4.8 Single source shortest path

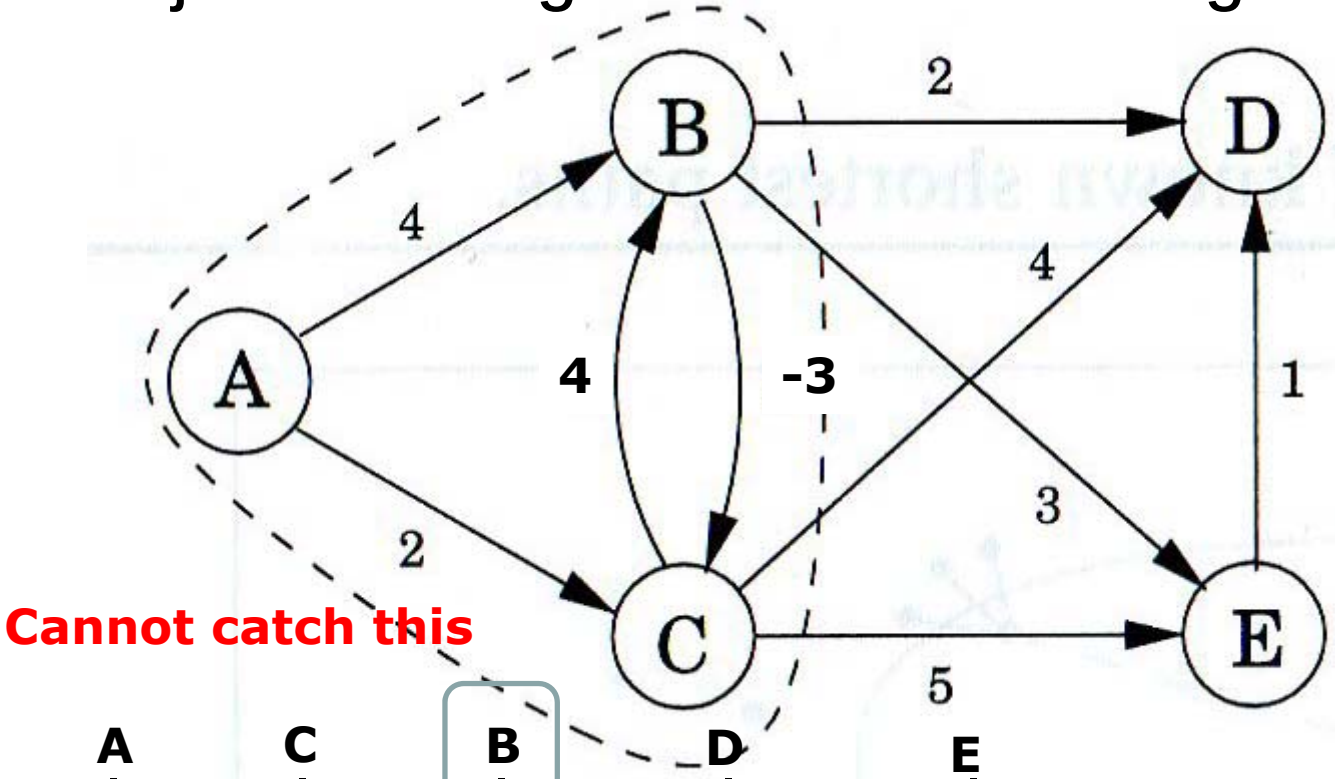
(4) Dijkstra's algorithm (2)

```
procedure Dijkstra ( G, s )

for all u ∈ V
    dist[u] = ∞;
    prev[u] = NULL;

dist[s] = 0;

H = makequeue ( V );
while ( H is not empty )
    u = get_min ( H );
    for all edges (u, v) ∈ E and v ∈ H
        if ( dist[v] > dist[u] + l(u, v) )
            dist[v] = dist[u] + l(u, v);
            prev[v] = u;
            modify_H ( H, v );
```

4.8 Single source shortest path

(4) Dijkstra's algorithm (6)

- Dijkstra's algorithm with a negative edge?



4.8 Single source shortest path

(5) Bellman-Ford's algorithm (1)

- A graph with a negative edge
- Execute edge relaxation for every edge for every path (from length 1 to n – 1)

```
for ( i = 1; i < n; i++ ) {
        for ( (u, v) in E ) {
                if ( dist[v] > dist[u] + w(u, v) )
                        dist[v] = dist[u] + w(u, v);
                        prev[v] = u;
        }
}
```

