Name: William A. Brannon

Assignment: Lab Assignment Report #12

Date Due: Monday, May, 06, 2019

Class: Fundamentals of Programming III Section #1

**Program Output**



**Lab13.cpp**

// Lab 13: Suffix Tree Construction

// By: William Brannon on 05/03/2019

#include "stdafx.h"

#include <iostream>

#include <vector>

#include <string>

using namespace std;

class Node {

public:

string label;

vector<Node> children;

int originalIndex;

vector<string> Family;

Node() {

label = "%"; //% indicates the root, $ indicates the end of a path that isnt a valid suffix;

originalIndex = -1;

}

void family(string s, Node\* root) {

if (label == "%") {

Family.clear();

}

s += label;

for (Node n : children) {

n.family(s, root);

}

if (children.size() == 0) {

size\_t found = s.find("$");

if (found == string::npos) {

root->Family.push\_back(s);

}

}

}

};

class SuffixTree {

public:

Node root;

SuffixTree() {

}

void insertString(string s, int a) {

//a is the original index of the substring that is being passed in

Node\* searcher = &root;

Node\* tempPtr = &root;

string temp = s;

for (unsigned int i = 0; i < s.length(); i++) {

bool found = false;

for (unsigned int i = 0; i < searcher->children.size(); i++) {

string compare(1, temp.at(0));

if (searcher->children.at(i).label == compare) {

found = true;

temp = temp.substr(1);

tempPtr = &searcher->children.at(i);

break;

}

}

if (!found) {

Node newNode;

newNode.label = temp.at(0);

newNode.originalIndex = a;

searcher->children.push\_back(newNode);

newNode.label = "$";

newNode.originalIndex = searcher->originalIndex;

searcher->children.push\_back(newNode);

temp = temp.substr(1);

for (unsigned int i = 0; i < searcher->children.size(); i++) {

if (searcher->children.at(i).originalIndex == a) {

tempPtr = &searcher->children.at(i);

break;

}

}

}

searcher = tempPtr;

}

}

void analyzeText(string s) {

for (unsigned int i = 1; i < s.length() + 1; i++) {

string temp = s.substr(s.length() - i);

insertString(temp, s.length() - i);

}

}

void display() {

cout << "Suffix Tree Paths: " << endl;

root.family("", &root);

for (string s : root.Family) {

cout << s << endl;

}

}

};

int main()

{

SuffixTree tester;

tester.analyzeText("CAGTCAGG"); //Brute-force creation of a suffix tree

tester.display(); //Brute-force traversal algorithm to print the tree

system("pause");

return 0;

}