Project 2: Genome Scale algorithms

Maria, Mateo and Jacob

2/20/2018

Project 2 - Suffix tree construction

Program description

The programs are implemented in Python (3) and we had no unsolved issues implementing the algorithms. Execution of the program on command line takes the form:

```
python suffix_tree_naive.py [text-filename] [string-pattern]
```

The output is directly to the console as a list of indexes of each first character of the pattern present in the text. If the pattern does not exist in the text `None` is returned.

Test of search-suffix-tree-naive

We used the data provided for testing which outputs the following results:

python suffix_tree_naive.py mississippi.txt ss [3, 6]

python suffix_tree_naive.py banana.txt ana [2, 4]

We have also compared those results to the Naive exact match approach, and they give the same indexes.

Time complexity

Building the suffix tree

The implementation of the algorithm was made using the naive approach, which has $O(n^2)$ time construction complexity. We defined the worst scenario to build a suffix tree by creating a text T that has alphabet size $|\sum| = 1$. For example: 'aaaaaaaa`$`', where the sentinel character (`$`) is included in order to ensure that every suffix of the text `T$` is represented by a leave. We increased the length of the text and stored the time of each tree-building operation.

The best-case time bounds was construct by including

Searching through the tree

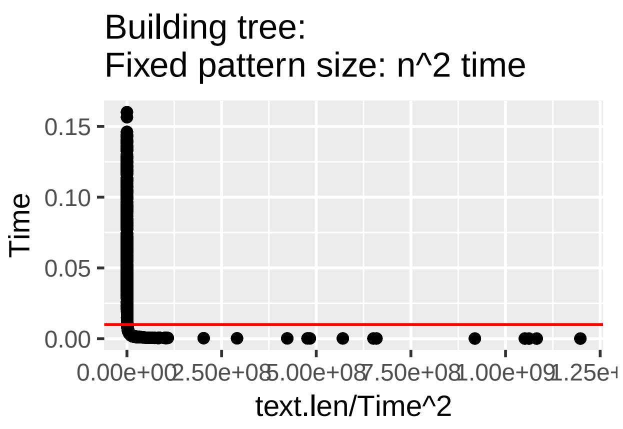
Once the suffix tree from T is build then search through the suffix tree. Since all the substring of the text T is present in the suffix tree it is easy to verify if a pattern P would be present in the text. It goes from the first character of the pattern until its last, if one cannot proceed through the next character of pattern through the tree, it means that the pattern does not occur in the suffix tree/text. The operation of searching for the pattern through the tree takes $O(m??log(|\sum|))$ time.

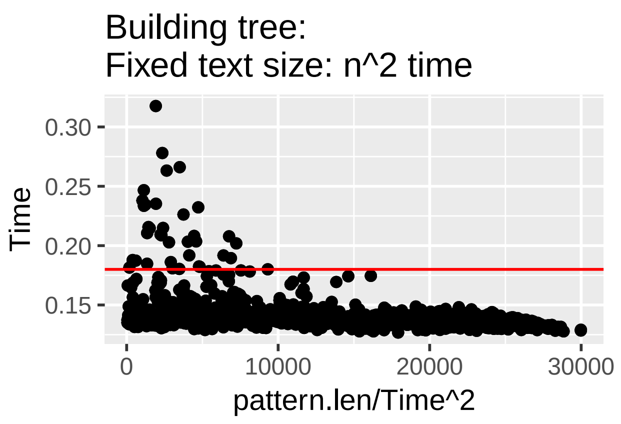
We have decided to evaluate the data based on 4 different scenarios: * Tx-Pc: where the text is variable and the pattern is constant * Tc-Px: where the text is constant and the pattern is variable * Tx-Px: both text and patterns are variable