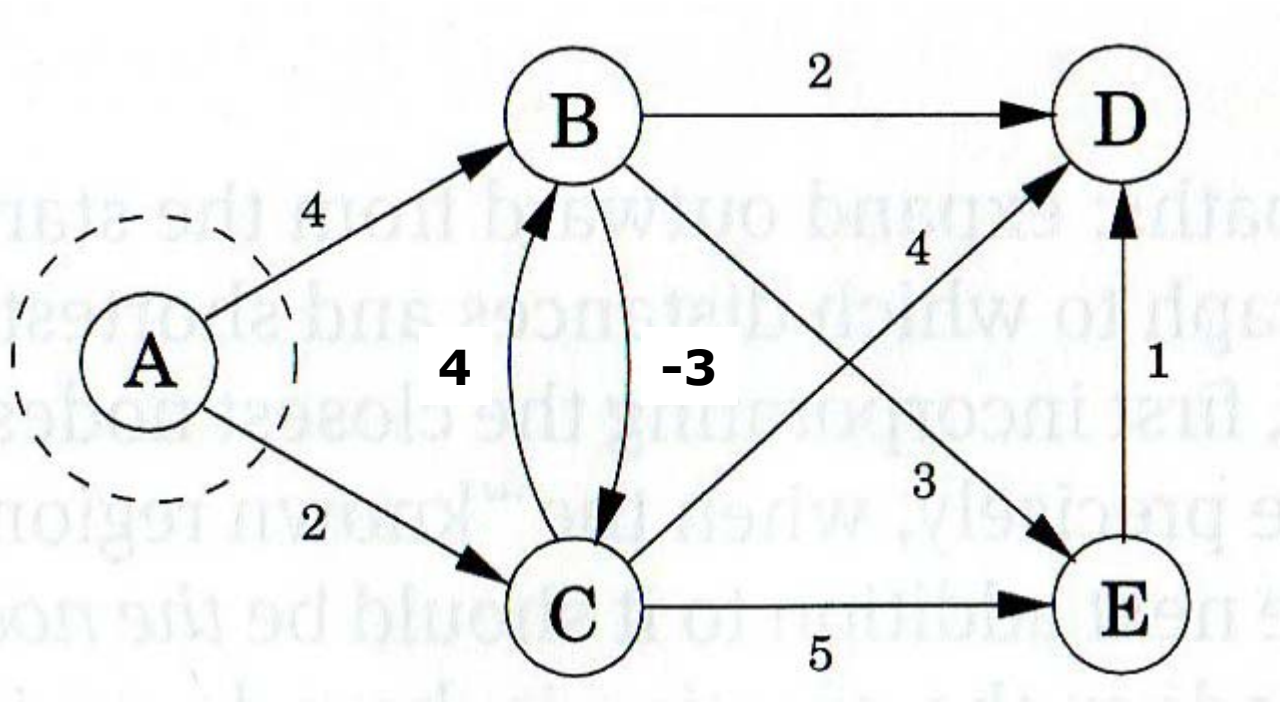
4.8 Single source shortest path

(5) Bellman-Ford's algorithm (2)

- How it works

  (0) Initially, dist[A] = 0 and all other vertices set ∞

	dist	prev
A		
B		
C		
D		
E		

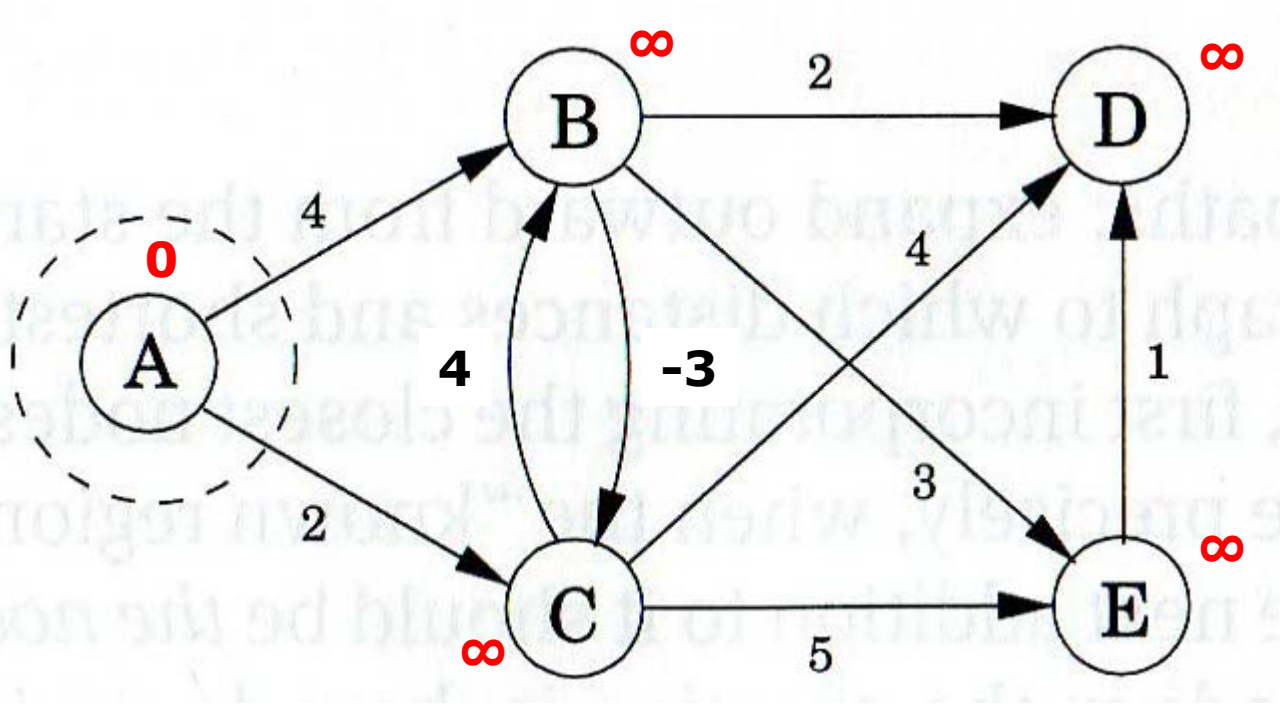


4.8 Single source shortest path

(5) Bellman-Ford's algorithm (2)

