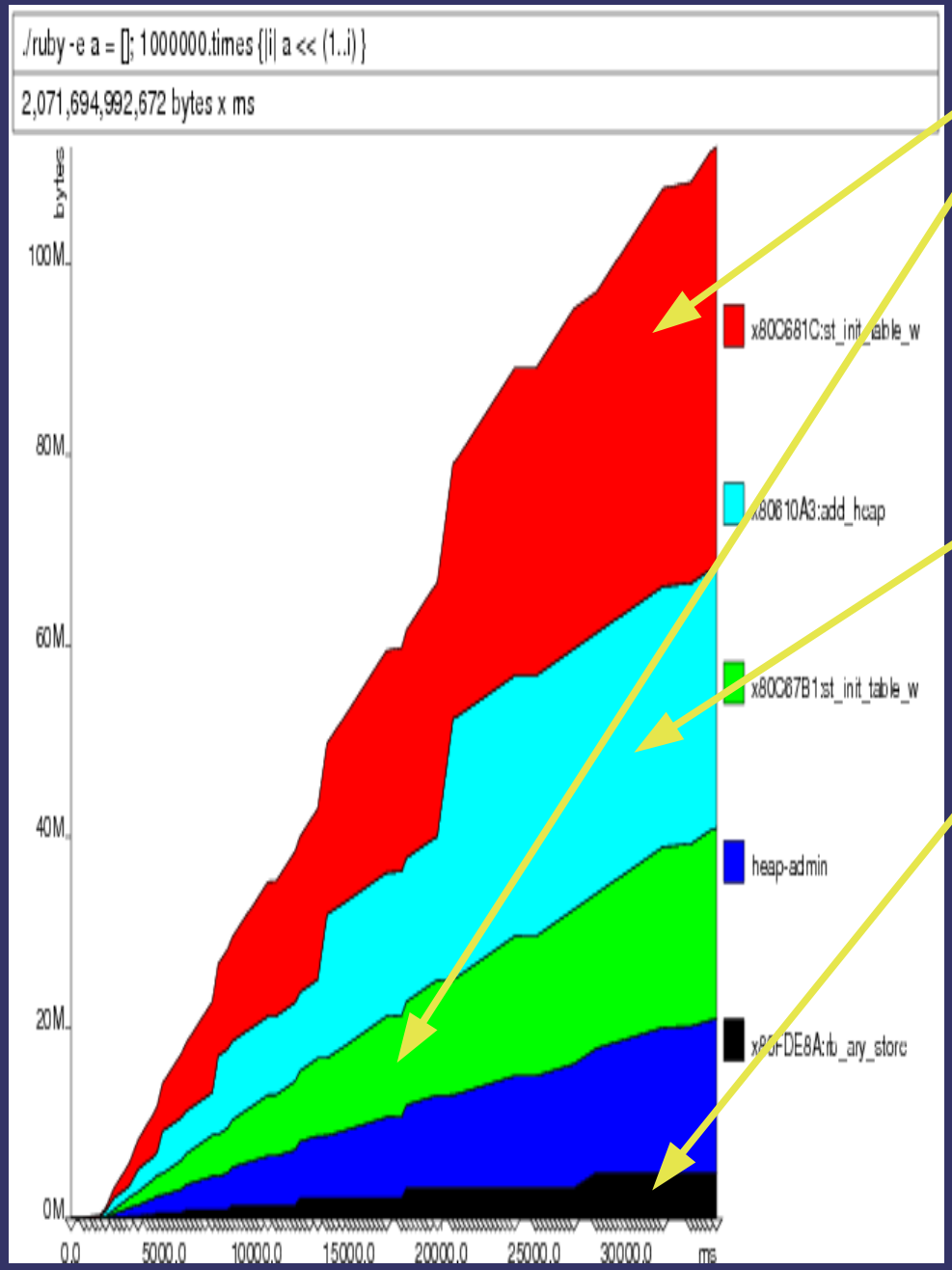
Range節約前


./ruby -e a = []; 1000000.times {|i| a << (1..i) }
2,071,694,992,672 bytes x ms
bytes
100M
80M
60M
40M
20M
0M
0.0
5000.0
10000.0
15000.0
20000.0
25000.0
30000.0
ms
x80C681C:st_init_table_w
x80610A3:add_heap
x80C67B1:st_init_table_w
heap-admin
x80FDE8A:rb_ary_store

