**COSC 320 – 001**

***Analysis of Algorithms***

2022/2023 Winter Term 2

**Project Topic Number: 2**

**Title of project: Plagiarism Detector**

**Group Lead:**

**Group Members:**

**Divyajot Kaur Dadiala**

**Jusnoor Kaur Sachdeva**

**Robert Barnstead**

**Abstract:**

For this milestone our team will be describing the first algorithm that we will use for our plagiarism detector. We shall be working on the Knuth-Morris-Pratt (KMP) string searching algorithm. We shall be describing the algorithm problem, writing a pseudo code, providing an analysis and listing the unexpected cases and difficulties.

**Problem Formulation**.

For the given problem we have to implement string matching and build a plagiarism detector. For the first milestone we are working on the **Knuth-Morris-Pratt (KMP)** string searching algorithm.

**Problem**: *Given a* ***text*** *of size t and a* ***pattern*** *of size p, we need to write a function that searches and prints the pattern in text.*

The ***input*** will be -

text[]: The text to be searched

pattern[]: The pattern we are looking for

The algorithm should return us with an ***output*** of –

An integer array **I[]** which contains the indexes in **text[]** at which **pattern[]** was found

An integer **m** returning the number of positions found.

**Pseudo-code**

**KMP:**

function search (text, pattern)

t = length of text

p = length of pattern

i = 0

j = 0

//calls the function to create the LPS array

createLPS(pattern)

while i < t

//iterates both i and j if a char match is found

if pattern[j] == text[i]

i++ j++

//prints that plagiarism is found when j = length of pattern or the entire pattern has been found in string

if j == p print "plagiarism found"

//check when the chars don't match

if i < t and pattern[j] != text[i]

//if j isn't 0 then set the value in the LPS array equal to j-1

if j != 0

j = LPS[j-1]

//otherwise iterate i

Else i++

function createLPS(pattern)

patternLength = length of pattern

length = 0

i = 1

LPS[patternLength] //create array of size m

LPS[0] = 0 //LPS at 0 is always zero

while (i < m)

//if pattern is equal to the pattern at length set the value in LPS array equal to length

if pattern[i] == pattern[length]

length++

LPS[i] = length

i++

else

if length != 0

length = LPS[length - 1]

else LPS[i] = length

i++

return LPS

**Algorithm Analysis.**

* Complexity of function **search** as per the pseudo codeis **O(n)**
* Complexity of function **createLPS** is **O(m)**

**Time complexity= O(n+m)** because we are using 2 functions to implement the algorithm.

**Hence the overall complexity is O(m+n),** Linear Time Complexity

**Running Time:**

* **Best Case: O(n)**
* **Worst Case: O(n)**

**Proof of Correctness**

For a given specific pattern P and finite text T , according to KMP it correctly reports each occurrence of P in T.

Keeping the algorithm in mind ,we know that the algorithm finishes in a finite number of steps.There is a prefix function called π which is basically defined on {1,2,…….,m}.

The algorithm does not try to evaluate the function π on integers that are not included in {1,2,…….,m} Now , we will prove that the statement is loon-invariant for the for loop, after each execution of the for loop statement the value of the variable q is set to j that belongs to {0,……..,m-1}……..

To be continued in next milestone. (We have written complete proof but not sure if it is to be added here. )

**Unexpected Cases/Difficulties.**

Combining the work from all the team members : It was difficult to coordinate.  
**SOLUTION** : Planning out a proper schedule so that it is not chaotic at the very last moment.

**Time management**: it was tough to schedule a group meeting keeping in mind the timings given by all the team members so there are tasks assigned to all the members.

**Solution**: Scheduled a zoom meeting so that there is a discussion keeping in mind the comfort level of all the team members

**Task Separation and Responsibilities**

Divyajot Kaur Dadiala: **Analysis of algorithm, Problem formation, Run time.** Created template and allocated tasks and took the responsibility of submission. Compiled the whole document.

Jusnoor Kaur Sachdeva: **Unexpected Cases/Difficulties, Proof of Correctness**

Robert Barnstead: **Pseudocode**