SEC 알고리즘 특강

Shortest Path
[최단경로알고리즘]

㈜한컴에듀케이션 이주현

수학은 패턴의 과학이다. 음악 역시 패턴들이다.
컴퓨터 과학은 추상화와 패턴의 형성에 깊은 관련이 있다.
컴퓨터 과학이 다른 분야들에 비해 특징적인 것은
지속적으로 차원이 급상승한다는 점이다.
미시적 관점에서 거시적 관점으로 도약하는 것이다.

-도널드 크누스

Shortest Paths

- 조건
  - 간선 가중치가 있는 유향 그래프
  - 무향 그래프는 각 간선에 대해 양쪽으로 유향 간선이 있는 유향 그래프로 생각할 수 있다
    - 즉, 무향 간선 (u,v)는 유향 간선 (u,v)와 (v,u)를 의미한다고 가정하면 된다
- 두 정점 사이의 최단경로
  - 두 정점 사이의 경로들 중 간선의 가중치 합이 최소인 경로
  - 간선 가중치의 합이 음인 싸이클이 있으면 문제가 정의되지 않는다

- 모든 쌍 최단경로
  - 모든 정점 쌍 사이의 최단경로를 모두 구한다
  - ➢ 플로이드-워샬 알고리즘

- 단일 시작점 최단경로
  - 단일 시작점으로부터 각 정점에 이르는 최단경로를 구한다
  - ➢ 다익스트라 알고리즘
    - 음의 가중치를 허용하지 않는 최단경로
  - ➢ 벨만-포드 알고리즘
    - 음의 가중치를 허용하는 최단경로
    - 싸이클이 없는 그래프의 최단경로

Floyd-Warshall Algorithm

- 모든 정점들 간의 상호 최단거리 구하기
- 응용 예
  - Road Atlas
  - 네비게이션 시스템
  - 네트웍 커뮤니케이션

Floyd-Warshall Algorithm – cont.

```
FloydWarshall(G)
{
    for i ← 1 to n
        for j ← 1 to n
            d^0_ij ← w_ij ;
    for k ← 1 to n                    ▷ 중간정점 집합 {1, 2, …, k}
        for i ← 1 to n                ▷ i : 시작 정점
            for j ← 1 to n            ▷ j : 마지막 정점
                d^k_ij ← min {d^(k-1)_ij , d^(k-1)_ik + d^(k-1)_kj};
}
```

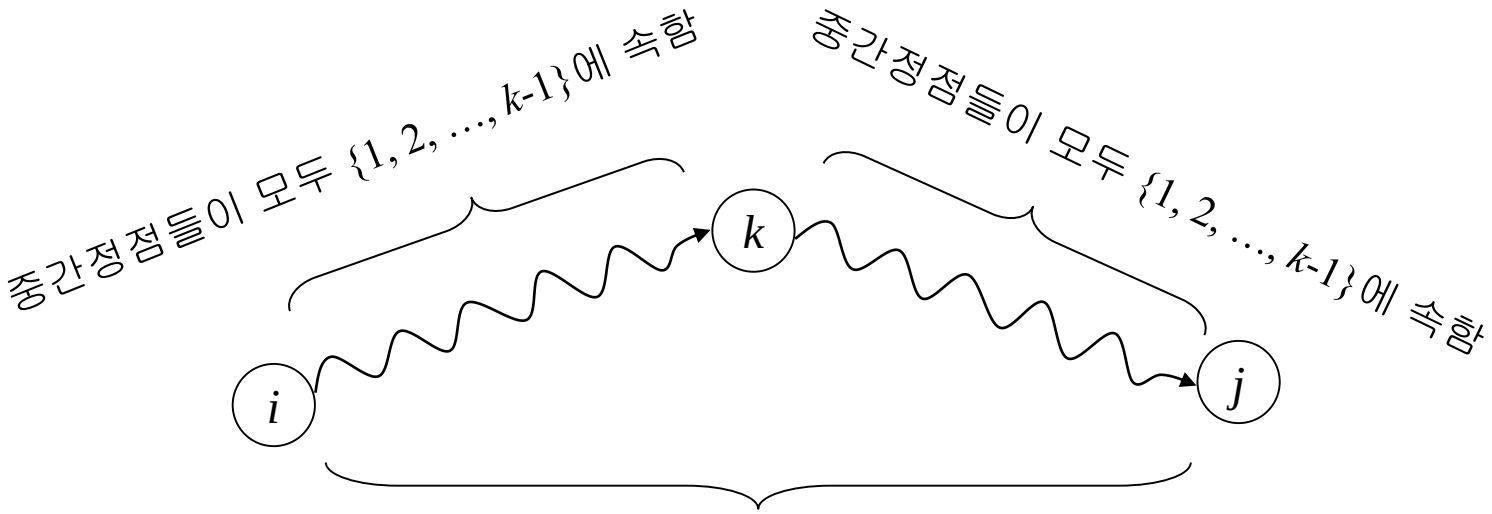
✓ d^k_{ij} : 중간 정점으로 정점 집합 $\{1, 2, \ldots, k\}$만을 사용하여 정점 i에서 정점 j에 이르는 최단경로

