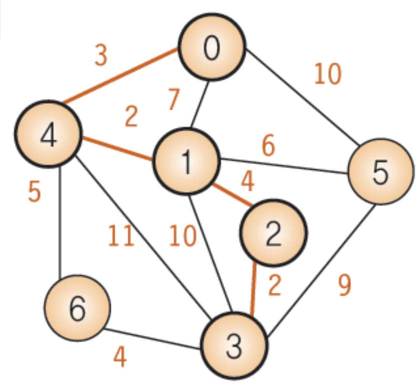
CSE2010 자료구조론

Week 10: Shortest Path

ICT융합학부 한진영

최단 경로(Shortest Path)

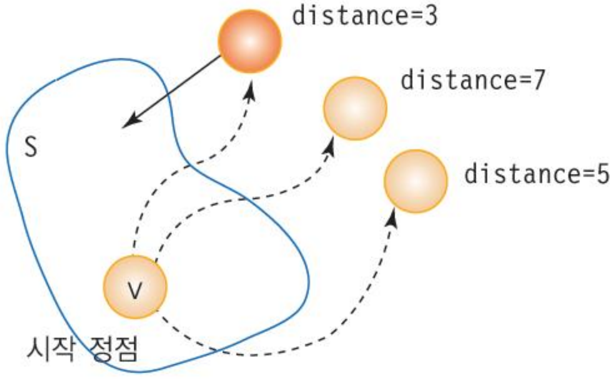
- 네트워크에서 정점 u와 정점 v를 연결하는 경로 중에서 간선들의 가중치 합이 최소가 되는 경로
- 간선의 가중치는 비용, 거리, 시간 등
- 정점 0에서 정점 3으로 가는 최단 경로 문제
  - 인접행렬에서 간선이 없는 노드쌍의 가중치는 ∞ 임
  - 0,4,1,2,3이 최단 경로
  - 최단경로 길이는 3+2+4+2=11



	0	1	2	3	4	5	6
0	0	7	∞	∞	3	10	∞
1	7	0	4	10	2	6	∞
2	∞	4	0	2	∞	∞	∞
3	∞	10	2	0	11	9	4
4	3	2	∞	11	0	∞	5
5	10	6	∞	9	∞	0	∞
6	∞	∞	∞	4	5	∞	0

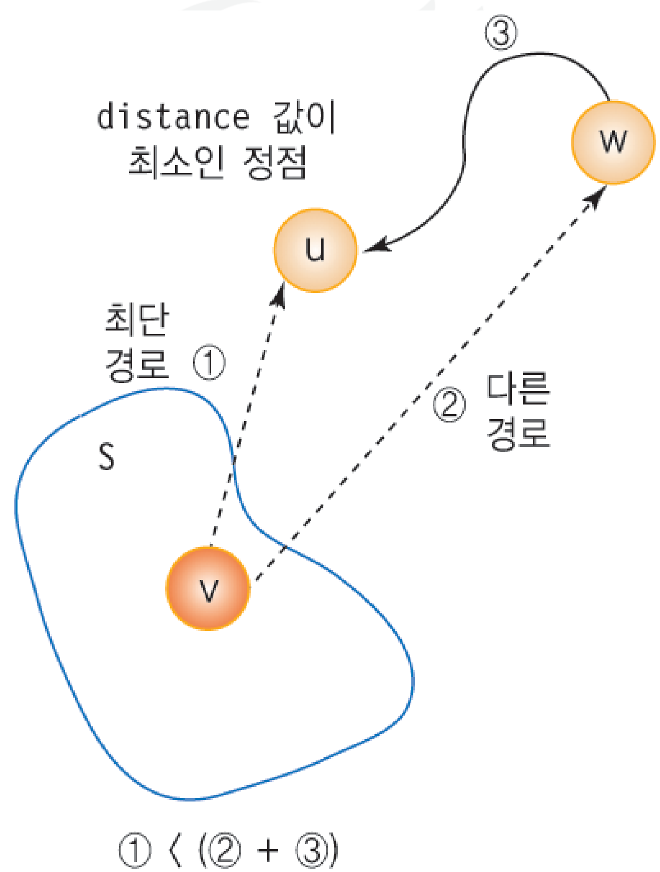
Dijkstra의 최단 경로 알고리즘(1)

