Lazy vs. non lazy

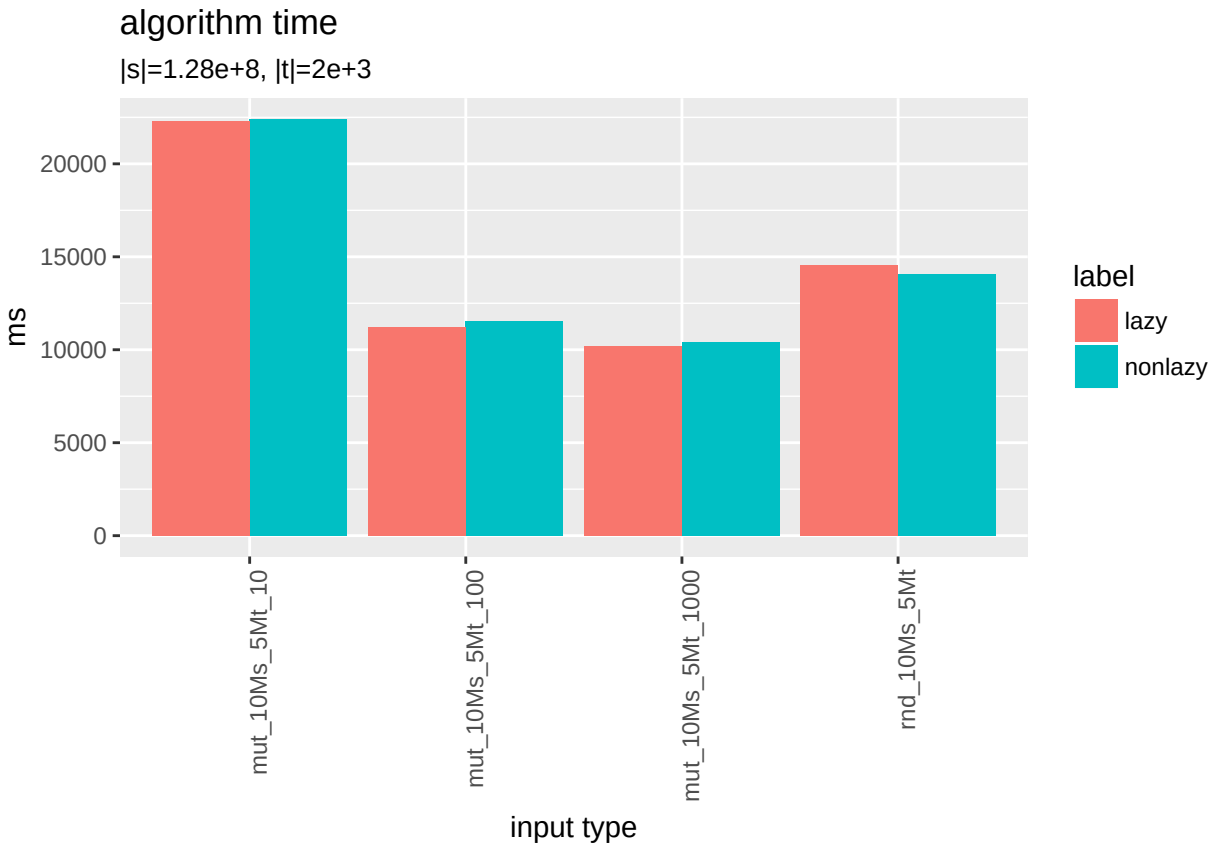
O. Denas

12/28/2016

Contents

1 Lazy vs non-lazy

