Nonlinear Data Structure

http://smart.hallym.ac.kr

Instructor: **Jin Kim**
010-6267-8189(033-248-2318)
jinkim@hallym.ac.kr
Office Hours:

Non Linear Data Structure

- Data structure we will consider this semister:
  - Tree(트리)
  - Binary Search Tree(이진탐색트리)
  - Graph(그래프)
  - Weighted Graph(가중치그래프)
  - Sorting(정렬)
  - Balanced Search Tree(균형탐색트리)

가중치 그래프
Weighted Graph

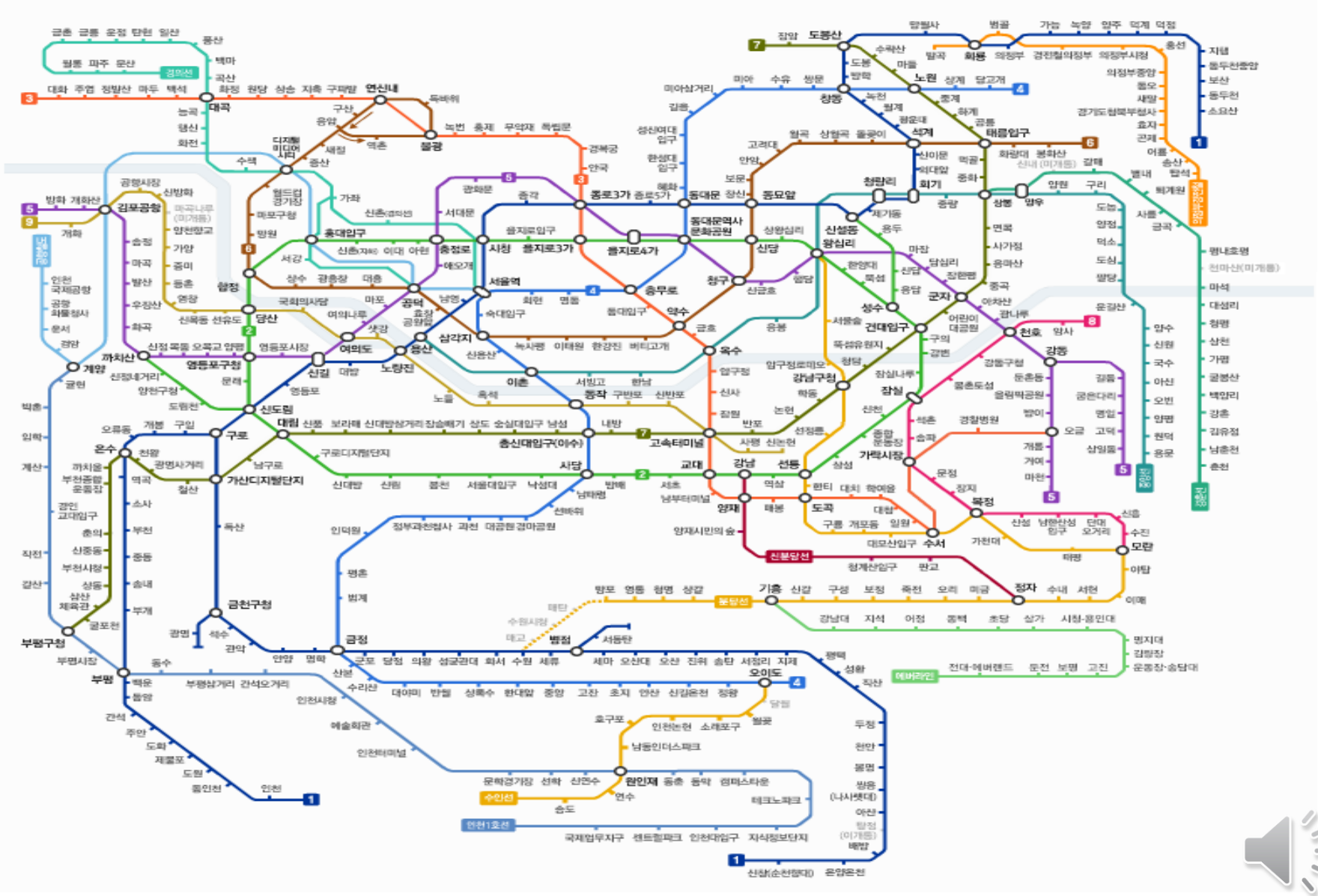
- Minimum Spanning Tree(최소비용간선트리)
- Shortest path(최단경로)
- Topological order(토폴로지컬(위상) 순서)
- Critical Path(임계경로)

Shortest Path(최단경로)

컴퓨터공학에서 가장 많이 사용되는 단 한 가지

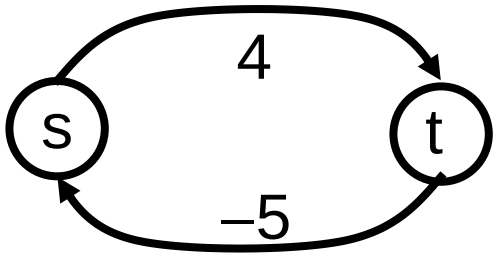
알고리즘을 고른다면 최단경로 찾는 알고리즘일 것임.

수도권 지하철 노선도



Shortest Paths Algorithms(최단경로알고리즘)

- Dijkstra's algorithm
- Bellman & Ford algorithm
- All Pairs Shortest Path – Floyd-Washall algorithm
- 가중치의 합이 음이 존재하는 사이클이 있는 그래프의 경우, 어떠한 알고리즘도 최단 경로를 해결할 수 없다.



Single-Source Shortest Paths
Dijkstra's algorithm

- **Given:** A single source vertex in a weighted, directed graph.
- Want to compute a shortest path for each possible destination.
  - Similar to BFS.
- We will assume either
  - no negative-weight edges
- Algorithm will compute a shortest-path tree.
  - Similar to BFS tree.

Dijkstra's Algorithm for Shortest Paths Algorithms(다익스트라의 최단경로알고리즘)

- Non-negative edge weight(음의 가중치를 허용하지 않음. 음의 가중치가 있는 그래프에서 잘못된 결과발생)
- Single-source shortest-paths problem: Find the shortest path from *s* to each vertex *v*. (하나의 출발점에서 다른 모든 정점까지의 최단 경로) 알고리즘은 트리모양의 출력을 제공한다.
- Like BFS: If all edge weights are equal, then use BFS, otherwise use this algorithm (너비우선탐색과 유사)
- Use Q = priority queue keyed on d[v] values(우선순위큐사용)(note: BFS uses FIFO)
- 다익스트라의 알고리즘은 greedy algorithm(탐욕적 알고리즘)

Important Aside: Greedy Algorithms

- A *greedy* algorithm always takes the best immediate or local solution while finding an answer. 탐욕 알고리즘은 특정단계에서 최적으로 보이는 답을 선택
- Greedy algorithms find optimal solutions for some optimization problems, but may find (far) less-than-optimal solutions for other optimization problems. 그러나 나중에 결과를 보면 최적의 답이 아닐수있다.
  - Dijkstra's algorithm is greedy.
- When greedy algorithms work, they are usually best. 다이스트라의 알고리즘은 다행히 탐욕알고리즘이지만 최적의 해를 제공한다.

Dijkstra's Algorithm

```
Dijkstra(G)
    for each v ∈ V
        d[v] = ∞;
    d[s] = 0; S = ∅; Q = V;
    while (Q ≠ ∅)
        u = ExtractMin(Q);//Q에서가장작은정점꺼냄
        S = S U{u};//결정된집합에 추가
        for each v ∈ u->Adj[]
            if (d[v] > d[u]+w(u,v))//경로 재계산
                d[v] = d[u]+w(u,v);
```

Dijkstra's Algorithm

Assumes **no negative-weight edges**.(음의 가중치가 없다고 가정)

Maintains a set S of vertices whose SP(shortest paths) from s has been determined.

Repeatedly selects u in V–S with minimum SP estimate (greedy choice).

Store V–S in priority queue Q.

```
Initialize(G, s);
S := ∅;
Q := V[G];
while Q ≠ ∅ do
u := Extract-Min(Q);
S := S ∪ {u};
for each v ∈ Adj[u] do
      Relax(u, v, w)
od
od
```

Dijkstra's Algorithm

- Set all distances initially to ∞, except the start node, which should be set to 0
- Construct a min priority queue(최소우선순위큐) of the nodes, with their distances as keys
- Repeatedly remove the minimum element, updating each of its adjacent node's distances if they are still in the queue and if the updated distance is less than the current distance

```
shortestPath(v, weight, n)
    // v는 시작점, weight는 가중치 인접 행렬, n은 정점수
    // create S[n], Dist[n]
  for (i←0; i<n; i←i+1) do {
       S[i] ← false;            // S를 초기화
       Dist[i] ← weight[v, i];      // Dist를 초기화
  }
  S[v] ← true;
  Dist[v] ← 0;
  for (i←0; i<n-2; i←i+1) do {  // n-2번 반복
       select u such that            // 새로운 최단 경로를 선정
            Dist[u] = min{Dist[j] | S[j] = false and 0≤j<n};
       S[u] ← true;
       for (w←0; w<n; w←w+1) do {  // 확정이 안된 경로들에 대해 다시 계산
            if (S[w] = false) then {
                 if (Dist[w] > (Dist[u] + weight[u, w])
                      then Dist[w] ← Dist[u] + weight[u, w];
  }}}
end shortestPath
```

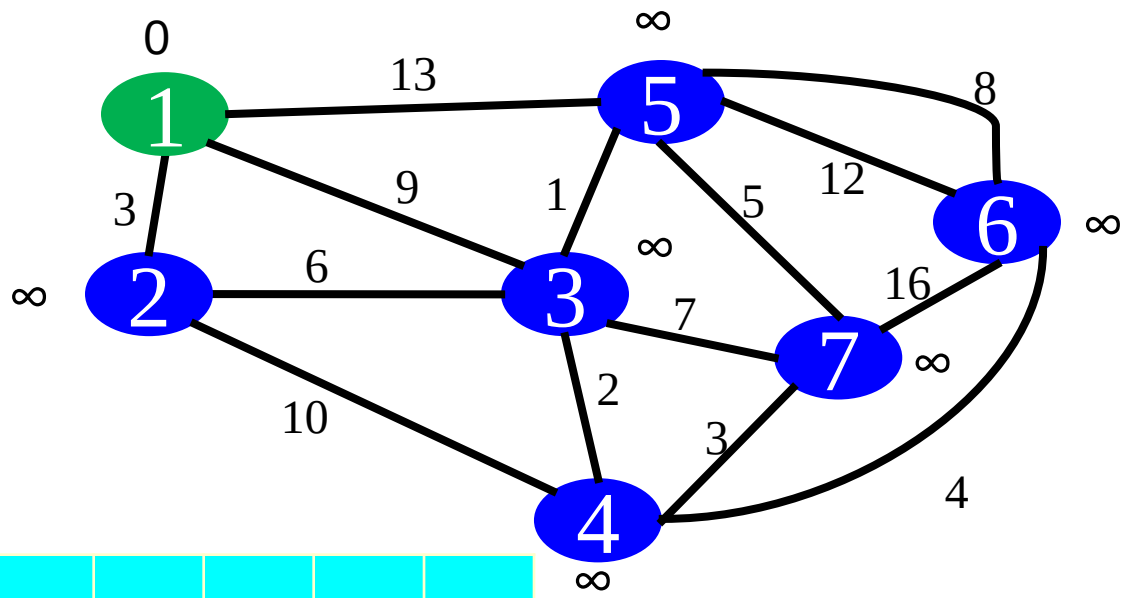
시간복잡도는 $O(E \lg V + E)$ V : 정점의 개수 E : 간선의 개수

$=O(V^2)$

Dijkstra's Algorithm
다익스트라의 알고리즘 예

Dijkstra's Algorithm Solution

- Find the shortest distances to each node from node 1

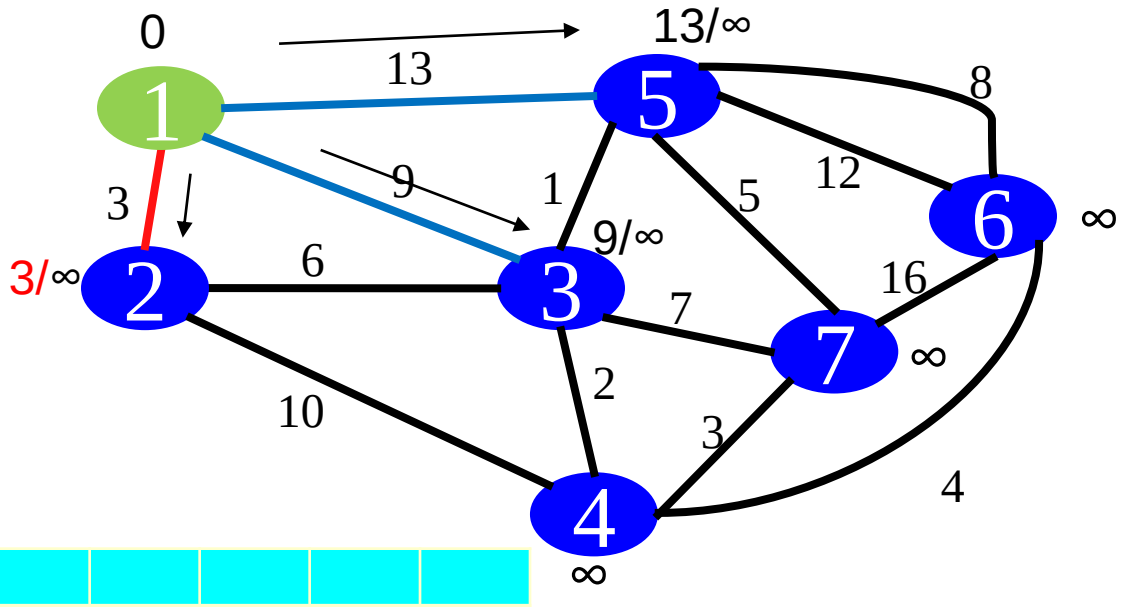


P	Q
1	(0)
2	(∞)
3	(∞)
4	(∞)
5	(∞)
6	(∞)
7	(∞)

S=							
Q=	1	2	3	4	5	6	7

Dijkstra's Algorithm Solution

- Find the shortest distances to each node from node 1

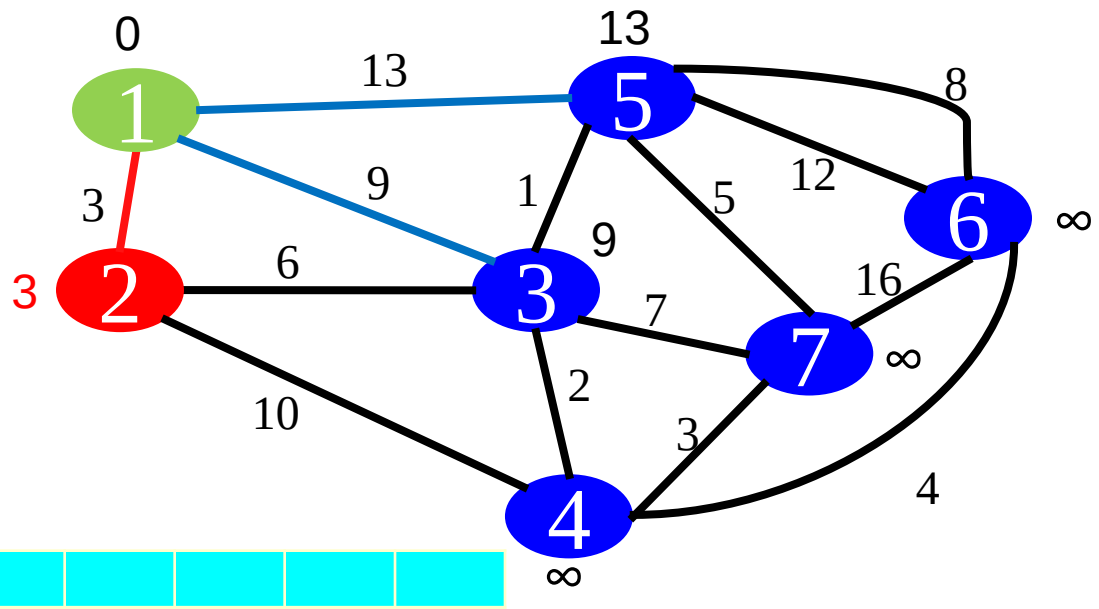


P	Q
x1	(0)
2	(3)
3	(9)
4	(∞)
5	(13)
6	(∞)
7	(∞)

S=	1						
Q=	2	3	4	5	6	7	

Dijkstra's Algorithm Solution

- Find the shortest distances to each node from node 1

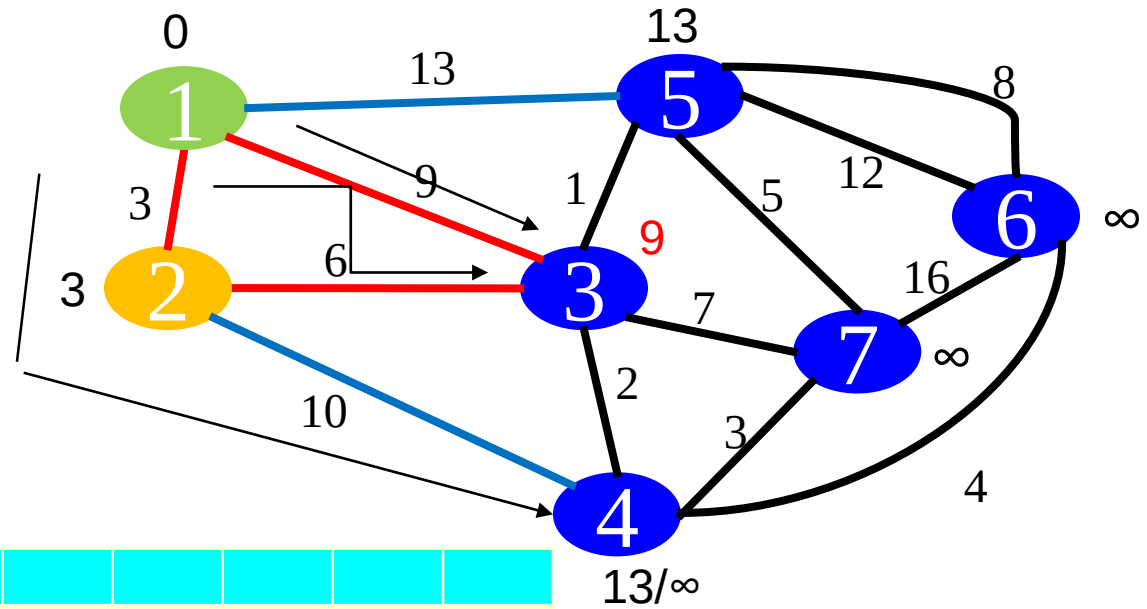


P	Q
x1	(0)
2	(3)
3	(9)
4	(∞)
5	(13)
6	(∞)
7	(∞)

S=	1	2					
Q=		3	4	5	6	7	

Dijkstra's Algorithm Solution

- Find the shortest distances to each node from node 1

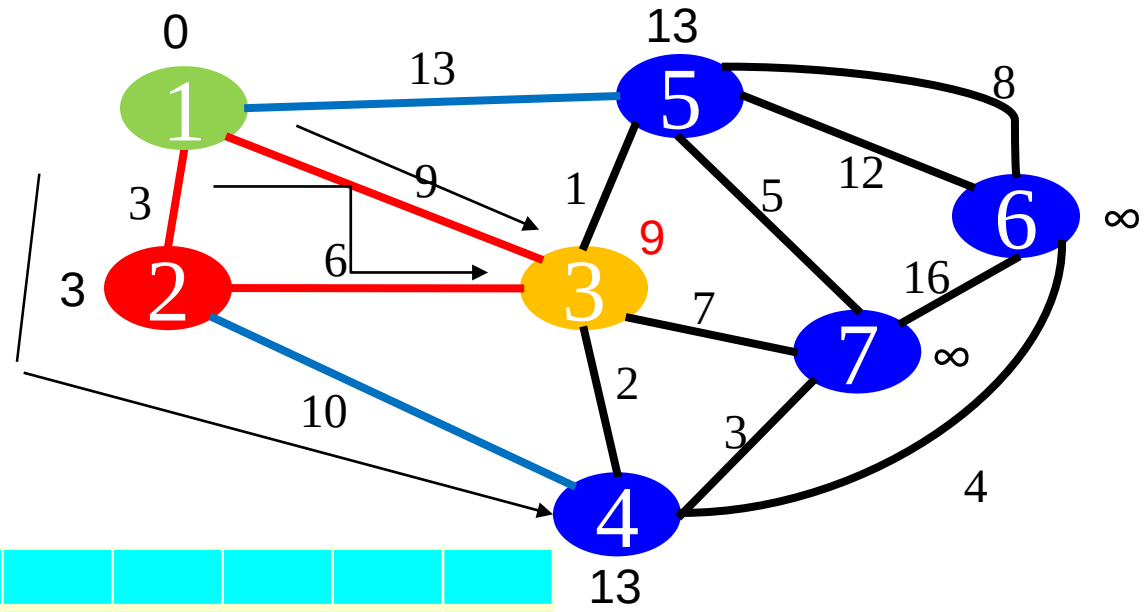


P	Q
x1	(0)
x2	(3)
3	(9)
4	(13)
5	(13)
6	(∞)
7	(∞)

