알고리즘\_과제\_5

(SHORTEST PATH ALGORITHM)



학부:컴퓨터학부

학번:20162518

출석번호 : 156번

이름 : 최승서

✥소스 코드

#include<stdio.h>

#include<stdlib.h>

#define INF 987654321

#define N 5

int W[N][N] = { //구하고자하는 그래프 기본값

{0, 10, 5, INF, INF},

{INF, 0, 2, 1, INF},

{INF, 3, 0, 9, 2},

{INF,INF,INF, 0, 4},

{7,INF, INF, 5, 0}

};

int D[N][N] = { //경로에 대한 비용

{0, 10, 5, INF, INF},

{INF, 0, 2, 1, INF},

{INF, 3, 0, 9, 2},

{INF,INF,INF, 0, 4},

{7,INF, INF, 5, 0}

};

/\*\*\*\*\*\*\*\*\*\*3,4번 그래프 변환값\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int W[N][N] = { //구하고자하는 그래프 기본값

{0, 4, INF, INF, INF},

{5, 0, 7, INF, INF},

{INF, INF, 0, 10, 5},

{1,INF,INF, 0, 2},

{9,2, INF, 3, 0}

};

int D[N][N] = { //경로에 대한 비용

{0, 4, INF, INF, INF},

{5, 0, 7, INF, INF},

{INF, INF, 0, 10, 5},

{1,INF,INF, 0, 2},

{9,2, INF, 3, 0}

};

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int P[N][N] = {0}; //(초기상태) 각 정점 직전에 연결되있는 정점

void floyd2(void);

void path(int q, int r){ //최단경로 출력 알고리즘

printf("path(%d, %d)\n", q, r);

if(P[q][r] != 0){

path(q, P[q][r]);

printf("[V%d] -> \n", P[q][r]);

path(P[q][r],r);

}

}

void floyd2(){ //플로이드 알고리즘

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

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

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

if(D[i][k] + D[k][j] < D[i][j]){

P[i][j] = k;

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

}

}

}

}

}

void print\_table(int A[N][N]){ //각 table 값을 출력

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

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

if(A[i][j] == INF)

printf("INF\t");

else

printf("%d\t", A[i][j]);

if(j == 4)

printf("\n");

}

}

}

int main(int argc, char\* argv[]) {

int q, r;

printf("출발점을 입력하시오: ");

scanf("%d", &q);

printf("도착점을 입력하시오: ");

scanf("%d", &r);

floyd2();

printf("\n\ntable W\n");

print\_table(W);

printf("\n\ntable D\n");

print\_table(D);

printf("\n\ntable P\n");

print\_table(P);

printf("\n\n[V%d] -> \n", q);

path(q,r);

printf("[V%d]\n", r);

return 0;

}

1,2)

Algorithm 3.5(최단거리 출력 알고리즘), Algorithm 3.4(floyd algorithm)을 이용해 두 정점 V0과 V3 사이의 최단경로를 찾고 찾는 과정을 단계적으로 보여라.

Algorithm 3.5(최단거리 출력 알고리즘)과 Algorithm 3.4(floyd algorithm) 의 의사코드는 아래와 같다

void path(int q, int r){

if(P[q][r] != 0){

path(q, P[q][r]);

printf("[V%d] -> \n", P[q][r]);

path(P[q][r],r);

}

}

void floyd2(){

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

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

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

if(D[i][k] + D[k][j] < D[i][j]){

P[i][j] = k;

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

}

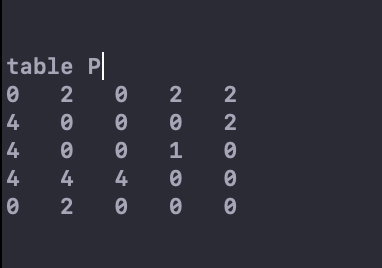
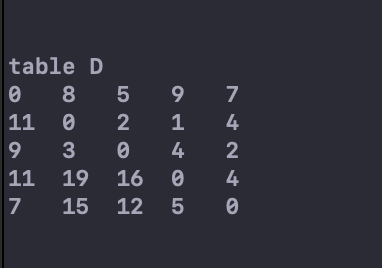
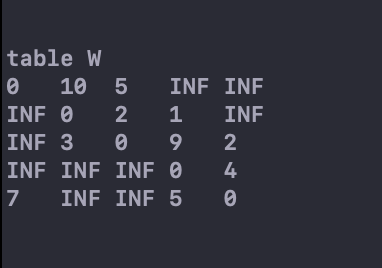
}