- 하나의 시작 정점으로부터 모든 다른 정점까지의 최단 경로 찾음
- 집합 S
  - 시작 정점 v로부터의 최단 경로가 이미 발견된 정점들의 집합
- distance 배열
  - 최단 경로가 알려진 정점들만을 이용한 다른 정점들까지의 최단 경로 길이
  - distance 배열의 초기값(시작 정점 v)
    - distance[v] = 0
    - 다른 정점에 대한 distance 값은 시작 정점과 해당 정점 간의 가중치 값
- 매 단계에서 가장 distance 값이 작은 정점을 S에 추가



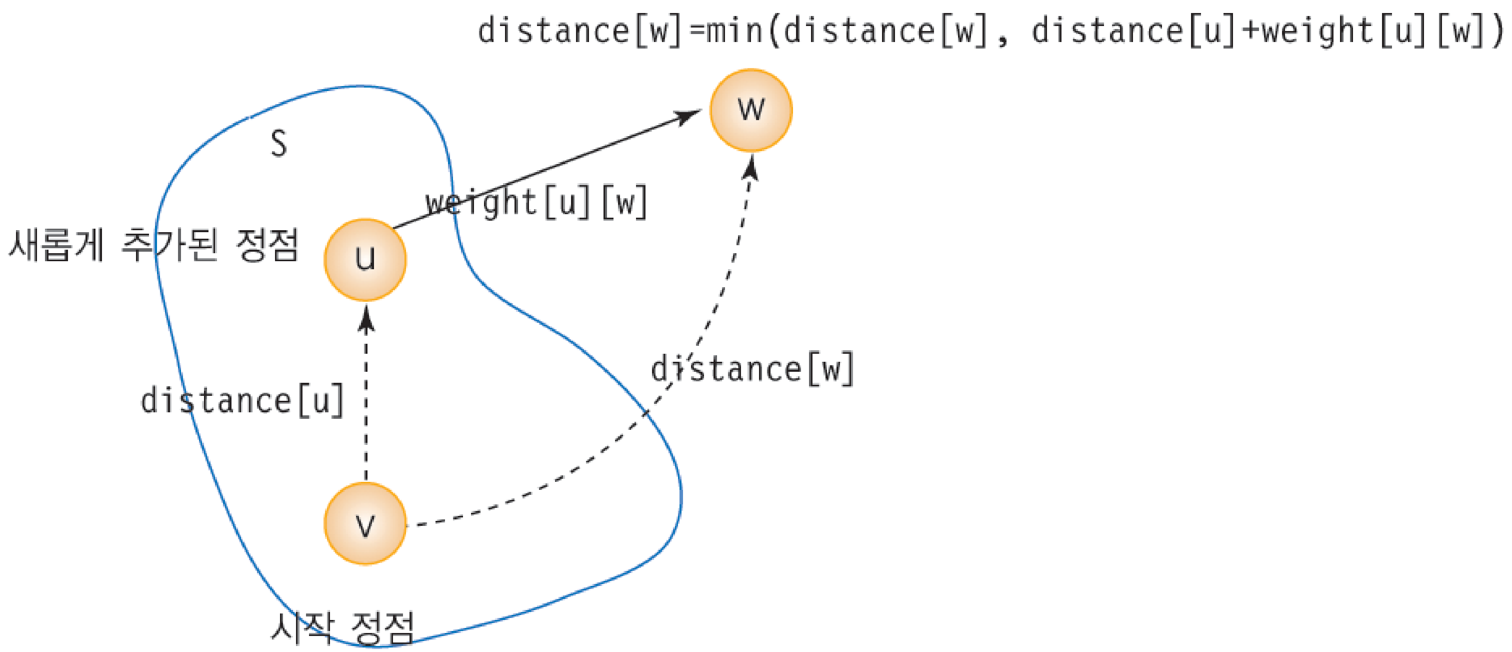
Dijkstra의 최단 경로 알고리즘(2)