S=	1	2					
Q=	3	4	5	6	7		

Dijkstra's Algorithm Solution

- Find the shortest distances to each node from node 1

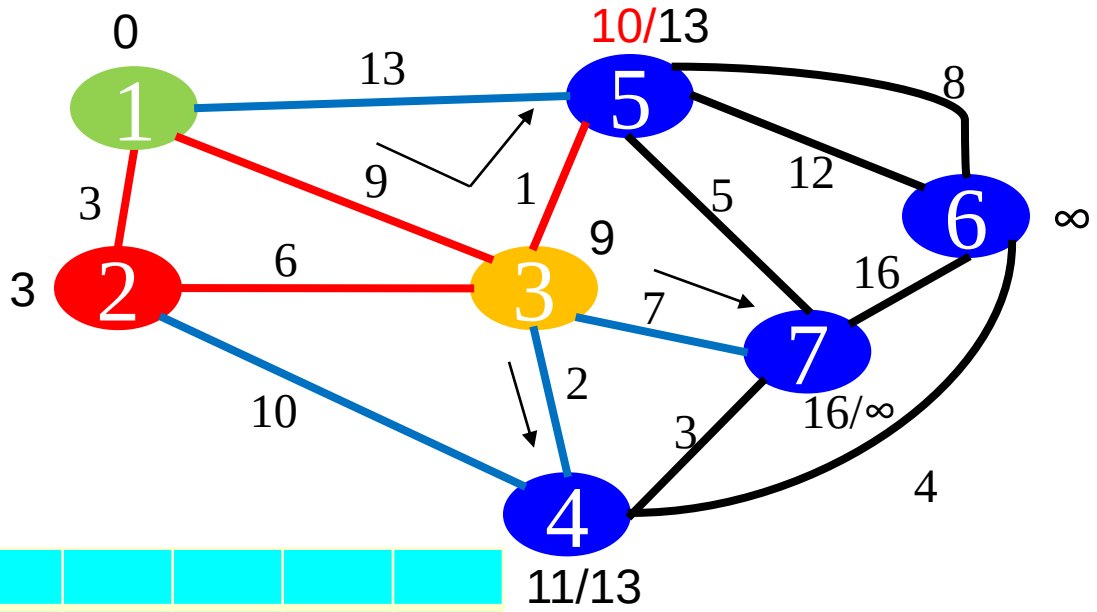


P	Q
x1	(0)
x2	(3)
3	(9)
4	(13)
5	(13)
6	(∞)
7	(∞)

S=	1	2					
Q=	3	4	5	6	7		

Dijkstra's Algorithm Solution

- Find the shortest distances to each node from node 1

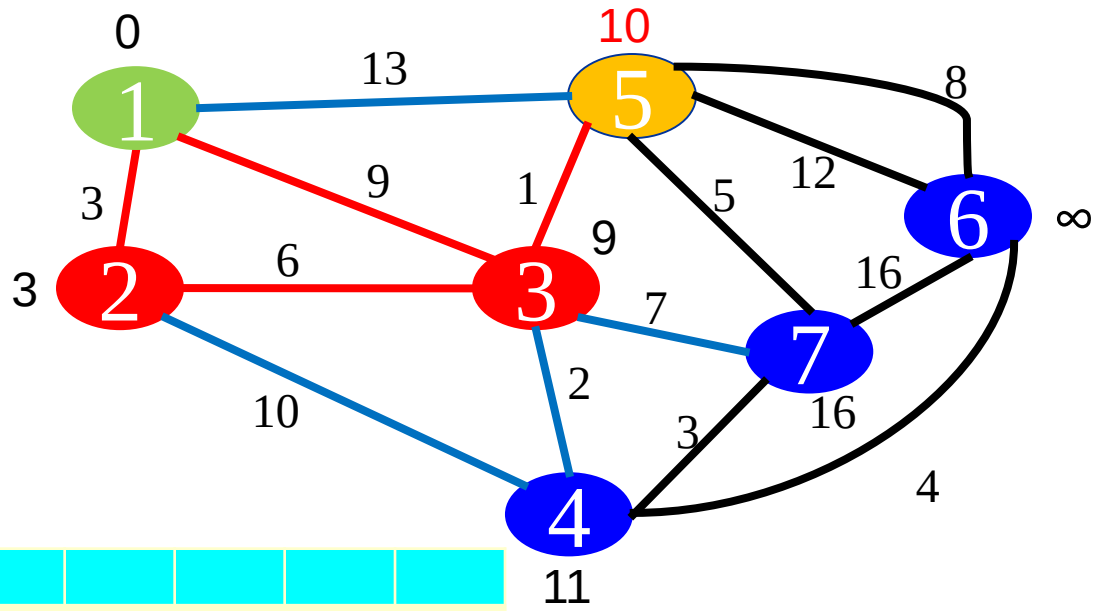


P	Q
x1	(0)
x2	(3)
x3	(9)
4	(11)
5	(10)
6	(∞)
7	(16)

S=	1	2	3				
Q=	4	5	6	7			

Dijkstra's Algorithm Solution

- Find the shortest distances to each node from node 1

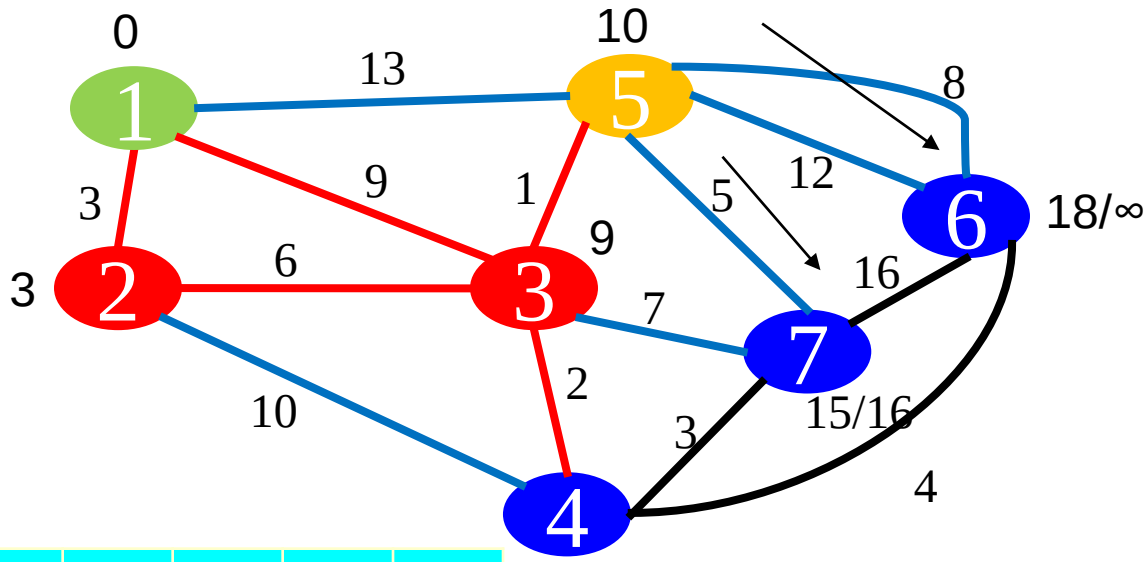


P	Q
x1	(0)
x2	(3)
x3	(9)
4	(11)
5	(10)
6	(∞)
7	(16)

S=	1	2	3				
Q=	4	5	6	7			

Dijkstra's Algorithm Solution

- Find the shortest distances to each node from node 1

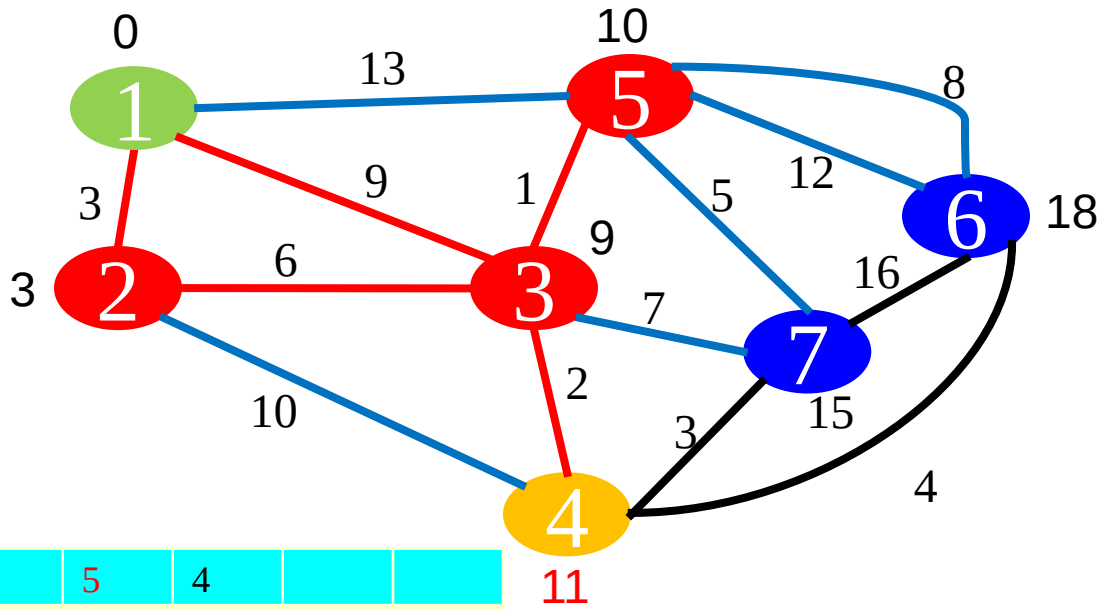


P	Q
x1	(0)
x2	(3)
x3	(9)
4	(11)
x5	(10)
6	(18)
7	(15)

S=	1	2	3	5			
Q=	4	6	7				

Dijkstra's Algorithm Solution

- Find the shortest distances to each node from node 1

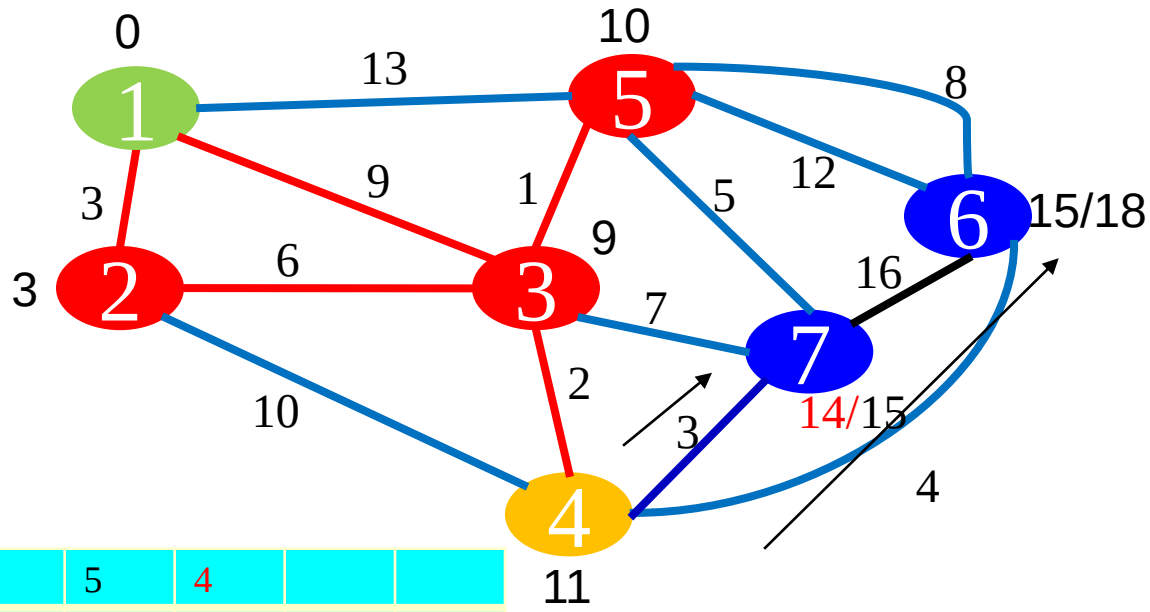


P	Q
x1	(0)
x2	(3)
x3	(9)
x4	(11)
x5	(10)
6	(18)
7	(15)

S=	1	2	3	5	4		
Q		6	7				

Dijkstra's Algorithm Solution

- Find the shortest distances to each node from node 1

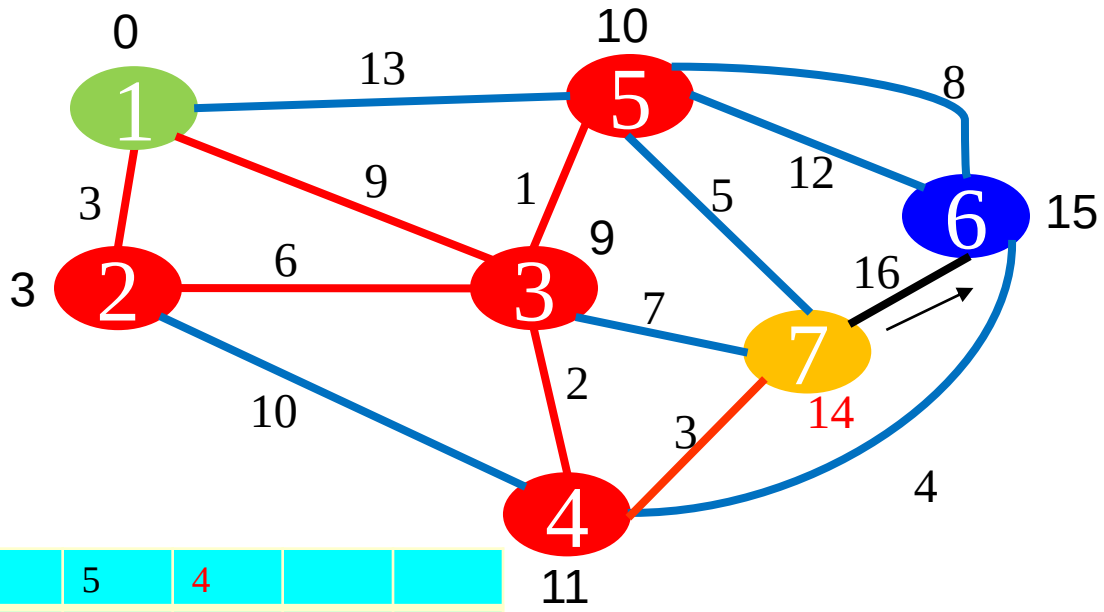


P	Q
x1	(0)
x2	(3)
x3	(9)
x4	(11)
x5	(10)
6	(15)
7	(14)

S=	1	2	3	5	4		
Q=	6	7					

Dijkstra's Algorithm Solution

- Find the shortest distances to each node from node 1

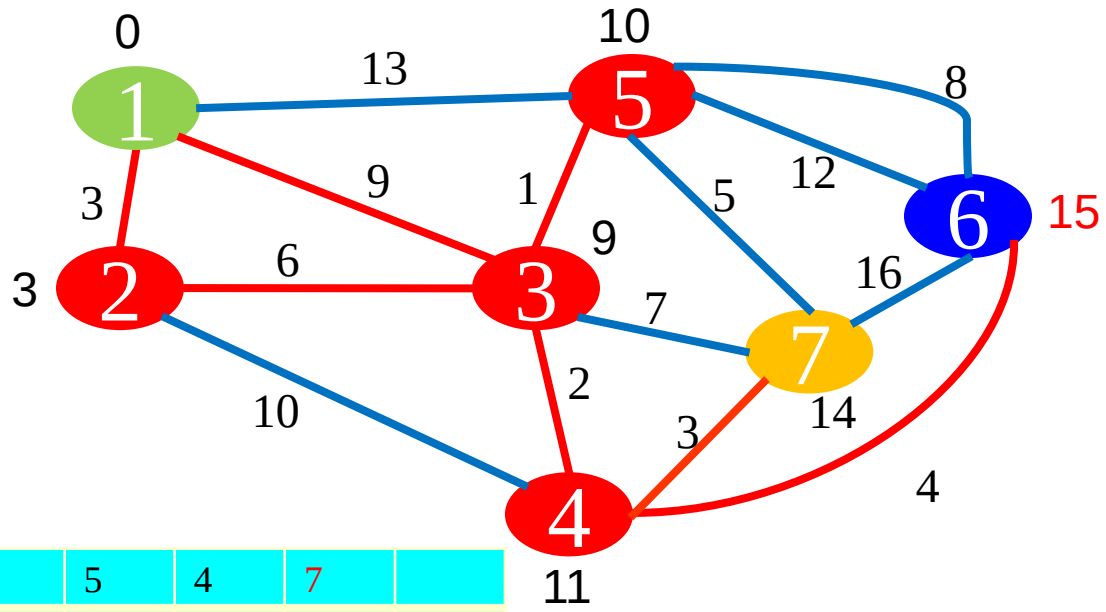


P	Q
x1	(0)
x2	(3)
x3	(9)
x4	(11)
x5	(10)
6	(15)
x7	(14)

S=	1	2	3	5	4		
Q=	6	7					

Dijkstra's Algorithm Solution

- Find the shortest distances to each node from node 1

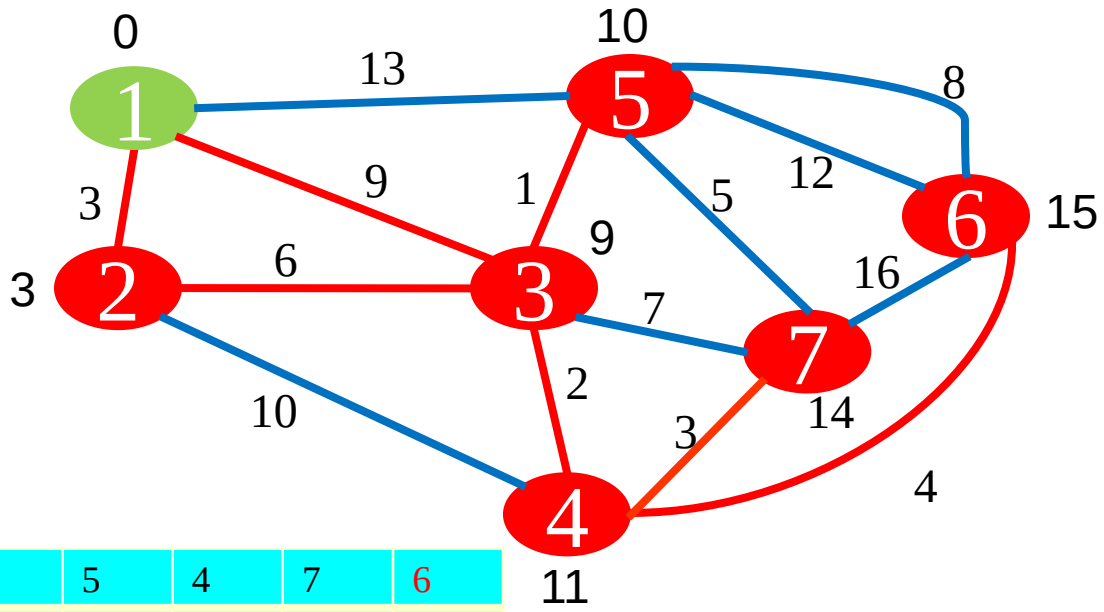


P	Q
x1	(0)
x2	(3)
x3	(9)
x4	(11)
x5	(10)
6	(15)
x7	(14)