- How it works

  (0) Initially, dist[A] = 0 and all other vertices set ∞



	dist	prev
A	0	-
B	∞	N
C	∞	N
D	∞	N
E	∞	N

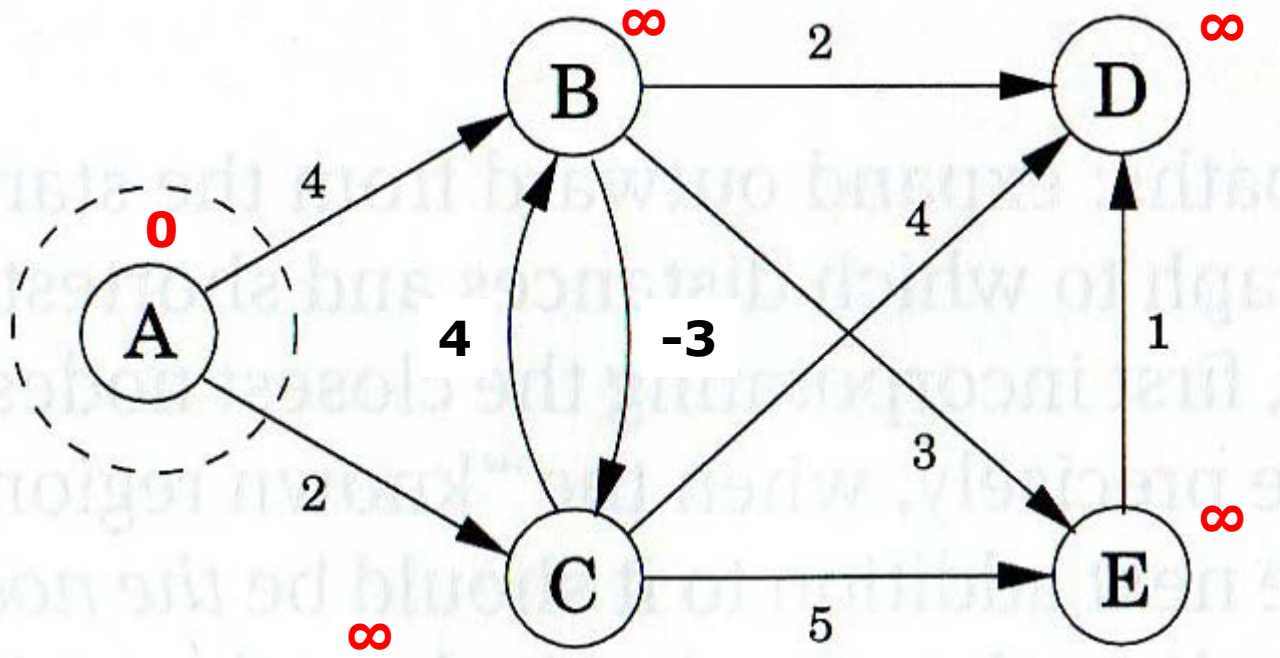
4.8 Single source shortest path

(5) Bellman-Ford's algorithm (2)

- How it works

  (1) Relaxation for all edges (i = A)

```
if ( dist[v] > dist[u] + w(u, v) )
    dist[v] = dist[u] + w(u, v);
```



