AI 알고리즘

네트워크 모형(최단경로, CPM/PERT 등)

수퍼스타를 키우는 곳
전주대학교

학습내용

- 다익스트라(Dijkstra) 알고리즘
- Floyd-Warshall 알고리즘

학습목표

- 다익스트라(Dijkstra) 알고리즘에 대해 설명할 수 있다.
- Floyd-Warshall 알고리즘에 대해 설명할 수 있다.

최단경로 문제

다익스트라(Dijkstra) 알고리즘

다익스트라(Dijkstra) 알고리즘

❖ 최단 경로(Shortest Path) 문제

- 주어진 가중치 그래프에서 어느 한 출발점에서 또 다른 도착점까지의 최단 경로를 찾는 문제
- 최단 경로를 찾는 가장 대표적인 알고리즘
  - 다익스트라(Dijkstra) 알고리즘
    - 주어진 네트워크 내에서 하나의 마디(출발 마디)로부터 모든 다른 마디로의 최단 경로를 구하기 위해 개발
  - Floyd-Warshall 알고리즘
    - 주어진 네트워크 내에서 임의의 두 마디 사이의 모든 최단 경로들을 구하기 위해 개발

❖ 기호 정의

- v_i
  - 출발 마디 1로부터 마디 i까지의 최단거리
- d_{ij}
  - 호(i,j) 사이의 길이
- 마디 i로부터 곧바로 연결되는 마디 j에 대한 라벨을 다음과 같이 정의
  - $[v_j, \mathrm{i}] = [v_i + d_{ij},\ \ i],\ \ d_{ij} \geq 0$
- 출발 마디에 대한 라벨은 [0,-]로 표시
  - 출발 마디의 선행마디는 없음을 나타냄

❖ 라벨

- 임시 라벨
  - 어떤 마디까지의 더 짧은 경로가 발견될 수 있다면 수정될 수 있음
- 영구 라벨
  - 더 좋은 경로가 발견되지 않을 때, 임시 라벨은 영구 라벨로 변경

다익스트라(Dijkstra) 알고리즘

❖ 절차

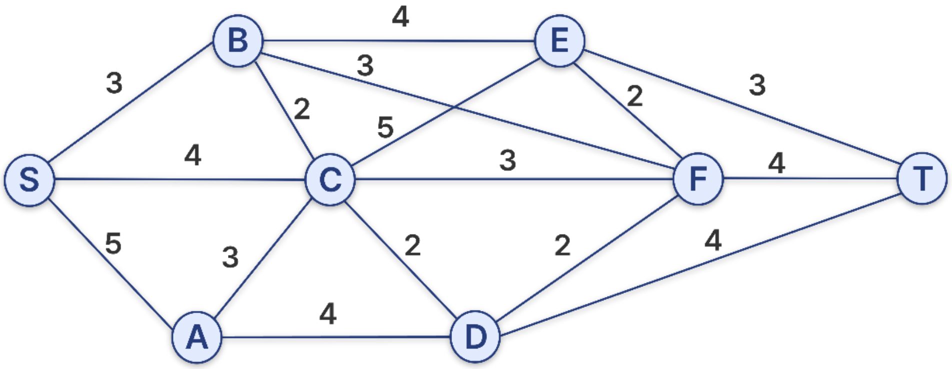
- 단계0
  - 출발마디(마디1)에 영구라벨 [0,-] 을 붙이고 i=1로 놓음
- 단계k

① 마디 j 가 영구라벨이 아닐 경우, 마디 i 로부터 직접 연결되는 마디 j 에 대해,
임시라벨 $[v_i + d_{ij}, i]$ 을 계산 마디 j 가 다른 마디 k 를 통해 라벨 $[v_j + k]$ 를 가지고 있는 경우,
$v_i + d_{ij} < v_j$ 이면 $[v_j + k]$ 를
$[v_i + d_{ij}, i]$ 로 대체

② 모든 마디들이 영구라벨을 가진다면 이 절차를 멈춤 아니면, 모든 임시 라벨들 중에 최단거리 $(= v_m)$를 갖는 라벨 $[v_m, l]$ 을 선택(동일한 값이 존재할 경우 임의로 선택). i=m으로 놓고 단계 k반복

❖ 예제

- S부터 T까지의 최단경로



- 단계0
  - 마디 S에 영구라벨 [0,-] 을 붙이고 i=1로 놓음
- 단계1
  - 마지막 영구라벨인 마디 S로부터 마디 A, B, C가 직접 연결되므로 각 마디에 대한 라벨을 다음과 같이 계산