✓수행시간: $\Theta(|V|^3)$

Floyd-Warshall Sample Code

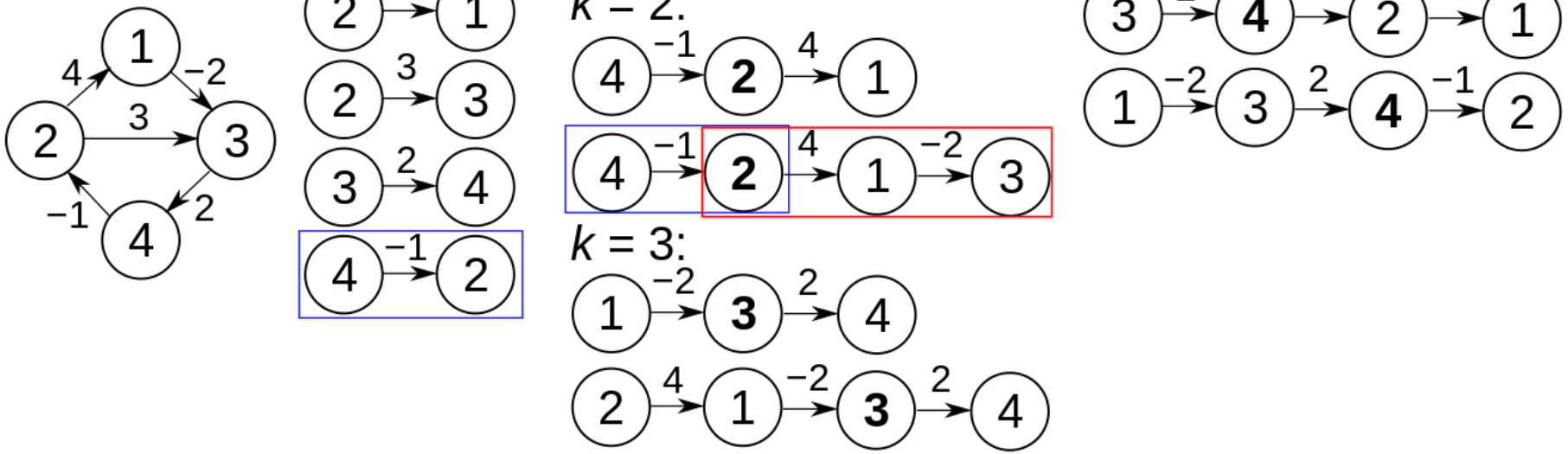
```
int adjMat [LM][LM];   /// 인접 행렬(비용, 거리 등등)
int path[LM][LM];  /// 경로 행렬
void  FloydWarshall(){
      for (int k =1; k<=n;k++){              /// k : 중간정점
            for (int i =1; i<=n; i++){       /// i : 시작 정점
                  for (int j=1; j<=n; j++){  /// j : 마지막 정점
                        if(adjMat[i][j]>adjMat [i][k]+adjMat [k][j]){
                              adjMat [i][j]=adjMat [i][k]+adjMat [k][j];
                              path[i][j] = k;
                        }
                  }
            }
      }
}
```

