**Frequency Analysis of C Program**

Problem Statement:

Take an extensive program written by others in C language, Perform the frequency analysis of characters and reserved words.

* Output Example:

{'#': 1, 'i': 13, 'n': 12, 'c': 2, 'l': 1, 'u': 2, 'd': 4, 'e': 2, ' ': 81, '<': 2, 's': 2, 't': 6, 'o': 2, '.': 1, 'h': 1, '>': 1, '\n': 14, 'm': 1, 'a': 2, '(': 5, ')': 5, '{': 3, ';': 6, 'f': 4, '"': 4, '%': 3, ',': 2, '&': 1, 'r': 4, '=': 3, '0': 3, '+': 2, '2': 1, 'p': 1, '\\': 1, '}': 3}

{'for': 1, 'while': 0, 'do': 0, 'if': 1, 'else': 0, 'switch': 0, 'int': 3, 'float': 0, 'char': 0, 'printf': 1, 'scanf': 1, 'include': 0, 'return': 1}

Algorithm:

IMPORT REGULAR EXPRESSION MODULE

DEFINE FUNCTION freq(lines):

counter={}

FOR line IN lines:

FOR char IN line:

if(char IN counter):

counter[char]=counter[char]+1;

ELSE:

counter[char]=1;

OUTPUT(counter)

RETURN

DEFINE FUNCTION freq\_word(lines):

counter={"for":0,"while" : 0,"do":0,"if": 0,"else":0,"switch":0,"int":0,"float":0,"char":0,"printf":0,"scanf":0,"include" : 0,"return":0}

FOR line IN lines:

SPLIT THE LINE WHEN [{(\s\d)}] APPEARS

FOR word IN line:

if(word IN counter):

counter[word]=counter[word]+1;

OUTPUT(counter);

RETURN

IF \_\_name\_\_ EQUALS "\_\_main\_\_":

C\_file=Enter C File name

SET main\_file TO open(C\_file, 'r+')

main\_file\_lines=main\_file.readlines();

freq(main\_file\_lines)

freq\_word(main\_file\_lines)

Proposed Python Code:

/\* ------- main.py ------- \*/

import re

def freq(lines):

counter={}

for line in lines:

for char in line:

if(char in counter):

counter[char]=counter[char]+1;

else:

counter[char]=1;

print(counter)

return

def freq\_word(lines):

counter={"for":0,"while" : 0,"do":0,"if": 0,"else":0,"switch":0,"int":0,"float":0,"char":0,"printf":0,"scanf":0,"include" : 0,"return":0}

for line in lines:

line=re.split(r"[{(\s\d)}]", line)

for word in line:

if(word in counter):

counter[word]=counter[word]+1;

print(counter);

return

if \_\_name\_\_ == "\_\_main\_\_":

C\_file='sample.c'

main\_file = open(C\_file, 'r+')

main\_file\_lines=main\_file.readlines();

freq(main\_file\_lines)

freq\_word(main\_file\_lines)

/\* ---------------------- \*/

Conclusion:

The proposed algorithm has a runtime of O(m\*n), where m is the number of lines & n is the length of the string.

Limitations and assumptions for this algorithm include:

1.The python and C file should be in same folder.

2.The reserved words, which are calculated are “for”, ”while”, ”do”, “if”,

“else”, ”switch”, “int”, “float” , “char” ,”printf”, “scanf”, “include” ,”return”.