Lazy vs. non lazy

O. Denas

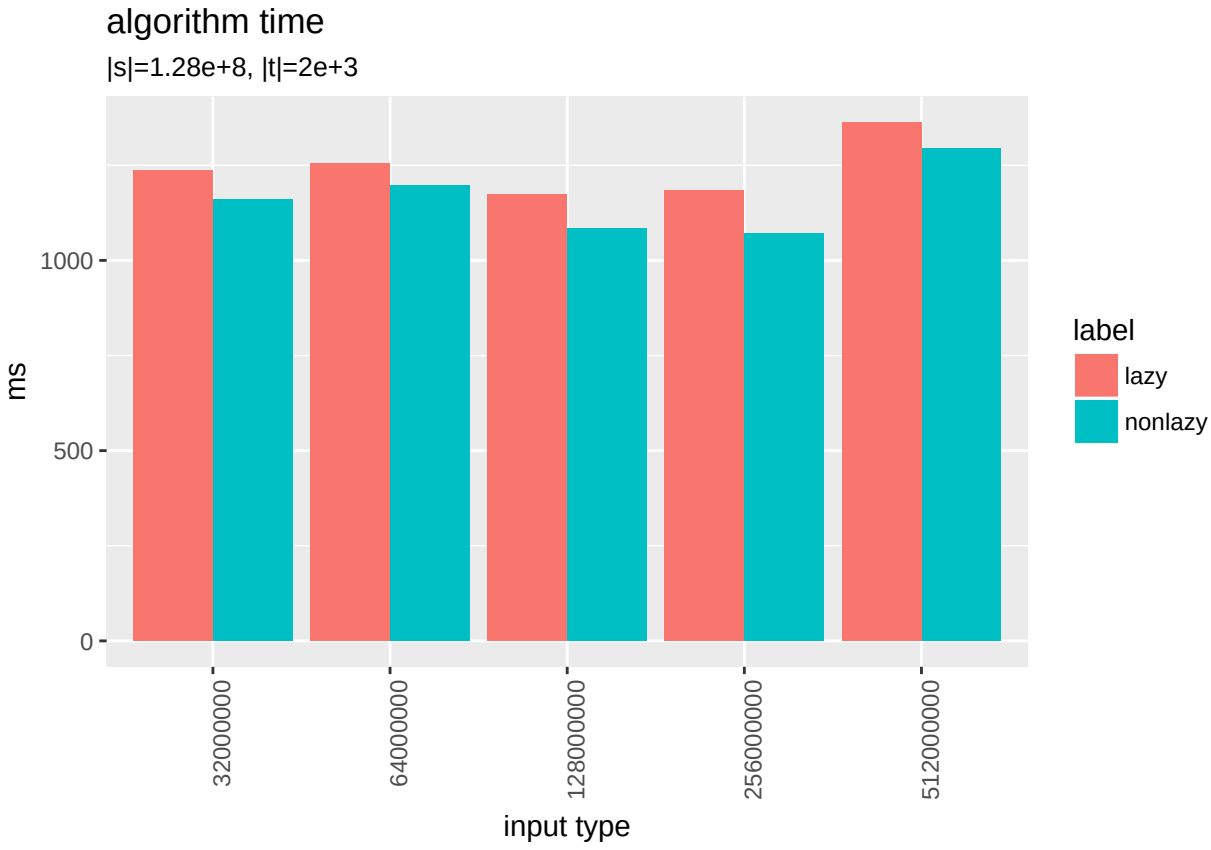
12/28/2016

Contents

1 Double vs. single rank

2 Lazy vs non-lazy

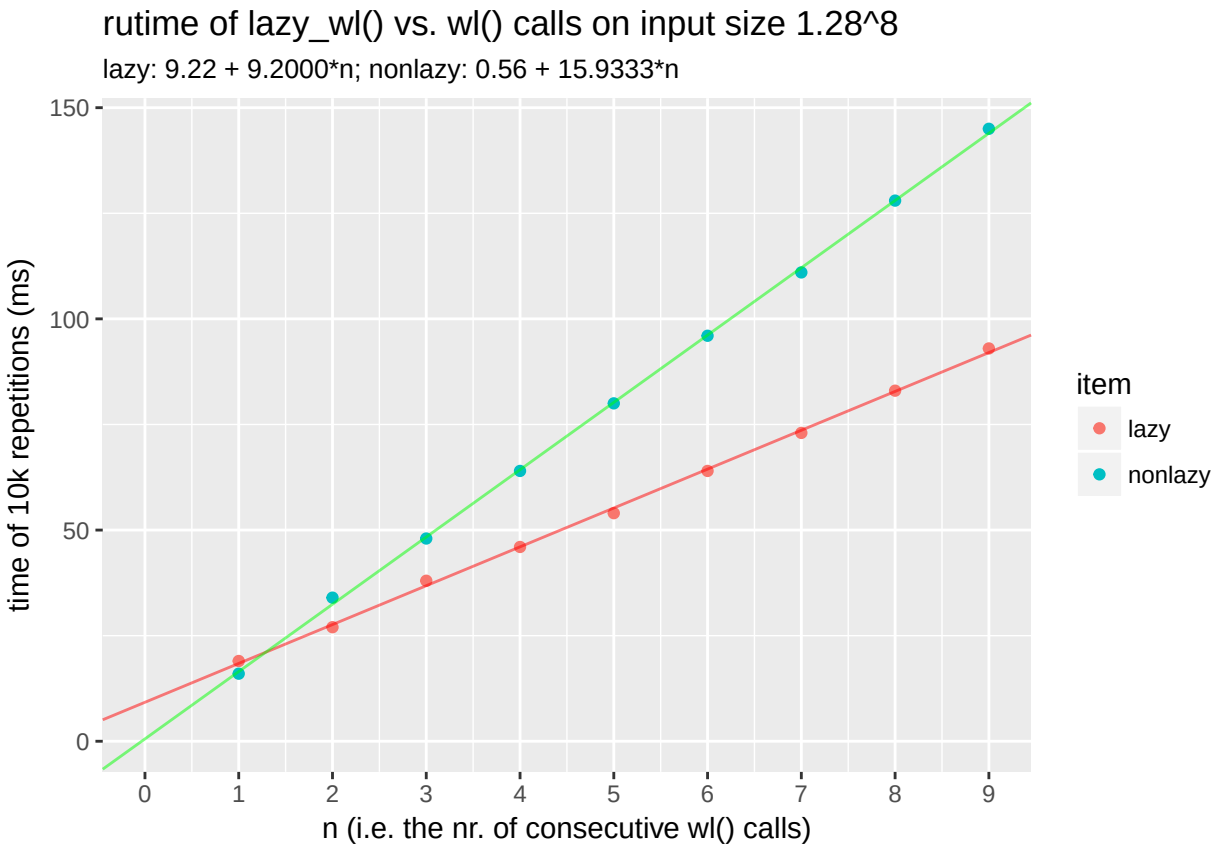
2.1 Run time



2.2 Sandbox timing

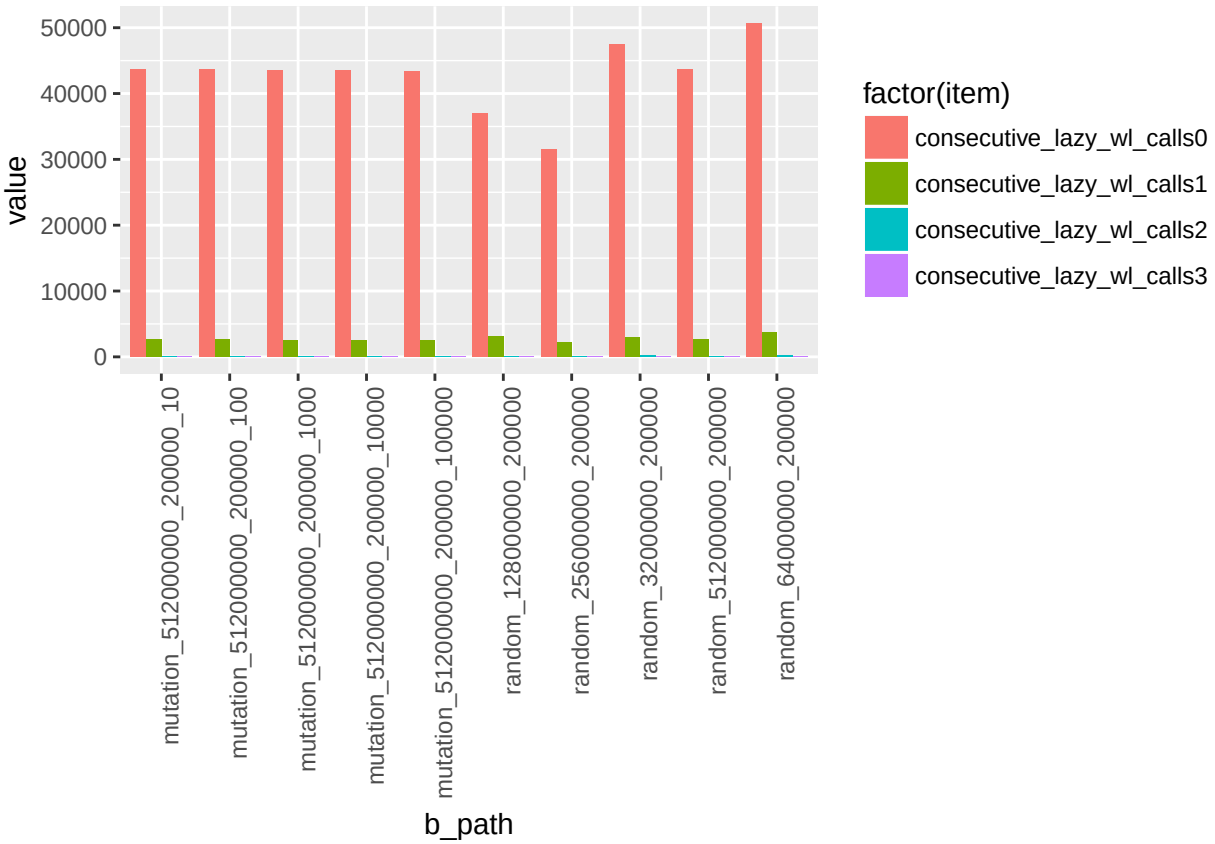
Measure the time of 10k repetitions of

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



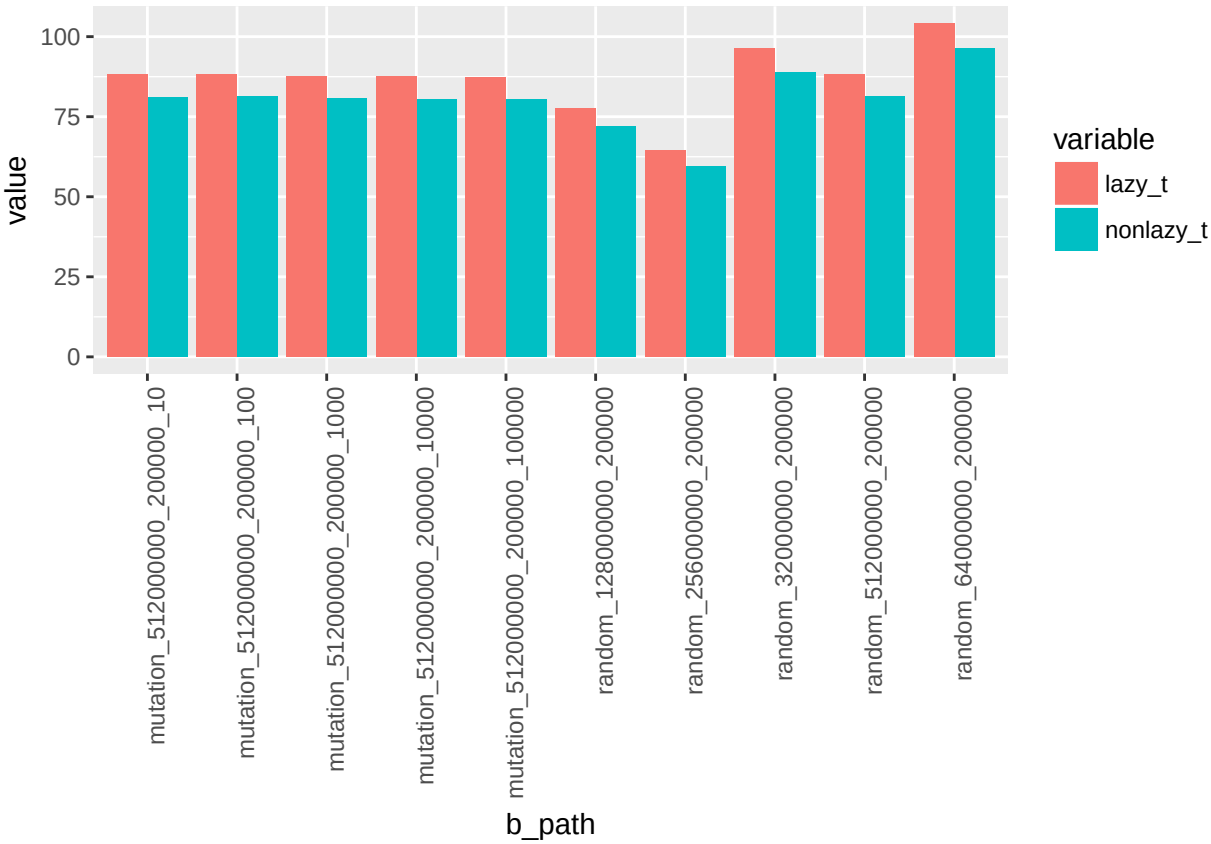
2.3 Input properties