}

}

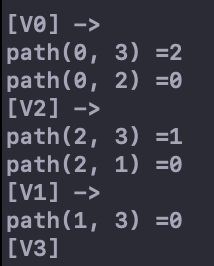
}

첫번째로 그래프를 행렬화 하여 W 값을 기입 해준다(연결되어 있지 않은 정점들은 INF로 표현했다). D는 경로의 대한 비용이 저장 되는데 초기값은 W와 같다. 입력한 W와 D의 값으로 floyd algorithm을 통해 D[a][b]는 정점 a,b의 경로에 대한 최소 비용값이 저장되고, P[a][b]는 a,b의 최단 경로에 있는 가장 큰 인덱스 값을 저장한다.



Path algorithm 코드를 보면 q와 r은 우리가 구하고자 하는 각각의 정점이고, 앞서 설명했듯이 이차원 배열 P[q][r]은 f 각 정점 q와 r사이에 놓여있는 정점 중에서 가장 큰 인덱스이다.

여기서 V0과 V3이 q, r일때 P[0][3]의 값이 0이면 V0,V3이 최단경로임을 뜻하며 값이 0이 아니면 q값에서 p[q][r]의 값으로 나온 정점으로 가는 경로를 찾아야한다. 재귀적인 절차를 통해 함수를 재호출하여 0값이 나오는 경로 즉 최단거리를 찾아가는 것이다.



이 결과 화면을 보면 P[0][3]의 값이 2가 나오고 다시 P[0][2]의 값이 0임을 확인해 정점 V2를 찾아냈다.

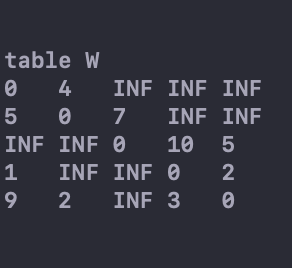
이 과정을 반복해 V0 과 V3의 최단 경로는 V0 -> V2 -> V1 -> V3임을 알 수 있었다.

3,4) 원래 그래프에서 V0, V1, V2, V3, V4를 a3, a4, a5, a1, a2으로

바꾼 뒤 Shortest Path Problem을 다시 해결하여 Matrix D와 P를 다시

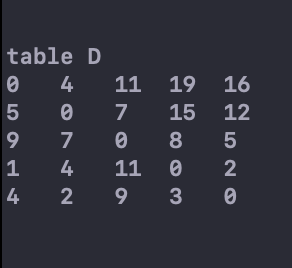
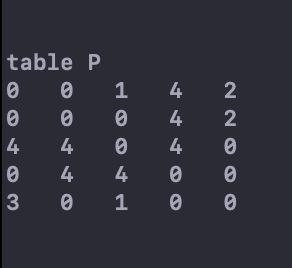
구한다.

먼저 바뀐 그래프를 이용해 Matrix W 값을 다시 구하면



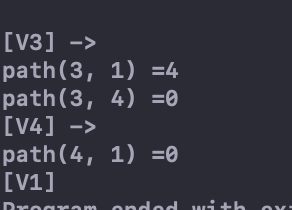
위의 실행화면과 같이 나온다.

이 Matrix W를 Floyd algorithm를 통해 Matrix D, P를 구한다

구해진 Matrix D,P를 이용해 Path algorithm으로 a3에서 a1로 가는 최단 경로를 찾는다.

찾는 과정은 아래와 같다(편의를 위해 a를 v로 표현했다).



a3에서 a1로 가는 최단경로중에 가장 큰 인덱스 값은 a4가 나온다.

다음으로 a3에서 a4의 가는 최단경로의 가장 큰 인덱스 값이 0이 나오므로 a3,a4가 최단 경로임을 뜻한다.

또 a4에서 a1로 가는 최단경로의 가장 큰 인덱스 값은 0이 나온다, 이것은 전과 동일하게 a4, a1이 최단 경로이다.

따라서 a3,a1의 최단경로는 a3 -> a4 -> a1이 된다.