S=	1	2	3	5	4	7	
Q=	6						

Dijkstra's Algorithm Solution

- Find the shortest distances to each node from node 1

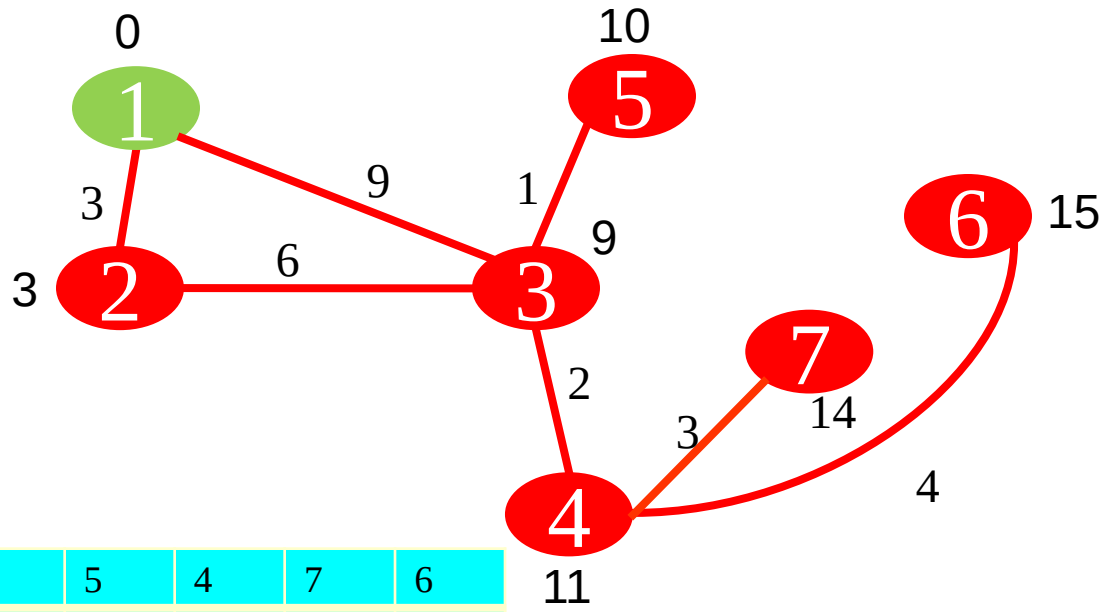


P	Q
x1	(0)
x2	(3)
x3	(9)
x4	(11)
x5	(10)
x6	(15)
x7	(14)

S=	1	2	3	5	4	7	6
Q=							

Dijkstra's Algorithm Solution

- Find the shortest distances to each node from node 1



P	Q
x1	(0)
x2	(3)
x3	(9)
x4	(11)
x5	(10)
x6	(15)
x7	(14)

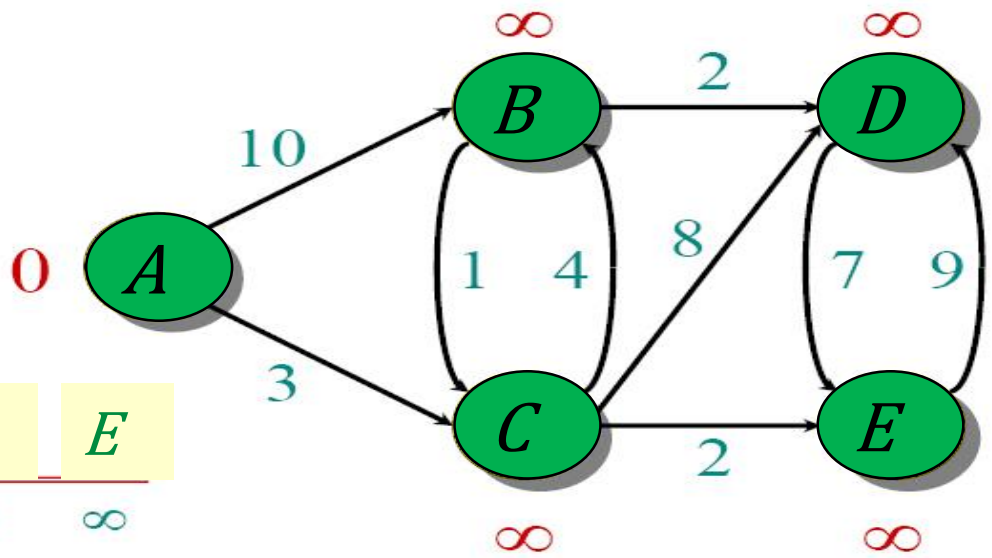
S=	1	2	3	5	4	7	6
Q=							

Spanning Tree (간선트리)

Dijkstra's Algorithm
다익스트라의 알고리즘 예2

Dijkstra Animated Example

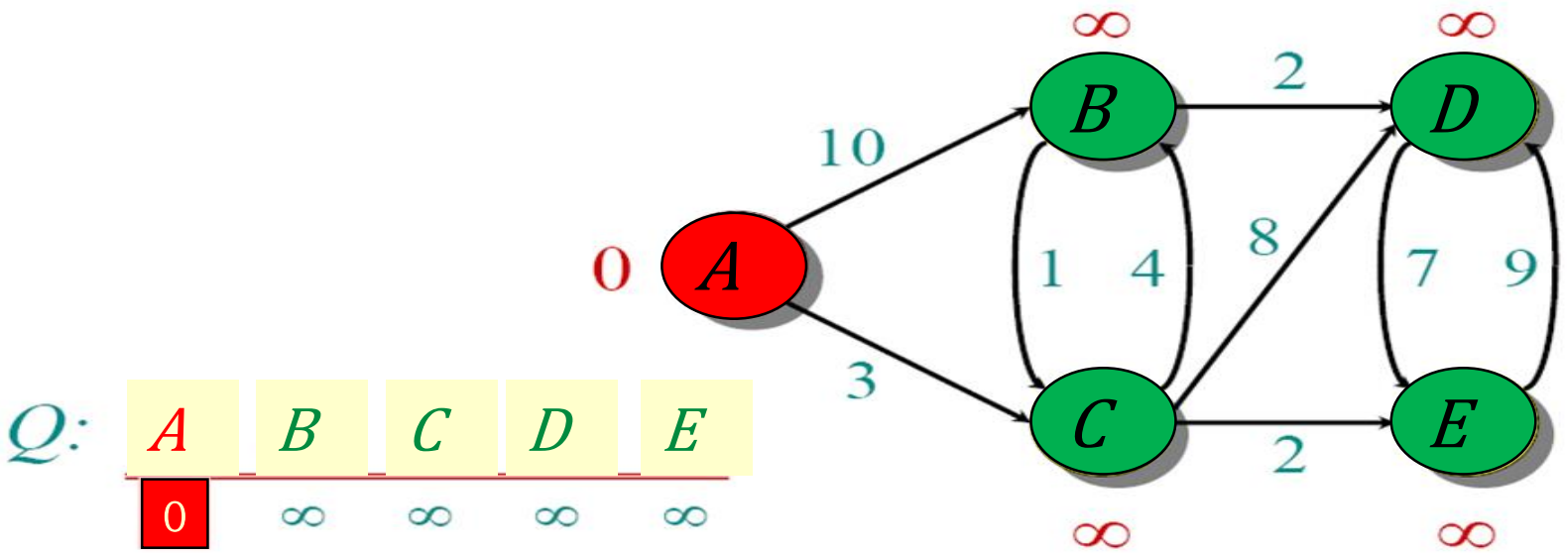
Initialize:



