

# Report of Progress

by Jiachen Yang

Jiachen Yang<sup>1</sup>

<sup>1</sup>Research Student in Osaka University

October 25, 2011

# Works in last week

- Implemented Generalized Suffix Tree to extract Maximal Common Sequence from multiple input files.
- Changed tokenize framework from "tokenize" package in Python to PLY(Python Lex-Yacc) project.  
Add other language support in future.

Paper reading ?, ?, ?.

# Generalized Suffix Tree

## What is GST

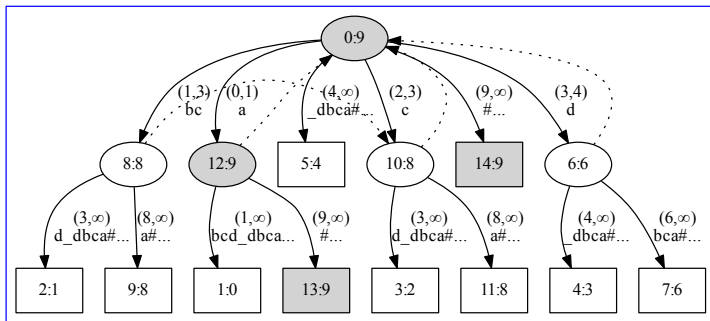
- Generalized Suffix Tree(GST): A Suffix Tree that contains multiple strings.
- Advantage: Find Maximal Common Sequence(MCS) in linear time of sum of length of strings.
- Disadvantage: All strings are stored in memory.

## Building GST:

- 1 Append each string with a unique terminal mark(\$)
- 2 Concat strings as a long string.  
STRING1\$<sub>1</sub>STRING2\$<sub>2</sub>STRING3\$<sub>3</sub>...
- 3 build normal Suffix Tree with this long string
- 4 find CS that terminate at different terminal marks.

# GST example

Figure: Suffix Tree of abcd\_dbca#



```
$ echo abcd_dbca# | ../filter.py
bc 2:{1, 6}
```

# Implement GST on multiple token sequence

- ① Append each token sequence with a unique ENDMARK.
- ② Build a list with positions of ENDMARKs.  
ENDMARK list looks like: {100,300,500,800}
- ③ A filter on generated MCS that
  - compare MCS end index with ENDMARK list.  
MCS looks like: 40, {120,330,600}  
end-index of above MCS is: {160,370,640}
  - extract those MCS that ends at different ENDMARK position.

This approach can extract MCS in different files as well as in different groups of files.

# Example of GST filter on python token sequence

```
$ ./st_token.py *.py
62:{7213, 8150} NAME,NEWLINE,DEDENT,DEDENT,CLASS,NAME,COLON,
NEWLINE,INDENT,DEF,NAME,LPAR,NAME,COMMA,NAME,COMMA,NAME,
COMMA,NAME,COMMA,NAME,RPAR,COLON,NEWLINE,INDENT,NAME,DOT,
NAME,EQUAL,NAME,NEWLINE,NAME,DOT,NAME,EQUAL,NAME,NEWLINE,
NAME,DOT,NAME,EQUAL,NAME,NEWLINE,NAME,DOT,NAME,EQUAL,NAME,
NEWLINE,DEDENT,DEF,NAME,LPAR,NAME,RPAR,COLON,NEWLINE,INDENT,
RETURN,NAME,DOT,NAME
60:{304, 6196} COMMA,STRING,COMMA,STRING,COMMA,STRING,COMMA,
STRING,COMMA,STRING,COMMA,STRING,COMMA,STRING,COMMA,STRING,
COMMA,STRING,COMMA,STRING,COMMA,STRING,COMMA,STRING,COMMA,
STRING,COMMA,STRING,COMMA,STRING,COMMA,STRING,COMMA,STRING,
COMMA,STRING,COMMA,STRING,COMMA,STRING,COMMA,STRING,COMMA,
STRING,COMMA,STRING,COMMA,STRING,COMMA,STRING,COMMA,STRING,
COMMA,STRING,COMMA,STRING,COMMA,STRING,COMMA,STRING
40:{7997, 10214} NAME,DOT,NAME,LSQB,NAME,RSQB,NEWLINE,DEDENT,
DEF,NAME,LPAR,NAME,COMMA,NAME,COMMA,NAME,RPAR,COLON,NEWLINE,
INDENT,NAME,DOT,NAME,LSQB,NAME,RSQB,EQUAL,NAME,NEWLINE,
DEDENT,DEF,NAME,LPAR,NAME,COMMA,NAME,RPAR,COLON,NEWLINE,
INDENT
```

# Experiment on GST

220 input files from Python 3.2 standard Library.

588,962 tokens.

in 1m43.2s

Cost 644MiB memory

# Works in this week

- Continue to read papers
- Add c lexer support and other language support
- idea of using n-gram to figure out "interesting" clones