For various types ("mutation_xxxx" means `s` and `t` are random identical strings with mutations inserted every xxxx characters) of inputs run the MS algorithm and count the number of consecutive `lazy_wl()` calls of length k for $k = 0, 1, 2, 3$.



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

```
## # A tibble: 10 × 4
##                               b_path    lazy_t nonlazy_t  diff_ms
##                                <chr>     <dbl>     <dbl>    <dbl>
## 1      mutation_512000000_200000_10  88.14617  81.14290 7.003273
## 2     mutation_512000000_200000_100  88.28988  81.26228 7.027593
## 3    mutation_512000000_200000_1000  87.70322  80.67022 7.033007
## 4   mutation_512000000_200000_10000  87.55585  80.53369 7.022160
## 5  mutation_512000000_200000_100000  87.40388  80.39074 7.013140
## 6           random_128000000_200000  77.58776  72.13446 5.453300
## 7           random_256000000_200000  64.53955  59.64030 4.899253
## 8            random_32000000_200000  96.36021  88.79362 7.566587
## 9           random_512000000_200000  88.24567  81.21810 7.027573
## 10           random_64000000_200000 104.27172  96.44919 7.822533
```


value
100
75
50
25
0
mutation_512000000_200000_10
mutation_512000000_200000_100
mutation_512000000_200000_1000
mutation_512000000_200000_10000
mutation_512000000_200000_100000
random_128000000_200000
random_256000000_200000
random_32000000_200000
random_512000000_200000
random_64000000_200000
b_path
variable
lazy_t
nonlazy_t