<insertion sort>

#include <stdio.h>

FILE \*inF, \*outF;

void main()

{

int arr[10] = {0};

inF = fopen("input.txt","r");

int b = 0;

while (feof(inF) == 0)

{

fscanf(inF, "%d", &arr[b]);

b++;

}

fclose(inF);

for (int i = 1; i < 10;i++)

{

int temp = arr[i];

int aux = i - 1;

while (aux>=0&&arr[aux]>temp)

{

arr[aux + 1] = arr[aux];

aux--;

}

arr[aux + 1] = temp;

}

for (int i = 0; i < 10; i++)

{

printf("%d ",arr[i]);

}

outF = fopen("output.txt","w");

fclose(outF);

}

<bubble sort>

#include <stdio.h>

FILE \*inF, \*outF;

void main()

{

int arr[10] = {0};

inF = fopen("input.txt","r");

int b = 0;

while (feof(inF) == 0)

{

fscanf(inF, "%d", &arr[b]);

b++;

}

fclose(inF);

for (int i = 0; i < 9; i++)

{

for (int j = 0; j < 9 - i; j++)

{

if (arr[j] > arr[j + 1])

{

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

for (int i = 0; i < 10; i++)

{

printf("%d ",arr[i]);

}

outF = fopen("output.txt","w");

fclose(outF);

}

<Longest Common Subsequence>

int LCS[20][20] = {0};

for (int i = 1; i <= strlen(str[1]); i++)

{

for (int j = 1; j <= strlen(str[0]);j++)

{

if (str[1][i - 1] == str[0][j - 1])

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

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

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

else

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

}

}

for (int i = 0; i <= strlen(str[1]); i++) //print

{

for (int j = 0; j <= strlen(str[0]); j++)

{

printf("%d ",LCS[i][j]);

}

printf("\n");

}

for (int i = 1; i <= strlen(str[1]); i++)

{

for (int j = 1; j <= strlen(str[0]); j++)

{

if (LCS[i][j] == LCS[i - 1][j - 1] + 1 && LCS[i][j] == LCS[i][j - 1] + 1 && LCS[i][j] == LCS[i - 1][j] + 1)

lcsString[stringCount++]=str[1][i - 1];

}

}

lcsString[stringCount] = '\n';

printf("\n%s",lcsString);

**<Matrix chain order>**

int Matrix[6][6] = {0};

int Index[6][6] = {0};

int colNum[6] = {4,3,2,10,6,5};

for (int i = 1; i <= 5; i++)

Matrix[i][i - 1] = 0;

for (int len = 2; len <= 5; len++) //행렬곱 길이마다 Matrix,Index에 값 채워넣기

{

for (int i = 1; i <= 6 - len; i++)

{

int j = i + (len - 1);

Matrix[i][j] = INF;

int minK = 0;

for (int k = i; k <j; k++)//get minimum

{

int cost = Matrix[i][k] + Matrix[k + 1][j] + colNum[i - 1] \* colNum[k] \* colNum[j];

if (Matrix[i][j]>cost)

{

Matrix[i][j] = cost;

minK = k;

}

}

Index[i][j] = minK;

}

}

for (int i = 0; i < 6; i++)

{

for (int j = 0; j < 6; j++)

{

printf("%d ",Matrix[i][j]);

}

printf("\n");

}

for (int i = 0; i < 6; i++)

{

for (int j = 0; j < 6; j++)

{

printf("%d ", Index[i][j]);

}

printf("\n");

}

<Prim’s algorithm>

int visitedNode[5] = {1};

int Node[5] = {0};

int visitedCount = 1;

int route[5] = {0};

int totalDistance = 0;

for (int i = 1; i < 5; i++)

Node[i] = 1;

while (visitedCount != 5)

{

int minDistance = 100;

int minI = 0, minJ = 0;

for (int i = 0; i < 5; i++) //get minimum route

{

if (visitedNode[i])

{

for (int j = 0; j < 5; j++)

{

if (Node[j] != 0)

{

if (distance[i][j]!=0&&minDistance>distance[i][j])

{

minDistance = distance[i][j];

minI = i;

minJ = j;

}

}

}

}

}

route[visitedCount] = minJ;

totalDistance += minDistance;

Node[minJ] = 0;

visitedNode[minJ] = 1;

visitedCount++;

}

for (int i = 0; i < 5; i++)

{

printf("%d ",route[i]);

}

0 1 1 3 0

1 0 2 1 0

1 2 0 1 0

3 1 1 0 7

0 0 0 7 0

<Bellman-Ford algorithm>

<kruskal algorithm>