마디	라벨	상태	마디	라벨	상태
S	[0, -]	영구 라벨	B	[0 + 3, S] = [3, S]	임시 라벨
A	[0 + 5, S] = [5, S]	임시 라벨	C	[0 + 4, S] = [4, S]	임시 라벨

- 3개의 임시 라벨 중 마디B가 최소거리를 제공함으로 마디 B를 영구 마디로 변경

다익스트라(Dijkstra) 알고리즘

❖ 예제

- 단계2
  - B로부터 마디 C, E, F가 직접 연결되므로 각 마디에 대한 라벨을 다음과 같이 계산

마디	라벨	상태	마디	라벨	상태
S	[0, -]	영구 라벨	C	[0 + 4, S] = [4, S] [3 + 2, B] = [5, B] X	임시 라벨
A	[0 + 5, S] = [5, S]	임시 라벨	E	[3 + 4, B] = [7, B]	임시 라벨
B	[3, S]	영구 라벨	F	[3 + 3, B] = [6, B]	임시 라벨

  - 4개의 임시 라벨 중 마디 C가 최소거리를 제공함으로 마디 C를 영구 마디로 변경

- 단계3
  - C로부터 마디 E, F가 직접 연결되므로 각 마디에 대한 라벨을 다음과 같이 계산

마디	라벨	상태	마디	라벨	상태
S	[0, -]	영구 라벨	D	[4 + 2, C] = [6, C]	임시 라벨
A	[0 + 5, S] = [5, S]	임시 라벨	E	[3 + 4, B] = [7, B] [4 + 5, C] = [9, C] X	임시 라벨
B	[3, S]	영구 라벨	F	[3 + 3, B] = [6, B] [4 + 3, C] = [7, C] X	임시 라벨
C	[0 + 4, S] = [4, S]	영구 라벨			

  - 4개의 임시 라벨 중 마디 A가 최소거리를 제공함으로 마디 A를 영구 마디로 변경

- 단계4
  - A로부터 마디 D가 직접 연결되므로 각 마디에 대한 라벨을 다음과 같이 계산

마디	라벨	상태	마디	라벨	상태
S	[0, -]	영구 라벨	D	[4 + 2, C] = [6, C] [5 + 4, A] = [9,A] X	임시 라벨
A	[0 + 5, S] = [5, S]	영구 라벨	E	[3 + 4, B] = [7, B]	임시 라벨
B	[3, S]	영구 라벨	F	[3 + 3, B] = [6, B]	임시 라벨
C	[0 + 4, S] = [4, S]	영구 라벨			

  - 3개의 임시 라벨 중 마디 D가 최소거리를 제공함으로 마디 D를 영구 마디로 변경

다익스트라(Dijkstra) 알고리즘

❖ 예제

- 단계5
  - A로부터 마디 D가 직접 연결되므로 각 마디에 대한 라벨을 다음과 같이 계산

마디	라벨	상태	마디	라벨	상태
S	[0, -]	영구 라벨	D	[4 + 2, C] = [6, C]	영구 라벨
A	[0 + 5, S] = [5, S]	영구 라벨	E	[3 + 4, B] = [7, B]	임시 라벨
B	[3, S]	영구 라벨	F	[3 + 3, B] = [6, B] [4 + 3, C] = [7, C] X [6 + 2, D] =[8, D] X	임시 라벨
C	[0 + 4, S] = [4, S]	영구 라벨	T	[6 + 4, D] = [10, D]	임시 라벨

  - 3개의 임시 라벨 중 마디 F가 최소거리를 제공함으로 마디 F를 영구 마디로 변경
- 단계6
  - 마지막 영구 라벨인 마디 F로부터 마디 T가 직접 연결되므로 각 마디에 대한 라벨을 계산

마디	라벨	상태	마디	라벨	상태
S	[0, -]	영구 라벨	D	[4 + 2, C] = [6, C]	임시 라벨
A	[0 + 5, S] = [5, S]	임시 라벨	E	[3 + 4, B] = [7, B] [4 + 5, C] = [9, C] X	임시 라벨
B	[3, S]	영구 라벨	F	[3 + 3, B] = [6, B] [4 + 3, C] = [7, C] X	임시 라벨
C	[0 + 4, S] = [4, S]	영구 라벨			

  - 2개의 임시 라벨 중 마디 E가 최소거리를 제공함으로 마디 E를 영구 마디로 변경
- 단계7
  - 마지막 영구 라벨인 마디 F로부터 마디 T가 직접 연결되므로 각 마디에 대한 라벨을 계산

