**TUGAS PRAKTIKUM 7**

**ANALISIS ALGORITMA**



**Muhammad Fahmi Alwan 140810170052**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS PADJADJARAN**

**TAHUN AJARAN 2018/2019**

1. Cari algoritma Matrix Chain Multiplication Problem, dan buat programnya
2. Cari algoritma Longest Common Subsequence, dan buat programnya, bandingkan dengan perhitungan manual di kertas

Jawab:

1. Algoritma

matrix-multiply(a,b)

//ncolumns = jumlah kolom

//nrows = jumlah baris

if ncolumns[A] =/= nrows[B]

then error "dimensi tidak kompetibel"

else for i<-1 to nrows[A]

do for j<-1 to ncolumns[B]

do C[i,j] <-0

for k<-1 to ncolumns[A]

do C[i,j] <- C[i,j] + A[i,k].B[k,j]

return C

Source Code

#include<stdio.h>

#include<limits.h>

// Matrix Ai has dimension p[i-1] x p[i] for i = 1..n

int MatrixChainOrder(int p[], int n)

{

/\* For simplicity of the program, one extra row and one

extra column are allocated in m[][]. 0th row and 0th

column of m[][] are not used \*/

int m[n][n];

int i, j, k, L, q;

/\* m[i,j] = Minimum number of scalar multiplications needed

to compute the matrix A[i]A[i+1]...A[j] = A[i..j] where

dimension of A[i] is p[i-1] x p[i] \*/

// cost is zero when multiplying one matrix.

for (i=1; i<n; i++)

m[i][i] = 0;

// L is chain length.

for (L=2; L<n; L++)

{

for (i=1; i<n-L+1; i++)

{

j = i+L-1;

m[i][j] = INT\_MAX;

for (k=i; k<=j-1; k++)

{

// q = cost/scalar multiplications

q = m[i][k] + m[k+1][j] + p[i-1]\*p[k]\*p[j];

if (q < m[i][j])

m[i][j] = q;

}

}

}

return m[1][n-1];

}

int main()

{

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

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

printf("Minimum number of multiplications is %d ",

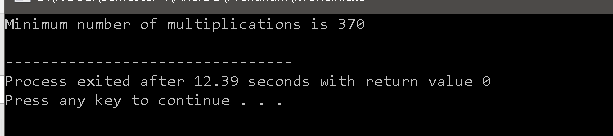
MatrixChainOrder(arr, size));

getchar();

return 0;

}

Screenshot



1. Algoritma

lcs-length(x,y)

m<-length[x]

n<-length[y]

for i<-1 to m do c[i,0]<-0

for j<-0 to n do c[0,j]<-0

for i<-1 to m

do for j<-1 to n

do if xi==yj

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

b[i,j]<-"panahkiriatas"

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

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

b[i,j]<-"panahatas"

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

b[i,j]<-"panahkiri"

return c and b

Source Code

/\* Dynamic Programming implementation of LCS problem \*/

#include<iostream>

#include<cstring>

#include<cstdlib>

using namespace std;

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

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

{

int L[m+1][n+1];

/\* Following steps build L[m+1][n+1] in bottom up fashion. Note

that L[i][j] contains length of LCS of X[0..i-1] and Y[0..j-1] \*/

for (int i=0; i<=m; i++)

{

for (int j=0; j<=n; j++)

{

if (i == 0 || j == 0)

L[i][j] = 0;

else if (X[i-1] == Y[j-1])

L[i][j] = L[i-1][j-1] + 1;

else

L[i][j] = max(L[i-1][j], L[i][j-1]);

}

}

// Following code is used to print LCS

int index = L[m][n];

// Create a character array to store the lcs string

char lcs[index+1];

lcs[index] = '.'; // Set the terminating character

// Start from the right-most-bottom-most corner and

// one by one store characters in lcs[]

int i = m, j = n;

while (i > 0 && j > 0)

{

// If current character in X[] and Y are same, then

// current character is part of LCS

if (X[i-1] == Y[j-1])

{

lcs[index-1] = X[i-1]; // Put current character in result

i--; j--; index--; // reduce values of i, j and index

}

// If not same, then find the larger of two and

// go in the direction of larger value

else if (L[i-1][j] > L[i][j-1])

i--;

else

j--;

}

// Print the lcs

cout << "LCS of " << X << " and " << Y << " is " << lcs;

}

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

int main()

{

char X[] = "RENDANG";

char Y[] = "NANGKA";

int m = strlen(X);

