알고리즘 과제

Practice.09

**학번 : 201402432**

**이름 : 조디모데**

**9-1 All-Pairs Shortest Paths and Matrix Multiplication**

* 알고리즘 설명

L’행렬을 INF로 초기화 후 L행렬의 원소 l과 W행렬의 원소 w를 이용해서 최소 비용을 L’에 저장한다. 같은 작업을 모든 행렬에 반복

* 컴파일 방법  
  graph\_sample\_bellman.txt 파일을 바탕화면에 넣는다  
  C:\\Users\\Administrator\\Desktop\\graph\_sample\_bellman.txt
* **Shortest Path**

**1**

**5**

**7**

**-1**

**-8**

**-4**

**2**

**10**

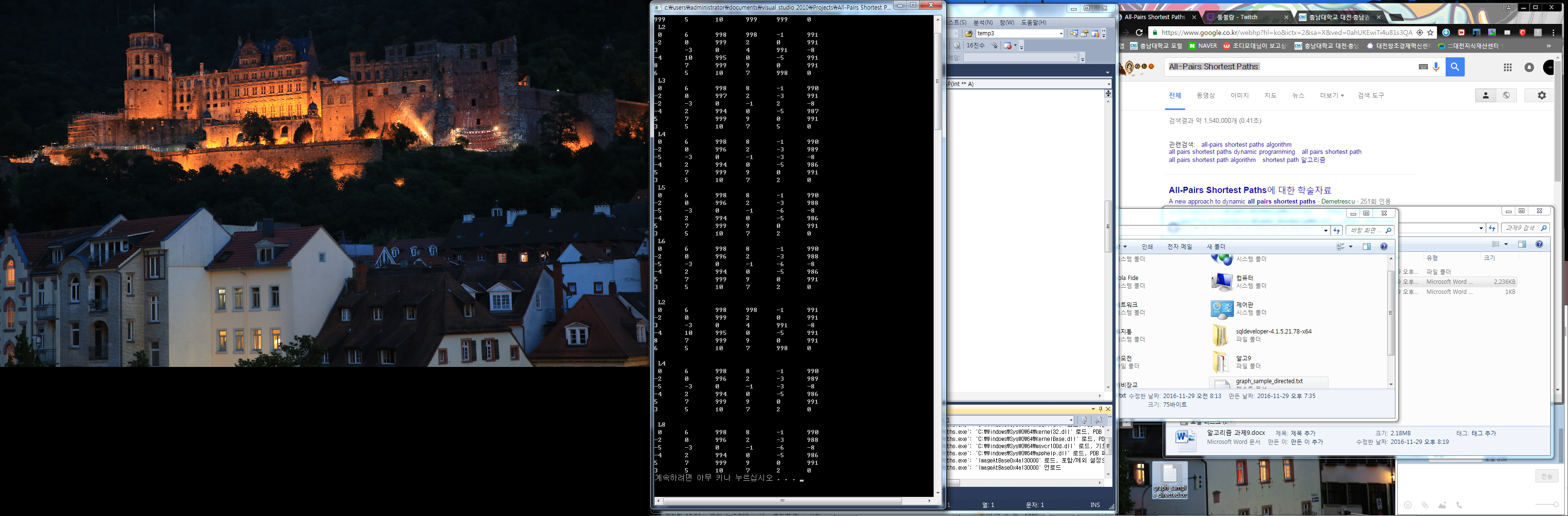
**3**

**2**

**<직접 찾은 값>**

**0 to 1 : 0 – 4 – 1 : 6 (-1+7)  
0 to 2 : 0 – 4 – 1 – 3 – 0 : 길 없음  
0 to 3 : 0 – 4 – 1 – 3 : 8 (-1+7+2)   
0 to 4 : 0 – 4 : -1  
0 to 5 : 0 – 4 – 1 – 3 – 0 : 길 없음**

**<프로그램 결과값>**



Code ( .C )

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#define INF 999 ; // Max, inf

int ver = 0 ;

void insertGraph() ;

int setViaVerSize(){

int ver = 0 ;

FILE \*fps ;

int num1 = 0;

fps = fopen("C:\\Users\\Administrator\\Desktop\\graph\_sample\_directed.txt","rt");

fscanf(fps,"%d",&num1);

ver = num1;

fclose(fps);

return ver ;

}

int \*\*setVia(int \*\*arr){

FILE \*fps ;

int i ;

int num1 = 0 ;

int num2 = 0 ;

int num3 = 0 ;

int num4 = 0 ;

fps = fopen("C:\\Users\\Administrator\\Desktop\\graph\_sample\_directed.txt","rt");

fscanf(fps,"%d",&num1) ;

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

fscanf(fps,"%d %d %d",&num2, &num3, &num4);

insertGraph(arr, num2, num3, num4);

}

fclose(fps);

return arr ;