마디	라벨	상태	마디	라벨	상태
S	[0, -]	영구 라벨	D	[4 + 2, C] = [6, C]	영구 라벨
A	[0 + 5, S] = [5, S]	영구 라벨	E	[3 + 4, B] = [7, B]	영구 라벨
B	[3, S]	영구 라벨	F	[3 + 3, B] = [6, B]	영구 라벨
C	[0 + 4, S] = [4, S]	영구 라벨	T	[6 + 4, D] = [10, D] [6 + 4, F] = [10, F] [7 + 3, E] = [10, E]	임시 라벨

  - 3개의 임시 라벨 중 마디 T가 최소거리를 제공함으로 마디 T를 영구 마디로 변경

다익스트라(Dijkstra) 알고리즘

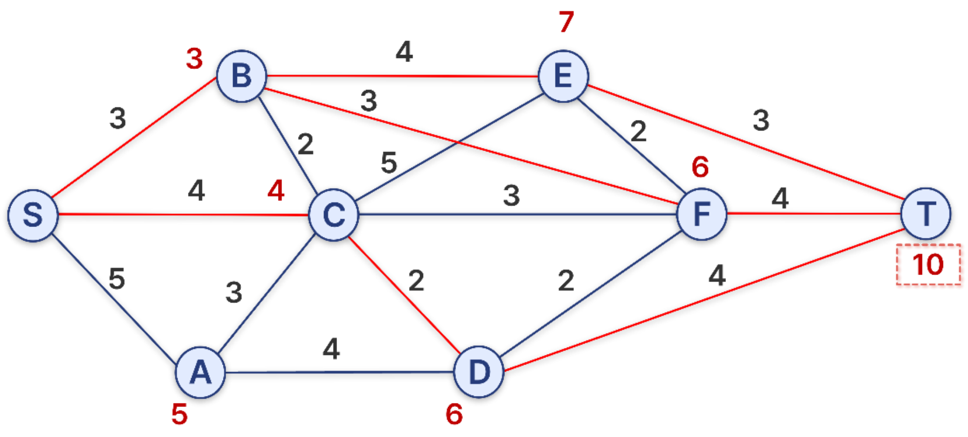
❖ 예제

- 단계8
  - 모든 마디가 영구라벨을 갖게 되었으므로 절차를 멈춤

마디	라벨	상태	마디	라벨	상태
S	[0, -]	영구라벨	D	[4 + 2, C] = [6, C]	영구라벨
A	[0 + 5, S] = [5, S]	영구라벨	E	[3 + 4, B] = [7, B]	영구라벨
B	[3, S]	영구라벨	F	[3 + 3, B] = [6, B]	영구라벨
C	[0 + 4, S] = [4, S]	영구라벨	T	[6 + 4, D] = [10, D] [6 + 4, F] = [10, F] [7 + 3, E] = [10, E]	영구라벨

