TUGAS PRAKTIKUM 7

ANALISIS ALGORITMA

ILHAM MUHARAM

140810170046

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

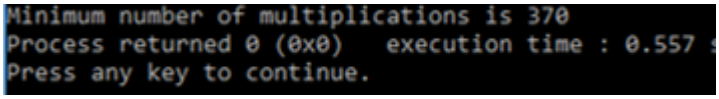
```
matrix-multiply(a,b)
//ncolumns = number of collumn
//nrows = number of rows
if ncolumns[A] =/= nrows[B]
  then error "incompatible dimensions"
  else for i<-1 to nrows[A]
         do for j<-1 to ncolumns[B]
              do C[i,j] <-0
                for k<-1 to ncolumns[A]
                  do C[i,j] <- C[i,j] + A[i,k].B[k,j]
    return C


//nkolom = panjang kolom
//nbaris = panjang baris
```

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[] = {1, 2, 3, 4};
	int size = sizeof(arr)/sizeof(arr[0]);

	printf("Minimum number of multiplications is %d ",
						MatrixChainOrder(arr, size));

	getchar();
	return 0;
}
```

Screenshot



2. Algoritma

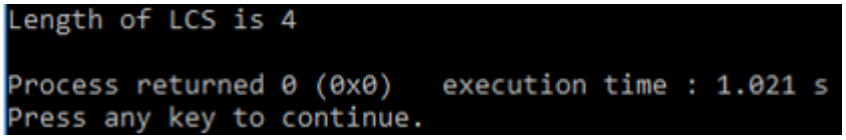
```
lcs-length(x,y)
m<-length[x]
n<-length[y]
for i<-1 to m do c[i,0]<-0
for j<-0 to n do c[0,j]<-0
for i<-1 to m
  do for j<-1 to n
       do if xi==yj
            then c[i,j]<-c[i-1,j-1]+1
                 b[i,j]<-"panahkiriatas"
            else if c[i-1,j]>=c[i,j-1]
                   then c[i,j]<-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

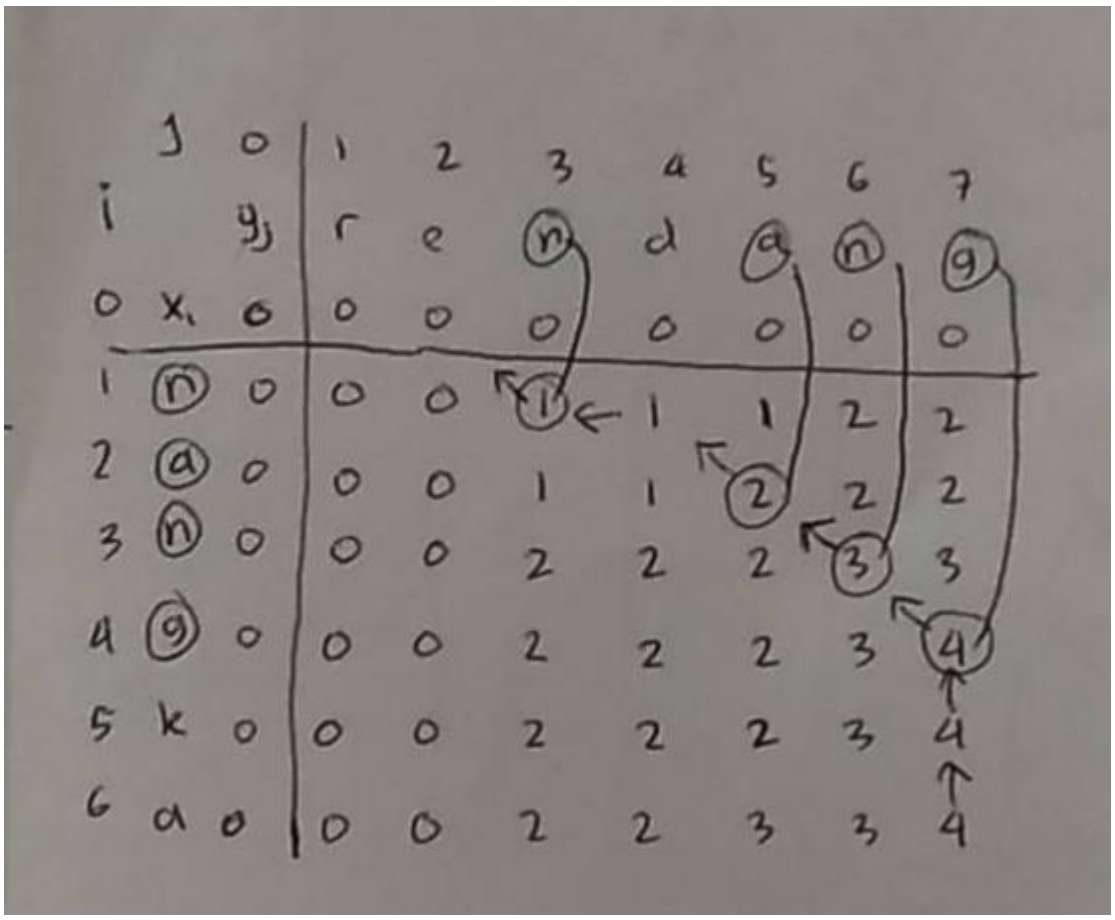
```
/* Dynamic Programming C/C++ implementation of LCS
problem */
#include<bits/stdc++.h>

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 )
{
int L[m+1][n+1];
int i, j;

/* 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 (i=0; i<=m; i++)
{
     for (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]);
        }
}

/* L[m][n] contains length of LCS for X[0..n-1] and
Y[0..m-1] */
return L[m][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", lcs( X, Y, m, n ) );

return 0;
}
```

Screenshot



Perhitungan manual



			1	2	3	4	5	6	7
	j	o							
i		g	r	e	n	d	a	n	g
0	x	o	0	0	0	0	0	0	0
1	n	o	0	0	1	1	1	2	2
2	a	o	0	0	1	1	2	2	2
3	n	o	0	0	2	2	2	3	3
4	g	o	0	0	2	2	2	3	4
5	k	o	0	0	2	2	2	3	4
6	a	o	0	0	2	2	3	3	4

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