**TUGAS MODUL PRAKTIKUM 7**

**STABLE MATCHING PROBLEM**

****

**Oleh:**

**Refa Annisatul Ilma 140810170060**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

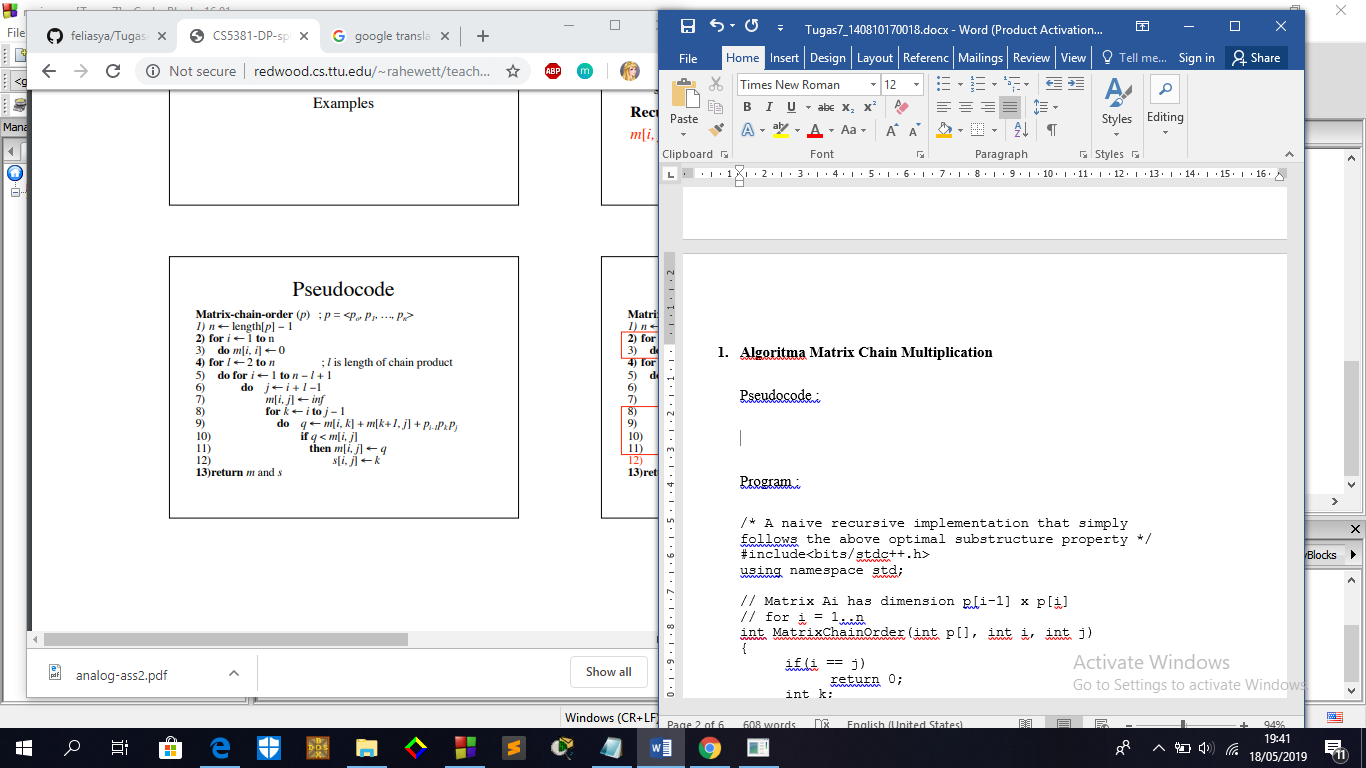
**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS PADJADJARAN**

**TAHUN AJARAN 2018/2019**

1. **Algoritma Matrix Chain Multiplication**

Pseudocode :



Program :

/\* A naive recursive implementation that simply

follows the above optimal substructure property \*/

#include<bits/stdc++.h>

using namespace std;

// Matrix Ai has dimension p[i-1] x p[i]

// for i = 1..n

int MatrixChainOrder(int p[], int i, int j)

{

if(i == j)

return 0;

int k;

int min = INT\_MAX;

int count;

// place parenthesis at different places

// between first and last matrix, recursively

// calculate count of multiplications for

// each parenthesis placement and return the

// minimum count

for (k = i; k < j; k++)

{

count = MatrixChainOrder(p, i, k) +

MatrixChainOrder(p, k + 1, j) +

p[i - 1] \* p[k] \* p[j];

if (count < min)

min = count;

}

// Return minimum count

return min;

