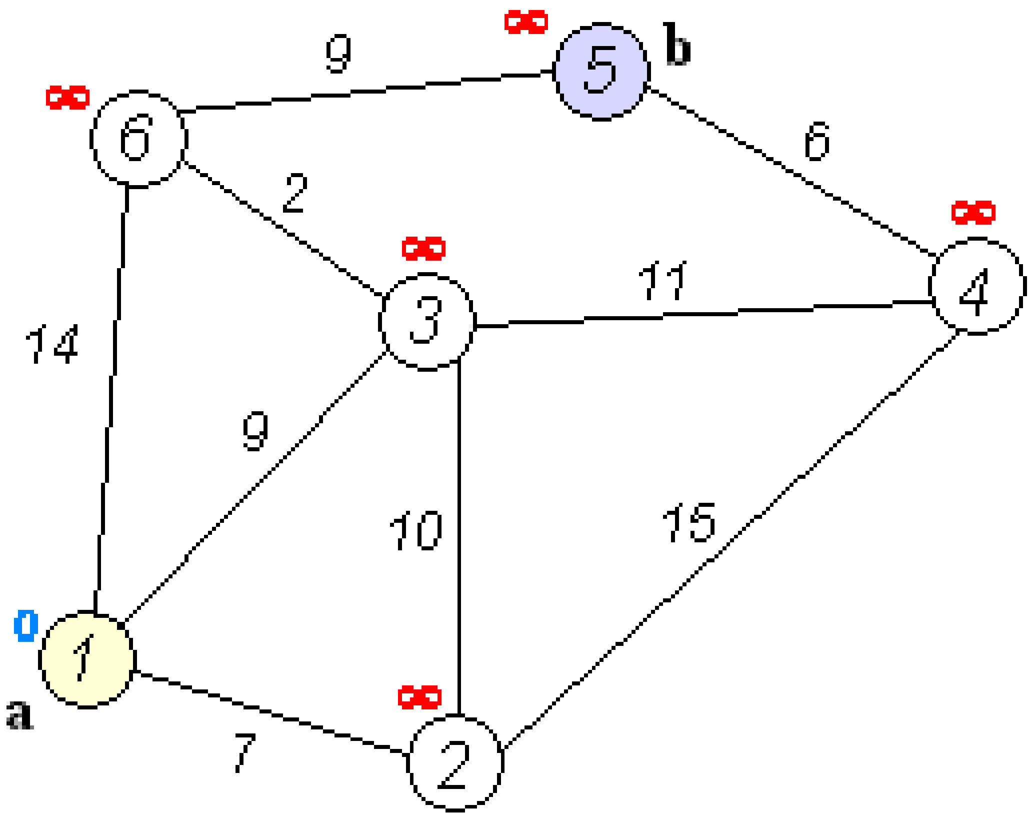
Shortest Path First Algorithm

충남대학교 컴퓨터공학과
이영석

최단거리 문제

- 문제
  - single source to all destination
  - single source to single destination
  - all source to all destination
  - 링크 가중치: 양, 음
- 알고리즘
  - Dijkstra: 링크의 가중치가 양의 값
  - Bellman-ford: 링크의 가중치가 음의 값에도 됨
  - Floyd-warshall: all source to all destination
  - A*: speed
- Sing source to all destination
  - Shortest-path tree
  - O(E + V log V)