dist prev

A	0	-
B	∞	N
C	∞	N
D	∞	N
E	∞	N

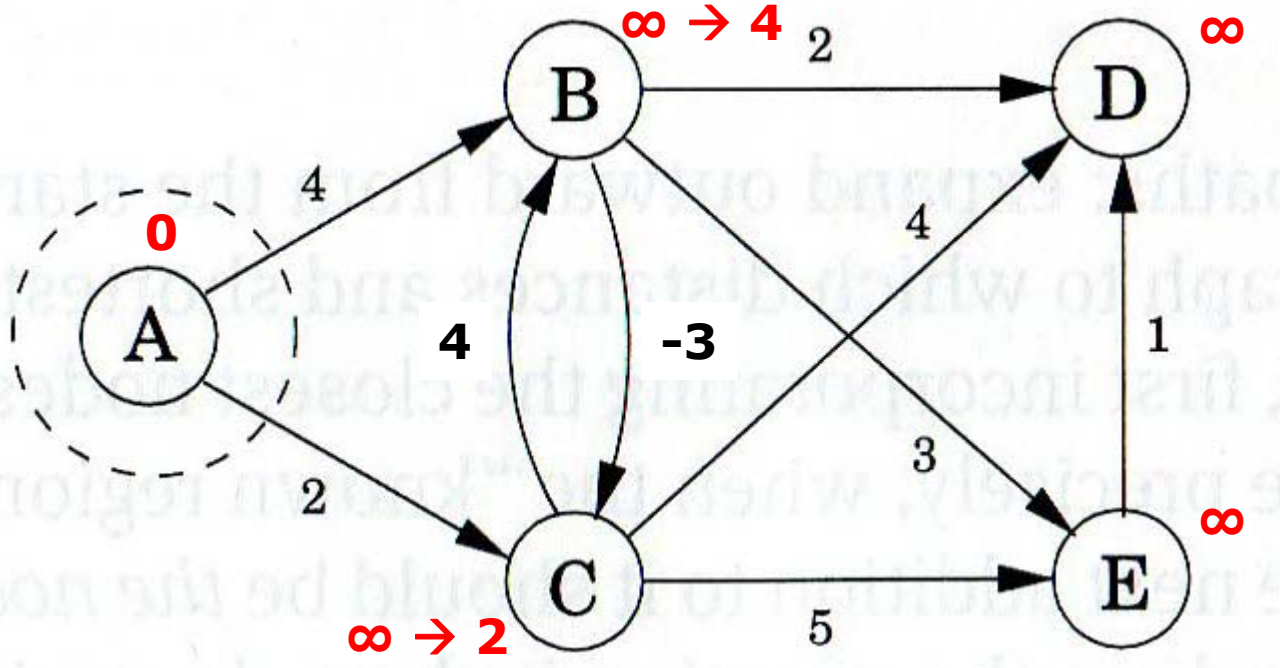
4.8 Single source shortest path

(5) Bellman-Ford's algorithm (2)

- How it works

  (1) Relaxation for all edges (i = A)

```
if ( dist[v] > dist[u] + w(u, v) )
    dist[v] = dist[u] + w(u, v);
```



	dist	prev
A	0	-
B	∞	4
C	∞	2
D	∞	N
E	∞	N

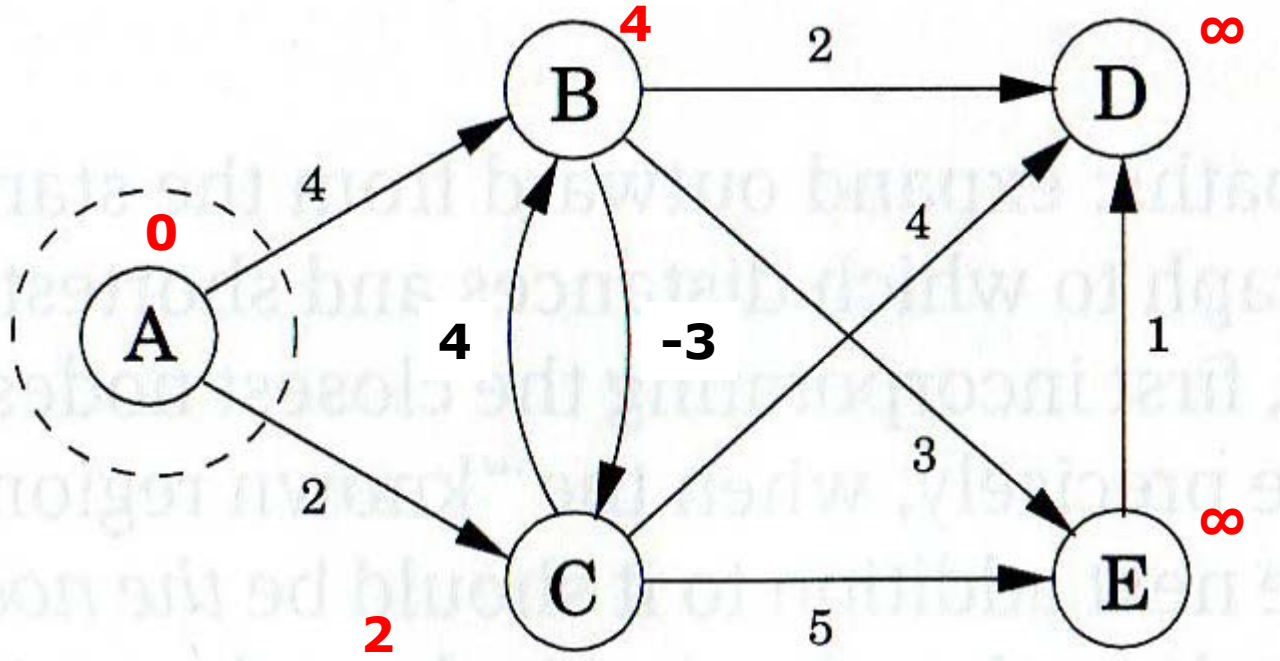
4.8 Single source shortest path

(5) Bellman-Ford's algorithm (2)