- distance 값이 가장 작은 정점을 u라고 하면 시작 정점 v에서 정점 u까지의 최단거리는 경로 ①
- 정점 w를 거쳐서 정점 u로 가는 가상적인 짧은 경로가 있다고 가정해보면 정점 v에서 정점 u까지의 거리는 정점 v에서 정점 w까지의 거리 ②와 정점 w에서 정점 u로 가는 거리③을 합한 값
- 그러나 경로 ②는 경로 ①보다 항상 길 수 밖에 없음. 왜냐하면 현재 distance 값이 가장 작은 정점은 u이기 때문
- 따라서 매 단계에서 distance 값이 가장 작은 정점들을 추가해 나가면 시작 정점에서 모든 정점까지의 최단거리를 구할 수 있음



Dijkstra의 최단 경로 알고리즘(3)

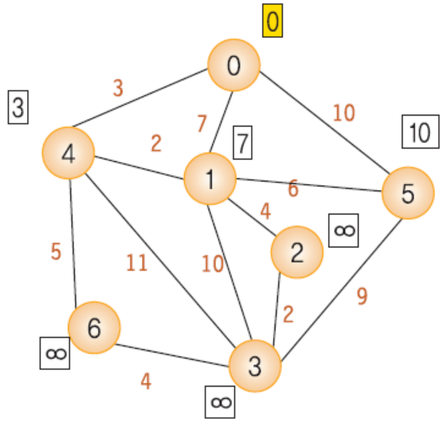
- 새로운 정점이 S에 추가되면 distance값 갱신



Dijkstra의 최단 경로 알고리즘(4)

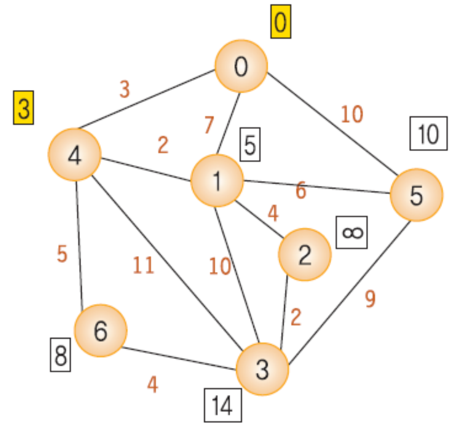
```
// 입력: 가중치 그래프 G, 가중치는 음수가 아님.
// 출력: distance 배열, distance[u]는 v에서 u까지의 최단 거리이다.
shortest_path(G, v)

S←{v}
for 각 정점 w∈G do
    distance[w]←weight[v][w];
while 모든 정점이 S에 포함되지 않으면 do
    u←집합 S에 속하지 않는 정점 중에서 최소 distance 정점;
    S←S∪{u}
    for u에 인접하고 S에 있는 각 정점 z do
        if distance[u]+weight[u][z] < distance[z]
            then distance[z]←distance[u]+weight[u][z];
```

Dijkstra의 최단 경로 알고리즘 예(1)