d^k_{ij} 관련



중간정점들이 모두 $\{1, 2, \ldots, k\}$에 속함

Example: Floyd-Warshall Algorithm



Dijkstra Algorithm

모든 간선의 가중치는 음이 아니어야 함

```
Dijkstra(G, r)
▷ G=(V, E): 주어진 그래프
▷ r: 시작으로 삼을 정점
{
    S ← Φ ;                  ▷ S : 정점 집합 : 초기 공집합
    for each u∈V             ▷ 모든 정점의 초기 거리 값을 ∞ 로 한다.
        d_u ← ∞ ;
    d_r ← 0 ;                ▷ 시작 정점의 거리 값을 0으로 한다.
    while (S≠V){             ▷ n회 순환된다
        u ← extractMin(V-S, d) ;
        S ← S ∪{u};
        for each v∈L(u) ▷ L(u) : u로부터 연결된 정점들의 집합
            if (v∈V-S and d_v<d_u+w_u,v) then d_v ← d_u+w_u,v ;
    }
}
extractMin(Q, d)
{
    집합 Q에서 d값이 가장 작은 정점 u를 리턴한다 ;
}
```

이완(relaxation)

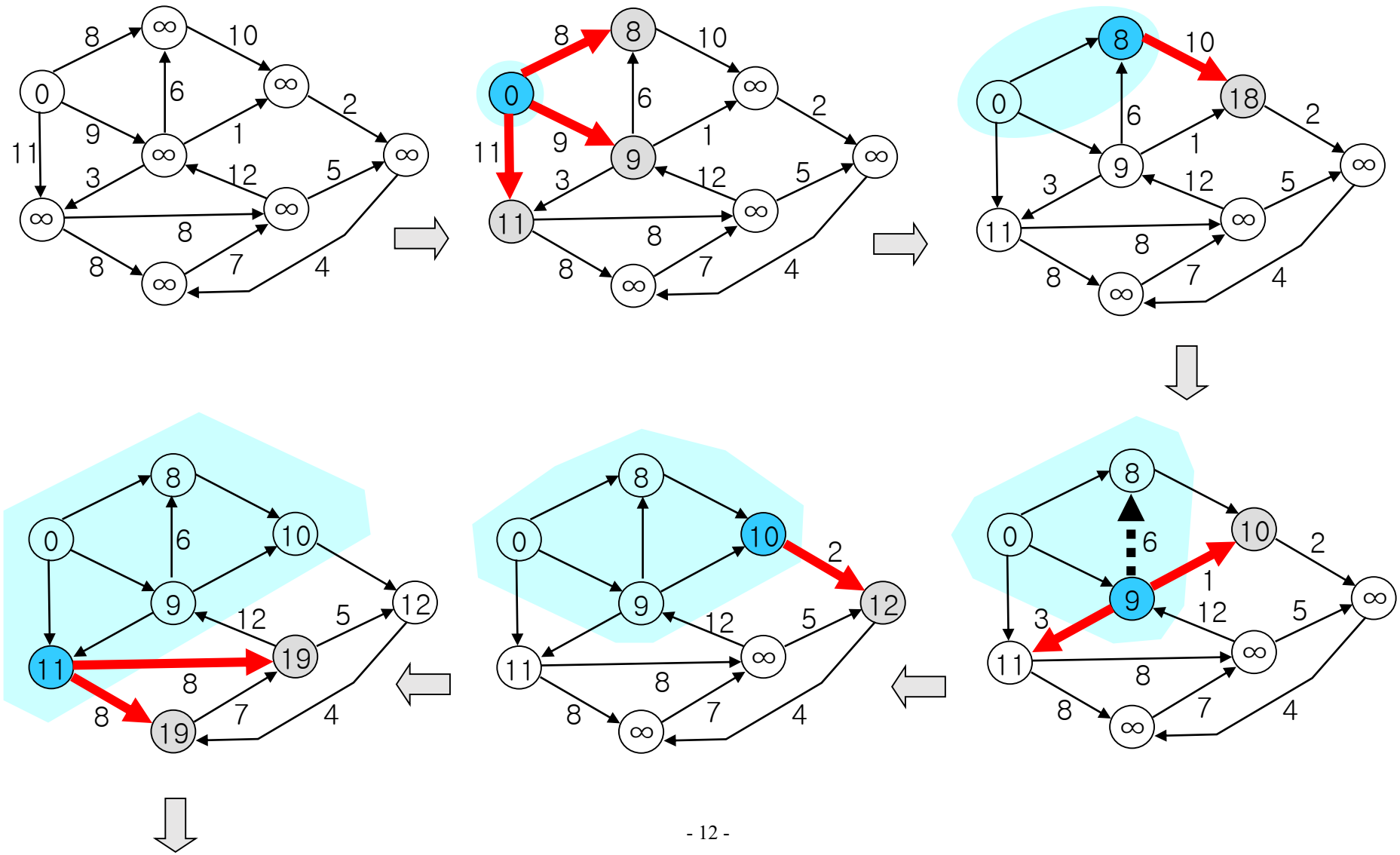
✓수행시간: $O(|E|\log|V|)$

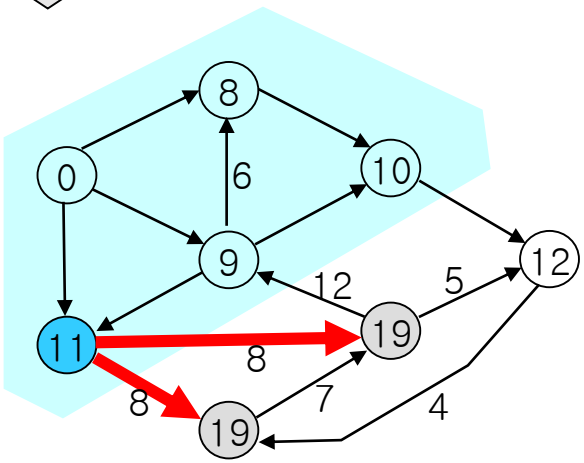
힙 이용

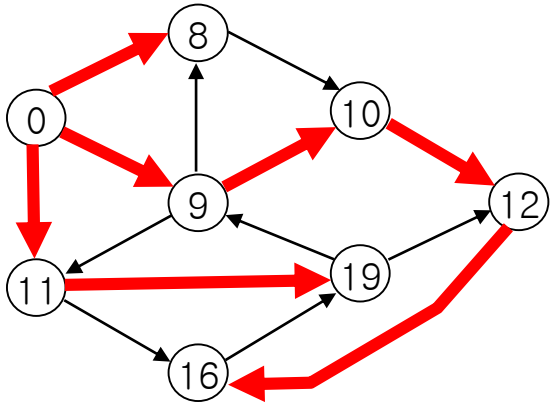
Dijkstra Sample Code

