

알고리즘 과제

Practice.10

학번 : 201402432

이름 : 조디모데

10-1 Matrix-chain multiplication

- 알고리즘 설명

$M[i][j]$ 는 matrix i 에서 matrix j 까지 곱셈 결과 값이 가장 작은 값을 가지는 변수

$M[i][j] = \text{Math.min}(M[i][j], M[i][k-1] + M[k][j] + d(i-1)*d(k)*d(j))$ 이 식을 이용하여 구한다.

d 는 행렬의 행 또는 열 값이고, i 행렬은 $d(i-1) * d(i)$ 로 구성되어 있다.

이때 $i \leq k \leq j$ 범위 내에서 가능한 모든 k 의 값을 넣어보면서 $M[i][j]$ 를 구한다.

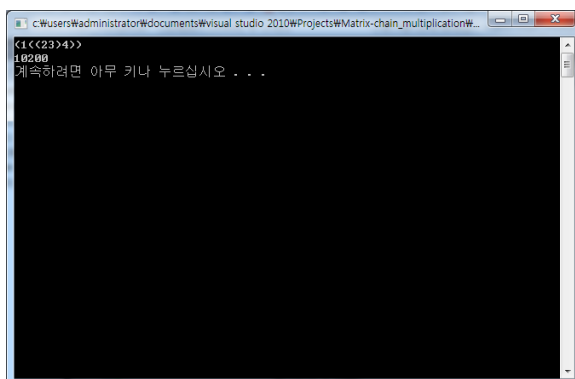
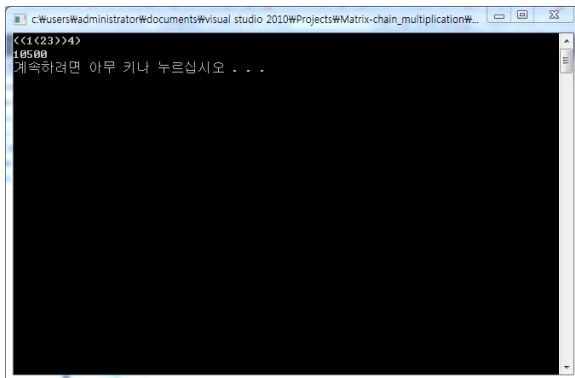
- 컴파일 방법

sample폴더를 바탕화면에 넣는다

"C:\Users\WWAdministrator\Desktop\WWsample\WWsample_mat1.txt"

"C:\Users\WWAdministrator\Desktop\WWsample\WWsample_mat2.txt"

- 프로그램 결과값



Code (.C)

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>

void getArr(int* num){
    FILE *fps ;
    int temp = 0 ;
    int temps[8] = {0} ;

    // input의 숫자들을 배열에 num에 저장하라는 부분이
    fps = fopen("C:\\Users\\Administrator\\Desktop\\sample_mat2.txt", "rt");
    fscanf(fps, "%d", &temp) ;
    num[7] = temp ;

    fscanf(fps, "%d %d %d %d %d %d %d", &num[0], &num[1], &num[2], &num[3], &num[4],
    &num[5], &num[6] ) ;

    fclose(fps) ;
}

void PRINT_OPIMAL_PARENS( int(*s)[7], int i, int j){
    if(i==j)
        printf("%d", i) ;
    else{
        printf("(") ;
        PRINT_OPIMAL_PARENS(s, i, s[i][j]) ;
        PRINT_OPIMAL_PARENS(s, s[i][j]+1, j) ;
        printf(")") ;
    }
}

/*
int find_min(int *M[7], int *p, int i, int j, int k){
    int temp = (M[i][k-1] + M[k][j] + p[i-1]*p[k]*p[j]) ;

    if(M[i][j] > temp){
        return temp ;
    }
    return M[i][j] ;
}
*/

int matrix_multiplication( int p[], int n){
    int L, i, k, j, temp;
    int M[7][7];
    int S[7][7] ;

    for(i=0; i<7 ; i++){
        for(j=0; j<7 ; j++){
            M[i][j] = 0 ;
            S[i][j] = 0 ;
        }
    }
}
```

```

        }
    }

    for(L=2; L<7 ; L++){
        for(i=1; i<7-L+1; i++){
            j = i+L-1 ;
            M[i][j] = 999999 ;
            for(k=i; k<=j-1; k++){
                temp = M[i][k] + M[k+1][j] + p[i-1] * p[k] * p[j];
                if(temp < M[i][j]){
                    M[i][j] = temp;
                    S[i][j] = k ;
                }
            }
        }
    }
}

    PRINT_OPIMAL_PARENS(S,1,4) ;
    printf("Wn") ;
    return M[1][4] ;
}

int main(){
    int *num ;
    int n ;
    num = (int*) malloc(sizeof(int)*9) ;

    getArr(num) ; // Get Input Data
    n = num[7] ;
    num[7] = 0 ;
    printf("%dWn", matrix_multiplication(num, n));
    system("pause") ;

