20150282

소프트웨어학부

최다연

알고리즘

신정훈 교수님

traveling salesman problem

1. Algorithm 3.11을 교재의 pseudo code에 충실하도록 구현한다.

#include <iostream>

#include <cmath>

using namespace std;

#define INF 900

int TSPBFway;

int TSPway;

//A = V의 모든 부분집합.

//D[Vi][A] = A에 속한 정점을 각각 한 번씩만 거쳐서 Vi에서 V1으로 가는 최단경로 길이.

typedef struct vertex {

int current;

int next;

}vertex;

int bitc(int s, int n){

int count=0;

int one = 1;

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

if(s & one)

count++;

one = one << 1;

}

return count;

}

int J(int s){

int an = (int)(log10((double)s)/log10(2.0));

if(an<=0)

return 0;

else

return an;

}

//버텍스가 4개 v-{v1} = {v2,v3,v4} 의 부분집합은 -> 공집합,v2,v3,v4, {v2,v3},{v2,v4},{v3,v4},{v2,v3,v4}

void travel(int n, int W[][100], vertex \*\*P, int \*minLength){

int i, j, k;

int sub = (int)pow(2.0,n);

/\*

P = new vertex\*[n];

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

P[i] = new vertex[(int)pow(2, n)];

\*/

int\*\* D = new int\*[n];

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

D[i] = new int[(int)pow(2, n)];

////// D, P 초기화 /////////////////

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

for(j=0; j<(int)pow(2, n); j++)

D[i][j] = 0;

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

for(j=0; j<(int)pow(2, n); j++){

P[i][j].current = 0;

P[i][j].next = 0;

}

}

///////////////////////////////////////

//아무것도 거쳐가지 않을때

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

D[i][0] = W[i][0];

}

/////////////////////////////////////////////////

for(k=1; k < n-1; k++){ // 거쳐가는 node 수

//cout << k << "개를 거칠때"<< endl;

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

j=2;

while( j < sub ){//부분집합 모두

int one = 1;

int temp = INF;

int min = INF\*3;

//s가 k개의 node이고, Vi 미포함, V0 미포함

if((bitc(j, n) == k) && !(j & (int)pow(2, i)) && !(j & 1) ){

//n비트만큼 시프트하면서 해당 부분집합 검사

for(int a=0; a<n; a++){

if(j & one){

temp = (W[i][J(one)] + D[J(one)][ j^one ]);

if(temp >= INF)

temp = INF;

if(temp <= min){

min = temp;

P[i][j].current = J(one); //거쳐갈 노드

P[i][j].next = j^one;

}

}

one = one << 1;

}

D[i][j] = min;

}

j++;

}

}

}

// D[0][V-{v0}]

j = pow(2, n)-2;

int one = 1;

int temp = INF;

int min = INF\*3;

for(int a=0; a<n; a++){

if(j & one){

temp = (W[0][J(one)] + D[J(one)][ j^one ]);

if(temp < min){

min = temp;

P[0][(int)pow(2, n)-2].current = J(one); // 최초경로

P[0][(int)pow(2, n)-2].next = j^one;

}

}

one = one << 1;

}

D[0][(int)pow(2, n)-2] = min;

\*minLength = min;

}

void print\_path(int n, vertex \*\*P){

cout << "P배열을 이용한 경로 출력: ";

// P 출력해보기

int p = 0;

int q = pow(2, n)-2;

int temp1, temp2;

cout << "V1 -> ";

for(int i=0; i<=n-2; i++){

cout << "V"<< P[p][q].current + 1<< " -> ";

temp1 = P[p][q].current;

temp2 = P[p][q].next;

p = temp1;

q = temp2;

}

cout << "V1" << endl<<endl;

}

int main(void){

int min=0;

vertex \*\*P;

int n = 0;

int W[4][100] = {

{ 0, 2, 9, INF },

{ 1, 0, 6, 4 },

{ INF, 7, 0, 8 },

{ 6, 3, INF, 0 } };

int W2[4][100] = {

{ 0,6,5,2},

{3,0,1,2},

{2,3,0,7},

{INF,1,INF,0},

};

int W4[4][100] = {

{ 0,5,4,3},

{1,0,INF,4},

{INF,5,0,6 },

{INF,1,2,0},

};

n=4;

P = new vertex\*[n];

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

P[i] = new vertex[(int)pow(2, n)];

travel(n,W,P,&min);

cout << "\* W최단거리 : "<<min << endl;

print\_path(n, P);

delete[] P;

P = new vertex\*[n];