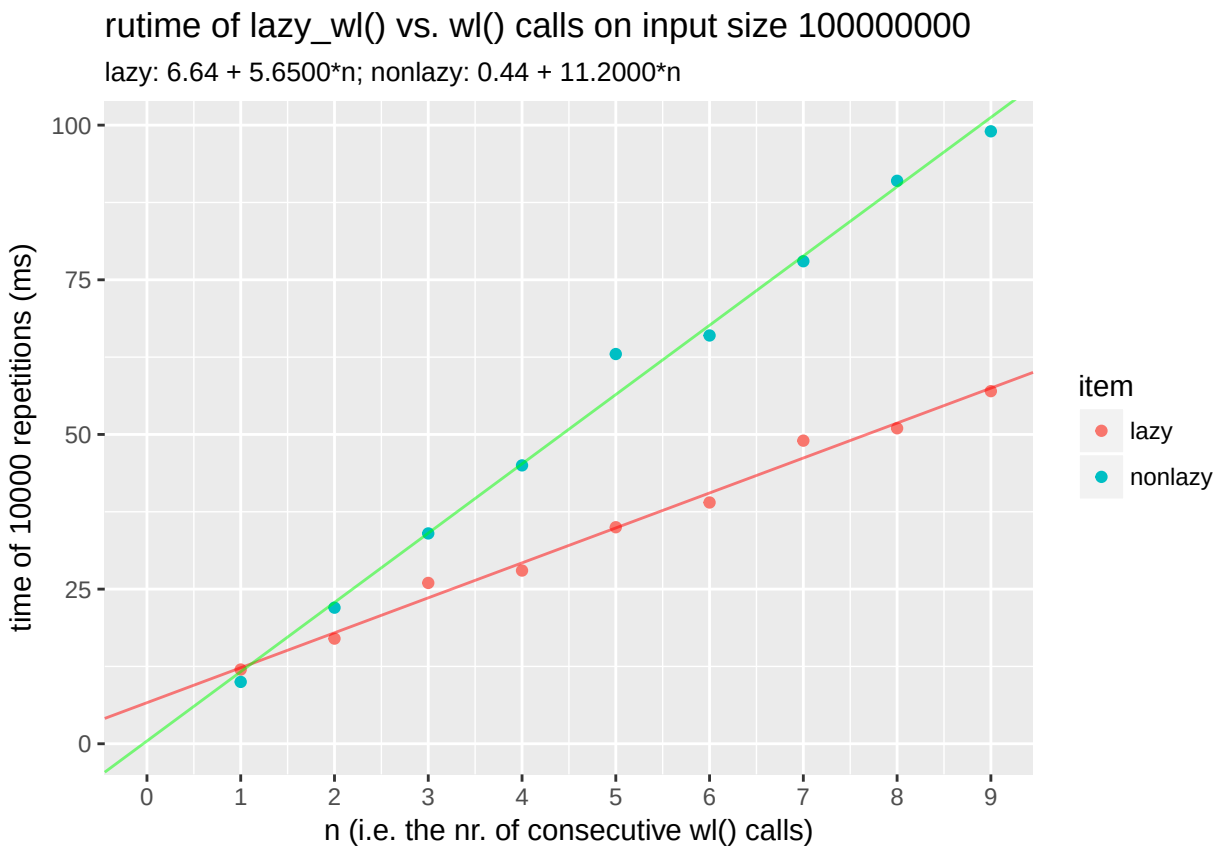
1.1 Run time



1.2 Sandbox timing

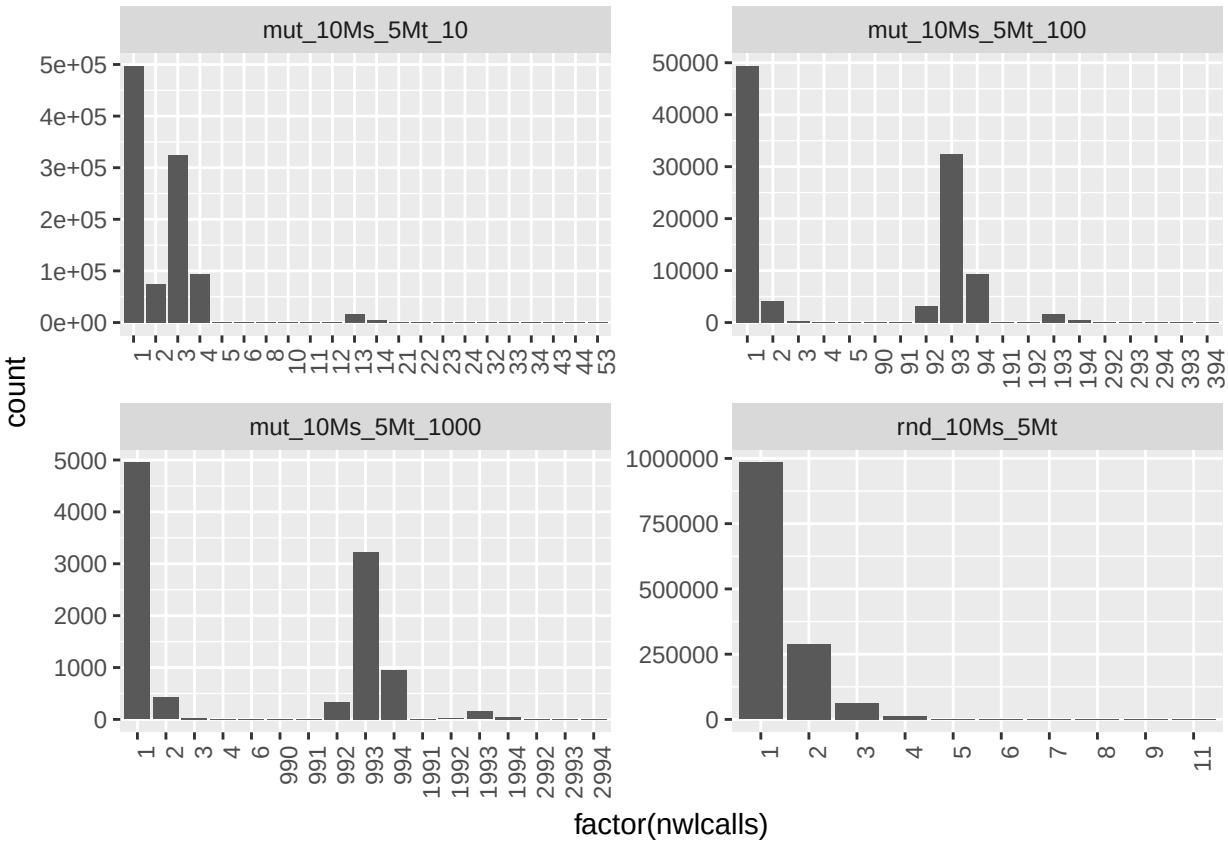
Measure the time of 10k repetitions of

(1) n consecutive `lazy_wl()` calls followed by a `lazy_wl_followup()` and
(2) n consecutive `wl()` calls