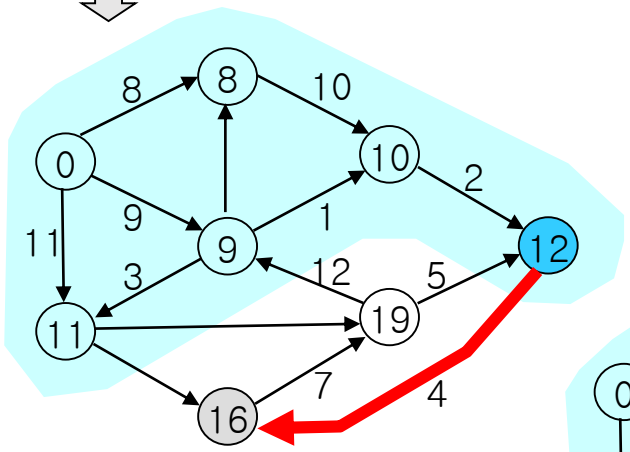
```
int adjMat[LM][LM];      /// 인접 행렬(비용, 거리 등등)
int dist[LM];            /// 출발지로부터 최단거리를 저장할 배열
int path[LM], visit[LM]; /// 경로, 방문 체크
void  Dijkstra(){
    for (int i =1; i <=n; i ++) dist[i ] = INF; /// initialize dist[]
    dist[1] = 0; /// initialize dist[start]
    for (int i =1; i <=n; i ++){
        int minNode, minVal =INF;
        for (int j =1; j <=n; j ++){ /// extract min
            if(visit[j]==0 && minVal>dist[j]){
                minNode = j, minVal = dist[j];
            }
        }
        visit[minNode] = 1;          /// check min node
        for (int j =1; j <=n; j ++){ /// relaxation
            if(dist[j] > minVal + adjMat[minNode][j]){
                dist[j] = minVal + adjMat[minNode][j];
                path[j] = minNode;
            }
        }
    }
}
```