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

P[i] = new vertex[(int)pow(2, n)];

travel(4,W2,P,&min);

cout << "\* W2최단거리 : "<<min << endl;

print\_path(n, P);

delete[] P;

P = new vertex\*[n];

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

P[i] = new vertex[(int)pow(2, n)];

travel(4,W4,P,&min);

cout << "\* W4최단거리 : "<<min << endl;

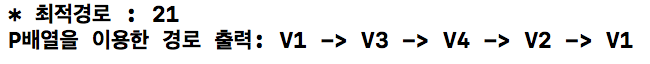
print\_path(n, P);

delete[] P;

return 0;

}

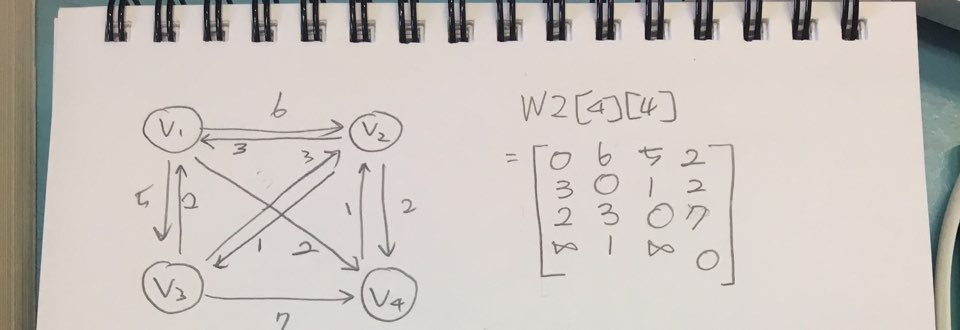
2. Figure 3.16을 입력으로 하여 구현한 코드를 검증한다.



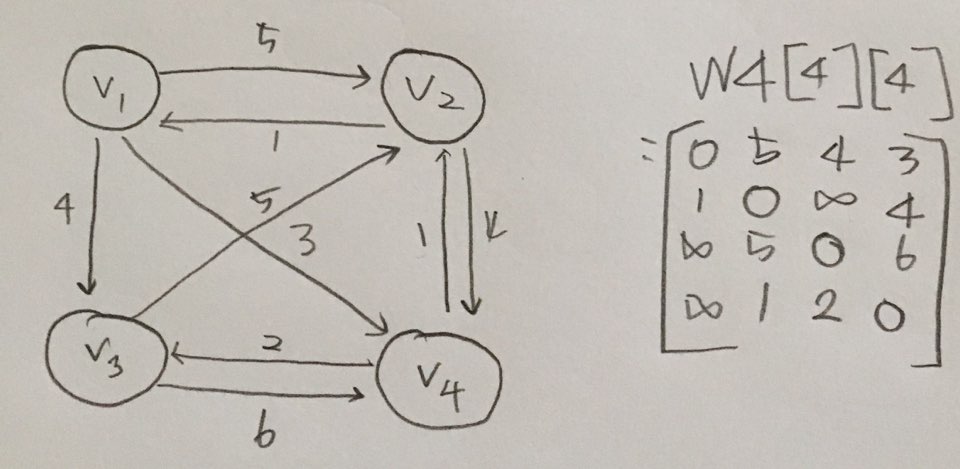
3. 임의의 추가적인 입력 그래프 2개를 생성한 후, 구현한 코드를 검증한다.

- 임의의 추가적인 입력 그래프를 보고서에 표현하고, 코드를 통한 검증 전에 solution에 해당하는 경로 또한 보고서에 넣어야 한다.

