Floyd's Algorithm

For Shortest Paths

소프트웨어학부

20170294 박해영

< 목 차 >

1. 목표

Dynamic Programming을 사용하여, 최단 경로 문제를 푸는 3차 시간 알고리즘을 만들어 내고, 더불어 하나의 정점에서 또 다른 정점으로 가는 최단경로를 출력할 수 있다. 또한 이 과정을 통해 동적계획 방법의 설계 과정을 이해하고, 최단경로 찾는 문제를 통해 최적화 문제가 무엇인지를 깨닫고자 하며, 직접 배열을 통해 구현해 보고, 계산해 봄으로써 상향식(bottom-up) 접근방법을 배우고자 한다.

2. Problem & Input / Output

*** Problem** : 가중치 포함 그래프의 각 정점에서 다른 모든 정점까지의 최단 거리를 계산하고, 각각의 최단경로를 구하라

*** Input** : 가중치 포함 방향성 그래프 W와 그 그래프에서의 정점의 수 v

*** Output** : 최단경로의 길이가 포함된 배열 D, 최단경로의 정점을 표시해 놓은 배열 P

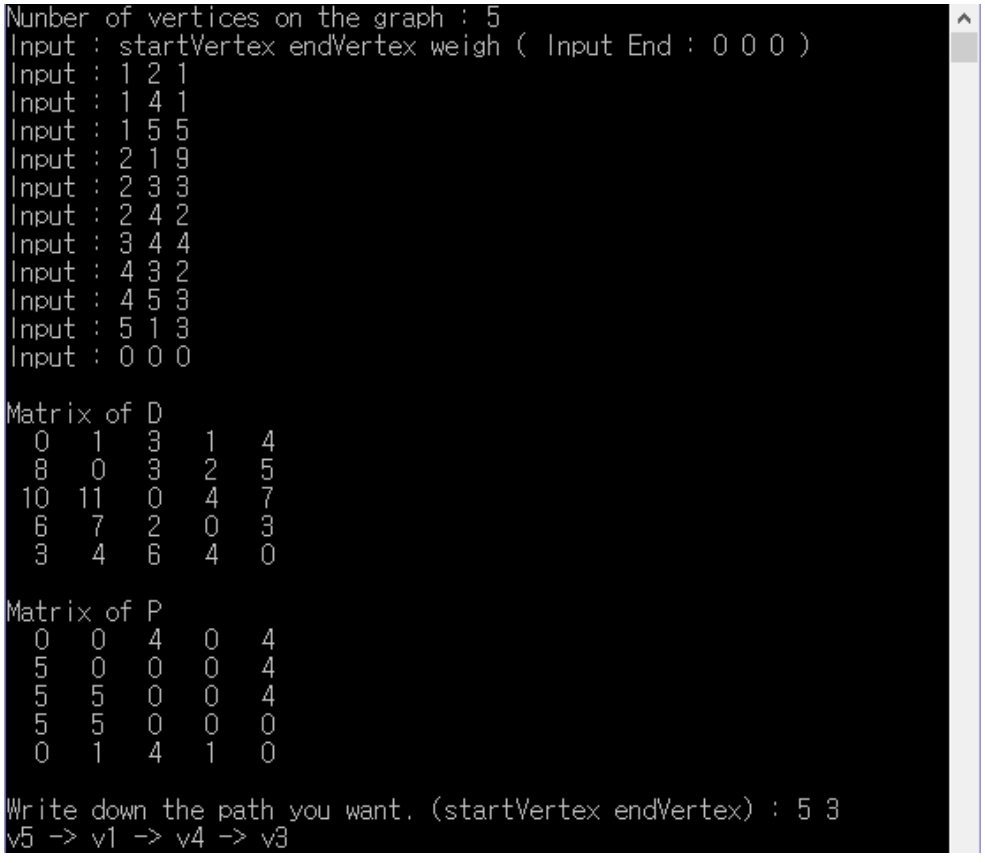
3. 구현 Language & 사용 Tool

* **구현 언어** : C language

* 사용 Tool : Visual Studio 2017

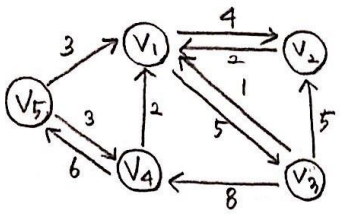
4. 교재의 입력 데이터 테스트

1) 가중치 포함 방향성 그래프 W



5. 자작 입력 데이터 생성 & 알고리즘의 과정 손계산

최단 경로의 출력.



W =	1	2	3	4	5
1	0	4	5	∞	∞
2	2	0	∞	∞	∞
3	1	5	0	8	∞
4	2	∞	∞	0	6
5	3	∞	∞	3	0

; 최초 W 구성.

P = 5X5 array 0으로 초기화

floyd2 함수 호출. ; n=5

① k=1

→ i=1, j=1.
D[1][1]+D[1][1] < D[1][1] ; false

→ i=1, j=2.
D[1][1]+D[1][2] < D[1][2] ; false.