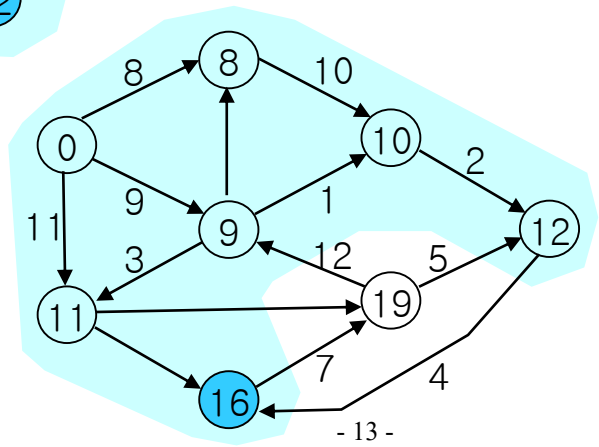
Dijkstra Algorithm의 작동 예

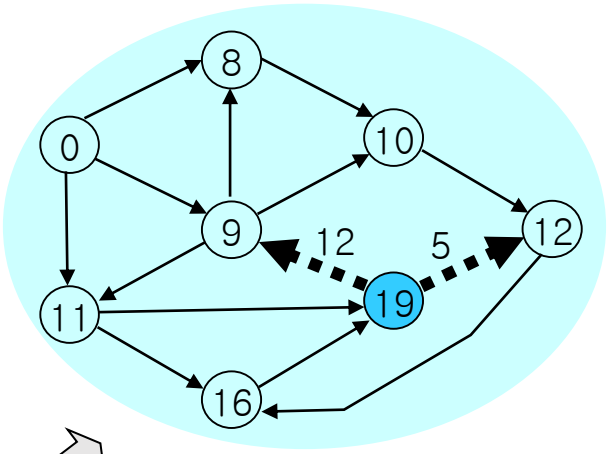



8
0
6
10
9
12
5
11
19
8
8
7
4
19


8
0
10
9
12
11
19
16


8
8
10
0
10
2
9
9
1
11
3
12
5
12
19
11
7
4
16


8
8
10
0
10
2
9
9
1
11
3
12
5
12
19
11
16
7
4


8
0
10
9
12
5
12
19
11
16

Bellman-Ford Algorithm

BellmanFord(G, r)

{

for each $u \in V$

$d_u \leftarrow \infty$;

$d_r \leftarrow 0$;

for $i \leftarrow 1$ **to** $|V|$-1

for each $(u, v) \in E$

if $(d_u + w_{u,v} < d_v)$ **then** $d_v \leftarrow d_u + w_{u,v}$;

}

음의 가중치를 허용한다

이완(relaxation)