}

// Driver Code

int main()

{

int arr[] = {10, 5, 3, 10, 2, 9};

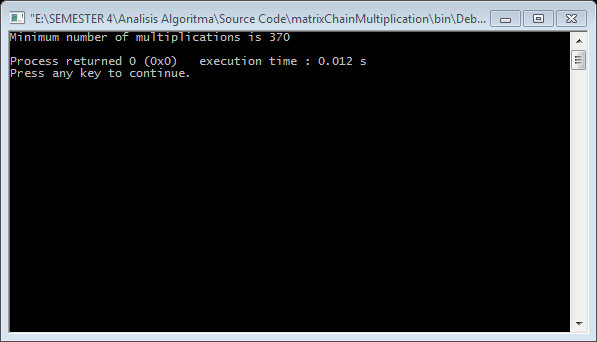
int n = sizeof(arr)/sizeof(arr[0]);

cout << "Minimum number of multiplications is "

<< MatrixChainOrder(arr, 1, n - 1);

}

Screenshot :



1. **Algoritma Longest Common Subsequence**

Pseudocode :

Mengambil X = <x\_1, ... x\_m> dan Y = <y\_1, ... y\_n> sebagai input. Menyimpan c [i, j] ke dalam tabel c [0..m, 0..n] dalam urutan baris-utama. Array b [i, j] menunjuk ke entri tabel untuk solusi subproblem optimal saat menghitung c [i, j].

LCS-Length(X, Y)

m <- length[X]

n <- length[Y]

for i <- 1 to m

c[i,0] <- 0

for j <- 1 to n

c[0,j] <- 0

for i <- 1 to m

for j <- 1 to n

if (x\_i == y\_j) {

c[i,j] <- c[i-1,j-1] + 1

b[i,j] <- NW

}

else if (c[i-1,j] >= c[i,j-1]) {

c[i,j] <- c[i-1,j]

b[i,j] <- N

}

else {

c[i,j] <- c[i,j-1]

b[i,j] <- W

}

Program :

/\* A Naive recursive implementation of LCS problem \*/

#include <bits/stdc++.h>

int max(int a, int b);

/\* Returns length of LCS for X[0..m-1], Y[0..n-1] \*/

int lcs(char\* X, char\* Y, int m, int n)

{

if (m == 0 || n == 0)

return 0;

if (X[m - 1] == Y[n - 1])

return 1 + lcs(X, Y, m - 1, n - 1);

else

return max(lcs(X, Y, m, n - 1), lcs(X, Y, m - 1, n));

}

/\* Utility function to get max of 2 integers \*/

int max(int a, int b)

{

return (a > b) ? a : b;

}

/\* Driver program to test above function \*/

int main()

{

char X[] = "RENDANG";

char Y[] = "NANGKA";

int m = strlen(X);

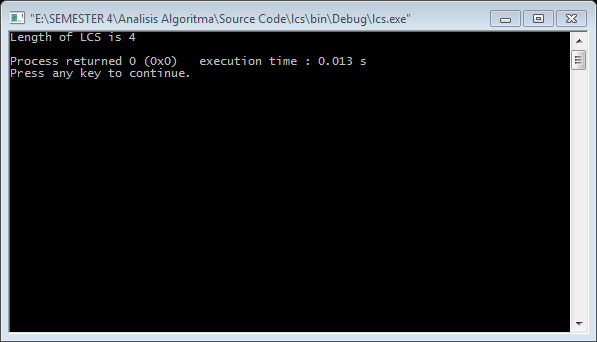
int n = strlen(Y);

printf("Length of LCS is %d\n", lcs(X, Y, m, n));

return 0;

}

Screenshot :



1. **Perbandingan perhitungan manual dengan perhitungan menggunakan program dari soal no 2 worksheet 5**

Perhitungan manual

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | j | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| i |  | Yj | r | e | n | d | a | n | g |
| 0 | Xi | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | n | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 2 | a | 0 | 0 | 0 | 1 | 1 | 2 | 2 | 2 |
| 3 | n | 0 | 0 | 0 | 1 | 1 | 2 | 3 | 3 |
| 4 | g | 0 | 0 | 0 | 1 | 1 | 2 | 3 | 4 |
| 5 | k | 0 | 0 | 0 | 1 | 1 | 2 | 3 | 4 |
| 6 | a | 0 | 0 | 0 | 1 | 1 | 2 | 3 | 4 |
|  |  |  |  |  | n |  | a | n | g |

Perhitungan dengan program :

X : r e n d a n g

Y : n a n g k a

Huruf yang sama ada 4 yaitu n a n g