- 최단경로
  - T-D-C-S
  - T-F-B-S
  - T-E-B-S

- S부터 T까지의 최단 경로



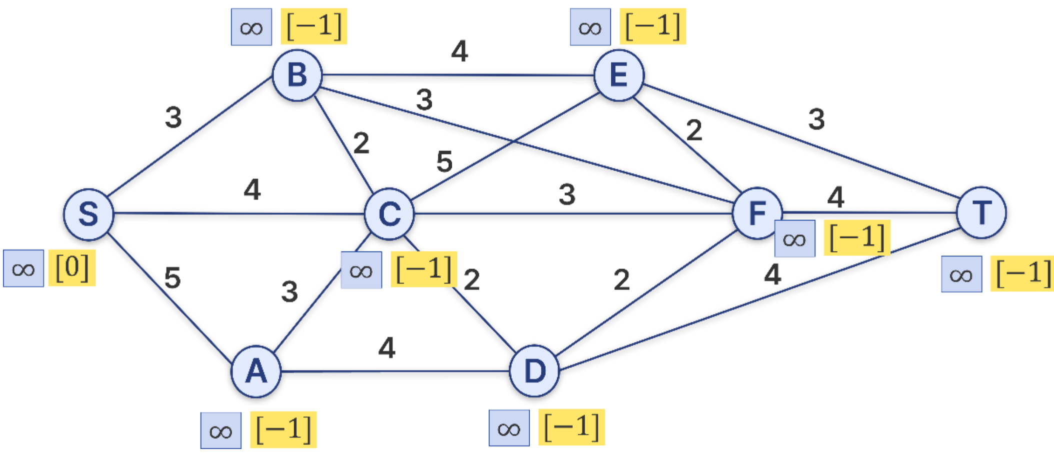
❖ 다익스트라 알고리즘과 Prim 알고리즘

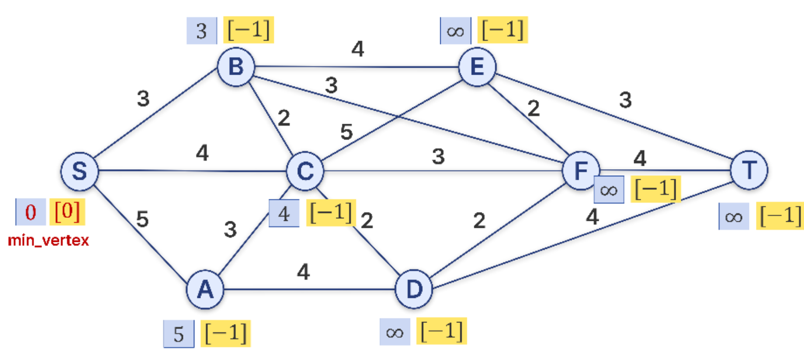
- 차이점
  - 다익스트라 알고리즘은 출발점이 주어지지만 Prim 알고리즘에서는 출발점이 주어지지 않음
  - Prim 알고리즘에서는 D의 원소에 간선의 가중치 저장, 다익스트라 알고리즘에서는 D의 원소에 출발점으로부터 각 정점까지의 경로 길이가 저장