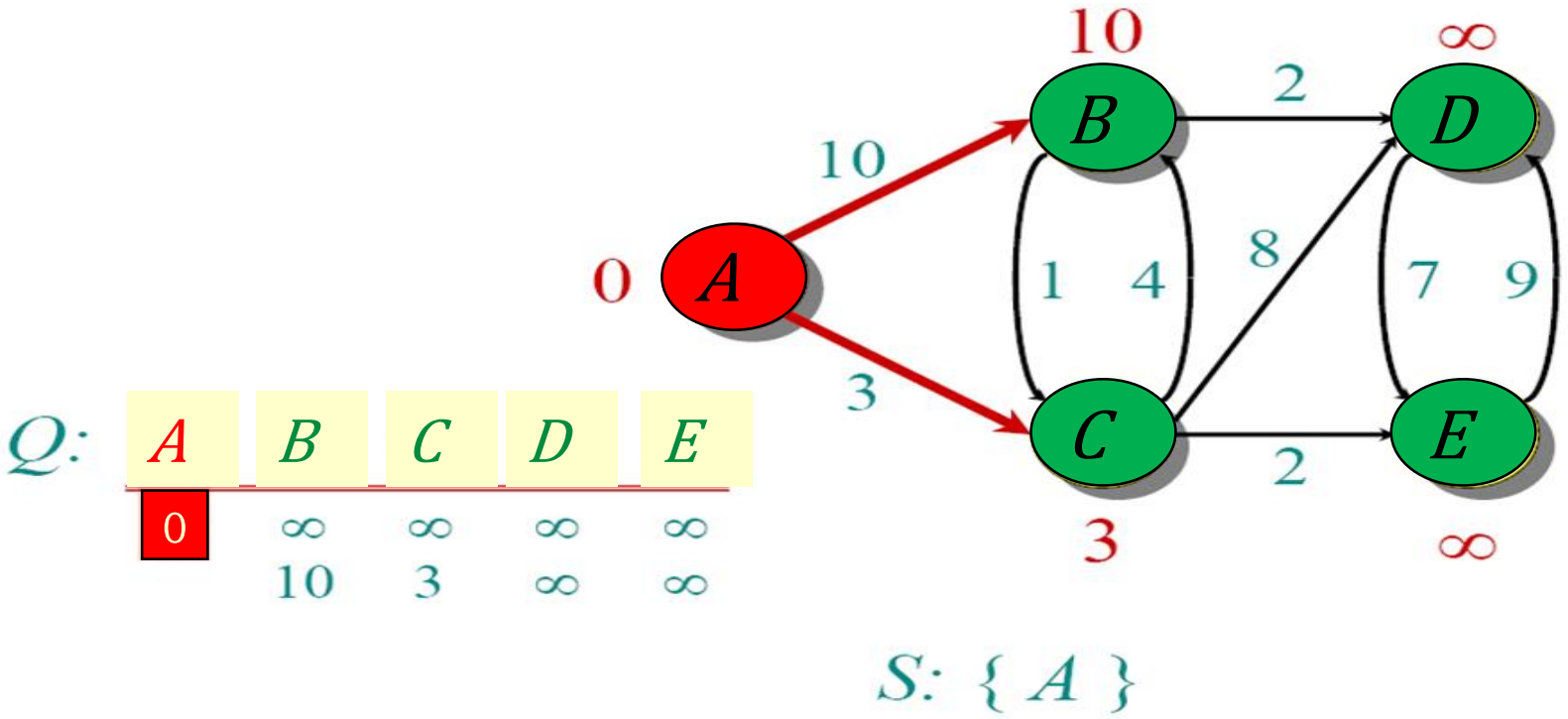
Q:	A	B	C	D	E
	0	∞	∞	∞	∞

S: {}

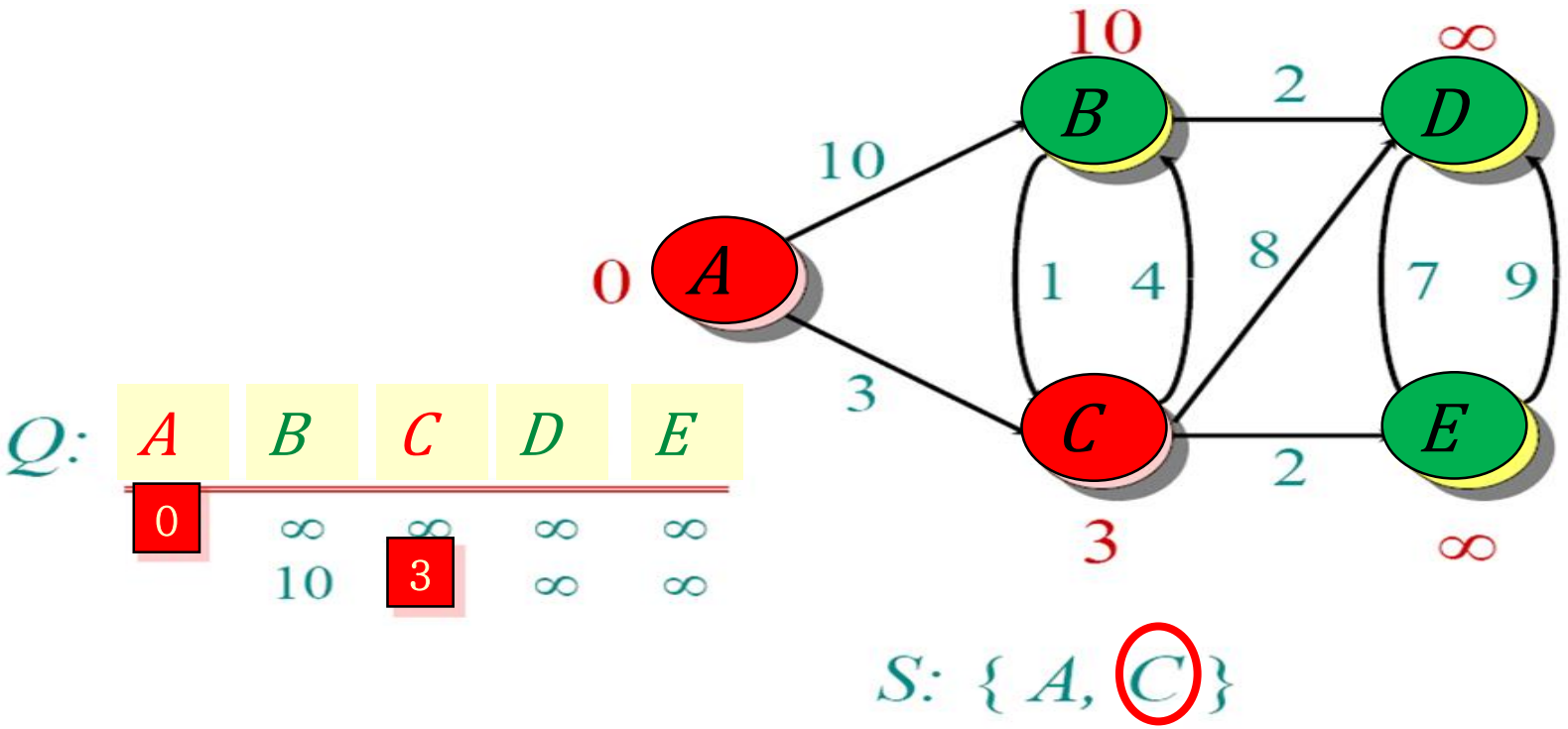
Dijkstra Animated Example



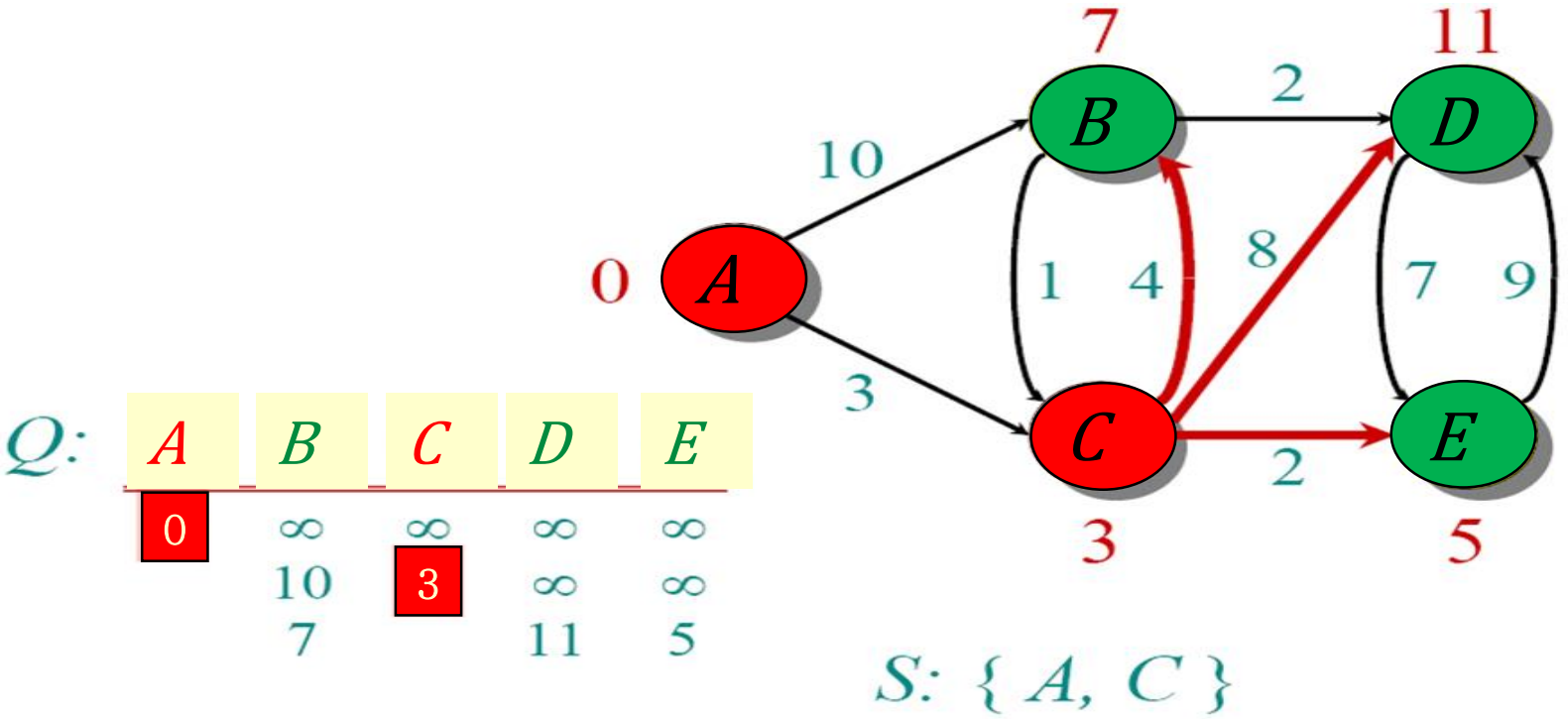
Dijkstra Animated Example



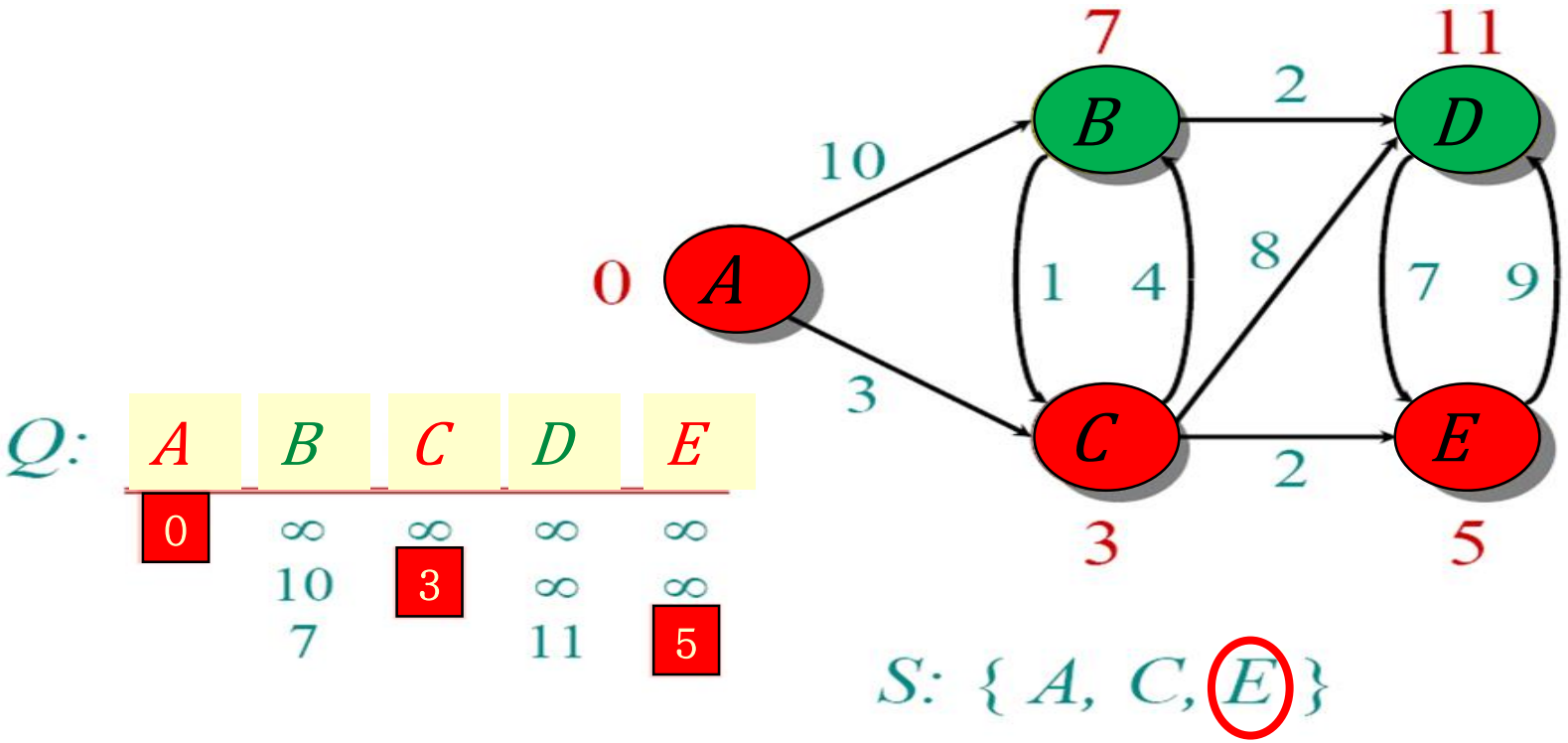
Dijkstra Animated Example



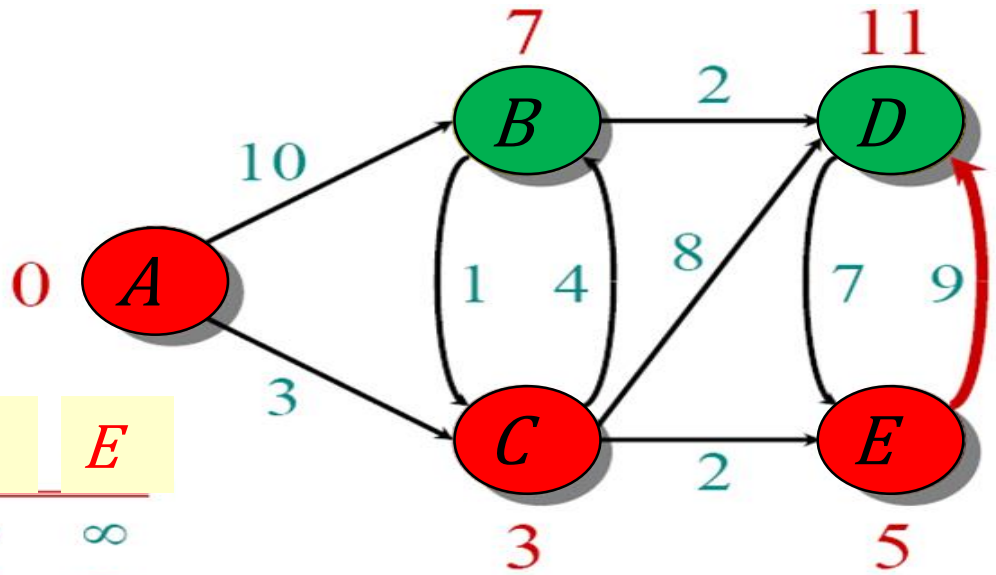
Dijkstra Animated Example



Dijkstra Animated Example



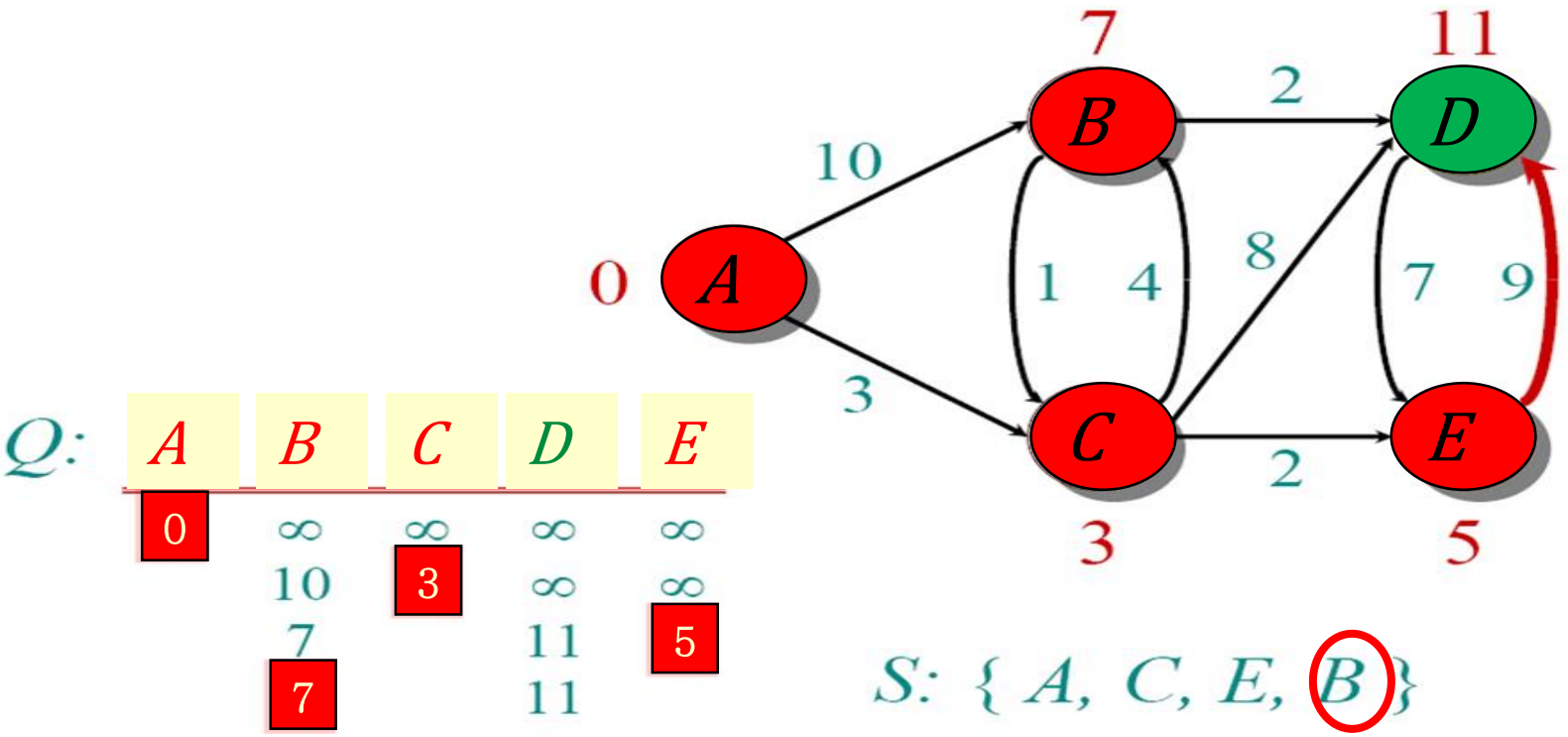
Dijkstra Animated Example



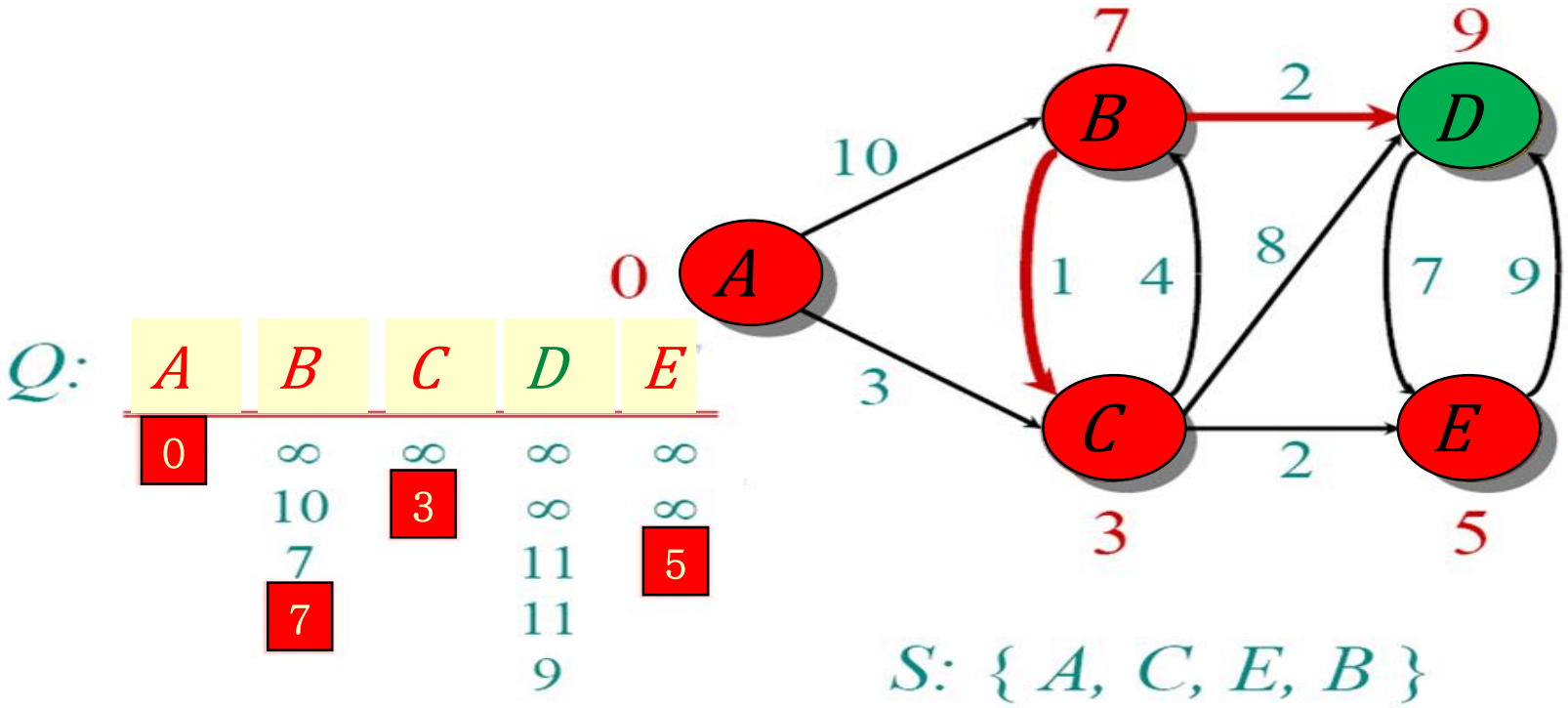
Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	

$S: \{ A, C, E \}$

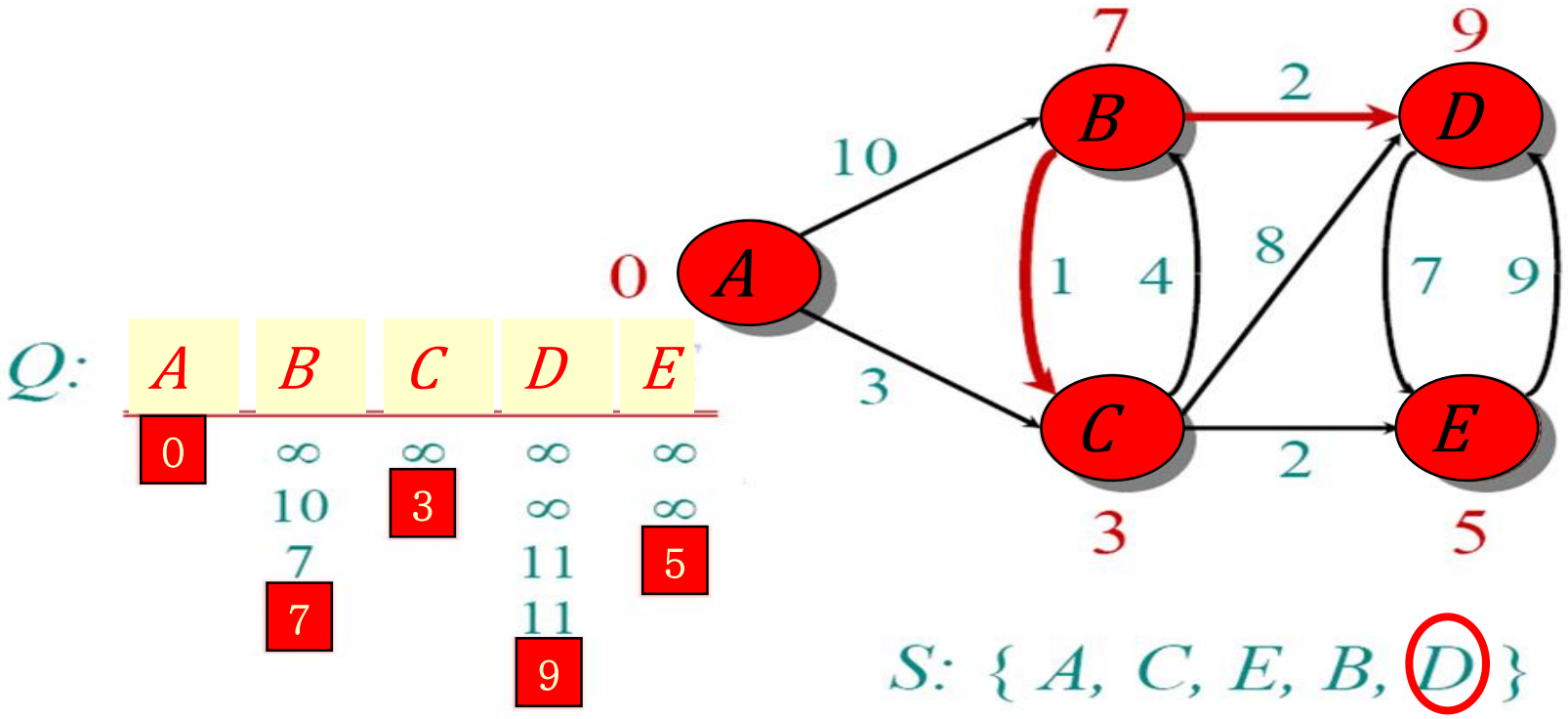
Dijkstra Animated Example



Dijkstra Animated Example

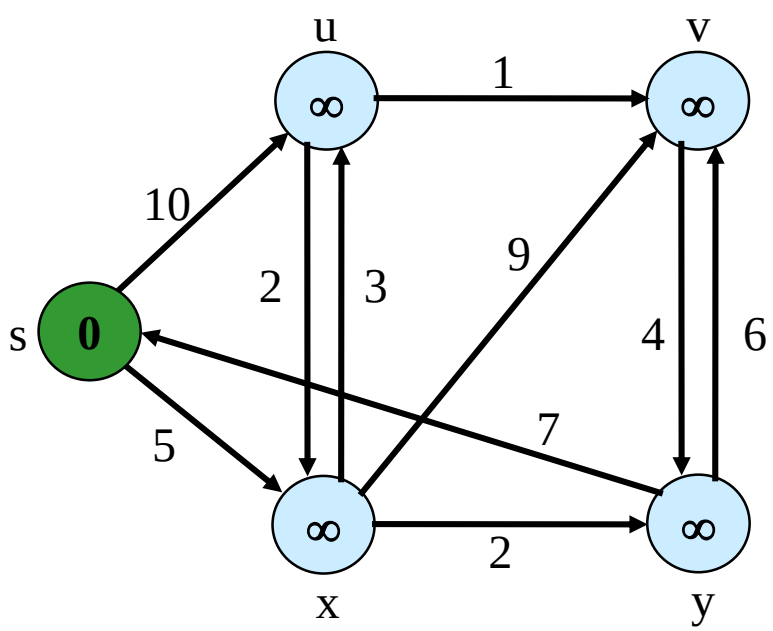


Dijkstra Animated Example



Dijkstra's Algorithm
다익스트라의 알고리즘 예3

Example

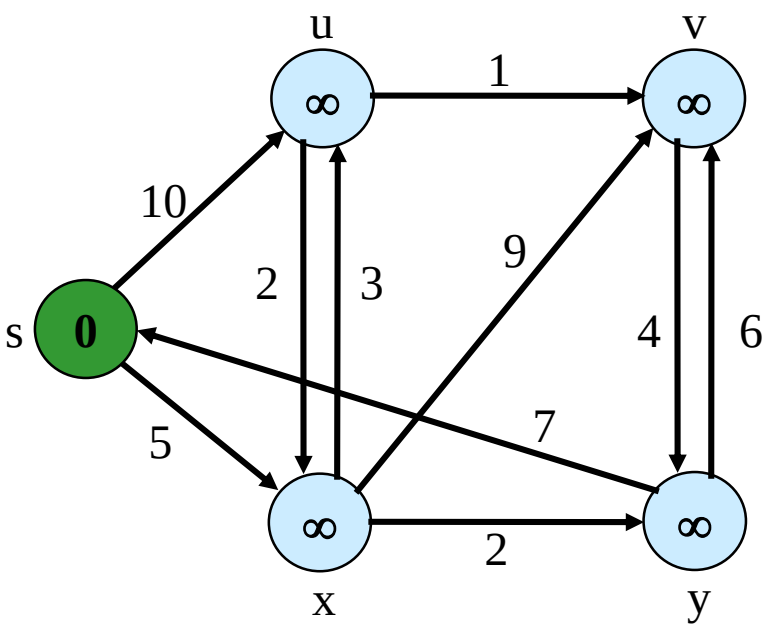


Q : S U X V Y

0 ∞ ∞ ∞ ∞

S : { S }

Example



Q : S U X V Y

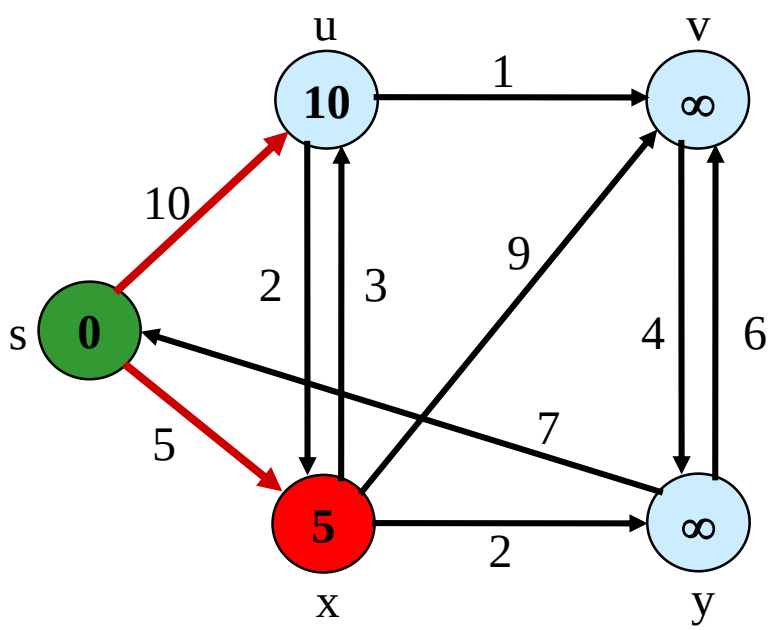
0 ∞ ∞ ∞ ∞

99:infinite

0

	S	**X**	**U**	**V**	**y**
S	0	5	10	99	99
X	99	0	3	9	2
U	99	2	0	1	99
v	99	99	99	0	4
y	7	99	99	6	0