- How it works

  (1) Relaxation for all edges (i = B)

```
if ( dist[v] > dist[u] + w(u, v) )
    dist[v] = dist[u] + w(u, v);
```



dist	prev
A	0	-
B	4	A
C	2	A
D	∞	N
E	∞	N

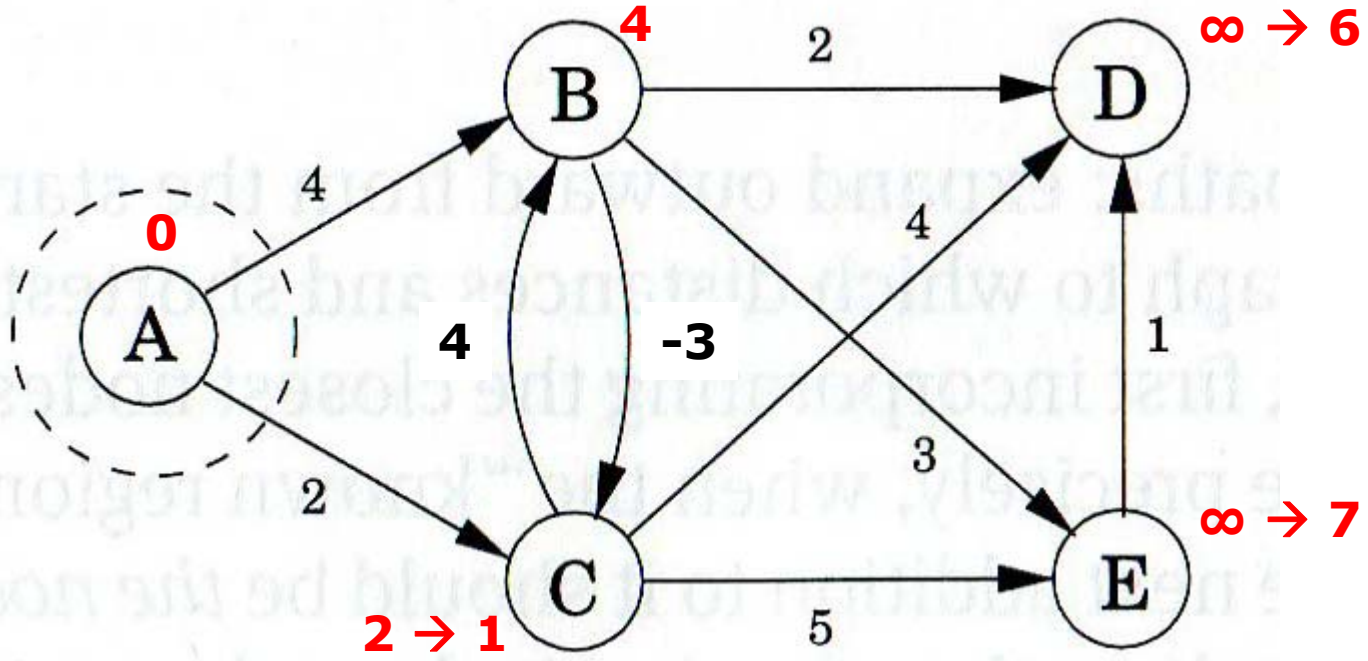
4.8 Single source shortest path

(5) Bellman-Ford's algorithm (2)

- How it works

  (1) Relaxation for all edges (i = B)

```
if ( dist[v] > dist[u] + w(u, v) )
    dist[v] = dist[u] + w(u, v);
```



