|  |  |
| --- | --- |
|  | |
| Assignment No: 08     1. A program of ‘Longest Common Subsequence’. 2. A program of ‘Matrix Chain Multiplication’. | |
| Date of Performance: 16/09/2019  Date of Submission: 22/09/2019 | Student ID: 17-02-04-058  Group: B1 |

**No.1:**

#include<bits/stdc++.h>

int max(int a, int b);

int i,j,m,n,c[20][20];

char x[20],y[20],b[20][20];

int max(int a, int b)

{

return (a > b)? a : b;

}

void print(int i,int j)

{

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

return;

if(b[i][j]=='c')

{

print(i-1,j-1);

printf("%c",x[i-1]);

}

else if(b[i][j]=='u')

print(i-1,j);

else

print(i,j-1);

}

void lcs()

{

m=strlen(x);

n=strlen(y);

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

c[i][0]=0;

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

c[0][i]=0;

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

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

{

if(x[i-1]==y[j-1])

{

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

b[i][j]='c';

}

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

{

c[i][j]=c[i-1][j];

b[i][j]='u';

}

else

{

c[i][j]=c[i][j-1];

b[i][j]='l';

}

}

}

void length()

{

int XLen = strlen(x);

int YLen = strlen(y);

int L[XLen + 1][YLen + 1];

int r, c, i;

for(r = 0; r <= XLen; r++)

{

for(c = 0; c <= YLen; c++)

{

if(r == 0 || c == 0)

{

L[r][c] = 0;

}

else if(x[r - 1] == y[c - 1])

{

L[r][c] = L[r - 1][c - 1] + 1;

}

else

{

L[r][c] = max(L[r - 1][c], L[r][c - 1]);

}

}

}

printf("\nLength of the LCS: %d\n", L[XLen][YLen]);

}

int main()

{

printf("Enter 1st sequence:");

scanf("%s",x);

printf("Enter 2nd sequence:");

scanf("%s",y);

printf("\nThe Longest Common Subsequence is : ");

lcs();

print(m,n);

length();

return 0;

}

**No.2:**

#include<bits/stdc++.h>

#define INFY 999999999

long int m[20][20];

int s[20][20];

int p[20],i,j,n;

void print\_optimal(int i,int j)

{

if (i == j)

printf(" A%d ",i);

else

{

printf("( ");

print\_optimal(i, s[i][j]);

print\_optimal(s[i][j] + 1, j);

printf(" )");

}

}

void matmultiply(void)

{

long int q;

int k;

for(i=n; i>0; i--)

{

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

{

if(i==j)

m[i][j]=0;

else

{

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

{

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

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

{

m[i][j]=q;

s[i][j]=k;

}

}

}

}

}

}

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

{

if(i == j)

return 0;

int k;

int min = INT\_MAX;

int 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 min;

}

int main()

{

int k;

printf("Enter the no. of elements: ");

scanf("%d",&n);

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

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

{

m[i][i]=0;

m[i][j]=INFY;

s[i][j]=0;

}

printf("\nEnter the dimensions: \n");

for(k=0; k<=n; k++)

{

printf("P%d: ",k);

scanf("%d",&p[k]);

}

matmultiply();

printf("\nCost Matrix M:\n");

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

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

printf("m[%d][%d]: %ld\n",i,j,m[i][j]);

i=1,j=n;

printf("\nMultiplication Sequence : ");

print\_optimal(i,j);

printf("\nMinimum number of multiplications is : %d ",

MatrixChainOrder(p, 1, n));

}