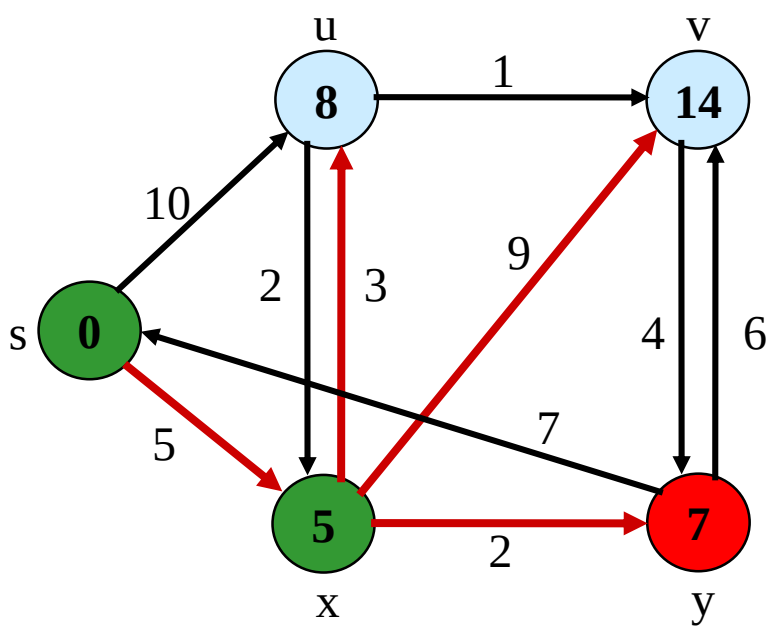
Example



Q : S	U	X	V	Y
0	∞	∞	∞	∞
10	5	∞	∞	

S : { S, X }

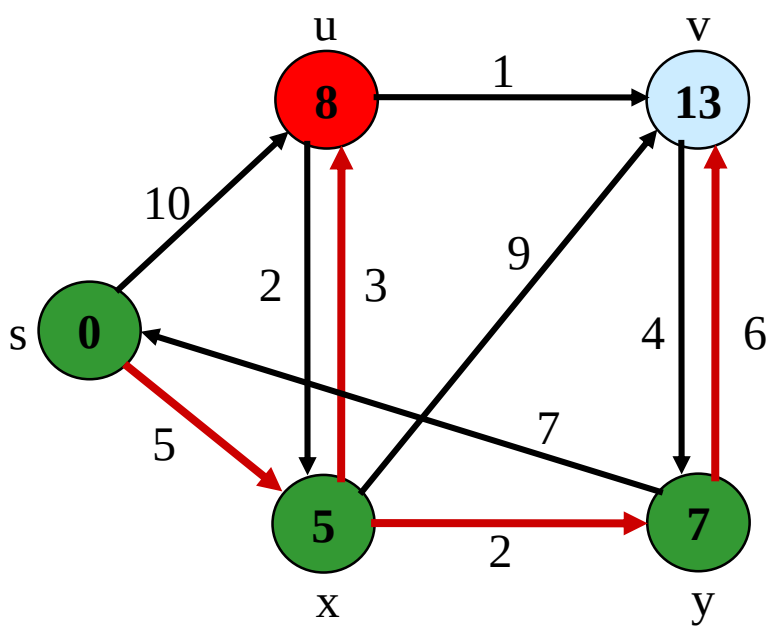
Example



Q : S	U	X	V	Y
0	∞	∞	∞	∞
	10	5	∞	∞
	8		14	7

S : { S, X, Y }

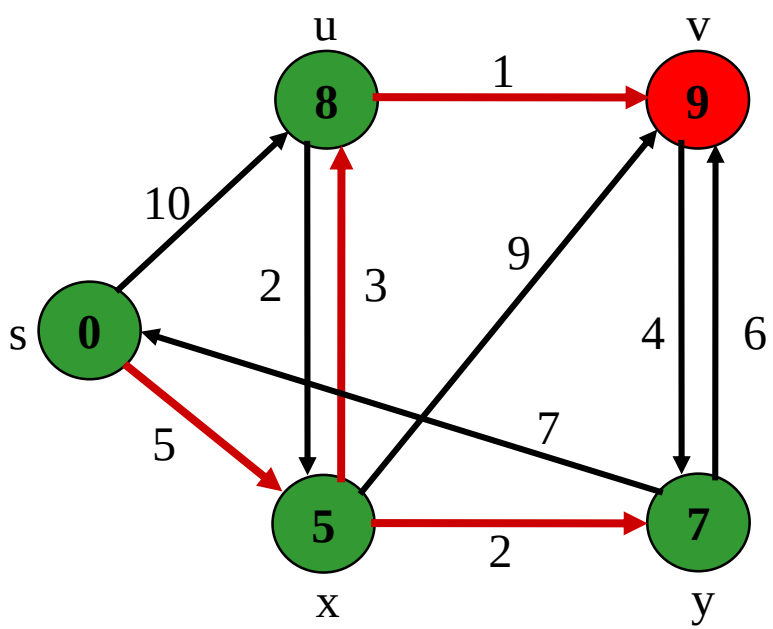
Example



Q : S	U	X	V	Y
0	∞	∞	∞	∞
	10	5	∞	∞
	8		14	7
	8		13	

S : { S, X, Y, U }

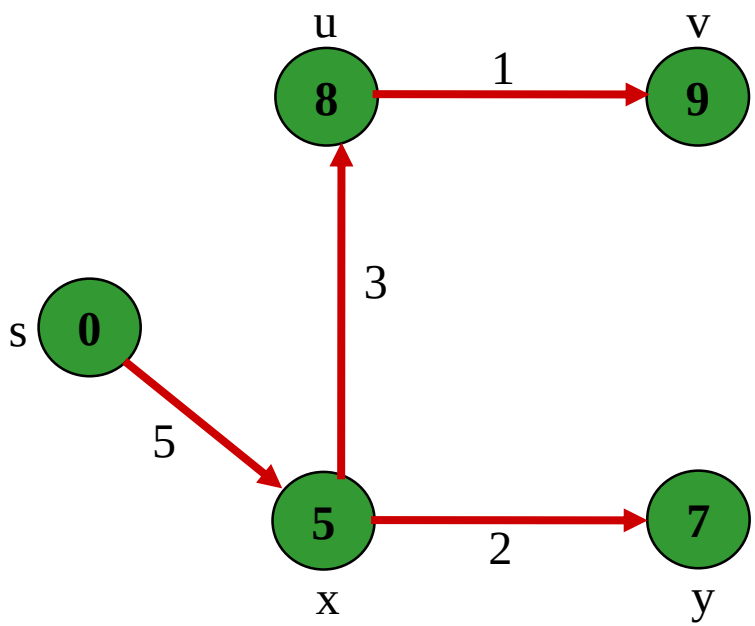
Example



Q : S	U	X	V	Y
0	∞	∞	∞	∞
	10	5	∞	∞
	8		14	7
	8		13	
			9	

S : { S, X, Y, U, V }

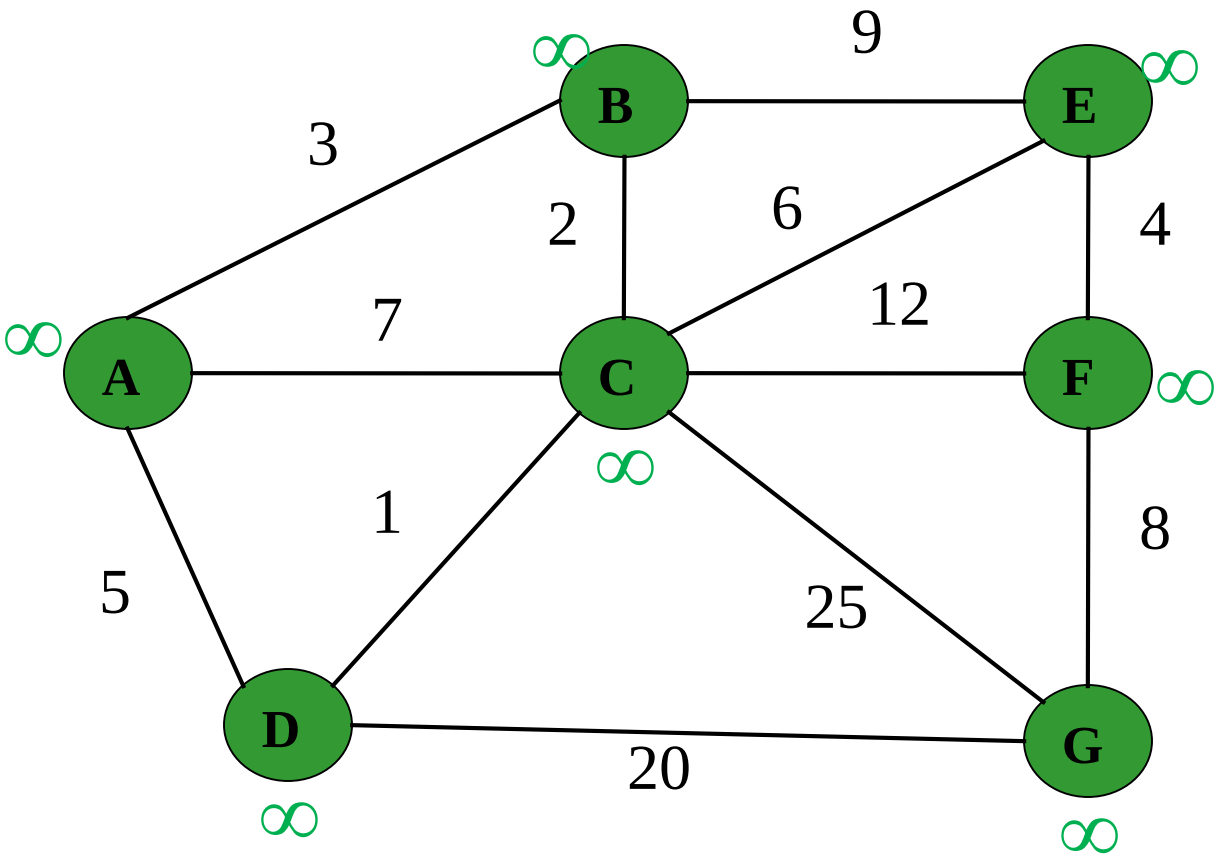
Example