	dist	prev
A	0	-
B	4	A
C	1	B
D	6	B
E	7	B

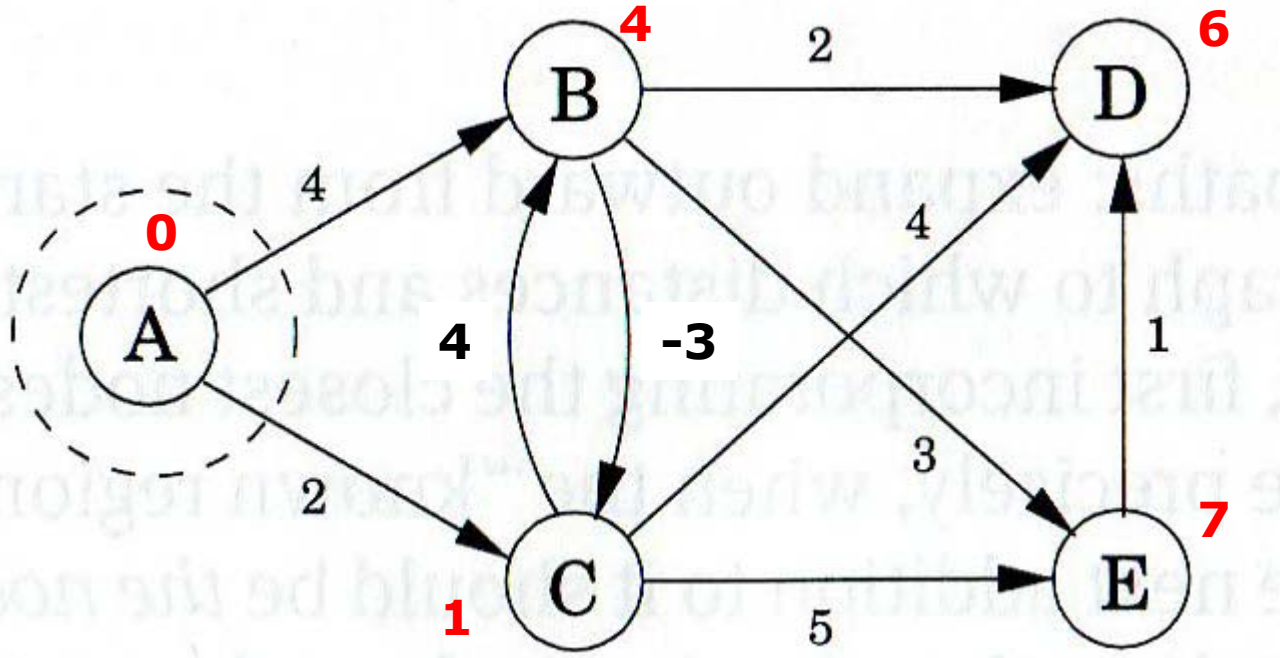
4.8 Single source shortest path

(5) Bellman-Ford's algorithm (2)

- How it works

  (1) Relaxation for all edges (i = C)

```
if ( dist[v] > dist[u] + w(u, v) )
    dist[v] = dist[u] + w(u, v);
```



dist prev

A	0	-
B	4	A
C	1	B
D	6	B
E	7	B

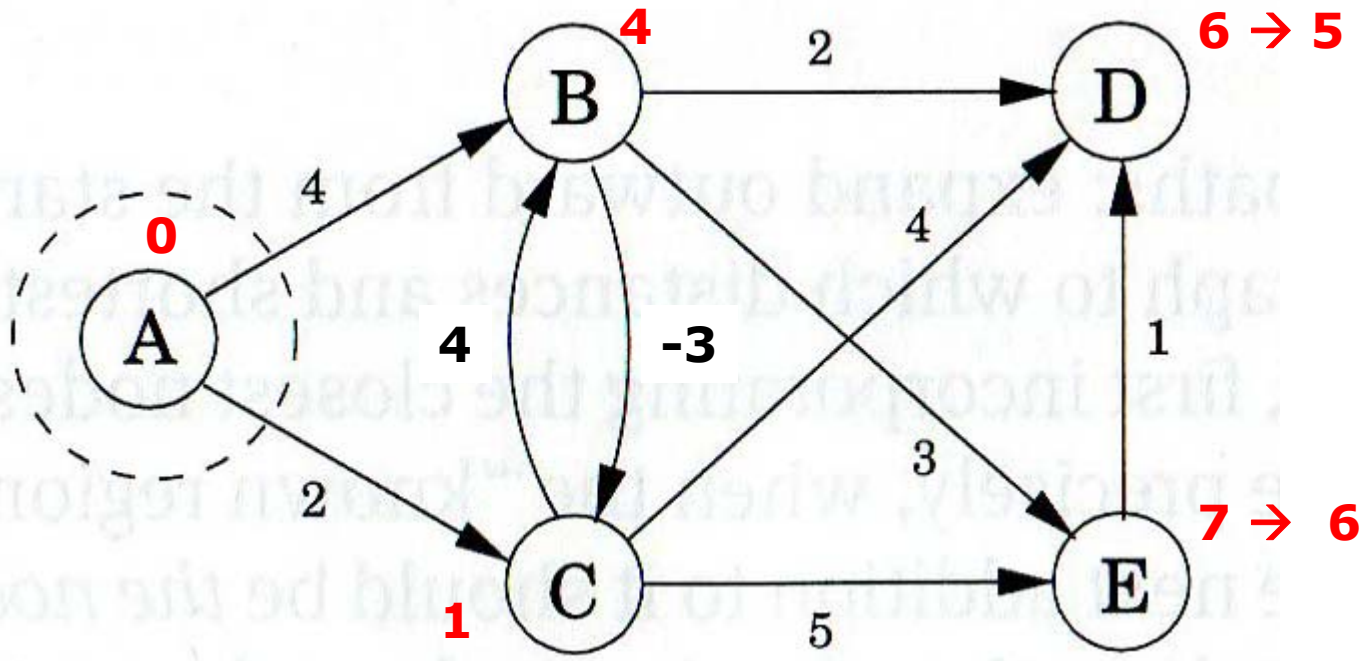
4.8 Single source shortest path

(5) Bellman-Ford's algorithm (2)

- How it works

  (1) Relaxation for all edges (i = C)

```
if ( dist[v] > dist[u] + w(u, v) )
    dist[v] = dist[u] + w(u, v);
```



