Practical 9

# Student Details

* Name : Tushar Harsora
* Roll : 19BCE509

# Aim

The aim is to implement the Suffix arrays structure. This structure is very useful in string matching and finding in the large source document. Here we implemented the SuffixArrays.

# Code

* SuffixArray.cpp

#include <iostream>

#include <string>

#include <algorithm>

#include <vector>

#include <cassert>

using namespace std;

vector<pair<string, int>> build\_suffix\_array(const string& s) {

auto n = s.size();

vector<pair<string, int>> suffixes;

for (int i = 0; i < n; i++) {

suffixes.emplace\_back(s.substr(i, n - i), i);

}

sort(suffixes.begin(), suffixes.end());

return suffixes;

}

auto get\_occurences(const vector<pair<string, int>>& suffix\_arr, const string& needle) {

int n = suffix\_arr.size(), m = needle.size();

int low = 0, high = n - 1;

int first\_of = INT\_MAX, last\_of = INT\_MIN;

while (low <= high) {

int mid = low + (high - low) / 2;

const string& curr = suffix\_arr[mid].first;

int mx = min((int)curr.size(), m);

bool done = false;

for (int i = 0; i < mx; i++) {

int res = curr[i] - needle[i];

if (res > 0) {

high = mid - 1;

done = true;

break;

}

if (res < 0) {

low = mid + 1;

done = true;

break;

}

}

if (!done) {

if (curr.size() < m) {

low = mid + 1;

}

else if (curr.size() > m) {

first\_of = mid;

high = mid - 1;

}

else {

first\_of = mid;

high = mid - 1;

}

}

}

low = first\_of, high = n - 1;

while (low <= high) {

int mid = low + (high - low) / 2;

const string& curr = suffix\_arr[mid].first;

int mx = min((int)curr.size(), m);

bool done = false;

for (int i = 0; i < mx; i++) {

int res = curr[i] - needle[i];

if (res > 0) {

high = mid - 1;

done = true;

break;

}

if (res < 0) {

low = mid + 1;

done = true;

break;

}

}

if (!done) {

if (curr.size() < m) {

low = mid + 1;

}

else if (curr.size() > m) {

last\_of = mid;

low = mid + 1;

}

else {

last\_of = mid;

low = mid + 1;

}

}

}

return last\_of - first\_of + 1;

}

int main() {

string s = "banana";

auto suffix\_array = build\_suffix\_array(s);

cout << "The \"an\" occurs " << get\_occurences(suffix\_array, "an") << " times in banana" << endl;

cout << "The \"na\" occurs " << get\_occurences(suffix\_array, "na") << " times in banana" << endl;

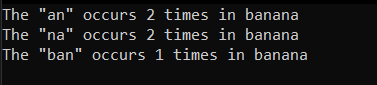
cout << "The \"ban\" occurs " << get\_occurences(suffix\_array, "ban") << " times in banana" << endl; return 0;

}

# Inputs

This Program Doesn’t take any input the output is attached below

# Screenshots of output



# Conclusion

Here we learned about the suffix trees and suffix arrays and how to implement them. What are different algorithms for implementing suffix arrays and how to efficiently implement the.