Dijsktra's Algorithm

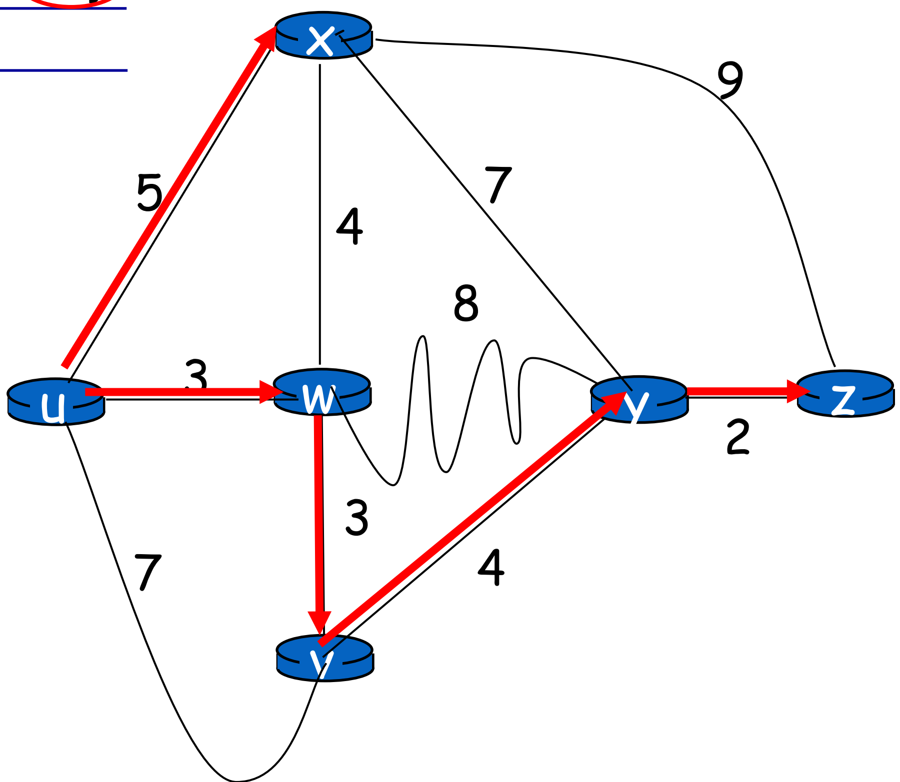
```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
        then D(v) = c(u,v)
    else D(v) = ∞

Loop
   find w not in N' such that D(w) is a minimum
    add w to N'
    update D(v) for all v adjacent to w and not in N' :
        D(v) = min( D(v), D(w) + c(w,v) )
    /* new cost to v is either old cost to v or known
     shortest path cost to w plus cost from w to v */
until all nodes in N'
```

Dijkstra's algorithm: example

Step	N'	D(v) p(v)	D(w) p(w)	D(x) p(x)	D(y) p(y)	D(z) p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

Notes:

- construct shortest path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)



```
function Dijkstra(Graph, source):
    dist[source] ← 0                                  // Initialization

    create vertex priority queue Q

    for each vertex v in Graph:
        if v ≠ source
            dist[v] ← INFINITY                        // Unknown distance from source to v
            prev[v] ← UNDEFINED                       // Predecessor of v

        Q.add_with_priority(v, dist[v])


    while Q is not empty:                             // The main loop
        u ← Q.extract_min()                           // Remove and return best vertex
        for each neighbor v of u:                     // only v that are still in Q
            alt ← dist[u] + length(u, v)
            if alt < dist[v]
                dist[v] ← alt
                prev[v] ← u
                Q.decrease_priority(v, alt)

    return dist, prev
```

$$O(|E| + |V| \log |V|)$$

경로 찾기 (source -> target (u))

```
S ← empty sequence
u ← target
if prev[u] is defined or u = source:          // Do something only if the vertex is reachable
    while u is defined:                       // Construct the shortest path with a stack S
        insert u at the beginning of S        // Push the vertex onto the stack
        u ← prev[u]                           // Traverse from target to source
```

Bellman-Ford Algorithm

- 음의 가중치 링크
- 순서

```
for v in V:
    v.distance = infinity
    v.p = None
source.distance = 0
for i from 1 to |V| - 1:
    for (u, v) in E:
        relax(u, v)
```

```
relax(u, v):
    if v.distance > u.distance + weight(u, v):
        v.distance = u.distance + weight(u, v)
        v.p = u
```

$O(|V| \cdot |E|)$

Bellman-Ford Algorithm

Bellman-Ford Equation (dynamic programming)

Define

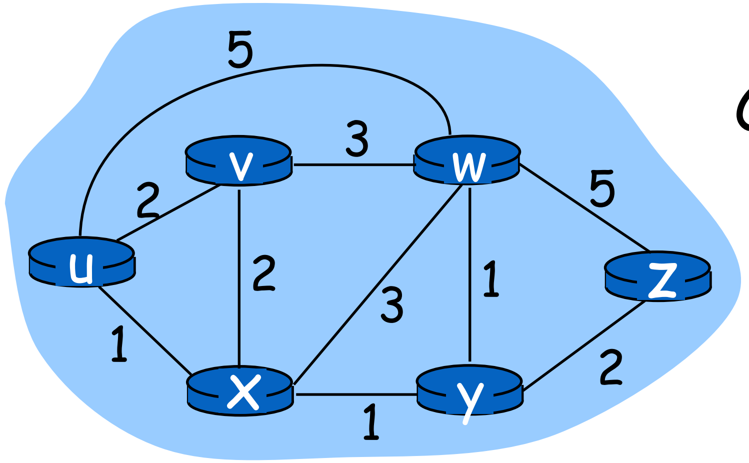
$d_x(y)$:= cost of least-cost path from x to y