✓수행시간: $\Theta(|E||V|)$

Bellman-Ford Sample Code - 1

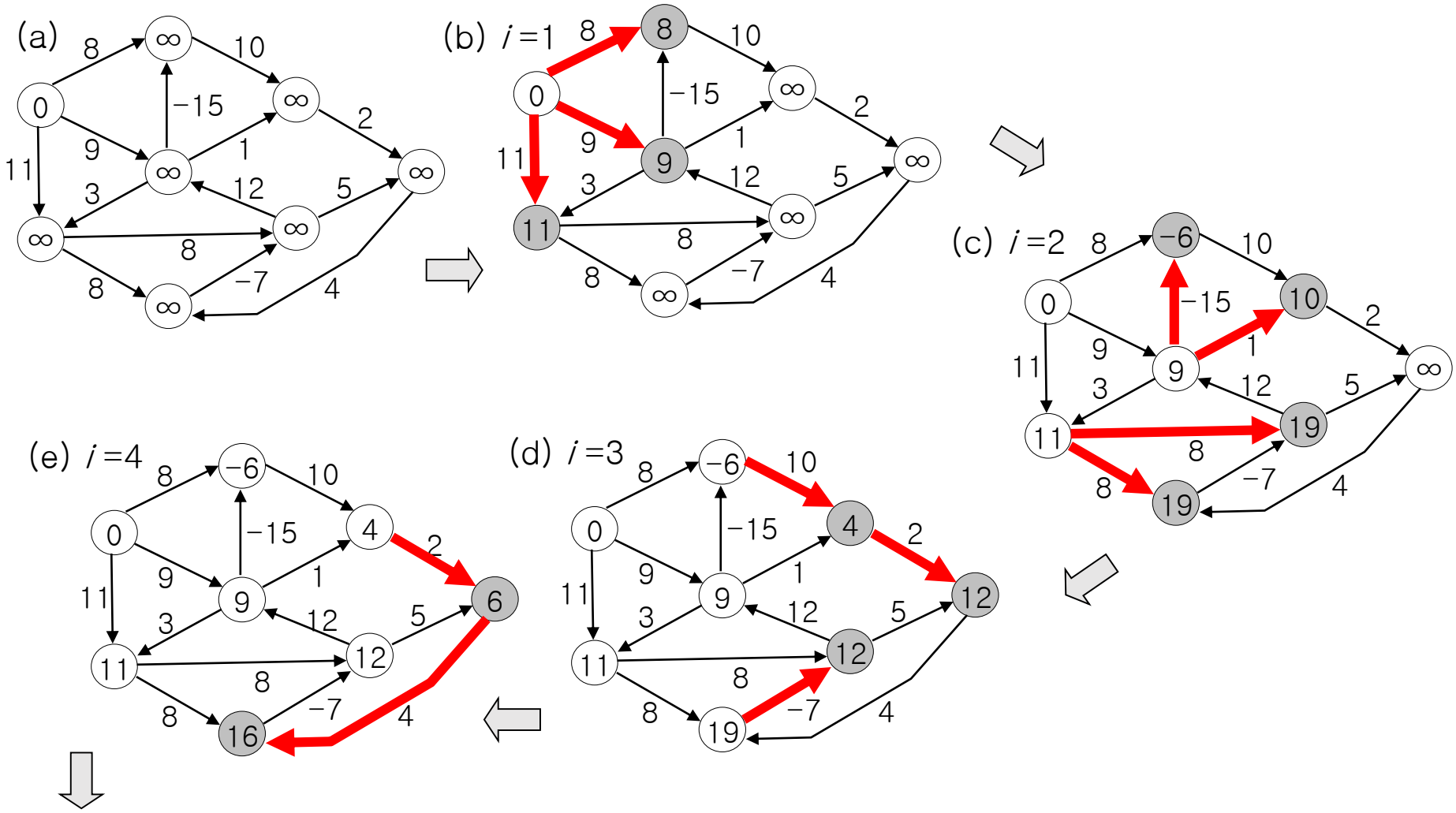
```
int adjMat[LM][LM];      /// 인접 행렬(비용, 거리 등등)
int dist[LM];            /// 출발지로부터 최단거리를 저장할 배열
int path[LM];            /// 경로, 방문 체크
void BellmanFord(){
    for (int i =1; i <=n; i ++) dist[i ] = INF; /// initialize dist[]
    dist[1] = 0; /// initialize dist[start]
    for (int k =1; k<n;k++){
        for (int i =1; i<=n; i++){
            for (int j=1; j<=n; j++){
                if(dist[j] > dist[i]+adjMat [i][j]){   /// relaxation
                    dist[j] = dist[i]+adjMat [i][j];
                    path[j] = i;
                }
            }
        }
    }
}
```

Bellman-Ford Sample Code - 2

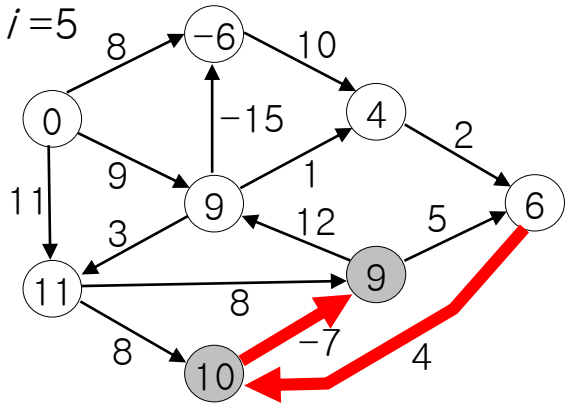
```
struct Data adjList[LM * LM];  /// 인접 리스트
int dist[LM];   /// 출발지로부터 최단거리를 저장할 배열
int path[LM];  /// 경로, 방문 체크
void BellmanFord(){
        for (int i =1; i <=n; i ++) dist[i ] = INF; /// initialize dist[]
        dist[1] = 0; /// initialize dist[start]

        for (int i =1; i<n; i++){
                for (int j =1; j<=an; j++){
                        int s = adjList[j].s, e = adjList[j].e;
                        if(dist[e] > dist[s]+ adjList[j].w){  /// relaxation
                                dist[e] = dist[s]+ adjList[j].w;
                                path[e] = s;
                        }
                }
        }
}
```