}

void insertGraph(int \*\*A,int num2,int num3,int num4){

\*(\*(A+num2)+num3) = num4;

}

int \*\*Extend\_Shortest\_Paths(int \*\*L,int \*\*A){

int \*\*L\_new ;

int i,j,k,temp ;

int min = INF ;

L\_new = (int \*\*)malloc (sizeof(int\*)\*ver) ;

for(i=0; i<ver; i++){

\*(L\_new + i) = (int\*)malloc(sizeof(int)\*ver) ;

}

for(i=0 ; i<ver ; i++){

for(j=0 ; j<ver ; j++){

L\_new[i][j] = INF ;

for(k=0 ; k<ver ; k++){

temp = L[i][k] + A[k][j] ;

if(L\_new[i][j] > temp) // 최OO소ùOùO비¬n¬®©­n용¯e¯e 저uu장aa

L\_new[i][j] = temp;

}

}

}

return L\_new ;

}

int \*\*SAPSP(int \*\*A){

int \*\*newL ;

int num=1 ;

int m,a,b,i ;

newL = (int \*\*)malloc (sizeof(int\*)\*ver);

for(i=0; i<ver; i++){

\*(newL + i) = (int\*)malloc(sizeof(int)\*ver) ;

}

for(a=0; a<ver; a++){

for(b=0; b<ver; b++){

newL[a][b]=A[a][b] ;

}

}

for(m=0; m<ver-1; m++){

newL = Extend\_Shortest\_Paths(newL,A) ;

num++ ;

printf(" L%d\n ",num) ;

for(a=0; a<ver; a++){

for(b=0; b<ver; b++){

printf("%d\t",newL[a][b]) ;

}

printf("\n") ;

}

}

return newL ;

}

int \*\*FAPSP(int \*\*A){

int \*\*newL ;

int \*\*LL ;

int num=1 ;

int m=1 ;

int a,b,i ;

newL = (int \*\*)malloc (sizeof(int\*)\*ver) ;

for(i=0; i<ver; i++){

\*(newL + i) = (int\*)malloc(sizeof(int)\*ver) ;

}

for(a=0; a<ver; a++){

for(b=0; b<ver; b++){

newL[a][b]=A[a][b] ;

}

}

while(m < ver-1){

newL = Extend\_Shortest\_Paths(newL,newL) ;

printf("\n") ;

m = 2\*m ;

printf(" L%d\n ",m) ;

for(a=0 ; a<ver ; a++){

for(b=0 ; b<ver ; b++){

printf("%d\t",newL[a][b]) ;

}

printf("\n") ;

}

}

return newL ;

}

int main(void){

int \*\*graph = NULL ;

int i, a, b ;

ver = setViaVerSize() ;

graph = (int \*\*)malloc (sizeof(int\*)\*ver);

for(i=0; i<ver; i++){

\*(graph + i) = (int\*)malloc(sizeof(int)\*ver);

}

for(a=0; a<ver; a++){

for(b=0; b<ver; b++){

if(a==b)

graph[a][b]=0;

else

graph[a][b]=INF;

}

}

\*\*setVia(graph);

printf(" L1\n ") ;

for(a=0; a<ver; a++){

for(b=0 ; b<ver ; b++){

printf("%d\t",graph[a][b]) ;

}

printf("\n") ;

}

\*\*SAPSP(graph) ;

\*\*FAPSP(graph) ;

system("pause") ;

}

**9-2 The Floyd-Warshall Algorithm**

* 알고리즘 설명  
  그래프에서 모든 꼭짓점 사이의 최단 경로의 거리를 구하여 가장 작은 값만 저장하는 알고리즘이다
* 컴파일 방법  
  graph\_sample\_dijkstra.txt 파일을 바탕화면에 넣는다.  
  C:\\Users\\Administrator\\Desktop\\ graph\_sample\_directed.txt
* **Shortest Path**