Then

$$\boxed{d_x(y) = \min_v \{c(x,v) + d_v(y)\}}$$

where min is taken over all neighbors v of x

Bellman-Ford example



Clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

$$
\begin{aligned}
d_u(z) &= \min \{\, c(u,v) + d_v(z),\\
&\qquad\quad c(u,x) + d_x(z),\\
&\qquad\quad c(u,w) + d_w(z) \,\}\\
&= \min \{2 + 5,\\
&\qquad\quad 1 + 3,\\
&\qquad\quad 5 + 3\} = 4
\end{aligned}
$$

Node that achieves minimum is next hop in shortest path ➔ forwarding table

$$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\} = \min\{2+0\ ,\ 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\} = \min\{2+1\ ,\ 7+0\} = 3$$

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

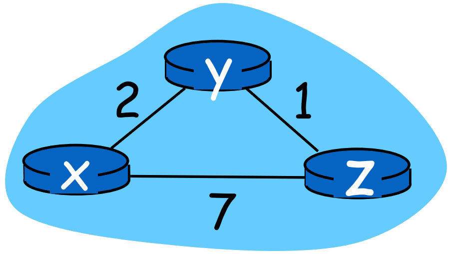
node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

time



$$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\} = \min\{2+0\ ,\ 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\} = \min\{2+1\ ,\ 7+0\} = 3$$

node x table

cost to

from	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

cost to

from	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

cost to

from	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

cost to

from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

cost to

from	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

cost to

from	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node z table

cost to

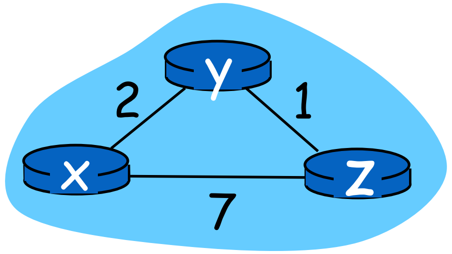
from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

