```
 b.report("lookup баркас ") do
   out = object.lookup("баркас")
   puts out.size
 end
# Аналогичные манипуляции со словом "война"
 b.report("lookup война ") do
   out = object.lookup("война")
   puts out.size
 end
# ... имя главного героя
 b.report("lookup Гриша ") do
   out = object.lookup("Гриша")
   puts out.size
 end
# и немного от себя
  b.report("lookup руби ") do
    out = object.lookup("руби")
    puts out.size
  end
end

Всё. Можно идти пить кофе. Результатом вывода программы будет:
           user     system      total        real
read...  0.000000   0.031000   0.031000 (  0.032000)
init... 84.454000   0.594000  85.048000 ( 85.109000)
lookup баркас 51
  0.000000   0.000000   0.000000 (  0.000000)
lookup война 63
  0.000000   0.000000   0.000000 (  0.000000)
lookup Гриша 164
  0.016000   0.000000   0.016000 (  0.016000)
lookup руби 120
  0.015000   0.000000   0.015000 (  0.016000)
```

Так мы опытным путём доказали, что Руби – очень древний язык. Первое число – как долго работала программа сама по себе. Второе – сколько работало ядро системы. Третье – арифметическая сумма первых двух. Последнее – сколько оттикало на настенных часах, то есть можно установить, как сильно отвлекался процессор от задачи выполнения нашей программы на другие задачи. Как видим, весьма быстро выполняются запросы, но зато весьма долго – препроцессинг (тест выполнялся на Celeron 2.40 GHz, 480 Мб ОЗУ). Полторы минуты для такой машины – это неэффективно. Надо улучшить результат. Действительно, быстрая сортировка – это, конечно хорошо, но только если нам мало что известно о сортируемых элементах. А это не так. Мы сортируем суффиксы, все их длины различны и меняются от 1 до m. Суть оптимизированной сортировки такова.

Отсортируем суффиксы по первой букве. Это можно сделать за время $O(m)$. Суффиксы с одинаковой первой буквой будут лежать в одной корзине. Суффикс $T[m \,..\, m]$ следует сразу поместить в начало своей кор-

зины. Пройдёмся по массиву Pos. Для Pos[i] верно, что суффикс Pos[i] – 1 будет первым в своей корзине. Ставим его на ближайшее к началу корзины место и помечаем это место как занятое. Получаем набор корзин с суффиксами, отсортированными по первым двум буквам. Дальше – проходим по Pos[i]. Уже для суффикса Pos[i] – 2 верно, что он будет на первом месте в своей корзине. Ставим его на ближайшее к началу корзины место, место помечаем как занятое. Получаем набор корзин с суффиксами, отсортированными по первым 4-м символам, и т.д. Весь процесс сортировки занимает время $O(m \cdot \log_2 m)$. Это уже приемлемо.

Ссылки

- http://ru.wikibooks.org/wiki/Ruby – книжка про Руби на русском языке.
- http://acm.mipt.ru/twiki/bin/view/Ruby/WebHome – материалы про Руби на сайте МФТИ.
- http://eigenclass.org/hiki/simple+full+text+search+engine – статья про реализацию суффиксных массивов на Руби.
- http://rain.ifmo.ru/cat/view.php/theory/unsorted/suffix-arrays-2005 – очень просто и понятно.
- Manber, Myers-Suffix arrays – A new method for on-line string searches. Книга про суффиксные массивы.

Юмор Юмор Юмор Юмор Юмор Юмор

♦ Теорема о глубине лужи

О глубине лужи нельзя судить до тех пор, пока в неё не упадёшь.

♦ I закон спора

Никогда не вступай в спор с дураком, так как люди могут не заметить разницы между вами.

♦ II закон спора

Кто громче всех кричит, тому и дают слово.