Report of Progress

by Jiachen Yang

Jiachen Yang[1]

[1]Research Student in Osaka University

October 25, 2011

Part Outline

Works in last week

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

Paper reading Roy and Cordy (2007), Rainer et al. (2008), Smith and Horwitz (2009).

A. Rainer, P. Lane, J. Malcolm, and S. Scholz. Using n-grams to rapidly characterise the evolution of software code. In *Automated Software Engineering-Workshops, 2008. ASE Workshops 2008. 23rd IEEE/ACM International Conference on*, pages 43–52. IEEE, 2008.

C. Roy and J. Cordy. A survey on software clone detection research. *Queen?s School of Computing TR*, 541:115, 2007.

R. Smith and S. Horwitz. Detecting and measuring similarity in code clones. *Proc. of IWSC*, 9, 2009.

Generalized Suffix Tree

What is GST

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

Generalized Suffix Tree

What is GST

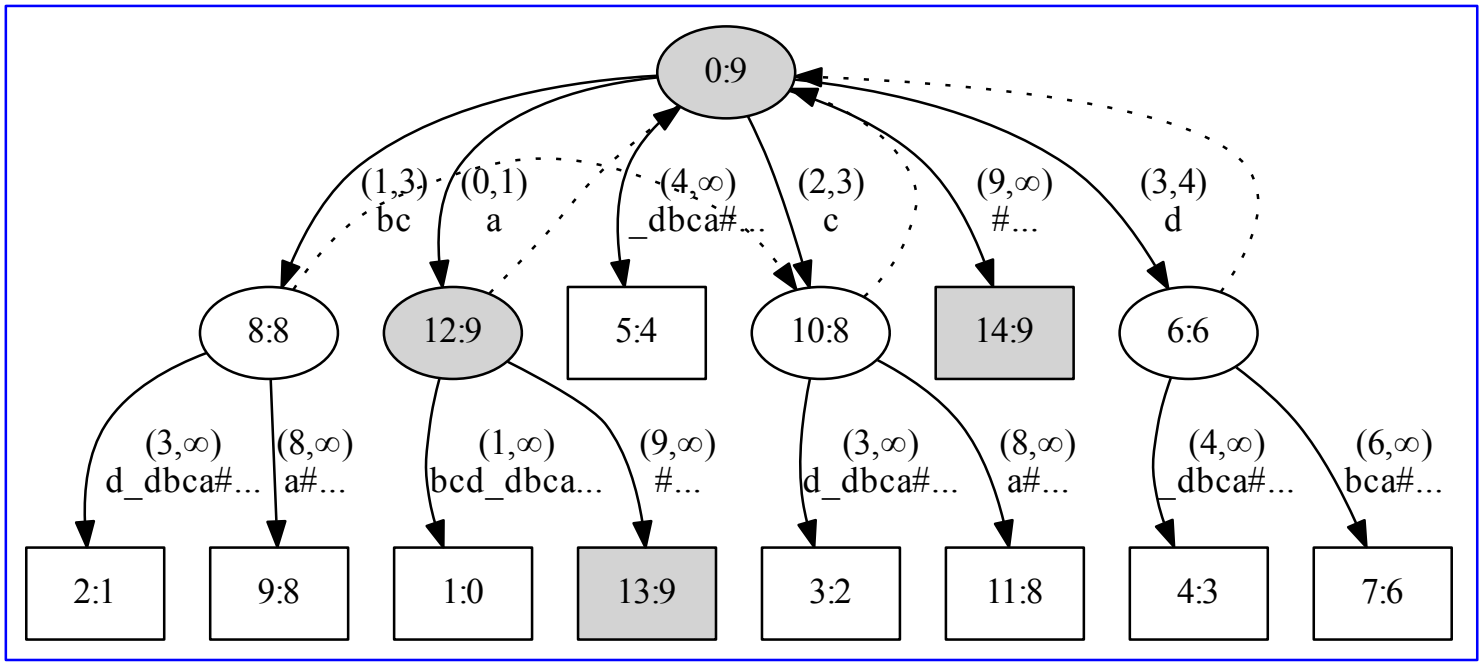
- Generalized Suffix Tree(GST): A Suffix Tree that contains multiple strings.
- Adventage: Find Maximal Common Sequence(MCS) in linear time of sum of length of strings.
- Disadvantage: All strings are stored in memory.

Building GST:

1. Append each string with a unique terminal mark($)
2. Concat strings as a long string.
   STRING1$\$_1$STRING2$\$_2$STRING3$\$_3$...
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