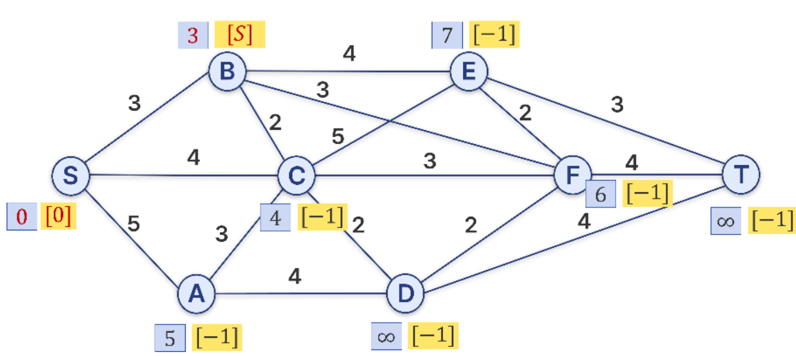
다익스트라(Dijkstra) 알고리즘

❖ 예제

- S부터 T까지의 최단 경로



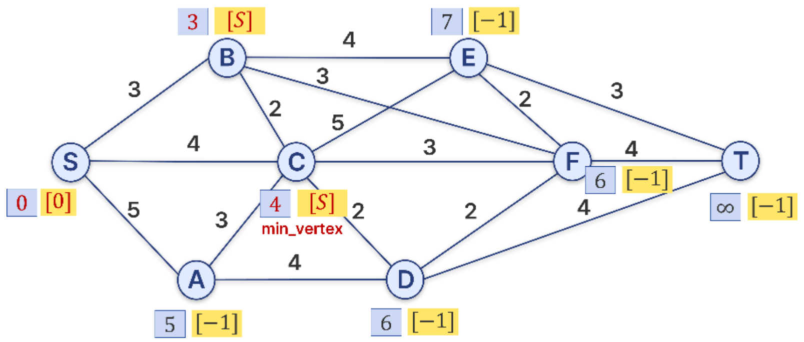


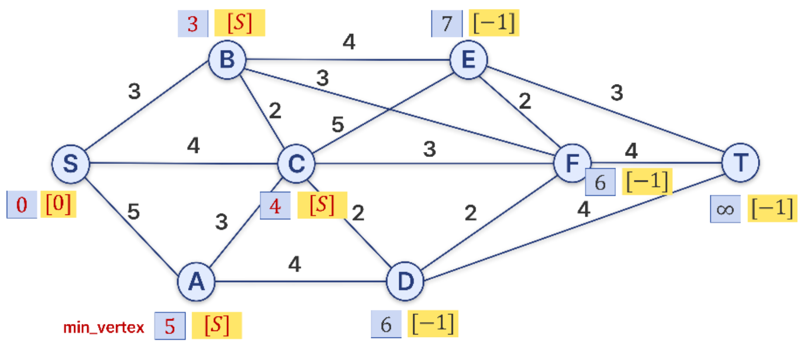


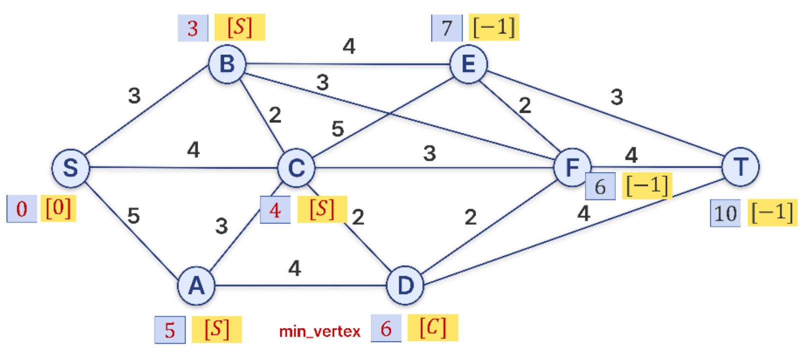
다익스트라(Dijkstra) 알고리즘

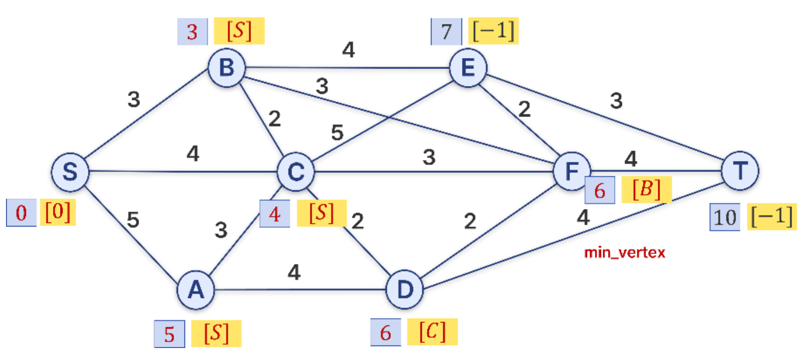
❖ 예제

- S부터 T까지의 최단 경로





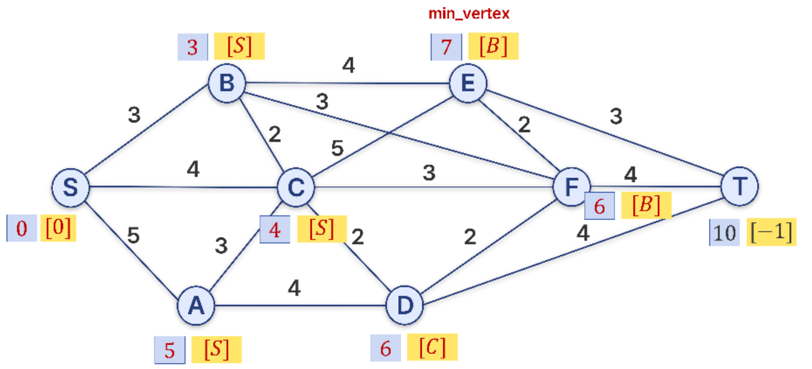


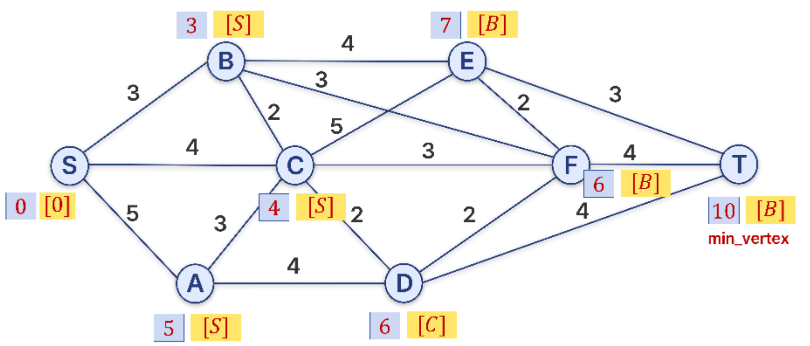


다익스트라(Dijkstra) 알고리즘

❖ 예제

- S부터 T까지의 최단 경로





❖ 수행시간

- Dijkstra 알고리즘은 N번의 반복을 거쳐 min_vertex를 찾고 min_vertex에 인접하면서 방문되지 않은 정점들에 대한 간선 완화를 시도
- 이후 D에서 min_vertex를 탐색하는데 O(N) 시간이 소요되고, min_vertex에 인접한 정점들을 검사하여 D의 원소들을 갱신하므로 추가로 O(N) 시간이 소요
- 따라서 총 수행 시간은 N *(O(N) +O(N)) = O(N2)