Range節約後


./ruby -e a = []; 1000000.times {|i| a << (1..i) }
133,433,308,758 bytes x ms
bytes
25M
20M
15M
10M
5M
0M
19.0
2019.0
4019.0
6019.0
8019.0 ms
x80611B3:add_heap
x80F68D1:th_init2
x80FE46E:rb_ary_store
x80C6DDC:st_init_table_w
x80C7572:st_add_direct
heap-admin
x80C737A:st_insert
x80C6D71:st_init_table_w

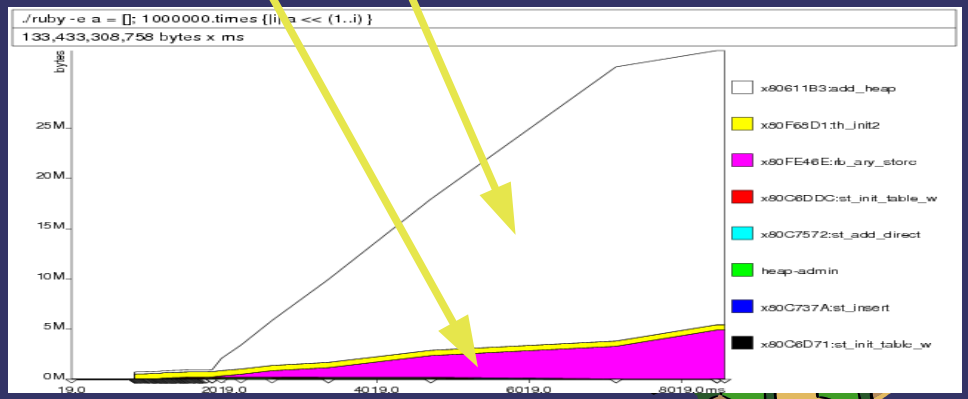
Range比較



ハッシュ表

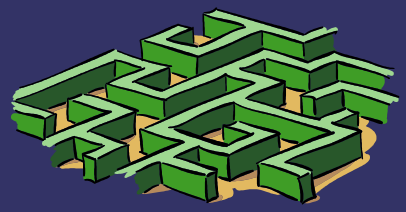
ヒープ

配列

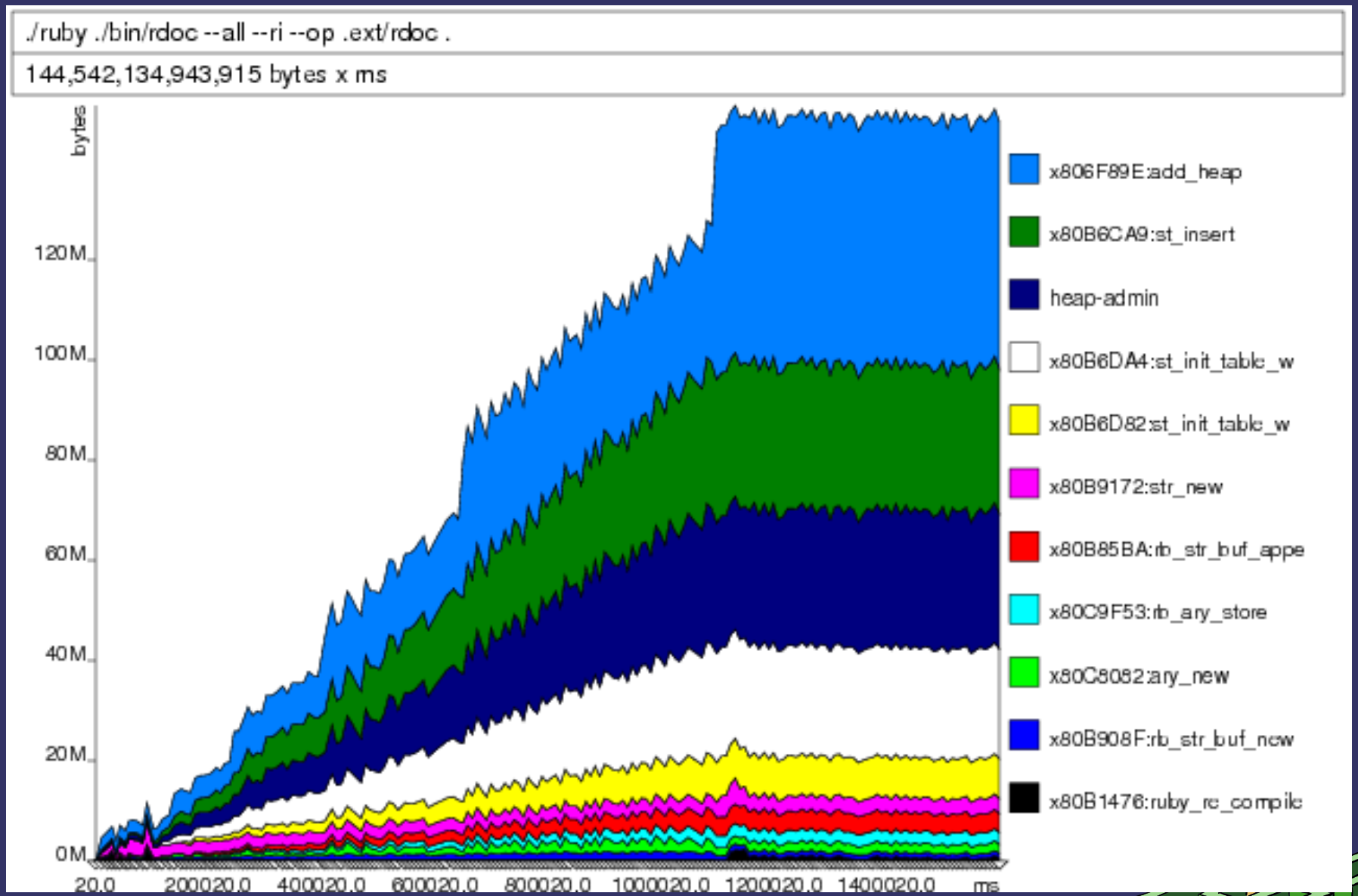


測定 (2)

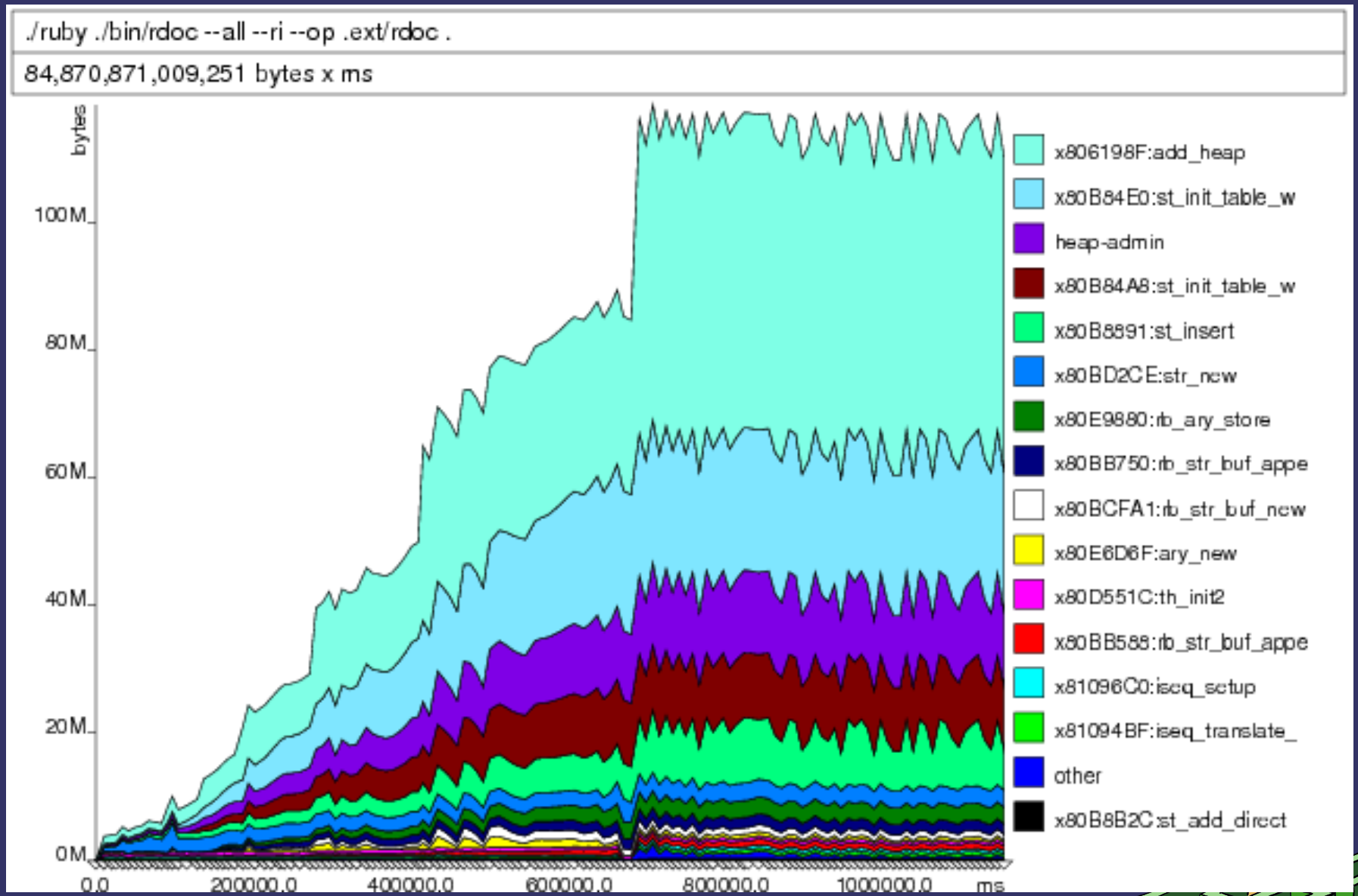
- make install-doc の rdoc
- rm -rf .ext/rdoc
  ruby ./bin/rdoc --all --ri --op .ext/rdoc .
- 1.8 と 1.9 の比較
- valgrind の massif でメモリ消費の時系列変化を測定する