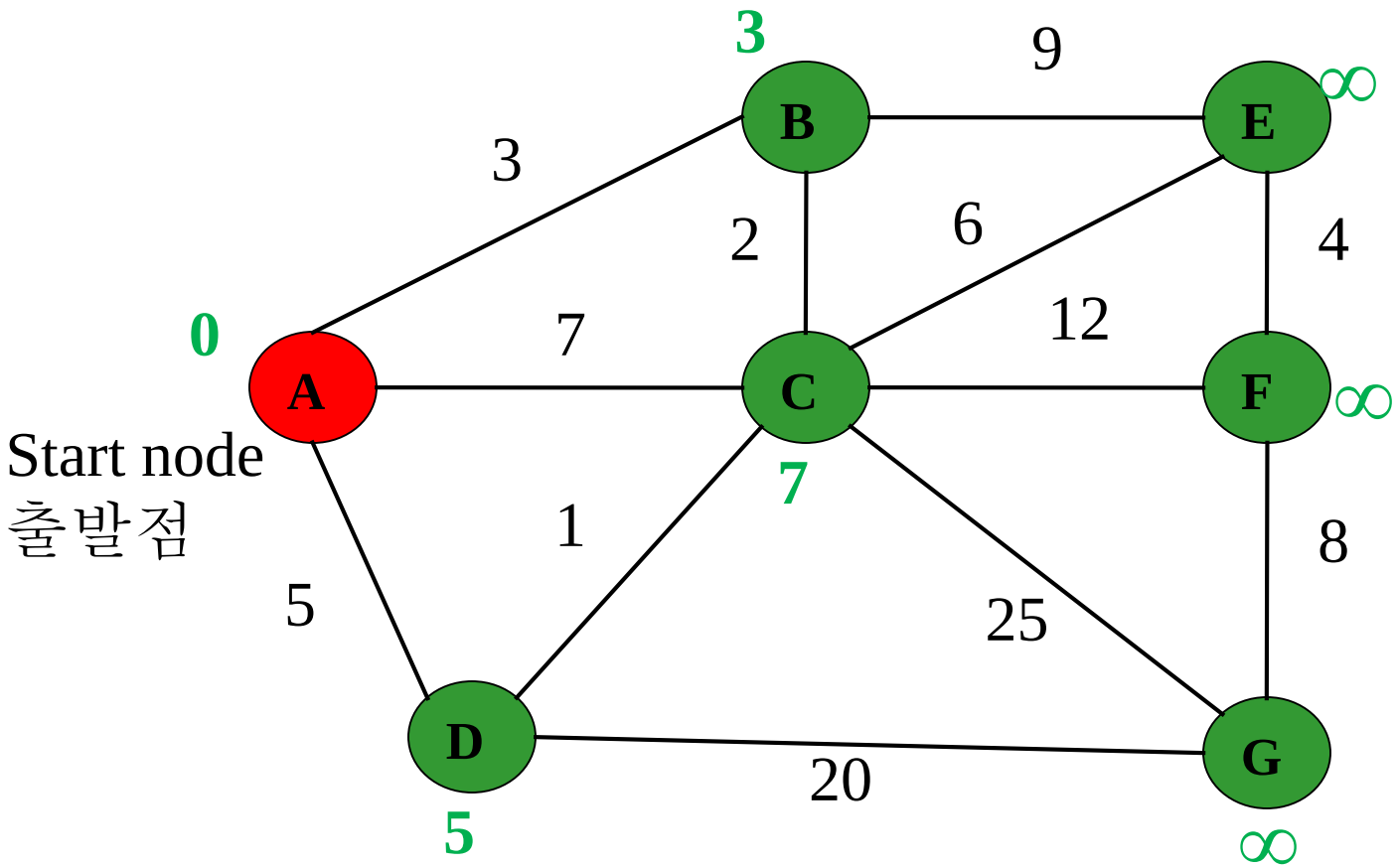
S : { S, X, Y, U, V }

Dijkstra's Algorithm
다익스트라의 알고리즘 예4

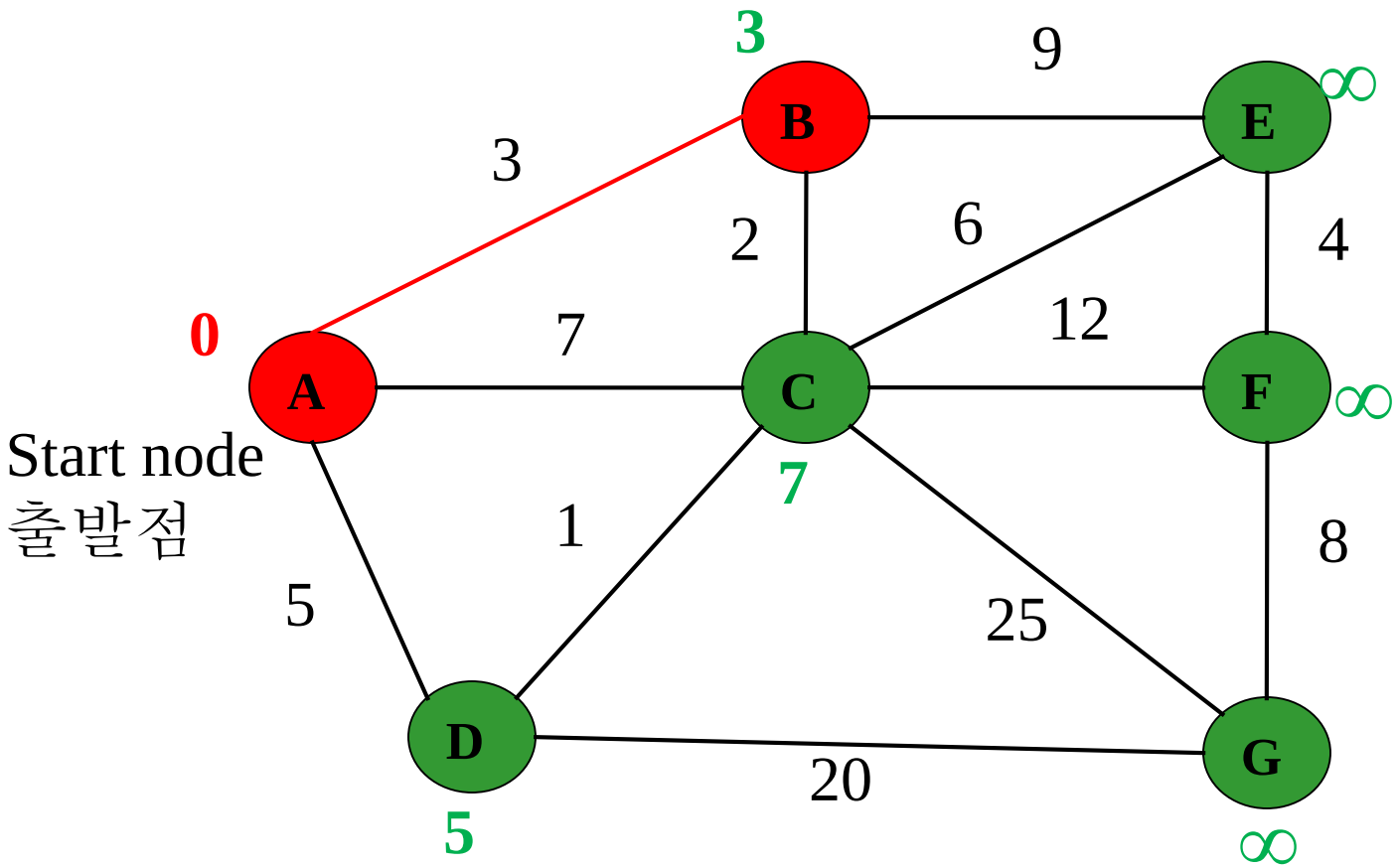
Dijkstra's algorithm



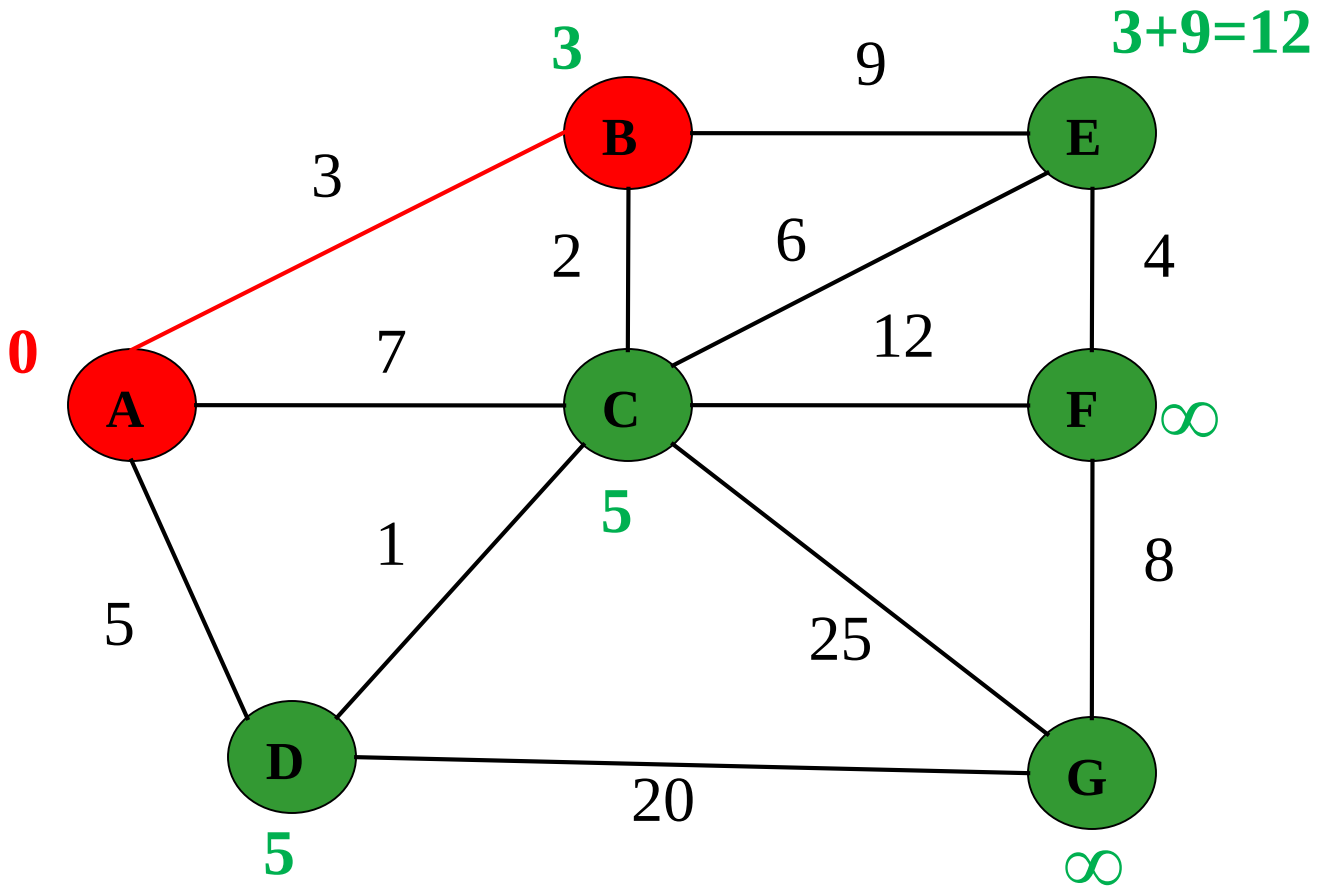
Dijkstra's algorithm



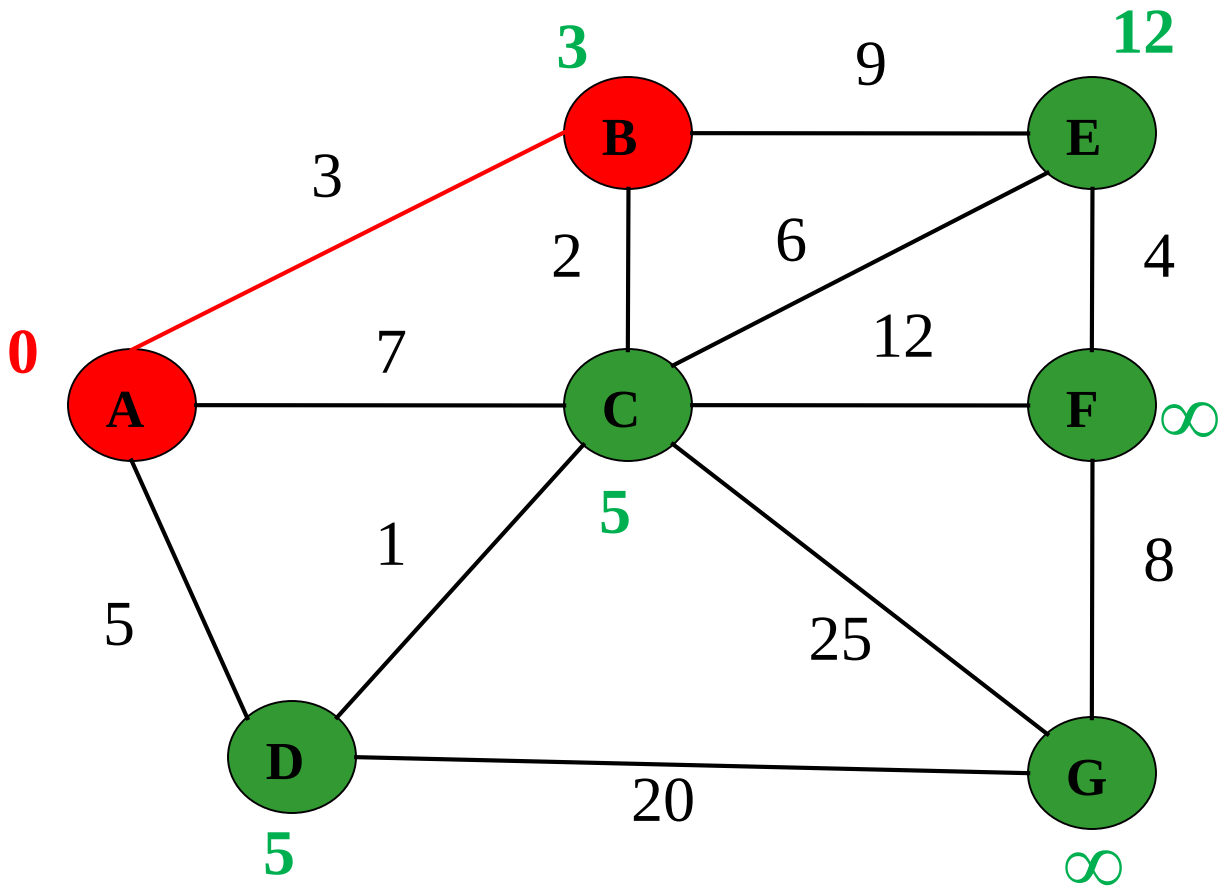
Dijkstra's algorithm



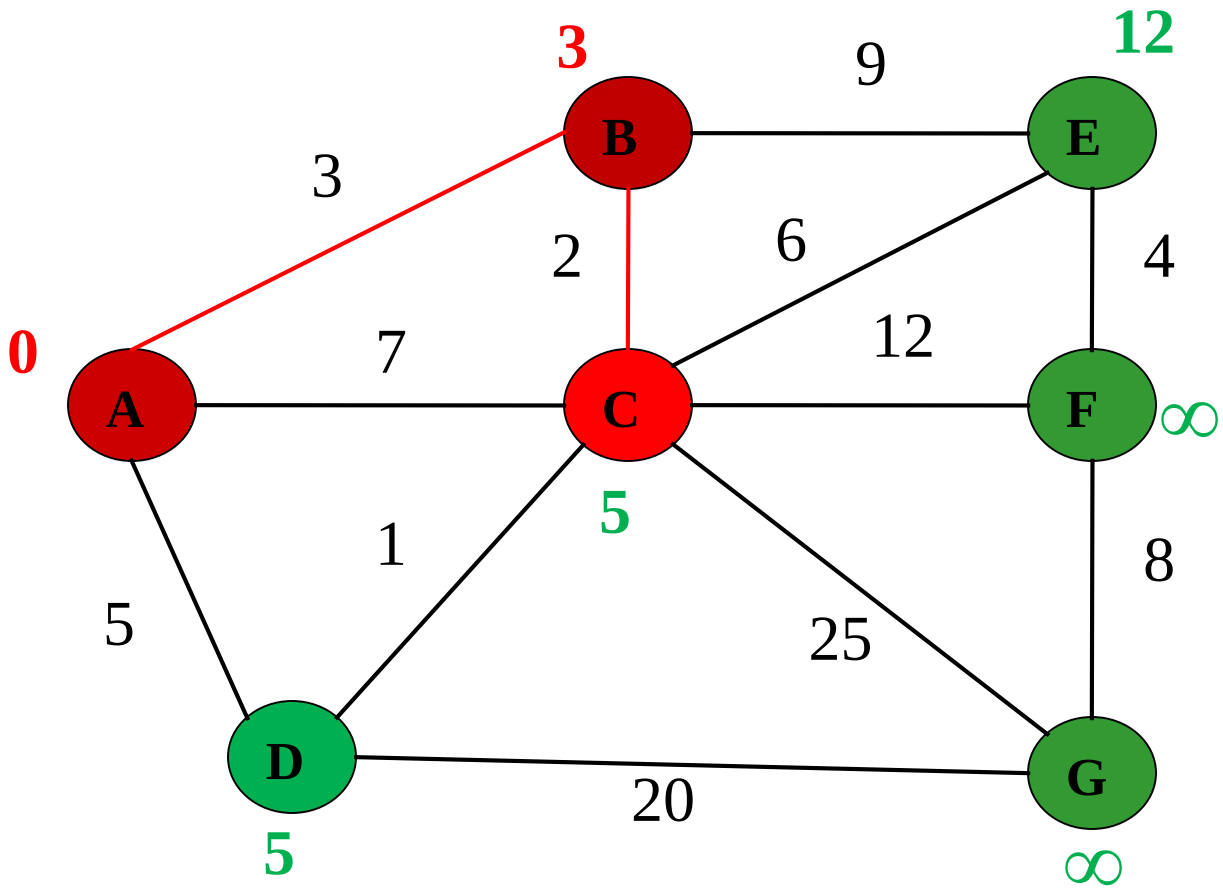
Dijkstra's algorithm



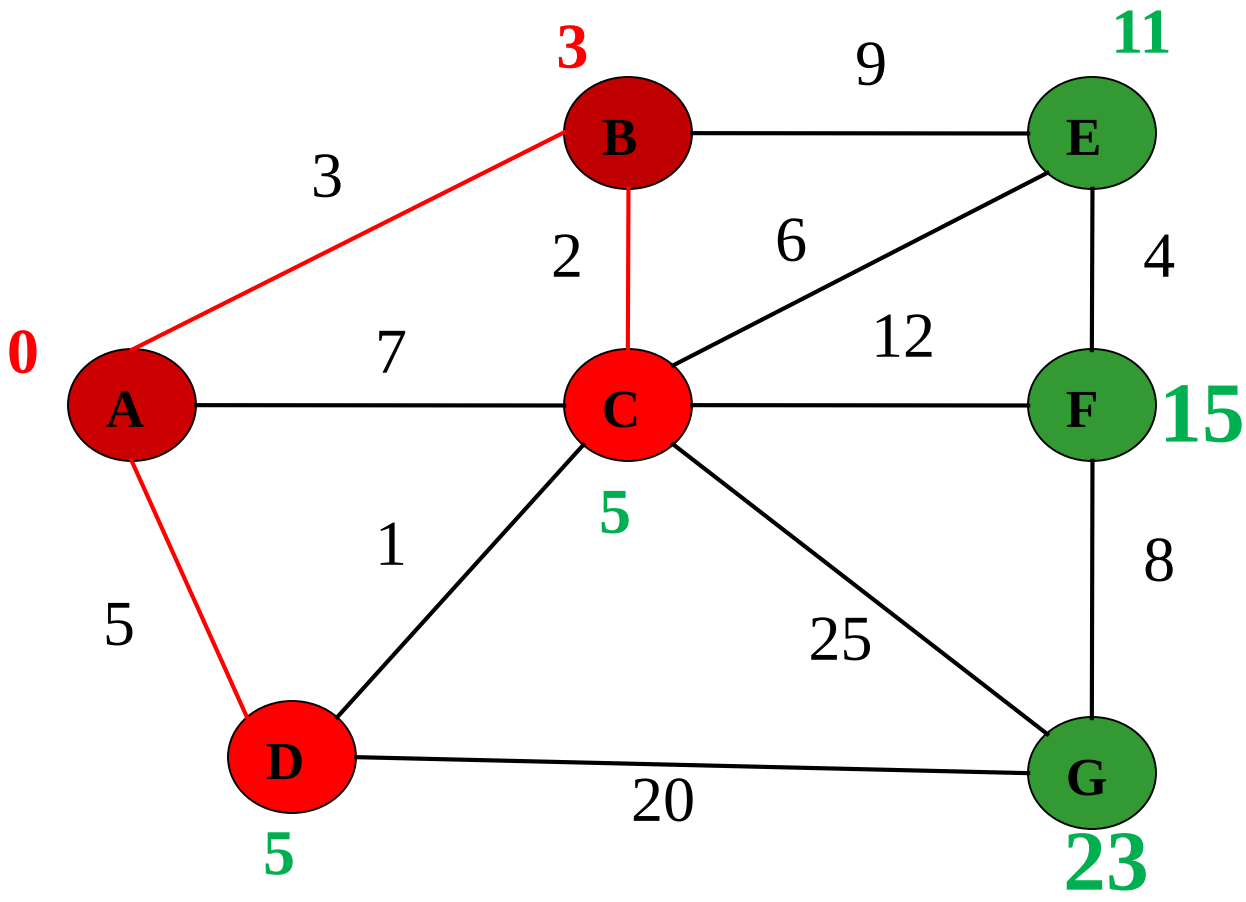
Dijkstra's algorithm



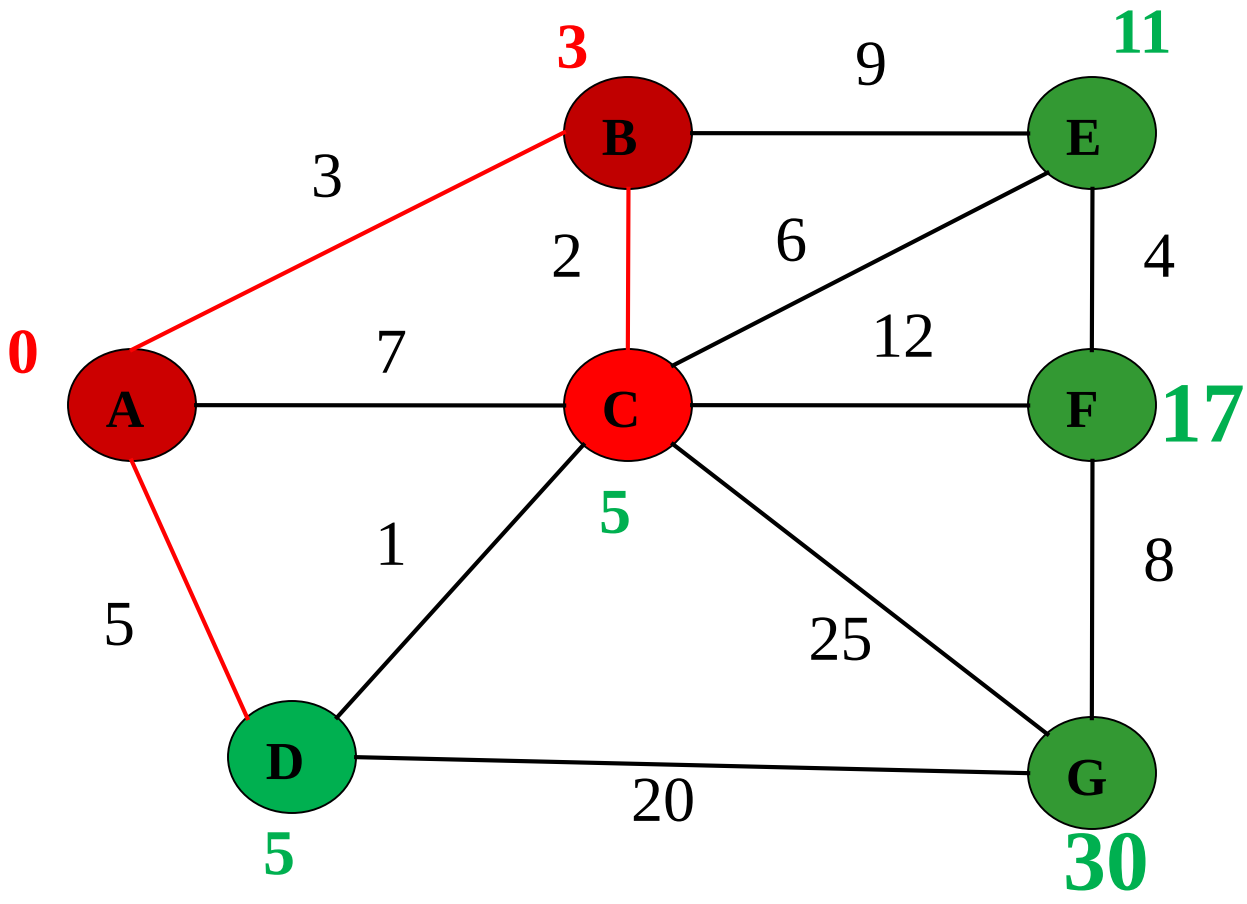
Dijkstra's algorithm



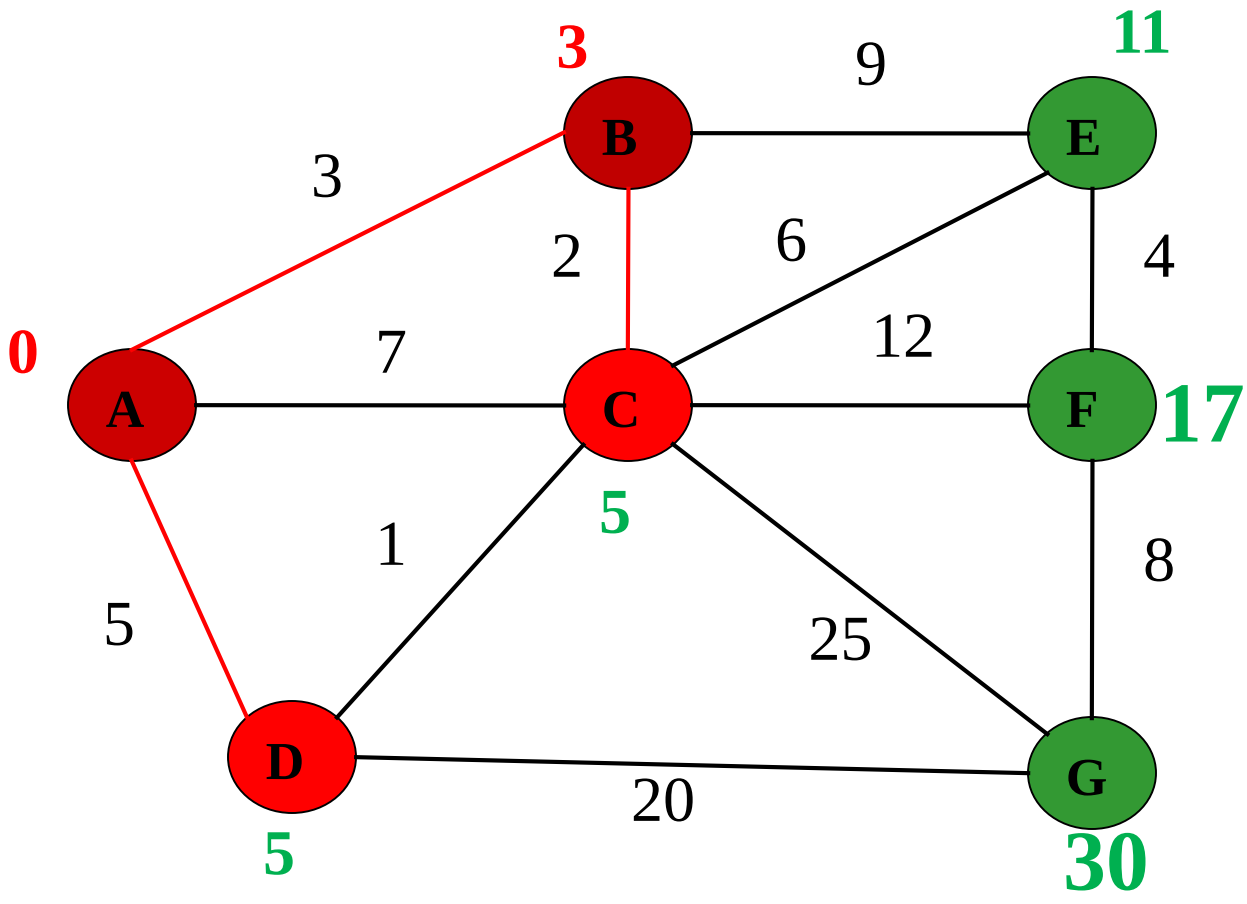