W2

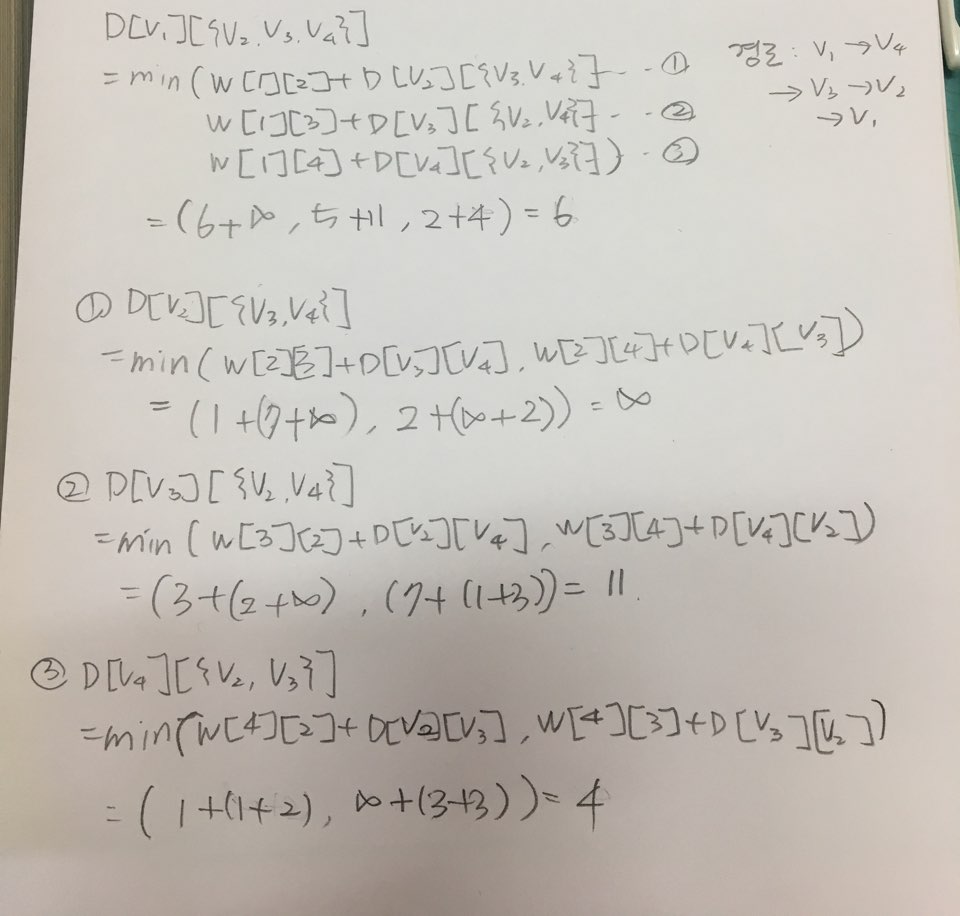


W4



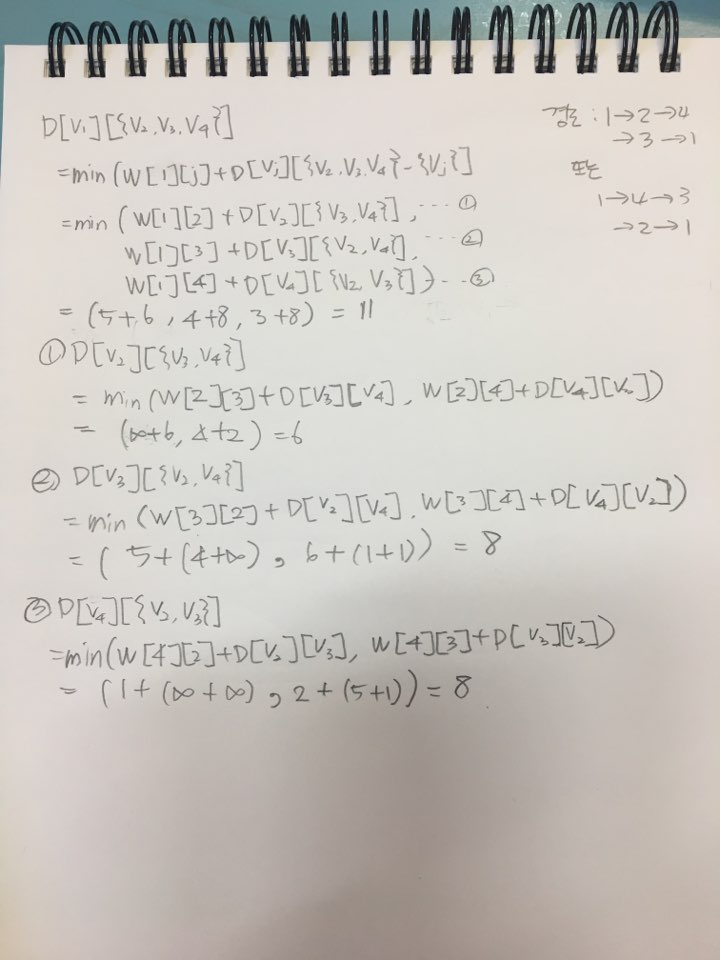
solution 경로

W2 :



최단거리 : 6, 경로 : v1 -> v4 -> v3 -> v2 ->v1

W4 :



최단거리 : 11, 최단경로 : v1 ->v2 -> v4 -> v3 -> v1 또는 v1 -> v4 -> v3 -> v2-> v1

4. P 배열을 이용하여 optimal tour를 출력하는 코드 또한 작성하여, 입력 그래프 3개를 대상으로 출력하시오.