    return 0;
}

```

10-2 Longest Common Subsequence

- 알고리즘 설명

LCS 알고리즘은 두 문자열이 주어졌을 때, 두 열에서 공통적으로 들어있는 부분열(다른 문자열에서 몇몇 문자가 빠져있어도 순서는 바뀌지 않은 문자열) 중에서 가장 긴 열의 길이를 찾아내는 알고리즘이다.

앞의 문자열을 기준으로 한 문자씩 비교해가며 $S1 \times S2$ 크기의 매트릭스를 채운다.

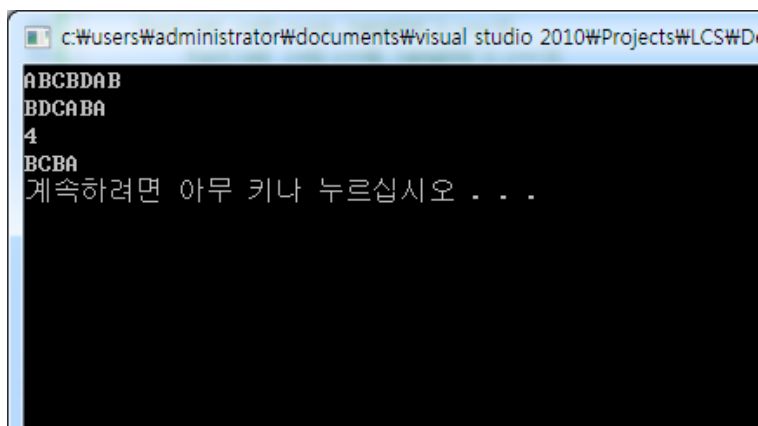
- 컴파일 방법

sample폴더를 바탕화면에 넣는다

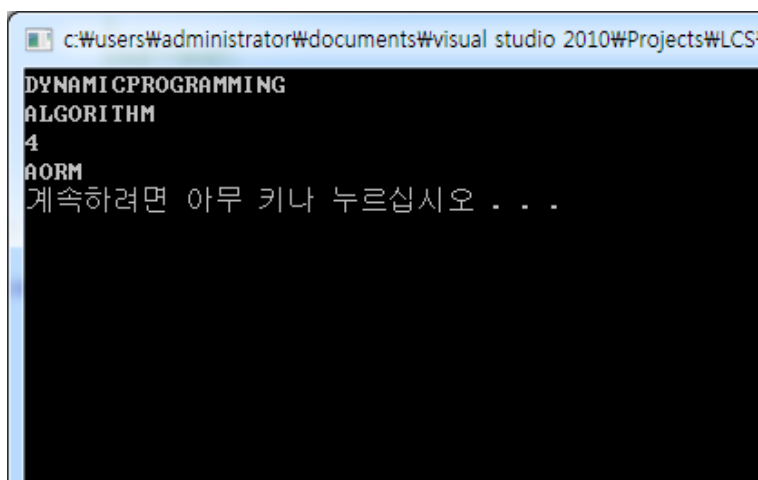
"C:\Users\Administrator\Desktop\sample\sample_lcs1.txt"

"C:\Users\Administrator\Desktop\sample\sample_lcs2.txt"

- 프로그램 결과



```
c:\Users\Administrator\documents\visual studio 2010\Projects\LCS\De
ABCBDAB
BDCABA
4
BCBA
계속하려면 아무 키나 누르십시오 . . .
```



```
c:\Users\Administrator\documents\visual studio 2010\Projects\LCS
DYNAMICPROGRAMMING
ALGORITHM
4
AORM
계속하려면 아무 키나 누르십시오 . . .
```

Code (.cpp)

```
#include <iostream>
#include <stdio.h>
#include <string>
#include <fstream>

using namespace std;

int max_num( int a, int b){
    if(a>=b) return a;
    else return b;
}

enum{
    LEFT = -1,
    UP = 1,
    CROSS = 0
};

void LCS_LENGTH(string A, string B){

    int **arr = new int*[A.length()+1]();
    int **s_arr = new int*[A.length()+1]();

    for(int i=0; i<=A.length(); i++){
        arr[i] = new int[B.length()+1]();
        s_arr[i] = new int[B.length()+1]();
    }

    int result = 0;

    for(int i=1; i<=A.length(); i++){
        for(int j=1; j<=B.length(); j++){
            if(A[i-1]==B[j-1]){
                arr[i][j] = arr[i-1][j-1]+1;
                s_arr[i][j] = CROSS;
            }
            else{
                arr[i][j] = max_num(arr[i-1][j], arr[i][j-1]);
                if(arr[i][j]==arr[i-1][j]) s_arr[i][j] = LEFT;
                else s_arr[i][j] = UP;
            }
        }
    }

    int k = A.length();
    int l = B.length();
    string answer="";
    while(arr[k][l]!=0){
        switch(s_arr[k][l]){
            case UP:
            {
                l--;
                break;
            }
        }
    }
```

```

        case CROSS:
        {
            answer = A[k-1]+answer;
            k--; l--;
            break;
        }
        case LEFT:
        {
            k--;
            break;
        }
    }
}

/*
for( int i=0; i<=A.length(); i++){
for( int j=0; j<=B.length(); j++){
    cout<<arr[i][j]<<" ";
}
cout<<endl;
}
*/
printf("%d\n", arr[A.length()][B.length()]) ;
cout<<answer<<endl ;

}

int main(){

    string A, B ;
    char temp[30] ;
    ifstream ifile ;

    ifile.open("C:\\Users\\Administrator\\Desktop\\sample\\sample_lcs2.txt");
    ifile.getline(temp, sizeof(temp)) ;
    ifile.getline(temp, sizeof(temp)) ;
    A = temp ;
    ifile.getline(temp, sizeof(temp)) ;
    ifile.getline(temp, sizeof(temp)) ;
    B = temp ;
    ifile.close(); // 파일 닫기

    cout << A << endl;
    cout << B << endl;

    LCS_LENGTH(A,B) ;

    system("pause") ;

}

```