다익스트라(Dijkstra) 알고리즘

❖ 최단 경로(Shortest Path) 문제

- 선형계획모형
  - x_{ij} =호(i,j) 사이의 흐름양
  - c_{ij} =호(i,j) 의 길이
  - 목적함수
    - 총 흐름양의 최소화

$$Mininmize \sum_{\{(i,j),i=1,\cdots,n,j=1,\cdots,n\}} c_{ij}x_{ij}$$

  - 제약식
    - 총 진입 흐름양=총 진출 흐름양

$$\sum_{\{(i,k),i=1,\cdots,n,\}} x_{ik} - \sum_{\{(k,j),i=1,\cdots,n,\}} x_{kj}$$

= 순수요량(−1, 0, 1)

최단경로 문제

Floyd-Warshall 알고리즘

Floyd-Warshall 알고리즘

❖ Floyd-Warshall 알고리즘

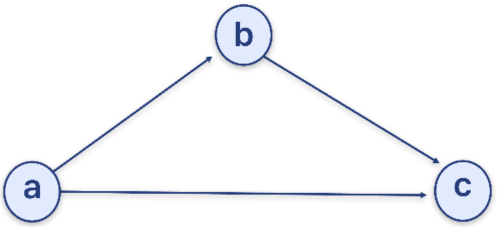
- 주어진 네트워크 내의 모든 서로 다른 두 마디 사이의 최단 경로
  - 내비게이션
  - 두 점 사이의 최단 경로 구하기

	서울	인천	수원	대전	전주	광주	대구	울산	부산
서울		40	40	155	230	320	300	410	430
안천			44	175	250	350	320	450	450
수원				130	190	300	270	355	390
대전					95	185	150	260	280
전주						105	220	330	320
광주							220	330	270
대구								110	135
울산									50
부산									

- 3개의 마디 사이에 비교를 근간으로 산출
  - 만일 $d\_kt+d\_tl<d\_kl$ 이 성립
    → 마디 k로부터 마디 l까지의 최단 거리는 마디 t를 거쳐 가는 것
    → 두 마디 k와 l 사이의 최단경로는 $k\to t\to l$로 대체하는 것이 최적

❖ 아이디어

- 작은 그래프에서 부분 문제들을 찾아보자
  - a에서 c까지의 최단 경로를 찾으려면 2가지 경로, 즉, a에서 c로 직접 가는 경로와 점 b를 경유하는 경로 중에서 짧은 것을 선택
  - N개의 노드의 경유 가능한 점을 모두 고려해보자



Floyd-Warshall 알고리즘

❖ 부분 문제

- 그래프의 점이 1, 2, 3, ⋯, n일 때
  - D_{ij}^{k} = 점 {1, 2, ⋯, k}를 경유 가능한 점 중
    점 i에서 점 j까지의 모든 경로 중에서 가장 짧은 경로의 거리
- 기호 정의
  - d_{kl}: 호(k, l)의 길이
  - D_t: 반복 t에서의 각 마디 간 최단 거리를 나타내는 거리행렬
  - S_t: 반복 t에서의 각 마디 간 최단 경로의 직전 마디를 나타내는 순서 행렬
- 초기 거리 행렬과 순서 행렬
  - 기본 아이디어 2개, 각각의 노드에서의 거리와 관련된 행렬과 순서행렬

$$\begin{bmatrix} - & d_{12} \cdots & d_{1n} \\ d_{21} & - & d_{2n} \\ \vdots & \ddots & \vdots \\ d_{n1} & \cdots & - \end{bmatrix} \qquad \begin{bmatrix} - & 2 \cdots & n \\ 1 & - & n \\ \vdots & \ddots & \vdots \\ 1 & \cdots & - \end{bmatrix}$$

❖ Floyd-Warshall 알고리즘

- 단계 0

$D_0 =$

	1	2	3	4	5	6	7	8
1	-	5	3	4	∞	∞	∞	∞
2	5	-	∞	3	4	∞	∞	∞
3	3	∞	-	2	∞	4	3	∞
4	4	3	2	-	2	5	3	∞
5	∞	4	∞	2	-	∞	2	4
6	∞	∞	4	5	∞	-	2	3
7	∞	∞	3	3	2	2	-	4
8	∞	∞	∞	∞	4	3	4	-

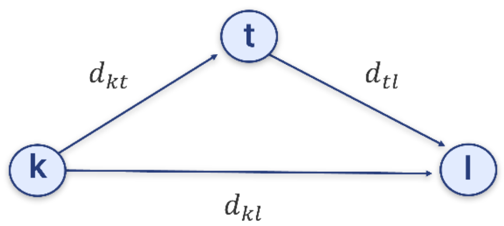
$S_0 =$

	1	2	3	4	5	6	7	8
1	-	A	B	C	D	E	F	T
2	S	-	B	C	D	E	F	T
3	S	A	-	C	D	E	F	T
4	S	A	B	-	D	E	F	T
5	S	A	B	C	-	E	F	T
6	S	A	B	C	D	-	F	T
7	S	A	B	C	D	E	-	T
8	S	A	B	C	D	E	F	-