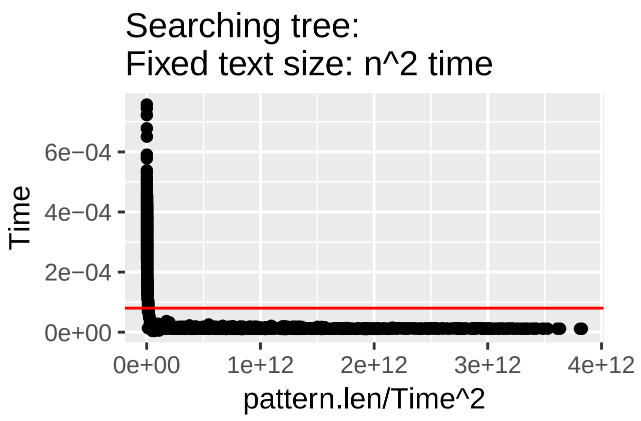
```r
setwd('/Users/PM/Dropbox/Genome_scale_algorithms/Genome_scale_algorithms/Project2')
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

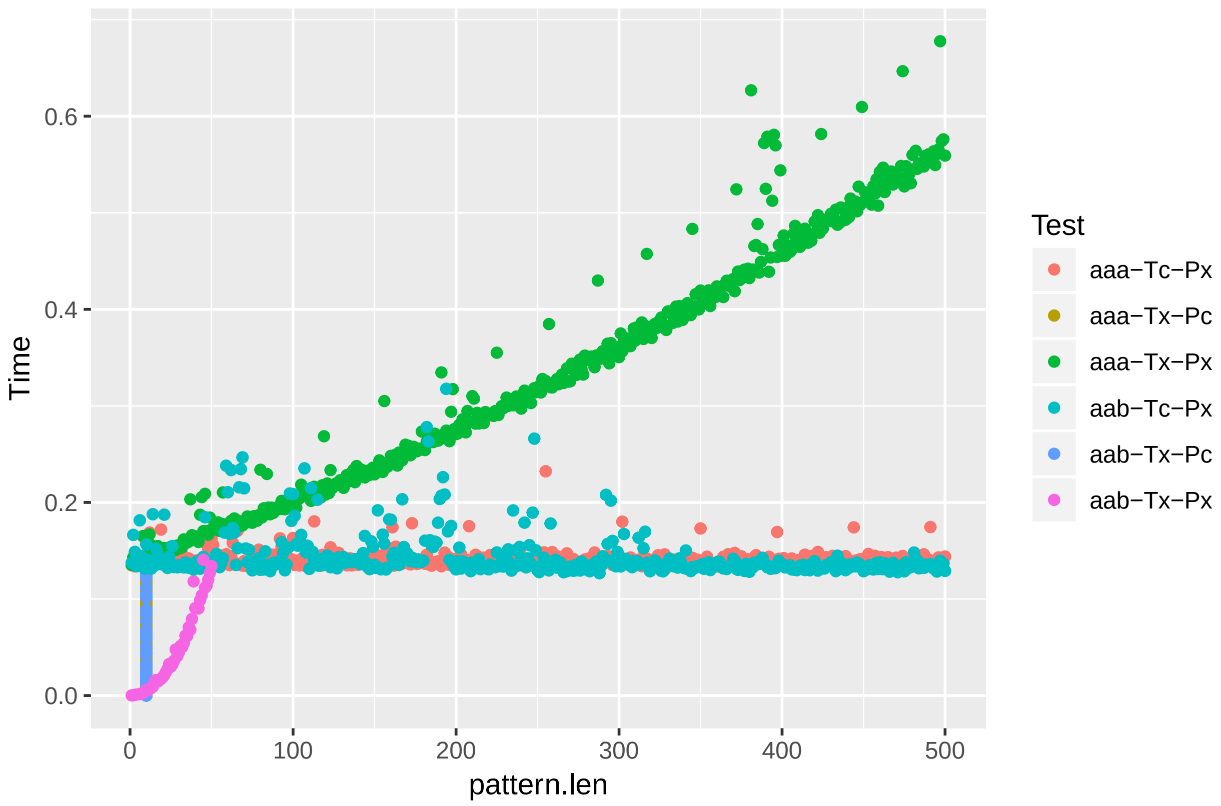
```
library(ggpubr)

df_build.naive = read.csv( "Test - SuffixTree Build - Naive.csv" , header = F)
colnames(df_build.naive) = c("Function", "Time", "Test", "text.len", "pattern.len", "occurrences", "?")



df_search = read.csv( "Test - SuffixTree Search.csv" )
colnames(df_search) = c("Function", "Time", "Test", "text.len", "pattern.len", "occurrences", "?")

fixed_text_size_build <- df_build.naive %>%
  filter(text.len == 500)

fixed_patter_size_build <- df_build.naive %>%
  filter(pattern.len == 10)

fixed_text_size_search <- df_search %>%
  filter(text.len == 500)

fixed_pattern_size_search <- df_search %>%
  filter(pattern.len == 10)


plot1 <- ggplot(data = fixed_patter_size_build) +
  geom_point(mapping = aes(x = text.len/Time^2, y = Time)) +
  ggtitle("Building tree: \nFixed pattern size: n^2 time") +
  geom_hline(yintercept = 0.01, color = "red")

plot2 <- ggplot(data = fixed_text_size_build) +
  geom_point(mapping = aes(x = pattern.len/Time^2, y = Time)) +
  geom_hline(yintercept = 0.18, color = "red") +
  ggtitle("Building tree: \nFixed text size: n^2 time")

plot3 <- ggplot(data = fixed_text_size_search) +
  geom_point(mapping = aes(x = pattern.len/Time^2, y = Time)) +
  geom_hline(yintercept = 0.00008, color = "red") +
  ggtitle("Searching tree: \nFixed text size: n^2 time")

#we dont need fourth plot since, searching the tree is not affected by text size, only the pattern

ggarrange(plot1,plot2,plot3,
      ncol = 2, nrow =2)
```







```
ggplot(data = df_build.naive) +
  geom_point(mapping = aes(x = pattern.len, y = Time, color = Test)) +
  ggtitle("")
```



Instruction of the project: https://github.com/mailund/gsa-exercises/tree/master/Project02