Dijkstra's algorithm



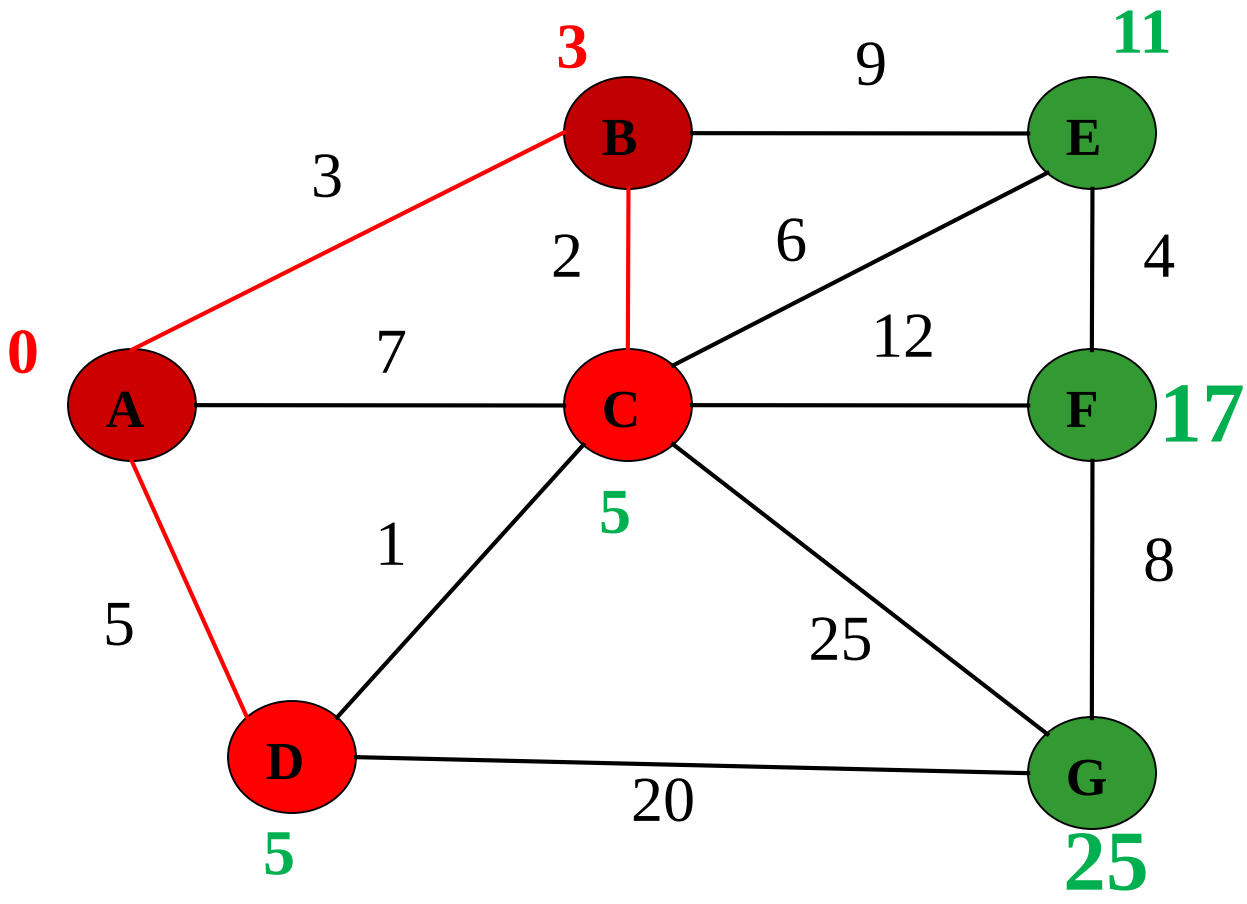
Dijkstra's algorithm



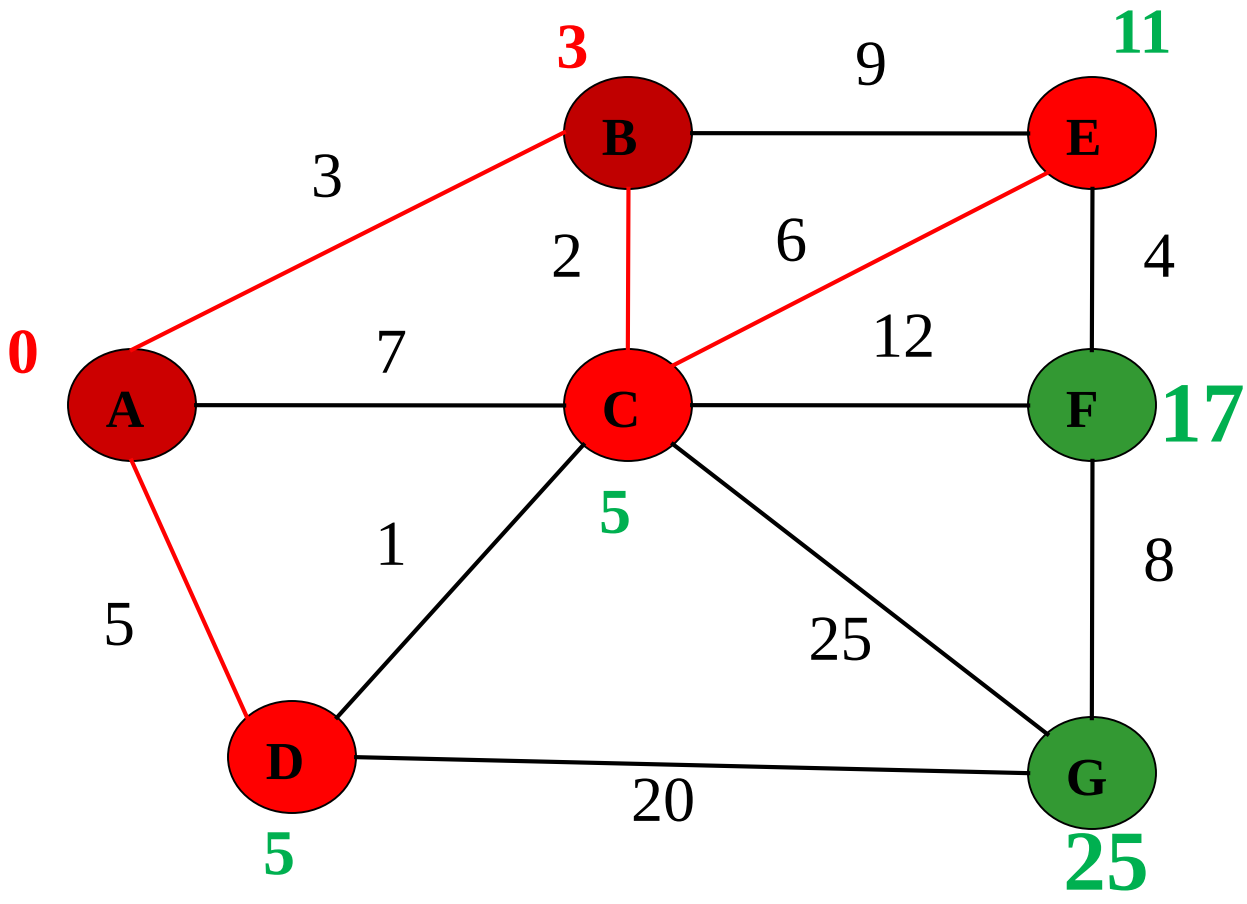
Dijkstra's algorithm



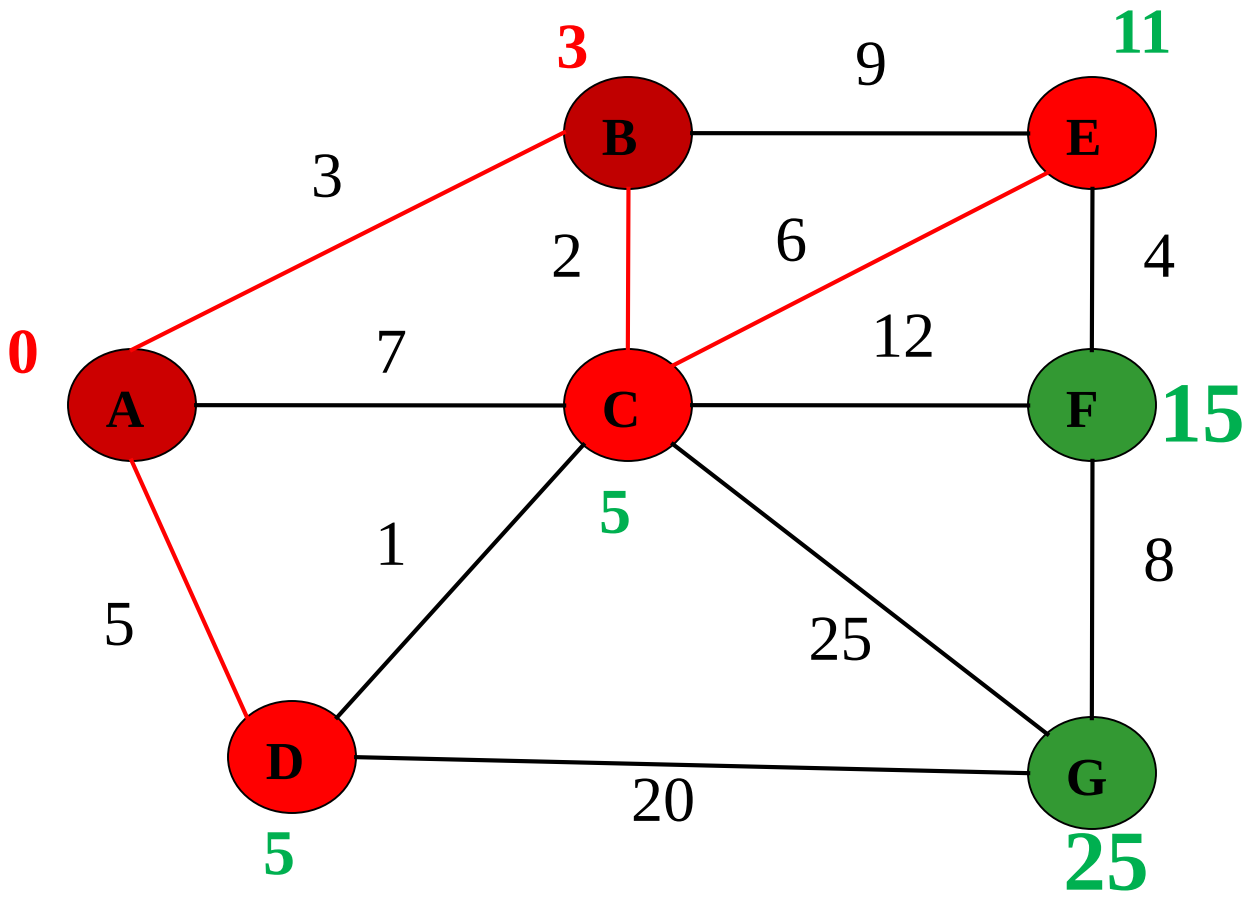
Dijkstra's algorithm



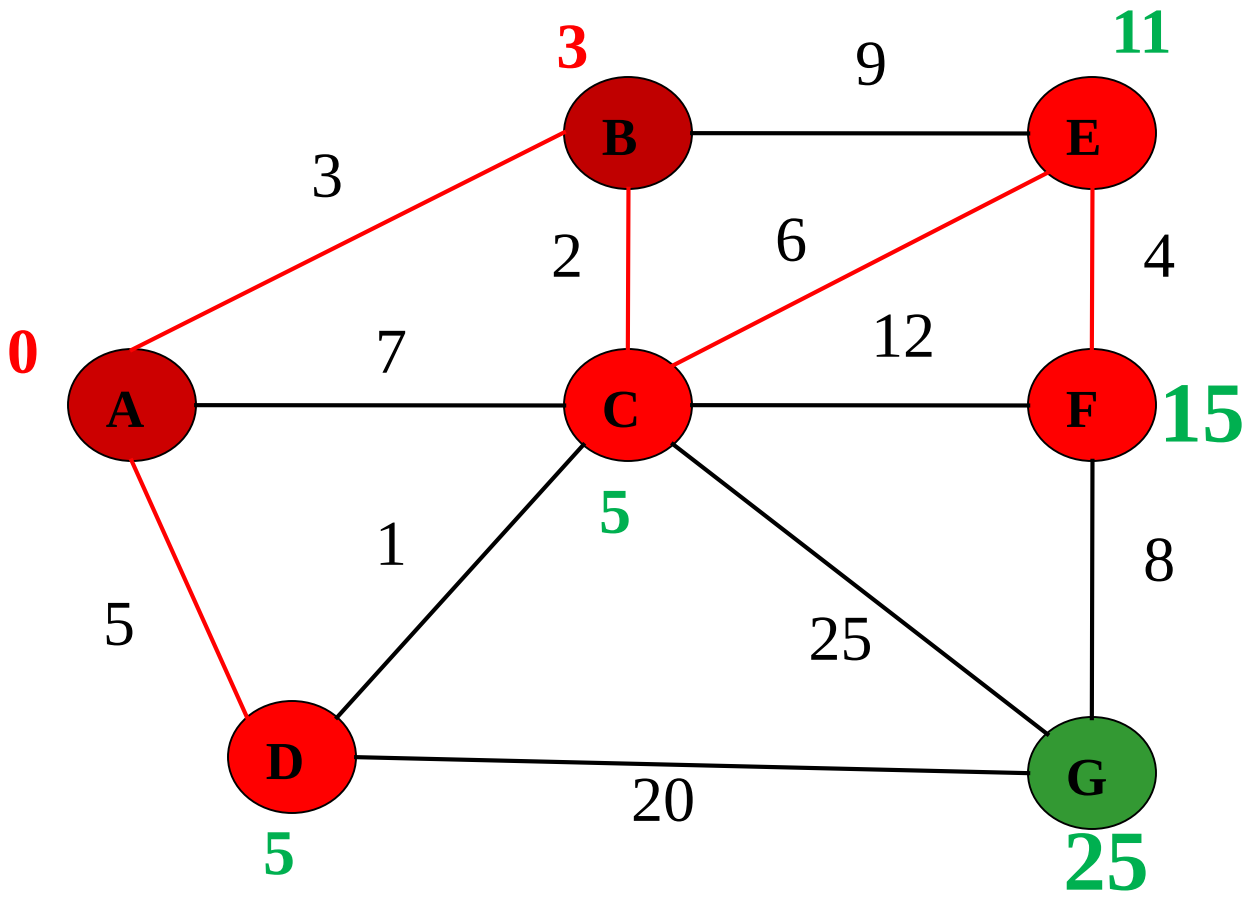
Dijkstra's algorithm



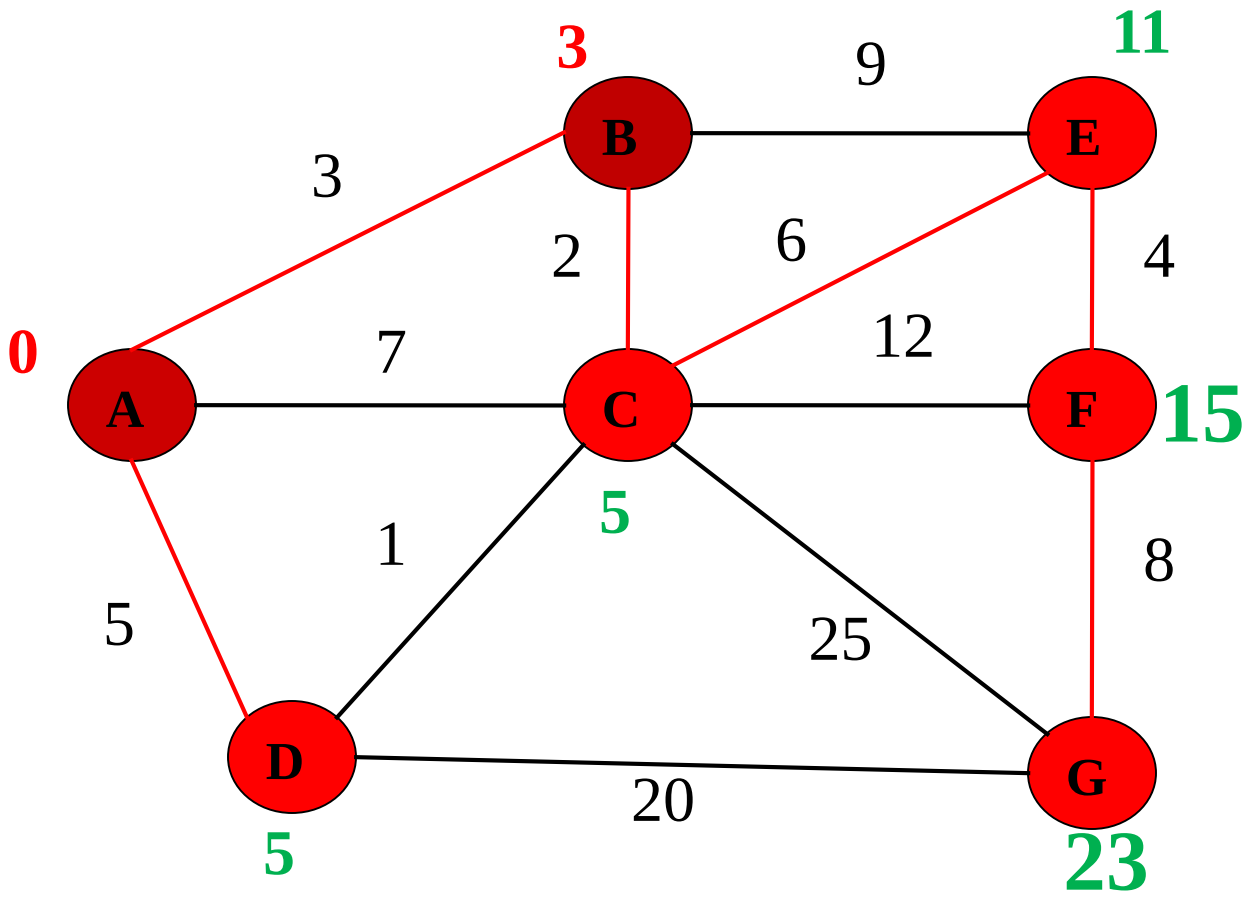
Dijkstra's algorithm



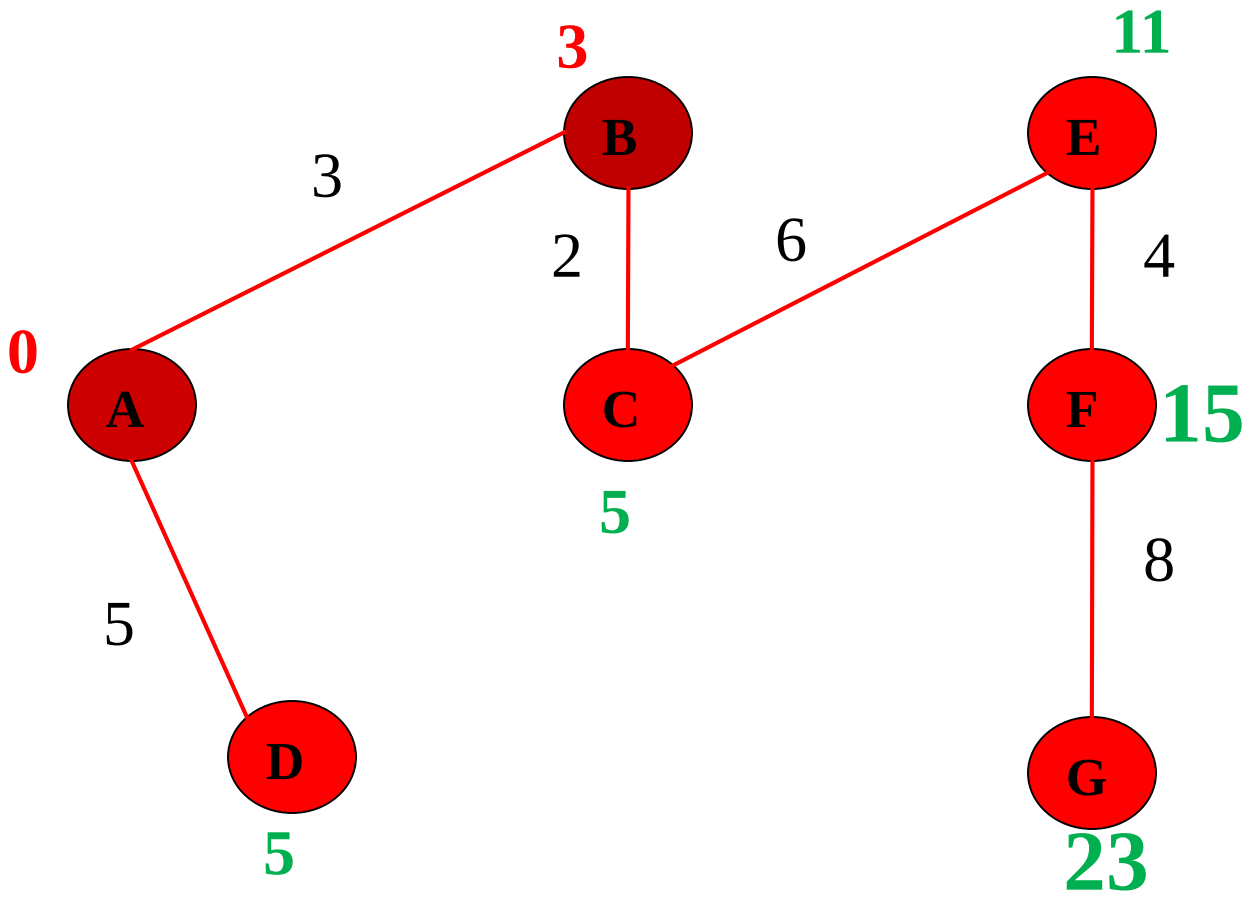
Dijkstra's algorithm



Dijkstra's algorithm



Dijkstra's algorithm

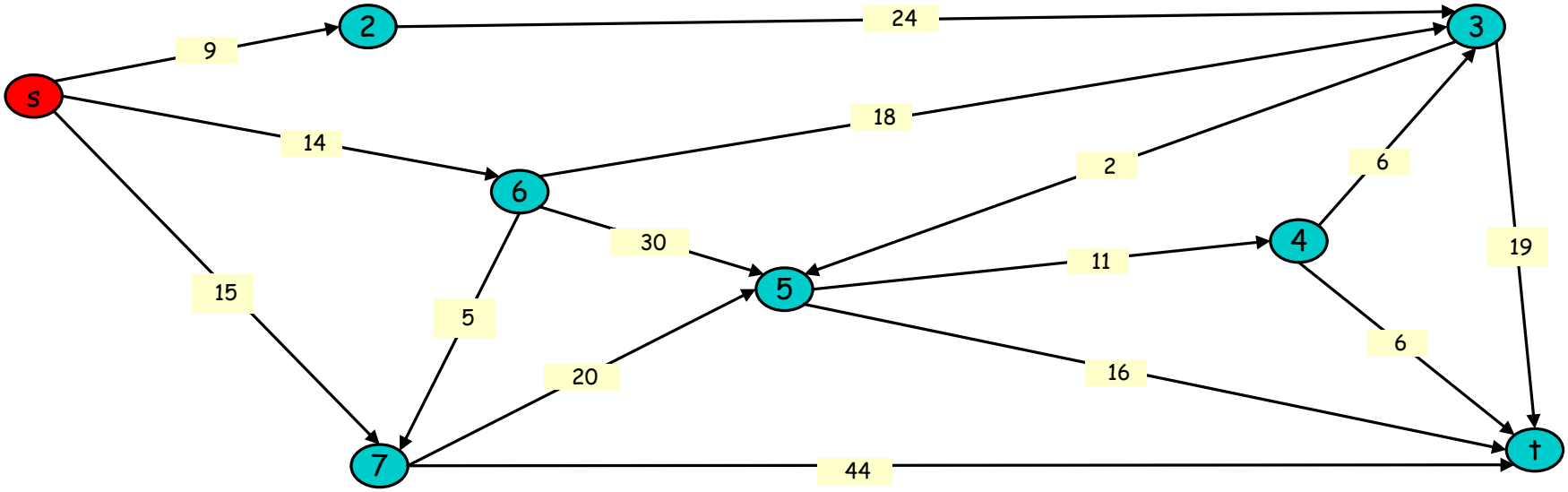


Spanning Tree (간선트리)