→ i=1, j=3
D[1][1]+D[1][3] < D[1][3] ; false

→ i=1, j=4
D[1][1]+D[1][4] < D[1][4] ; false

→ i=1, j=5
D[1][1]+D[1][5] < D[1][5] ; false

→ i=2, j=1
D[2][1]+D[1][1] < D[2][1] ; false

→ i=2, j=2
D[2][1]+D[1][2] < D[2][2] ; false

→ i=2, j=3
D[2][1]+D[1][3] < D[2][3] ; true
{ P[2][3] = 1.
D[2][3] = D[2][1]+D[1][3] = 7.

→ i=2, j=4
D[2][1]+D[1][4] < D[2][4] ; false

→ i=2, j=5
D[2][1]+D[1][5] < D[2][5] ; false

→ i=3, j=1.
D[3][1]+D[1][1] < D[3][1] ; false

→ i=3, j=2.
D[3][1]+D[1][2] < D[3][2] ; false

→ i=3, j=3
D[3][1]+D[1][3] < D[3][3] ; false

→ i=3, j=4
D[3][1]+D[1][4] < D[3][4] ; false

→ i=3, j=5
D[3][1]+D[1][5] < D[3][5] ; false

→ i=4, j=1.
D[4][1]+D[1][1] < D[4][1] ; false

→ i=4, j=2
D[4][1]+D[1][2] < D[4][2] ; true
{ P[4][2] = 1.
D[4][2] = D[4][1]+D[1][2] = 6.

→ i=4, j=3.
D[4][1]+D[1][3] < D[4][3] ; true
{ P[4][3] = 1.
D[4][3] = D[4][1]+D[1][3] = 7.

→ i=4, j=4
D[4][1]+D[1][4] < D[4][4] ; false

→ i=4, j=5
D[4][1]+D[1][5] < D[4][5] ; false

→ i=5, j=1.
D[5][1]+D[1][1] < D[5][1] ; false

→ i=5, j=2
D[5][1]+D[1][2] < D[5][2] ; true
P[5][2] = 1
D[5][2] = D[5][1]+D[1][2] = 7.

→ i=5, j=3

D[5][1]+D[1][3] < D[5][3] ; true

P[5][3] = 1,

D[5][3] = D[5][1]+D[1][3] = 8

→ i=5, j=4

D[5][1]+D[1][4] < D[5][4] ; false

→ i=5, j=5

D[5][1]+D[1][5] < D[5][5] ; false

∴ $D=\begin{bmatrix} 0 & 4 & 5 & \infty & \infty \\ 2 & 0 & 7 & \infty & \infty \\ 1 & 5 & 0 & 8 & \infty \\ 2 & 6 & 7 & 0 & 6 \\ 3 & 7 & 8 & 3 & 0 \end{bmatrix}$

$P=\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}$

위와 같은 방법으로 k=2, 3, 4, 5 일때를 실행.

k=3

→ i=1, j=4

D[1][3]+D[3][4] < D[1][4] ; true

P[1][4] = 3

D[1][4] = D[1][3]+D[3][4] = 13

→ i=2, j=4

D[2][3]+D[3][4] < D[2][4] ; true

P[2][4] = 3

D[2][4] = D[2][3]+D[3][4] = 15

$D=\begin{bmatrix} 0 & 4 & 5 & 13 & \infty \\ 2 & 0 & 7 & 15 & \infty \\ 1 & 5 & 0 & 8 & \infty \\ 2 & 6 & 7 & 0 & 6 \\ 3 & 7 & 8 & 3 & 0 \end{bmatrix}$, $P=\begin{bmatrix} 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 1 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}$

k=4

→ i=1, j=5

D[1][4]+D[4][5] < D[1][5] ; true

P[1][5] = 4

D[1][5] = D[1][4]+D[4][5] = 19

→ i=2, j=5

D[2][4]+D[4][5] < D[2][5] ; true

P[2][5] = 4

D[2][5] = D[2][4]+D[4][5] = 21

→ i=3, j=5

D[3][4]+D[4][5] < D[3][5] ; true

P[3][5] = 4

D[3][5] = D[3][4]+D[4][5] = 14

$D=\begin{bmatrix} 0 & 4 & 5 & 13 & 19 \\ 2 & 0 & 7 & 15 & 21 \\ 1 & 5 & 0 & 8 & 14 \\ 2 & 6 & 7 & 0 & 6 \\ 3 & 7 & 8 & 3 & 0 \end{bmatrix}$, $P=\begin{bmatrix} 0 & 0 & 0 & 3 & 4 \\ 0 & 0 & 1 & 3 & 4 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}$

∴ 최종 D & P 가 구해졌다

⇒ 최단경로 찾기 : $V_5 \rightarrow V_3$

① path(5, 3)

P[5][3] == 1

② path(5, 1)

P[5][1] == 0

③ path(1, 3)

P[1][3] == 0

출력결과 : P배열의 값이 0이면 값출력

① ② ③

$V_5 \rightarrow V_1 \rightarrow V_3$,,

V_5에서 V_3으로 가는 최단경로

⇒ [V_5 V_1 V_3]

6. 자작 입력 데이터 테스트