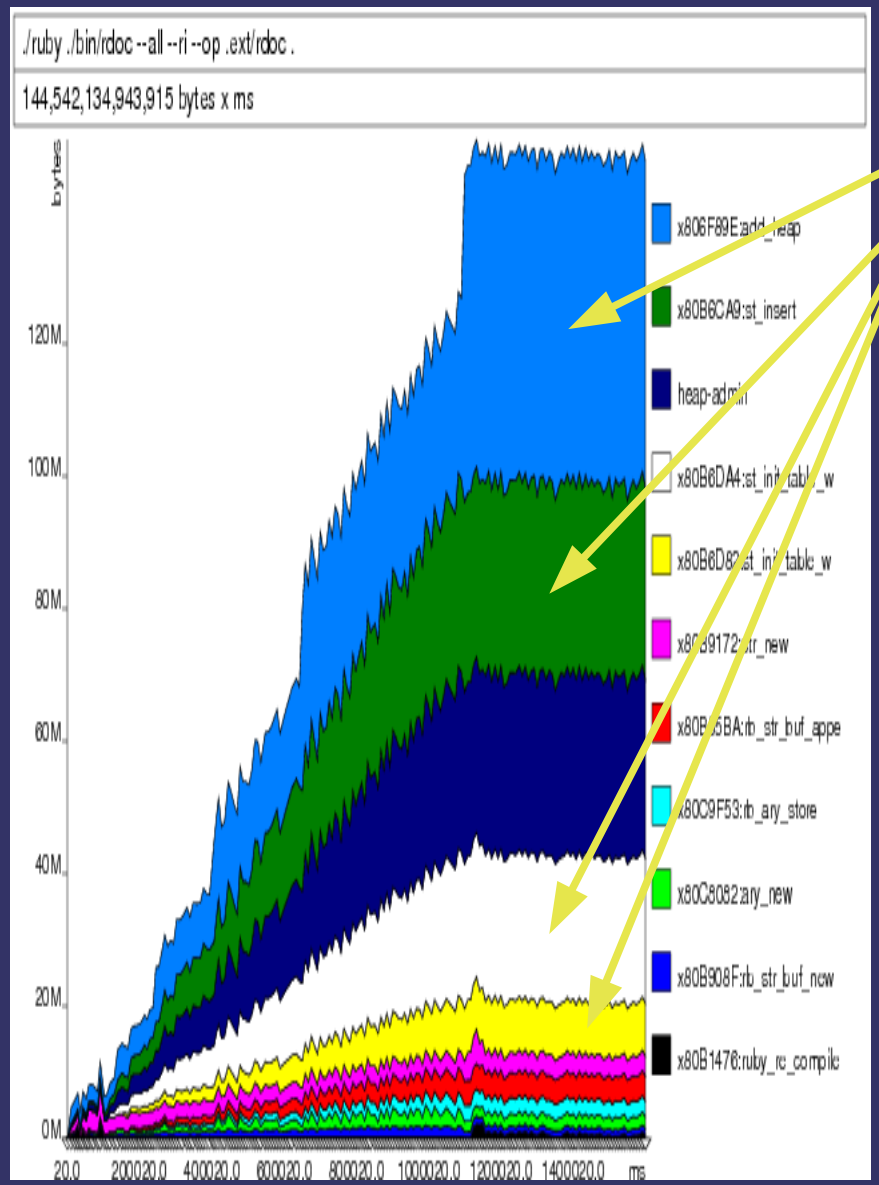
1.8


./ruby ./bin/rdoc --all --ri --op .ext/rdoc .
144,542,134,943,915 bytes x ms
bytes
120M
100M
80M
60M
40M
20M
0M
20.0
200020.0
400020.0
600020.0
800020.0
1000020.0
1200020.0
1400020.0
ms
x806F89E:add_heap
x80B6CA9:st_insert
heap-admin
x80B6DA4:st_init_table_w
x80B6D82:st_init_table_w
x80B9172:str_new
x80B85BA:rb_str_buf_appe
x80C9F53:rb_ary_store
x80C8082:ary_new
x80B908F:rb_str_buf_new
x80B1476:ruby_re_compile

1.9

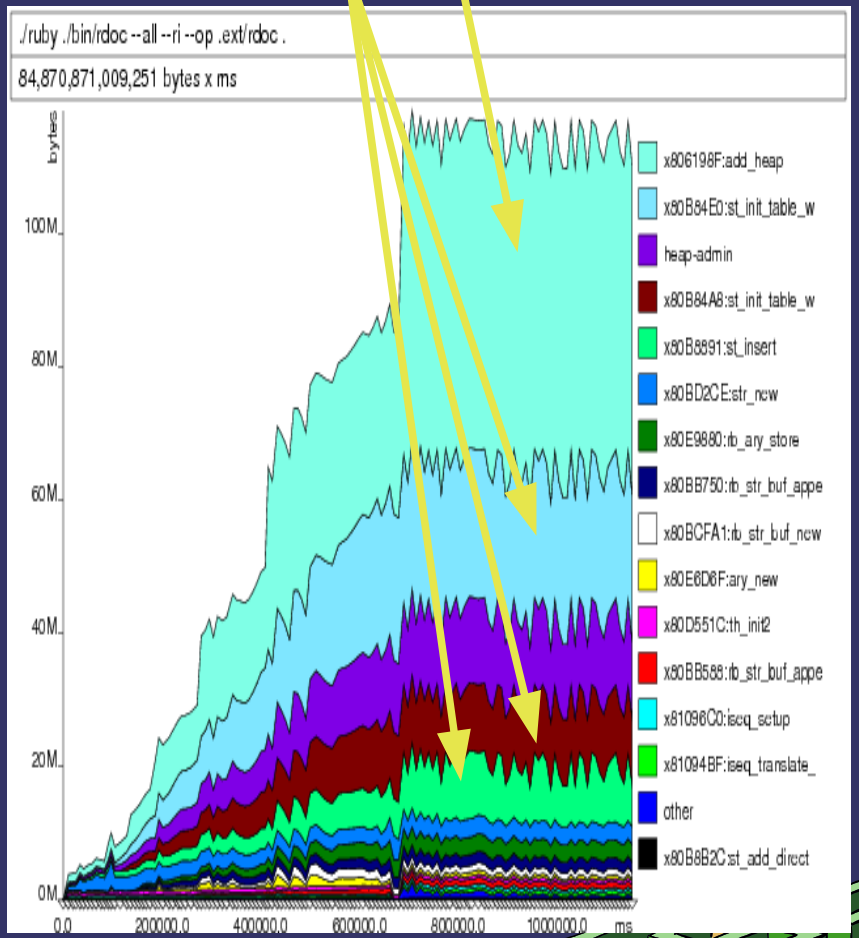

./ruby ./bin/rdoc --all --ri --op .ext/rdoc .
84,870,871,009,251 bytes x ms
bytes
100M
80M
60M
40M
20M
0M
0.0
200000.0
400000.0
600000.0
800000.0
1000000.0
ms
x806198F:add_heap
x80B84E0:st_init_table_w
heap-admin
x80B84A8:st_init_table_w
x80B8891:st_insert
x80BD2CE:str_new
x80E9880:rb_ary_store
x80BB750:rb_str_buf_appe
x80BCFA1:rb_str_buf_new
x80E6D6F:ary_new
x80D551C:th_init2
x80BB588:rb_str_buf_appe
x81096C0:iseq_setup
x81094BF:iseq_translate_
other
x80B8B2C:st_add_direct

rdoc 比較



ヒープ

ハッシュ表



まとめ

- Ruby のメモリ消費削減を行った
- Struct はメモリ効率がいいので活用しましょう