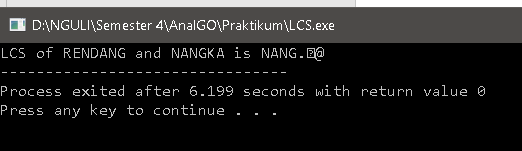
int n = strlen(Y);

lcs(X, Y, m, n);

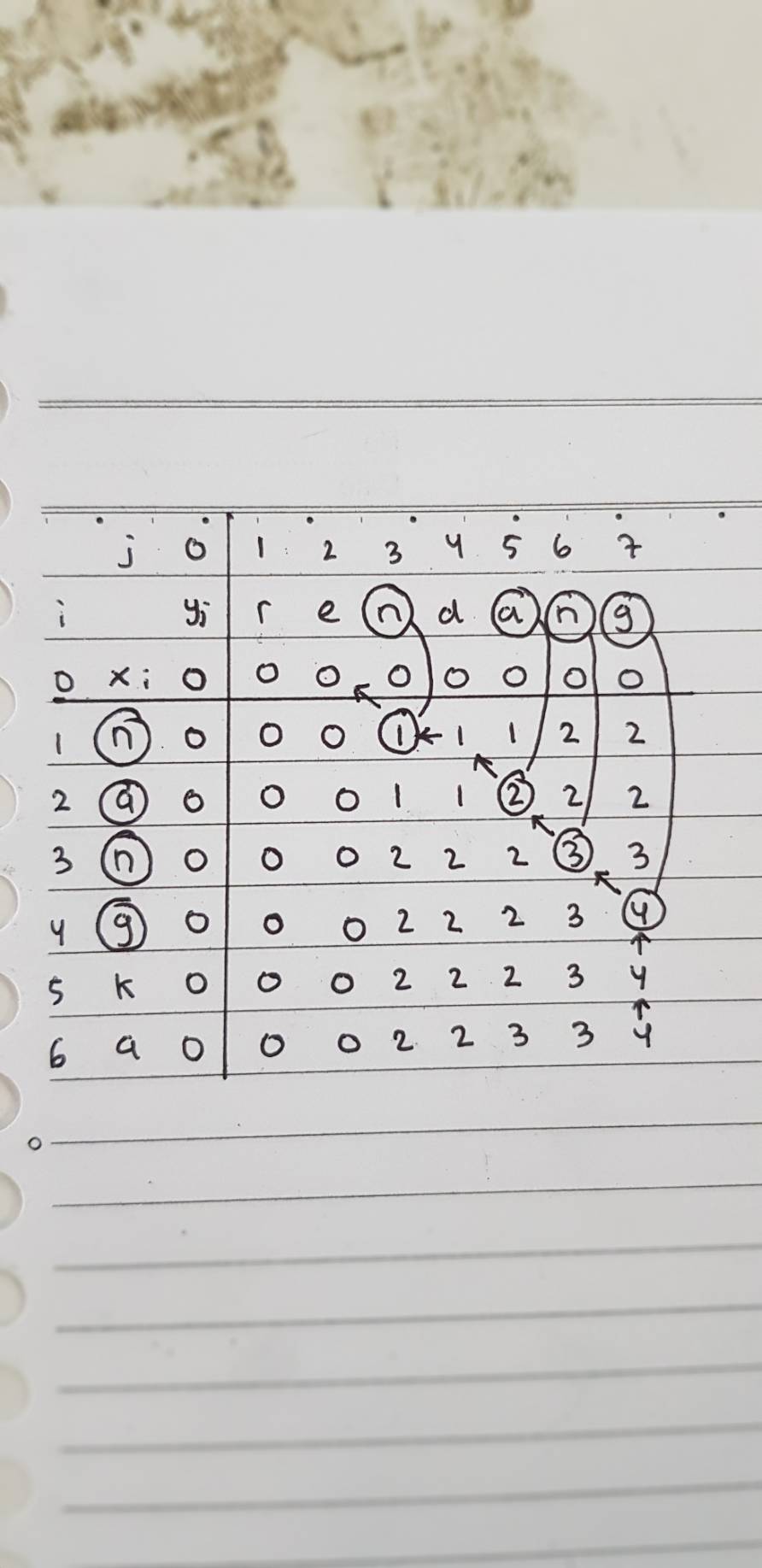
return 0;

}

Screenshot



Perhitungan manual

****

Hasil: Sama menghasilkan Panjang 4 karakter yaitu N A N G