dist	prev
A	0	-
B	4	A
C	1	B
D	5	C
E	6	C

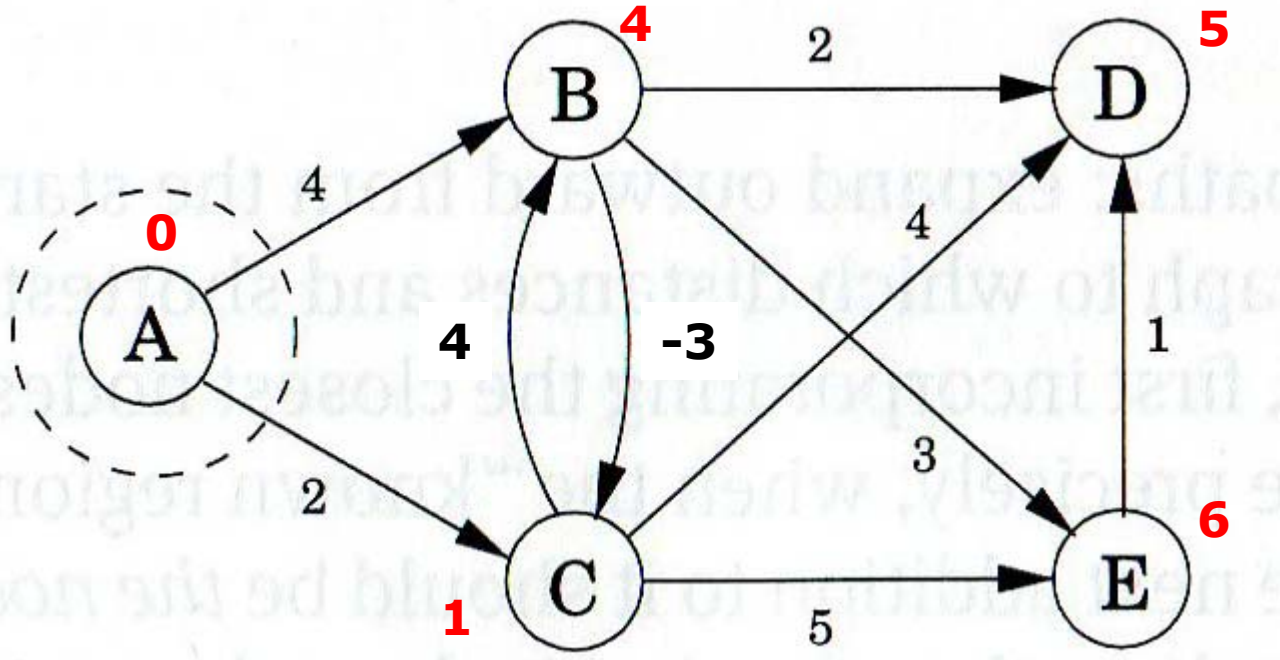
4.8 Single source shortest path

(5) Bellman-Ford's algorithm (2)

- How it works

  (1) Relaxation for all edges (i = D)

```
if ( dist[v] > dist[u] + w(u, v) )
    dist[v] = dist[u] + w(u, v);
```



dist prev

A	0	-
B	4	A
C	1	B
D	5	C
E	6	C

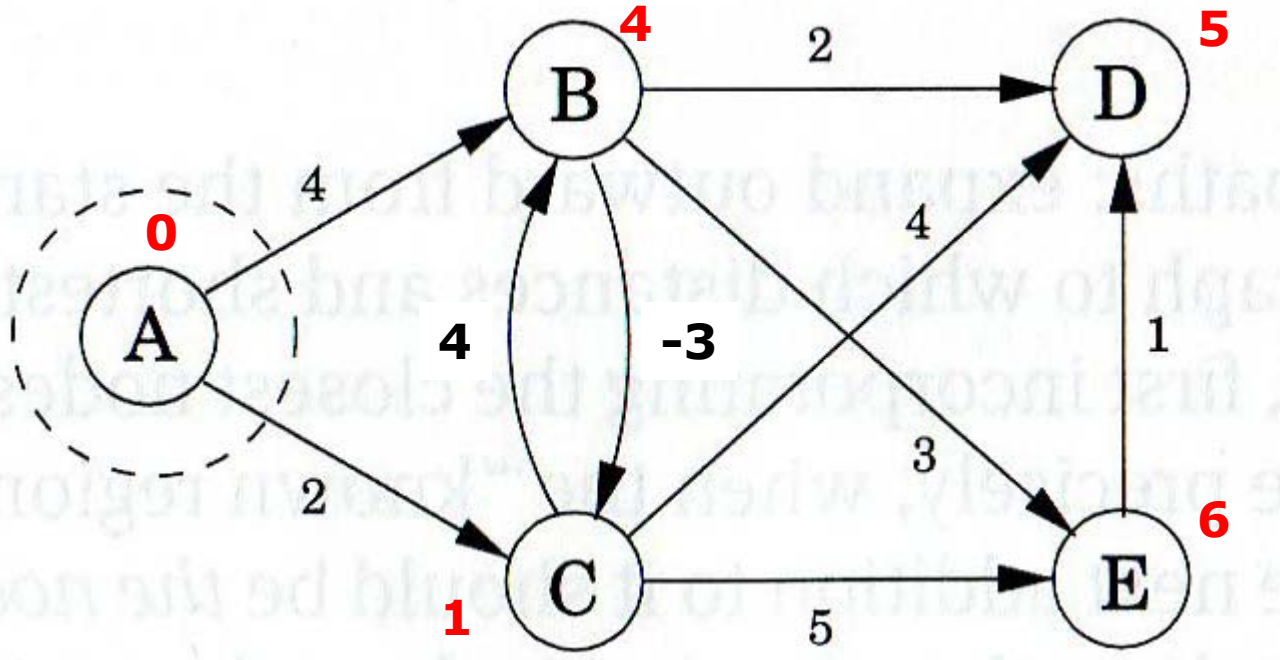
4.8 Single source shortest path

(5) Bellman-Ford's algorithm (2)