Floyd-Warshall 알고리즘

❖ Floyd-Warshall 알고리즘

- 단계1
  - 거리행렬 D_0 로부터 피봇행과 피봇열로써 첫 번째 행과 열을 선택하고, 거리행렬 내의 모든 k 와 l 에 대해 삼각연산을 적용
  - t=1 이므로 $d_{k1}+d_{1l}$(단, $k\neq t\neq l$)을 계산한 후 각 요소 d_{kl} 과 비교. 이 계산 결과를 이용하여 아래와 같은 조치를 취하고, 새로운 거리행렬 D_1 과 순서행렬 S_1 을 생성
  - 거리 행렬과 순서 행렬을 동시에 변형 진행



$D_1 =$

	1	2	3	4	5	6	7	8
1	-	5	3	4	∞	∞	∞	∞
2	5	-	∞8	3	4	∞	∞	∞
3	3	∞8	-	2	∞	4	3	∞
4	4	3	[illegible]	[illegible]	[illegible]	5	3	∞
5	∞	4	∞	2	-	∞	2	4
6	∞	∞	4	5	∞	-	2	3
7	∞	∞	3	3	2	2	-	4
8	∞	∞	∞	∞	4	3	4	-

대칭으로 동일

$S_1 =$

	1	2	3	4	5	6	7	8
1	-	A	B	C	D	E	F	T
2	S	-	S	C	D	E	F	T
3	S	S	-	C	D	E	F	T
4	S	A	B	-	D	E	F	T
5	S	A	B	C	-	E	F	T
6	S	A	B	C	D	-	F	T
7	S	A	B	C	D	E	-	T
8	S	A	B	C	D	E	F	-

- 단계2

$D_2 =$

	1	2	3	4	5	6	7	8
1	-	5	3	4	∞9	∞	∞	∞
2	5	-	∞8	3	4	∞	∞	∞
3	3	∞8	-	2	∞12	4	3	∞
4	4	3	2	-	2	5	3	∞
5	∞9	4	∞12	2	-	∞	2	4
6	∞	∞	4	5	∞	-	2	3
7	∞	∞	3	3	2	2	-	4
8	∞	∞	∞	∞	4	3	4	-

$S_2 =$

	1	2	3	4	5	6	7	8
1	-	A	B	C	A	E	F	T
2	S	-	S	C	D	E	F	T
3	S	S	-	C	A	E	F	T
4	S	A	B	-	D	E	F	T
5	A	A	A	C	-	E	F	T
6	S	A	B	C	D	-	F	T
7	S	A	B	C	D	E	-	T
8	S	A	B	C	D	E	F	-

Floyd-Warshall 알고리즘

❖ Floyd-Warshall 알고리즘

- 단계3

$D_3 =$

	1	2	3	4	5	6	7	8
1	-	5	3	4	∞9	∞7	∞6	∞
2	5	-	∞8	3	4	∞12	∞11	∞
3	3	∞8	-	2	∞12	4	3	∞
4	4	3	2	-	2	5	3	∞
5	∞9	4	∞12	2	-	∞	2	4
6	∞7	∞12	4	5	∞	-	2	3
7	∞6	∞11	3	3	2	2	-	4
8	∞	∞	∞	∞	4	3	4	-

$S_3 =$

	1	2	3	4	5	6	7	8
1	-	A	B	C	A	B	B	T
2	S	-	S	C	D	B	B	T
3	S	S	-	C	A	E	F	T
4	S	A	B	-	D	E	F	T
5	A	A	A	C	-	E	F	T
6	B	B	B	C	D	-	F	T
7	B	B	B	C	D	E	-	T
8	S	A	B	C	D	E	F	-

- 단계4

$D_4 =$

	1	2	3	4	5	6	7	8
1	-	5	3	4	∞9 6	∞7	∞6	∞
2	5	-	∞8	3	4	∞12 8	∞11 6	∞
3	3	∞8	-	2	∞12 4	4	3	∞
4	4	3	2	-	2	5	3	∞
5	∞9 6	4	∞12 4	2	-	∞7	2	4
6	∞7	∞12 8	4	5	∞7	-	2	3
7	∞6	∞11 6	3	3	2	2	-	4
8	∞	∞	∞	∞	4	3	4	-