cost to

from	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

cost to

from	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

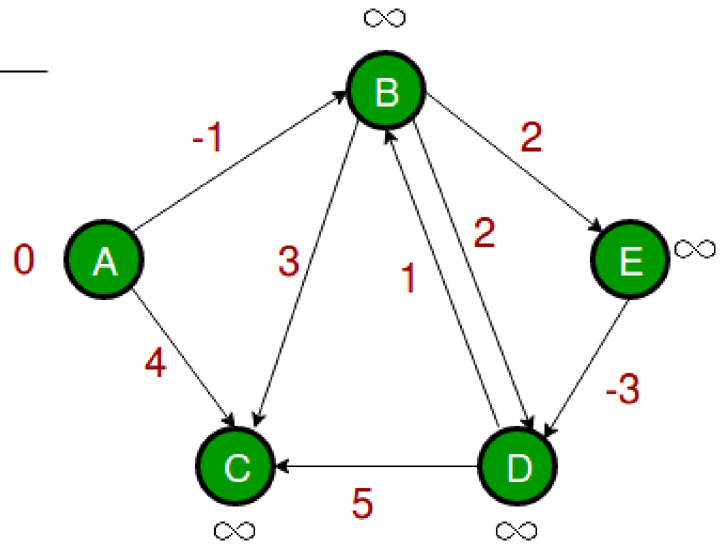


time

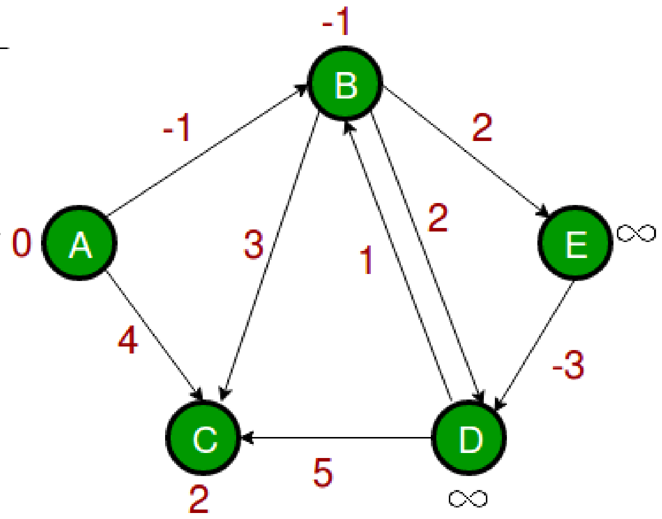
Bellman-Ford 다른 예

- 링크 처리 순서: (B, E), (D, B), (B, D), (A, B), (A, C), (D, C), (B, C), (E, D)
    - (A,B), (A, C), (B, C)

A	B	C	D	E
0	∞	∞	∞	∞



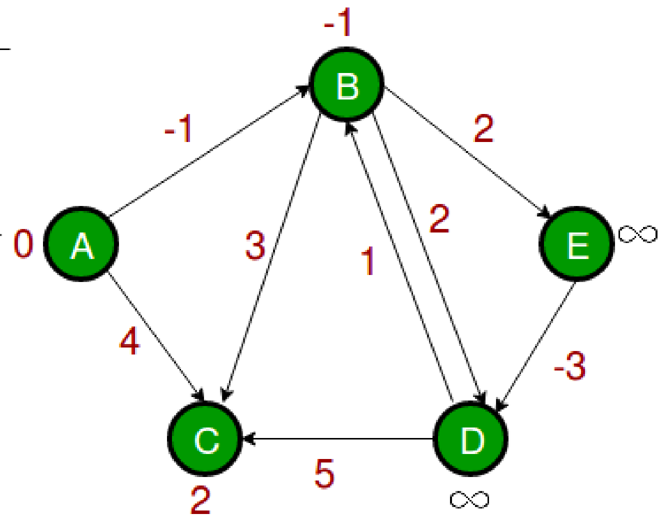
A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞



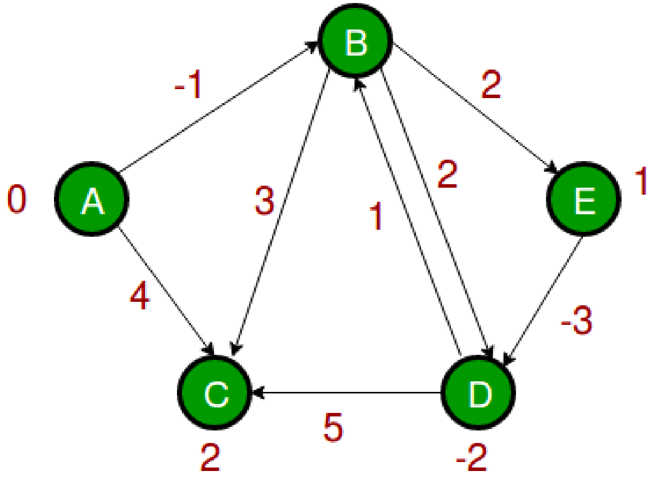
https://www.geeksforgeeks.org/bellman-ford-algorithm-dp-23/

- 링크 처리 순서: (B, E), (D, B), (B, D), (A, B), (A, C), (D, C), (B, C), (E, D)
    - (B, E), (B, D), (E, D)