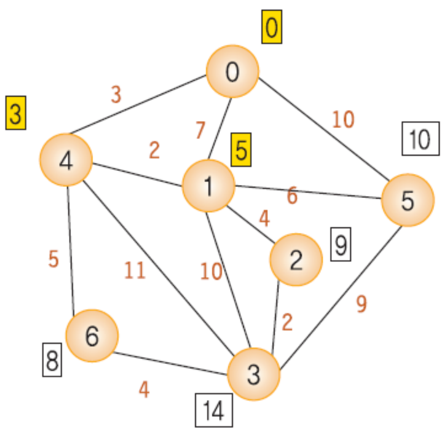
S={0}

	0	1	2	3	4	5	6
distance[]=	0	7	∞	∞	3	10	∞



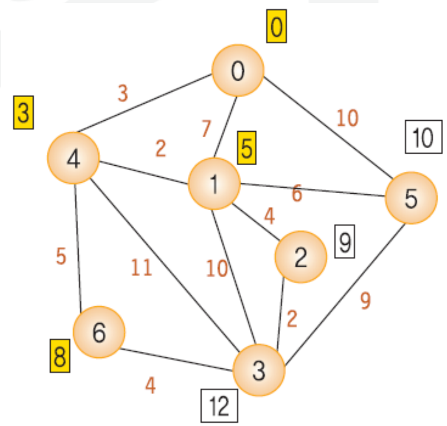
S={0, 4}

	0	1	2	3	4	5	6
distance[]=	0	5	∞	14	3	10	8



S={0, 4, 1}

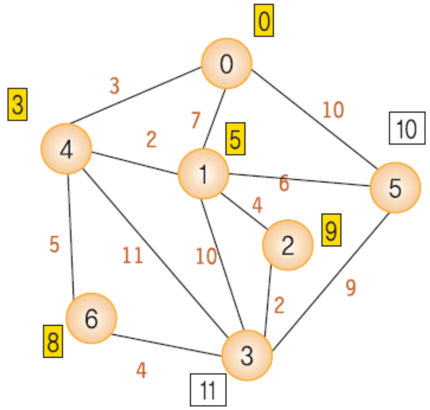
	0	1	2	3	4	5	6
distance[]=	0	5	9	14	3	10	8



S={0, 4, 1, 6}

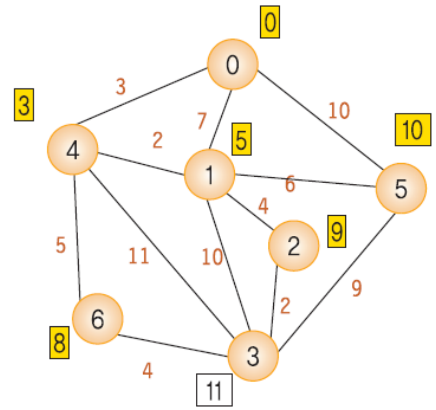
	0	1	2	3	4	5	6
distance[]=	0	5	9	12	3	10	8

Dijkstra의 최단 경로 알고리즘 예(2)



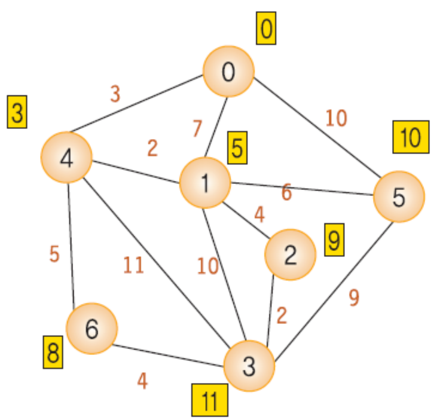
S={0, 4, 1, 6, 2}

	0	1	2	3	4	5	6
distance[]=	0	5	9	11	3	10	8



S={0, 4, 1, 6, 2, 5}

	0	1	2	3	4	5	6
distance[]=	0	5	9	11	3	10	8



S={0, 4, 1, 6, 2, 5, 3}

	0	1	2	3	4	5	6
distance[]=	0	5	9	11	3	10	8

Dijkstra의 최단 경로 알고리즘 복잡도

- 네트워크에 n개의 정점이 있다면, Dijkstra의 최단경로 알고리즘은 주 반복문을 n번 반복하고 내부 반복문을 2n번 반복하므로 $O(n^2)$의 복잡도를 가짐

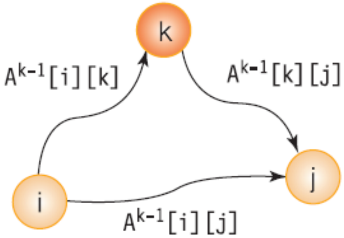
Floyd의 최단 경로 알고리즘(1)

- 모든 정점 사이의 최단경로를 찾음
- 2차원 배열 A를 이용하여 3중 반복을 하는 루프로 구성
- 인접 행렬 weight 구성
  - i==j이면, weight[i][j]=0
  - 두개의 정점 i,j 사이에 간선이 존재하지 않으면, weight[i][j]=∞
  - 정점 i,j 사이에 간선이 존재하면, weight[i][j]는 간선 (i, j)의 가중치
  - 배열 A의 초기 값은 인접 행렬 weight임

```
floyd(G)

for k ← 0 to n - 1
    for i ← 0 to  n - 1
       for j ← 0 to n - 1
          A[i][j] = min(A[i][j], A[i][k] + A[k][j])
```

Floyd의 최단 경로 알고리즘(2)

- A^k [i][j]
  - 0부터 k까지의 정점만을 이용한 정점 i에서 j까지의 최단 경로 길이
- $A^{-1} \rightarrow A^0 \rightarrow A^1 \rightarrow \ldots \rightarrow A^{n-1}$순으로 최단 경로 구해감
- A^{k-1}까지 구해진 상태에서 k번째 정점이 추가로 고려됨