$S_4 =$

	1	2	3	4	5	6	7	8
1	-	A	B	C	C	B	B	T
2	S	-	S	C	D	C	C	T
3	S	S	-	C	C	E	F	T
4	S	A	B	-	D	E	F	T
5	C	A	C	C	-	C	F	T
6	B	C	B	C	C	-	F	T
7	B	C	B	C	D	E	-	T
8	S	A	B	C	D	E	F	-

- 단계5

$D_5 =$

	1	2	3	4	5	6	7	8
1	-	5	3	4	∞9 6	∞7	∞6	∞10
2	5	-	∞8	3	4	∞12 8	∞11 6	∞8
3	3	∞8	-	2	∞12 4	4	3	∞8
4	4	3	2	-	2	5	3	∞6
5	∞9 6	4	∞12 4	2	-	∞7	2	4
6	∞7	∞12 8	4	5	∞7	-	2	3
7	∞6	∞11 6	3	3	2	2	-	4
8	∞10	∞8	∞8	∞6	4	3	4	-

$S_5 =$

	1	2	3	4	5	6	7	8
1	-	A	B	C	C	B	B	D
2	S	-	S	C	D	C	C	D
3	S	S	-	C	C	E	F	D
4	S	A	B	-	D	E	F	D
5	C	A	C	C	-	C	F	T
6	B	C	B	C	C	-	F	T
7	B	C	B	C	D	E	-	T
8	D	D	D	D	D	E	F	-

Floyd-Warshall 알고리즘

❖ Floyd-Warshall 알고리즘

- 단계6

$D_6 =$

	1	2	3	4	5	6	7	8
1	-	5	3	4	∞9 6	∞7	∞6	∞10
2	5	-	∞8	3	4	∞12 8	∞11 6	∞8
3	3	∞8	-	2	∞12 4	4	3	∞8 7
4	4	3	2	-	2	5	3	∞6
5	∞9 6	4	∞12 4	2	-	∞7	2	4
6	∞7	∞12 8	4	5	∞7	-	2	3
7	∞6	∞11 6	3	3	2	2	-	4
8	∞10	∞8	∞8 7	∞6	4	3	4	-

$S_6 =$

	1	2	3	4	5	6	7	8
1	-	A	B	C	C	B	B	D
2	S	-	S	C	D	C	C	D
3	S	S	-	C	C	E	F	E
4	S	A	B	-	D	E	F	D
5	C	A	C	C	-	C	F	T
6	B	C	B	C	C	-	F	T
7	B	C	B	C	D	E	-	T
8	D	D	E	D	D	E	F	-

- 단계7

$D_7 =$

	1	2	3	4	5	6	7	8
1	-	5	3	4	∞9 6	∞7	∞6	∞10
2	5	-	∞8	3	4	∞12 8	∞11 6	∞8
3	3	∞8	-	2	∞12 4	4	3	∞8 7
4	4	3	2	-	2	5	3	∞6
5	∞9 6	4	∞12 4	2	-	∞7	2	4
6	∞7	∞12 8	4	5	∞7	-	2	3
7	∞6	∞11 6	3	3	2	2	-	4
8	∞10	∞8	∞8 7	∞6	4	3	4	-

$S_7 =$

	1	2	3	4	5	6	7	8
1	-	A	B	C	C	B	B	D
2	S	-	S	C	D	C	C	D
3	S	S	-	C	C	E	F	E
4	S	A	B	-	D	E	F	D
5	C	A	C	C	-	C	F	T
6	B	C	B	C	C	-	F	T
7	B	C	B	C	D	E	-	T
8	D	D	E	D	D	E	F	-

- 단계8

$D_8 =$

	1	2	3	4	5	6	7	8
1	-	5	3	4	∞9 6	∞7	∞6	∞10
2	5	-	∞8	3	4	∞12 8	∞11 6	∞8
3	3	∞8	-	2	∞12 4	4	3	∞8 7
4	4	3	2	-	2	5	3	∞6
5	∞9 6	4	∞12 4	2	-	∞7	2	4
6	∞7	∞12 8	4	5	∞7	-	2	3
7	∞6	∞11 6	3	3	2	2	-	4
8	∞10	∞8	∞8 7	∞6	4	3	4	-

$S_8 =$

	1	2	3	4	5	6	7	8
1	-	A	B	C	C	B	B	D
2	S	-	S	C	D	C	C	D
3	S	S	-	C	C	E	F	E
4	S	A	B	-	D	E	F	D
5	C	A	C	C	-	C	F	T
6	B	C	B	C	C	-	F	T
7	B	C	B	C	D	E	-	T
8	D	D	E	D	D	E	F	-