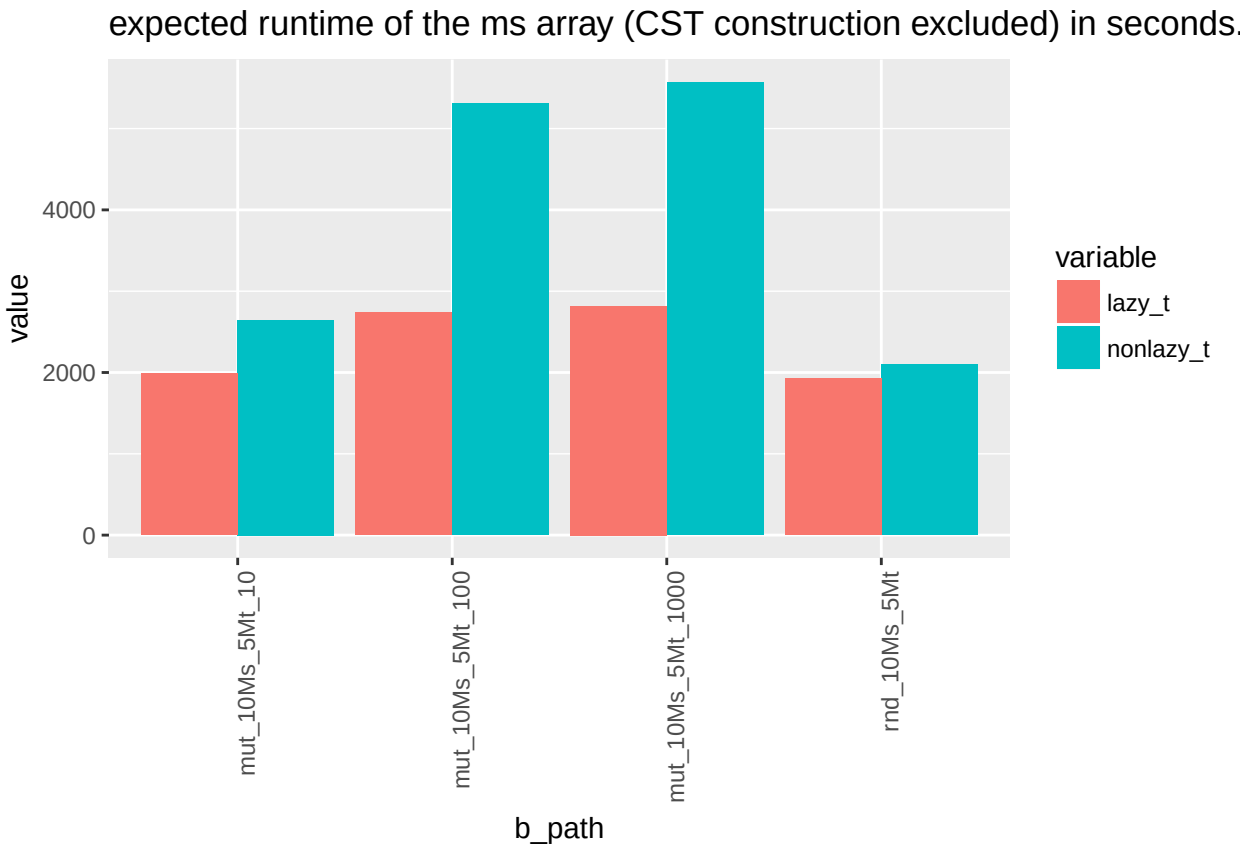
1.3 Input properties

For various types ("mut_XMs_YMt_Z" means `s` and `t` are random identical strings of length X, and Y million respectively with mutations inserted every Z characters. "rnd_XMs_YMt" means `s` and `t` are random strings of length X, and Y million respectively) of inputs run the MS algorithm and count the number of consecutive `lazy_wl()` calls.



Using the linear fits above, this is the expected toal time the `wl()` or `lazy_wl()` calls should take (in ms).

```
## # A tibble: 4 × 5
##              b_path   lazy_t nonlazy_t abs_diff_ms rel_diff_ms
##               <chr>    <dbl>     <dbl>       <dbl>       <dbl>
## 1   mut_10Ms_5Mt_10 1986.532  2646.988   -660.4558  -24.951223
## 2  mut_10Ms_5Mt_100 2740.874  5304.468  -2563.5940  -48.328952
## 3 mut_10Ms_5Mt_1000 2816.587  5570.411  -2753.8233  -49.436630
## 4      rnd_10Ms_5Mt 1926.194  2098.330   -172.1368   -8.203512
```


expected runtime of the ms array (CST construction excluded) in seconds.
value
4000
2000
0
mut_10Ms_5Mt_10
mut_10Ms_5Mt_100
mut_10Ms_5Mt_1000
rnd_10Ms_5Mt
b_path
variable
lazy_t
nonlazy_t