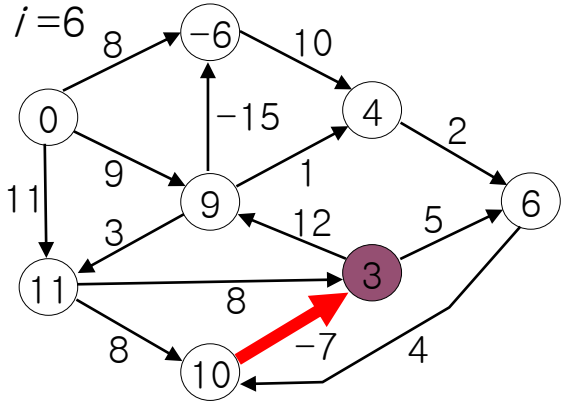
Bellman-Ford Algorithm의 작동 예



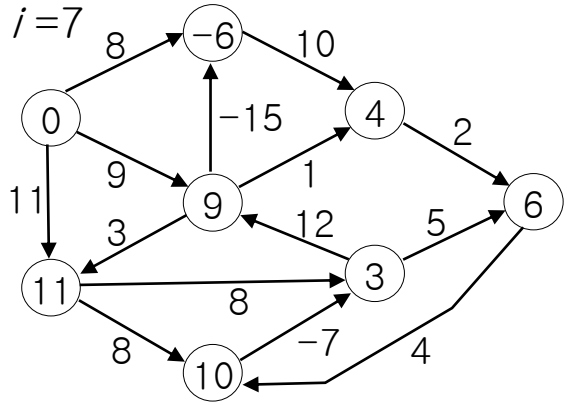
(f) $i=5$


8
−6
10
0
−15
4
2
11
9
1
9
6
3
12
5
11
9
8
8
−7
10
4

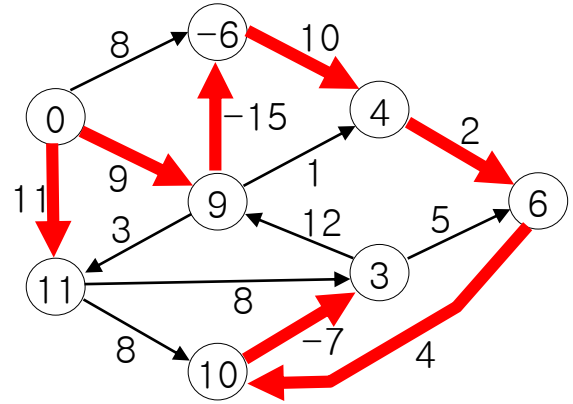
(g) $i=6$


8
−6
10
0
−15
4
2
11
9
1
9
6
3
12
5
11
3
8
8
−7
10
4

(h) $i=7$


8
−6
10
0
−15
4
2
11
9
1
9
6
3
12
5
11
3
8
8
−7
10
4

(i)


8
−6
10
0
−15
4
2
11
9
1
9
6
3
12
5
11
3
8
8
−7
10
4

DP로 본 Bellman-Ford Algorithm

- d_t^k : 중간에 최대 k 개의 간선을 거쳐
  정점 r로부터 정점 t에 이르는 최단거리
- 목표: d_t^{n-1}

✓ 재귀적 관계

$$\begin{cases} d_v^k = \min_{\text{for 모든 간선 } (u,\,v)} \{d_u^{k-1} + w_{u,\,v}\}, & k > 0 \\ d_r^0 = 0 \\ d_t^0 = \infty, \quad t \neq r \end{cases}$$

Thank you