- 0부터 k까지의 정점만을 사용하여 정점 i에서 정점 j로 가는 최단 경로는 다음의 2 가지의 경우로 나뉘어짐
  - 정점 k를 거치지 않는 경우:
    - A^k[i][j] 는 k보다 큰 정점은 통과하지 않으므로 최단거리는 여전히 A^{k-1}[i][j]]임
  - 정점 k를 거치는 경우:
    - i에서 k까지의 최단거리 A^{k-1}[i][k]에 k에서 j까지의 최단거리 A^{k-1}[k][j]를 더한 값

Floyd의 최단 경로 알고리즘 예

	0	1	2	3	4	5	6
0	0	7	INF	INF	3	10	INF
1	7	0	4	10	2	6	INF
2	INF	4	0	2	INF	INF	INF
3	INF	10	2	0	11	9	4
4	3	2	INF	11	0	13	5
5	10	6	INF	9	13	0	INF
6	INF	INF	INF	4	5	INF	0

	0	1	2	3	4	5	6
0	0	7	11	17	3	10	INF
1	7	0	4	10	2	6	INF
2	11	4	0	2	6	10	INF
3	17	10	2	0	11	9	4
4	3	2	6	11	0	8	5
5	10	6	10	9	8	0	INF
6	INF	INF	INF	4	5	INF	0

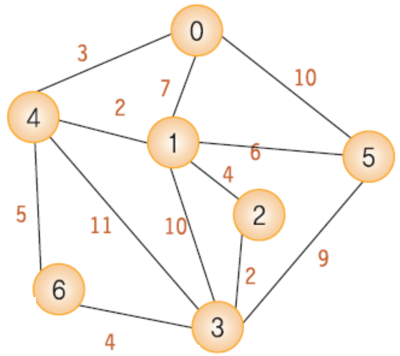
	0	1	2	3	4	5	6
0	0	7	11	13	3	10	INF
1	7	0	4	6	2	6	INF
2	11	4	0	2	6	10	INF
3	13	6	2	0	8	9	4
4	3	2	6	8	0	8	5
5	10	6	10	9	8	0	INF
6	INF	INF	INF	4	5	INF	0

	0	1	2	3	4	5	6
0	0	7	11	13	3	10	17
1	7	0	4	6	2	6	10
2	11	4	0	2	6	10	6
3	13	6	2	0	8	9	4
4	3	2	6	8	0	8	5
5	10	6	10	9	8	0	3
6	17	10	6	4	5	13	0

	0	1	2	3	4	5	6
0	0	5	9	11	3	10	8
1	5	0	4	6	2	6	7
2	9	4	0	2	6	10	6
3	11	6	2	0	8	9	4
4	3	2	6	8	0	8	5
5	10	6	10	9	8	0	13
6	8	7	6	4	5	13	0

	0	1	2	3	4	5	6
0	0	5	9	11	3	10	8
1	5	0	4	6	2	6	7
2	9	4	0	2	6	10	6
3	11	6	2	0	8	9	4
4	3	2	6	8	0	8	5
5	10	6	10	9	8	0	13
6	8	7	6	4	5	13	10

	0	1	2	3	4	5	6
0	0	5	9	11	3	10	8
1	5	0	4	6	2	6	7
2	9	4	0	2	6	10	6
3	11	6	2	0	8	9	4
4	3	2	6	8	0	8	5
5	10	6	10	9	8	0	13
6	8	7	6	4	5	13	0



Floyd의 최단 경로 알고리즘 복잡도

- 네트워크에 n개의 정점이 있다면, Floyd의 최단경로 알고리즘은 3중 반복문을 실행되므로 시간 복잡도는 $O(n^3)$
- 모든 정점상의 최단경로를 구하려면 Dijkstra의 알고리즘 $O(n^2)$을 n번 반복해도 되며, 이 경우 전체 복잡도는 $O(n^3)$
- 모든 정점 쌍의 최단경로를 구하는데 있어 두 알고리즘 모두 동일한 $O(n^3)$의 복잡도를 가지지만 Floyd의 알고리즘은 매우 간결한 반복구문을 사용하므로 Dijkstra의 알고리즘 보다 효율적일 수 있음

Week 10: Shortest Path