A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞



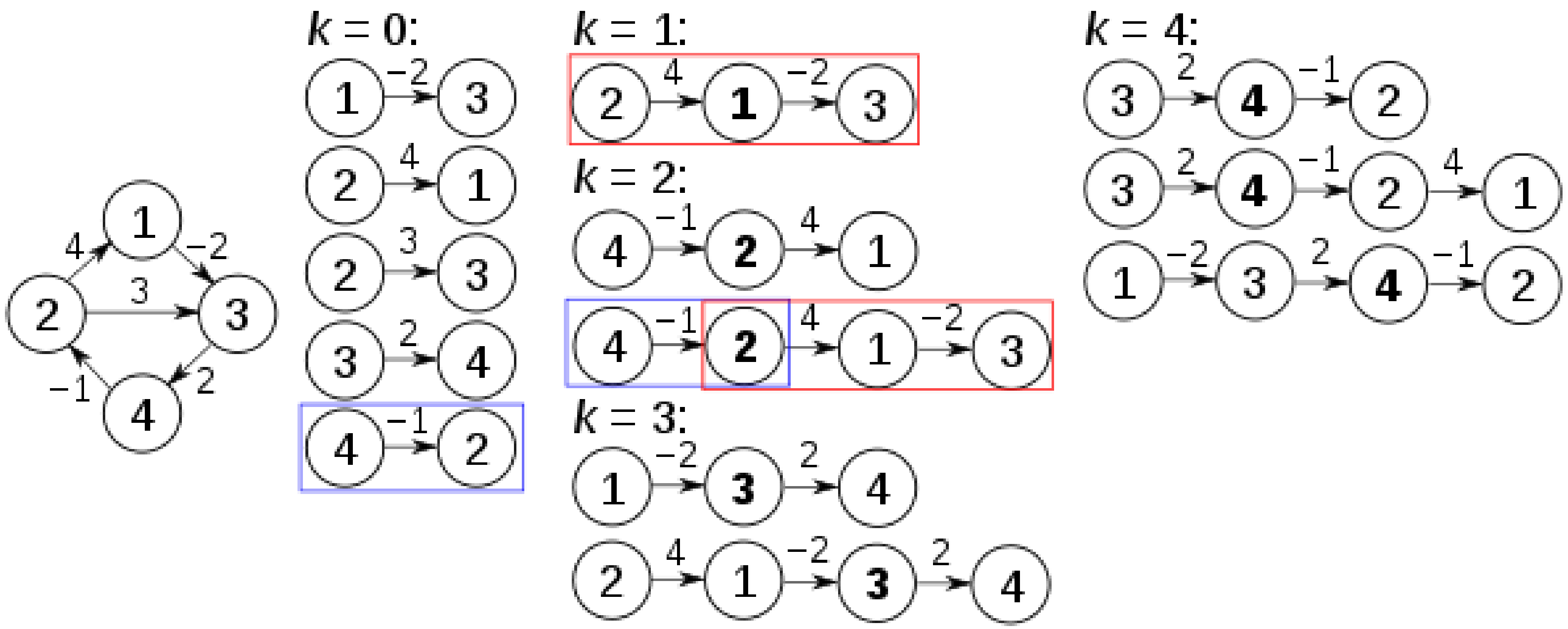
A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞
0	-1	2	∞	1
0	-1	2	1	1
0	-1	2	-2	1



Floyd-Warshall Algorithm

```
let dist be a |V| × |V| array of minimum distances initialized to ∞ (infinity)
for each edge (u,v)
   dist[u][v] ← w(u,v)  // 변 (u,v)의 가중치
for each vertex v
   dist[v][v] ← 0
for k from 1 to |V|
   for i from 1 to |V|
      for j from 1 to |V|
         if dist[i][j] > dist[i][k] + dist[k][j]
             dist[i][j] ← dist[i][k] + dist[k][j]
         end if
```

$\Theta(|V|^3)$



k = 0	j			
i	**1**	**2**	**3**	**4**
1	0	∞	-2	∞
2	4	0	3	∞
3	∞	∞	0	2
4	∞	-1	∞	0

k = 1	j			
i	**1**	**2**	**3**	**4**
1	0	∞	-2	∞
2	4	0	2	∞
3	∞	∞	0	2
4	∞	-1	∞	0

k = 2	j			
i	**1**	**2**	**3**	**4**
1	0	∞	-2	∞
2	4	0	2	∞
3	∞	∞	0	2
4	3	-1	1	0

k = 3	j			
i	**1**	**2**	**3**	**4**
1	0	∞	-2	0
2	4	0	2	4
3	∞	∞	0	2
4	3	-1	1	0

k = 4	j			
i	**1**	**2**	**3**	**4**
1	0	-1	-2	0
2	4	0	2	4
3	5	1	0	2
4	3	-1	1	0

최단경로 응용

- 라우터 SW
  - Dijkstra -> OSPF https://en.wikipedia.org/wiki/Open_Shortest_Path_First
  - Bellman-Ford -> RIP https://en.wikipedia.org/wiki/Routing_Information_Protocol