**1**

**5**

**7**

**-1**

**-8**

**-4**

**2**

**10**

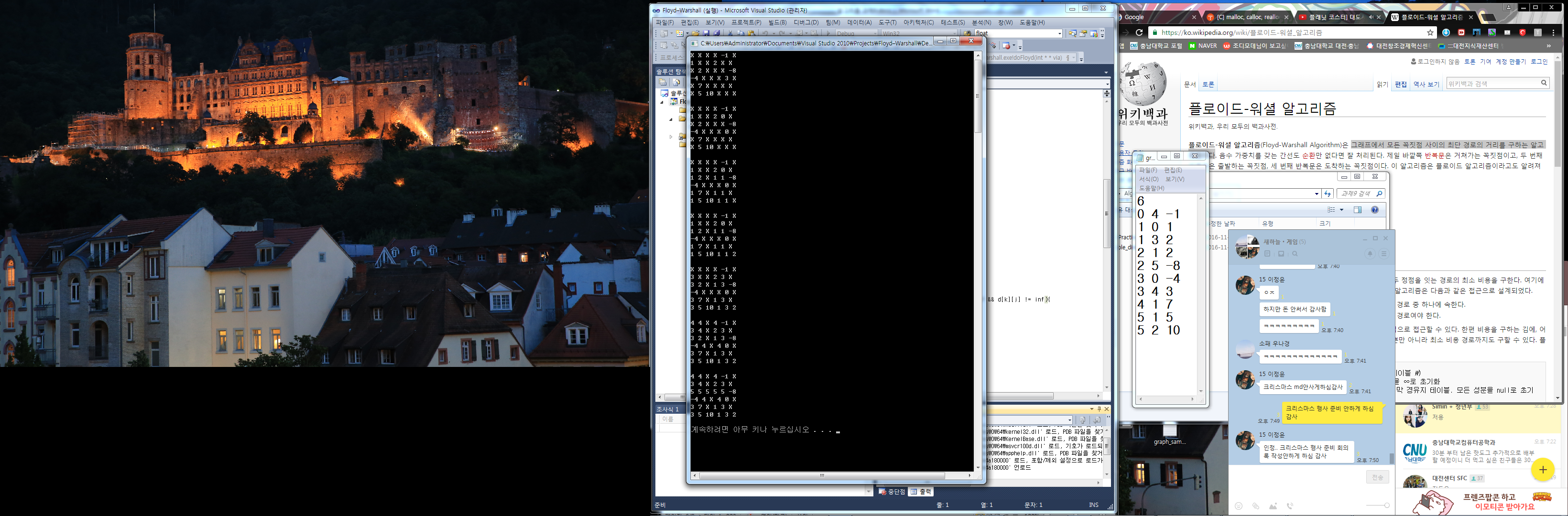
**3**

**2**

**<직접 찾은 값>**

**0 to 1 : 0 – 4 – 1 : 6 (-1+7)  
0 to 2 : 0 – 4 – 1 – 3 – 0 : 길 없음  
0 to 3 : 0 – 4 – 1 – 3 : 8 (-1+7+2)   
0 to 4 : 0 – 4 : -1  
0 to 5 : 0 – 4 – 1 – 3 – 0 : 길 없음**

**<프로그램 결과값>**



Code ( .C )

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <memory.h>

#define N 6

#define inf 999

void printArray(int\*\* arr)

{

int i,j ;

for(i=0 ; i<N ; i++){

for(j=0 ; j<N ; j++){

if(arr[i][j]==inf)

printf("X ") ;

else

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

}

printf("\n") ;

}

printf("\n") ;

}

void getArr(int\*num){

FILE \*fps ;

int temp = 0 , i ;

int n1, n2, n3 ;

// input의C 숫ùy자U들ìe을¡í 배öe열¯¡© num에¯¢® 저u장a하I는¥A 부¬I분¬¨¢

fps = fopen("C:\\Users\\Administrator\\Desktop\\graph\_sample\_directed.txt","rt");

fscanf(fps,"%d",&temp) ;

num[0] = temp ;

for(i=1; i < 31 ; i=i+3){

fscanf(fps,"%d %d %d",&n1, &n2, &n3) ;

num[i] = n1 ;

num[i+1] = n2 ;

num[i+2] = n3 ;

}

fclose(fps) ;

}

int\*\* doFloyd(int\*\* via){

// int\*\* d = cost table

int i, j, k ;

int \*\*d ; // Via Table.

d = (int\*\*)malloc(sizeof(int \*)\*N);

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

d[i] = (int \*)malloc(sizeof(int)\*N);

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

for(j=0 ; j<N ; j++)

d[i][j] = inf ;

// Init

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

{

for (j = 0; j < N; ++j)

{

d[i][j] = via[i][j];

}

}

// Floyd-Warshall Algorithm

for (k = 0; k < N; ++k){

for (i = 0; i < N; ++i){

for (j = 0; j < N; ++j){

if (d[i][j] > d[i][k] + d[k][j] && d[i][k] != inf && d[k][j] != inf){

d[i][j] = d[i][k] + d[k][j] ;

via[i][j] = k ;

}

}

}

printArray(via) ;

}

return d ;

}

int\*\* setCostTable(int\* arr){

int \*\*array ;

int i, j ;

array = (int\*\*)malloc(sizeof(int \*)\*N);

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

array[i] = (int \*)malloc(sizeof(int)\*N);

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

for(j=0 ; j<N ; j++)

array[i][j] = inf ;

for(i=1 ; i<30 ; i=i+3 ){

array[arr[i]][arr[i+1]] = arr[i+2] ;

}

return array ;

}

int main(void){

int num[31], \*\*d ;

getArr(num) ; // Get Input Data

d = setCostTable(num) ;

printArray(d) ;

doFloyd(d) ;

system("pause") ;

return 0;

}