- How it works

  (1) Relaxation for all edges (i = E)

```
if ( dist[v] > dist[u] + w(u, v) )
    dist[v] = dist[u] + w(u, v);
```



dist	prev
A	0	-
B	4	A
C	1	B
D	5	C
E	6	C

4.8 Single source shortest path

(5) Bellman-Ford's algorithm (3)

- A graph with a negative edge
- Execute edge relaxation for every edge for every path (from length 1 to n – 1)

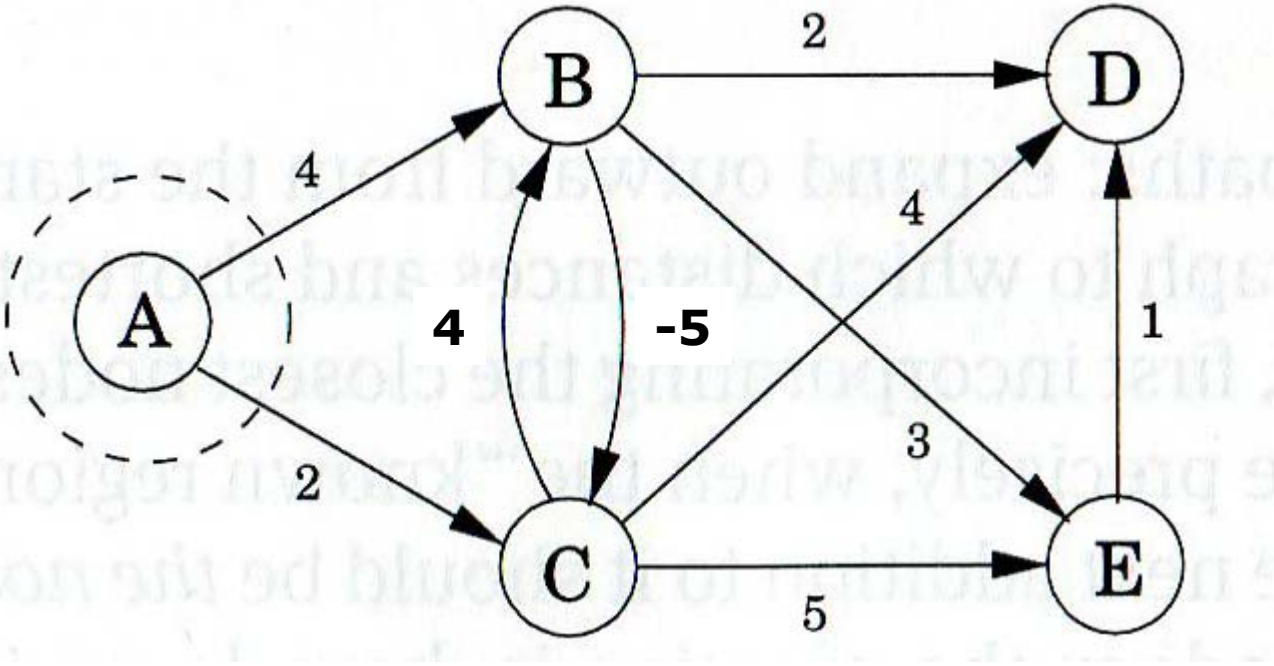
- Time complexity: O(n * m)

```
for ( i = 1; i < n; i++ ) {
       for ( (u, v) in E ) {
              if ( dist[v] > dist[u] + w(u, v) )
                     dist[v] = dist[u] + w(u, v);
                     prev[v] = u;
       }
}
```

4.8 Single source shortest path

(5) Bellman-Ford's algorithm (4)

- A graph with a negative cycle?
  - Bellman-Ford algorithm can solve this case?
  - A minimum cost path in a negative cycle exists?



All about graph

Type	Purpose	Operations	Performance
DFS	Traverse all vertices	Visiting all vertices & visiting all edges	O(n) + O(m)
SCC	Finding SCC	DFS on G^R and G	O(DFS)
BFS	Traverse all vertices	Visiting all vertices & visiting all edges	O(n) + O(m)
Dijkstra	Single source shortest path	Visiting all edges & managing queue	$O(n^2)$ (original) → O(m) + O(n log n)
Floyd			
Kruskal (Greedy)			
Prim (Greedy)			
MultiStage (Dynamic)			

4.8 Single source shortest path

- 다음은 Bellman-Ford algorithm에 대한 설명이다. 잘못된 것을 모두 고르시오.

(a) Bellman-Ford algorithm은 negative edge가 있는 graph에 대해서도 최단 거리를 계산할 수 있다.

(b) Bellman-Ford algorithm은 negative cycle이 있는 graph에 대해서도 최단 거리를 계산할 수 있다.

(c) Bellman-Ford algorithm의 시간 복잡도는 Dijkstra algorithm의 시간 복잡도와 같다.

(d) Sparse graph에 대한 Bellman-Ford algorithm의 시간 복잡도는 O(n^2)이다.