Dijkstra's Algorithm
다익스트라의 알고리즘 예5

Dijkstra's Shortest Path Algorithm

- Find shortest path from s to t.
- BFS(너비우선탐색)

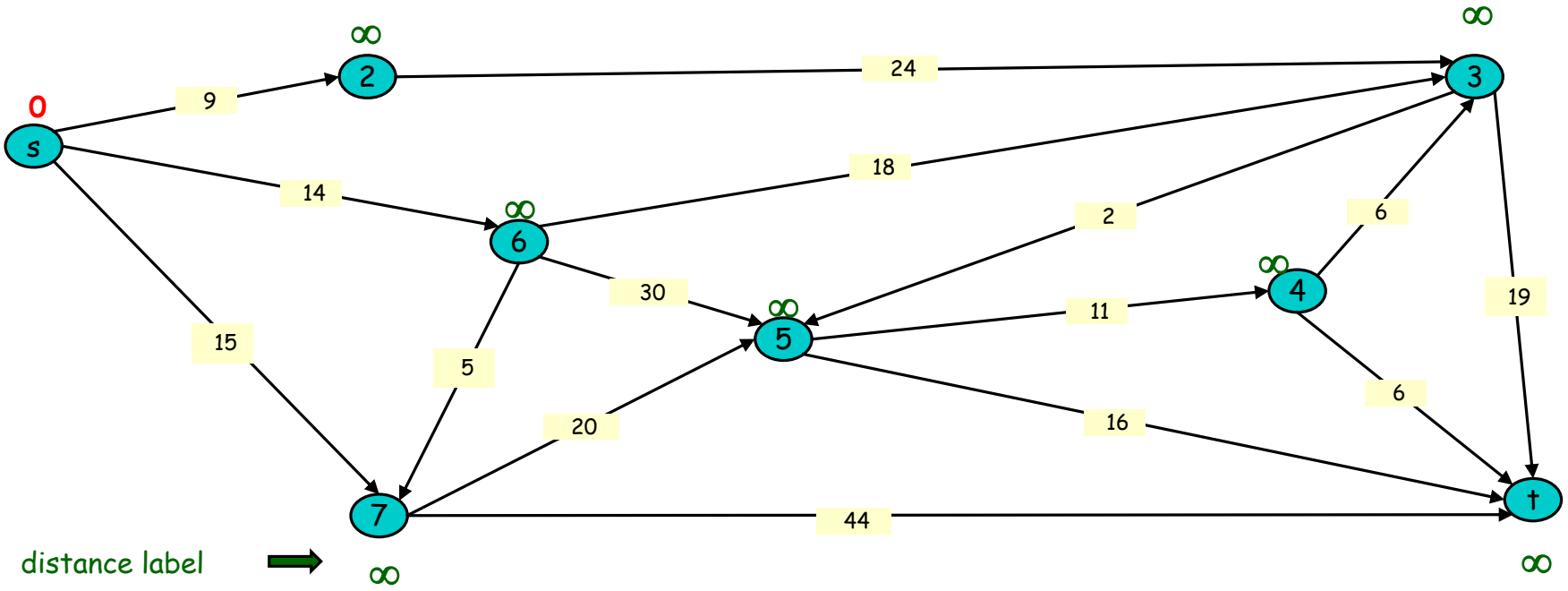


Dijkstra's Shortest Path Algorithm

S	2	3	4	5	6	7	t
0	∞	∞	∞	∞	∞	∞	∞

S = { } 해답집합

PQ = { s, 2, 3, 4, 5, 6, 7, t } 우선순위큐

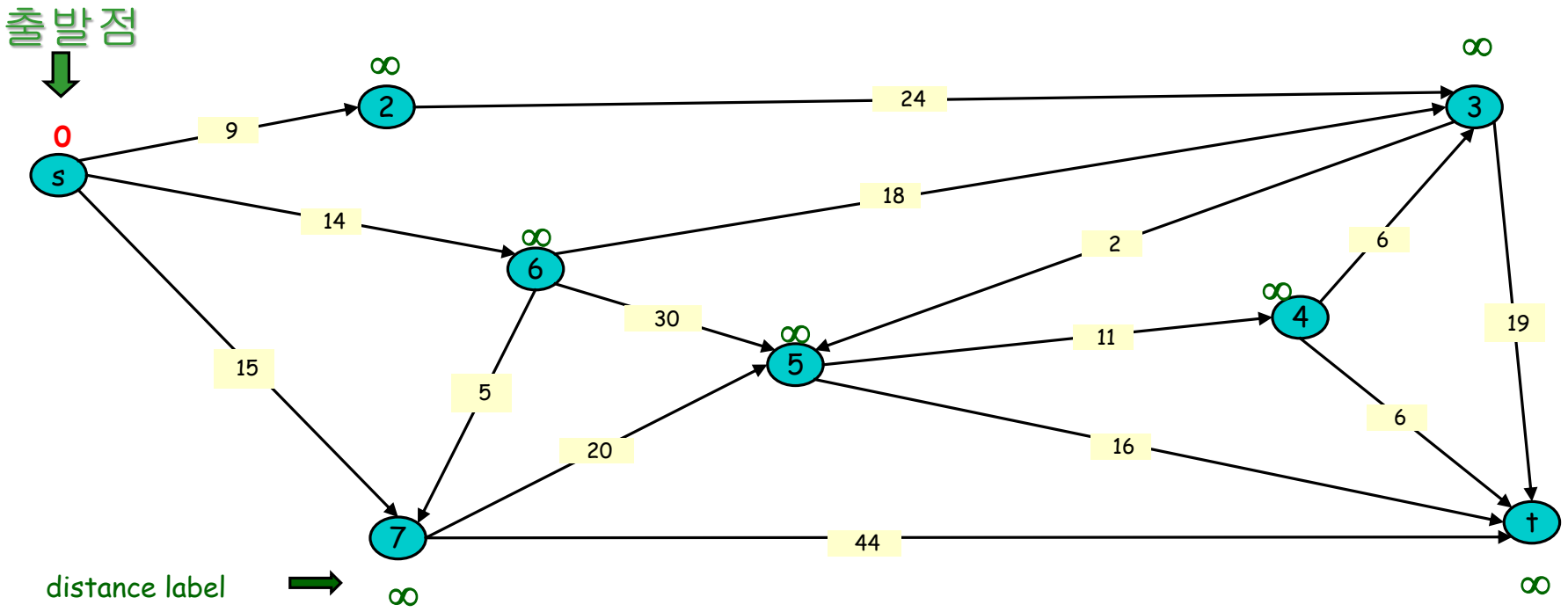


Dijkstra's Shortest Path Algorithm

S	2	3	4	5	6	7	t
0	∞	∞	∞	∞	∞	∞	∞

S = { }

PQ = { s, 2, 3, 4, 5, 6, 7, t }

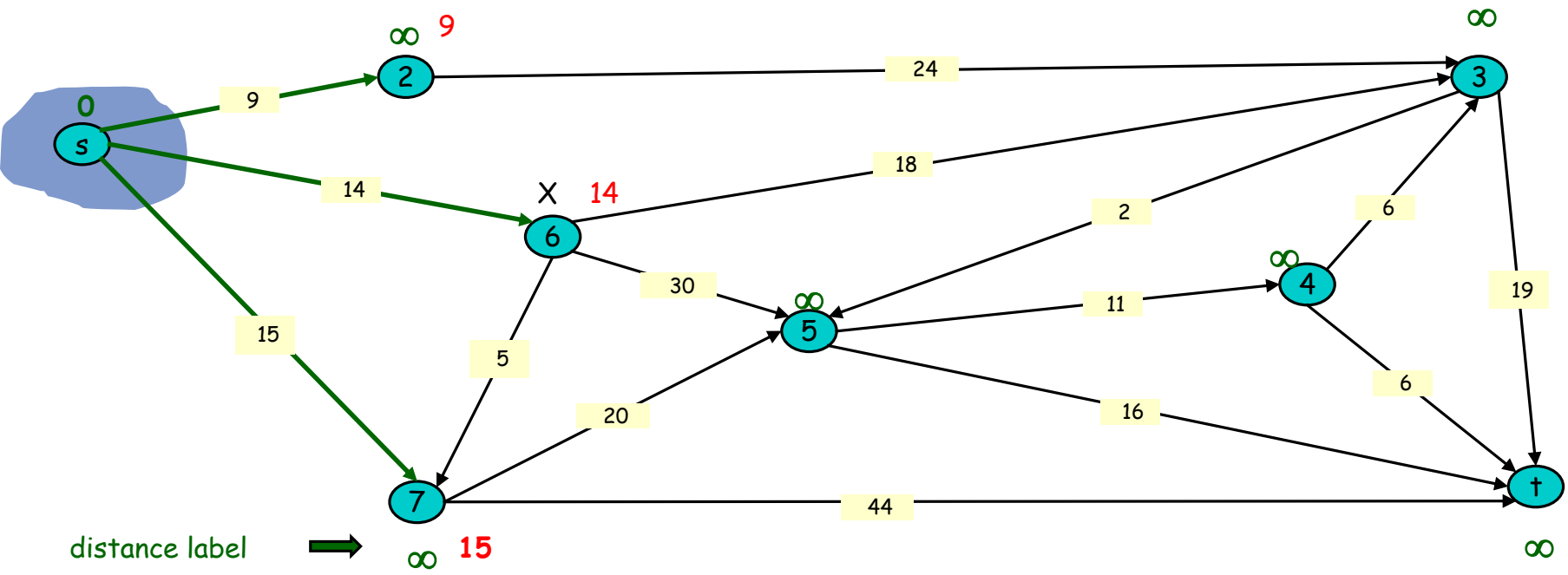


Dijkstra's Shortest Path Algorithm

S	2	3	4	5	6	7	t
0	9	∞	∞	∞	14	15	∞

S = { s }

PQ = { 2, 3, 4, 5, 6, 7, t }

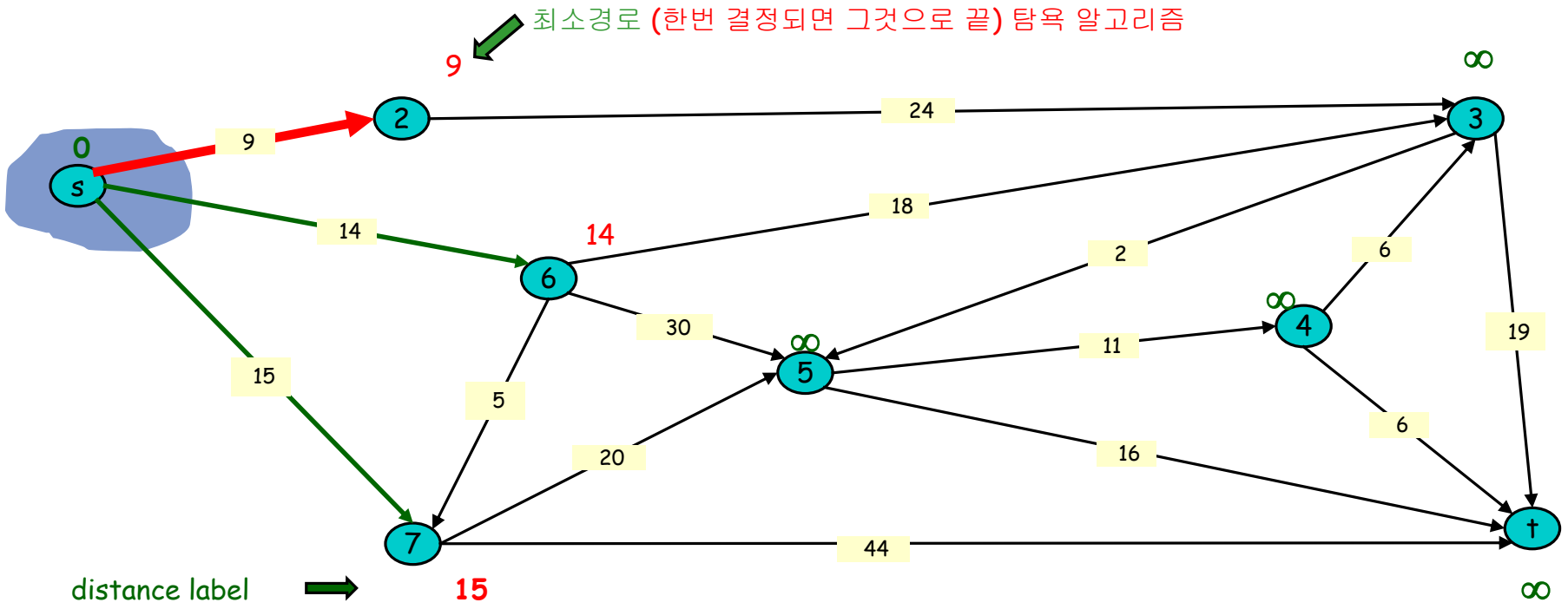


Dijkstra's Shortest Path Algorithm

S	2	3	4	5	6	7	t
0	9	∞	∞	∞	14	15	∞

S = { s }

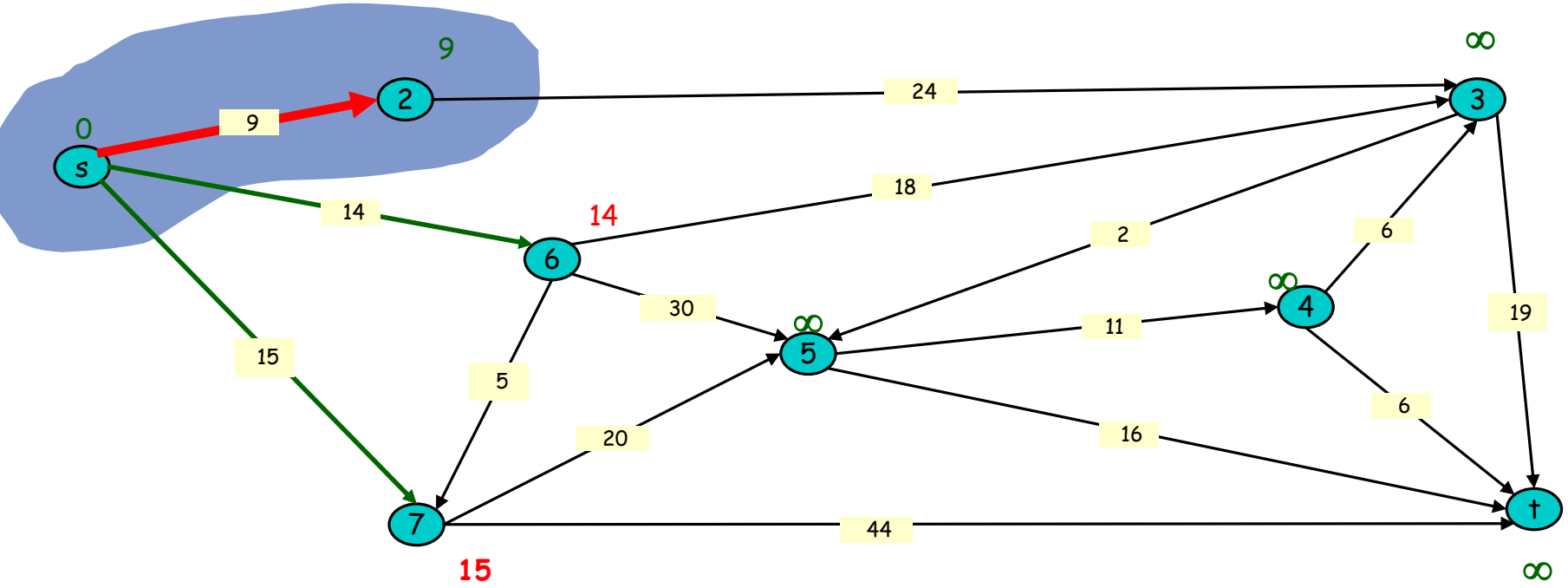
PQ = { 2, 3, 4, 5, 6, 7, t } 우선순위큐



Dijkstra's Shortest Path Algorithm

S	2	3	4	5	6	7	t
0	9	∞	∞	∞	14	15	∞

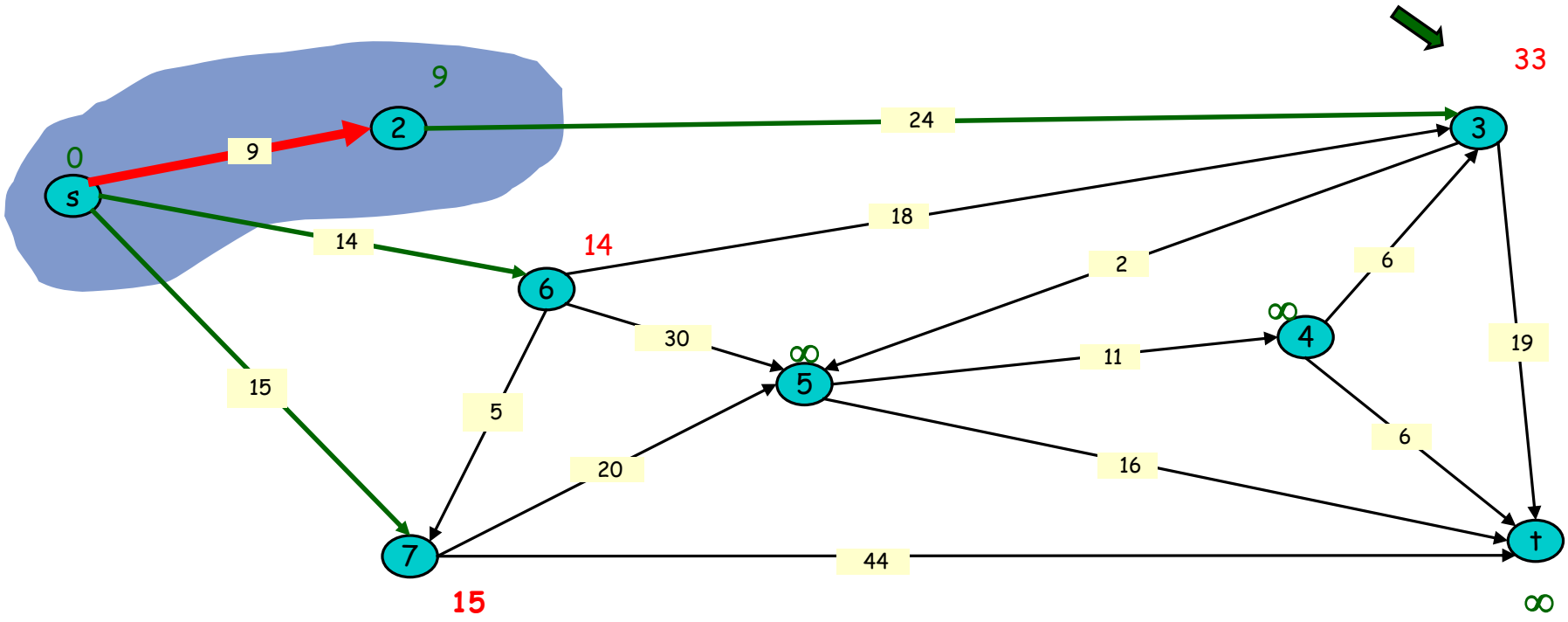
S = { s, 2 }
PQ = { 3, 4, 5, 6, 7, t }



Dijkstra's Shortest Path Algorithm

S	2	3	4	5	6	7	t
0	9	33	∞	∞	14	15	∞

S = { s, 2 }
PQ = { 3, 4, 5, 6, 7, t }

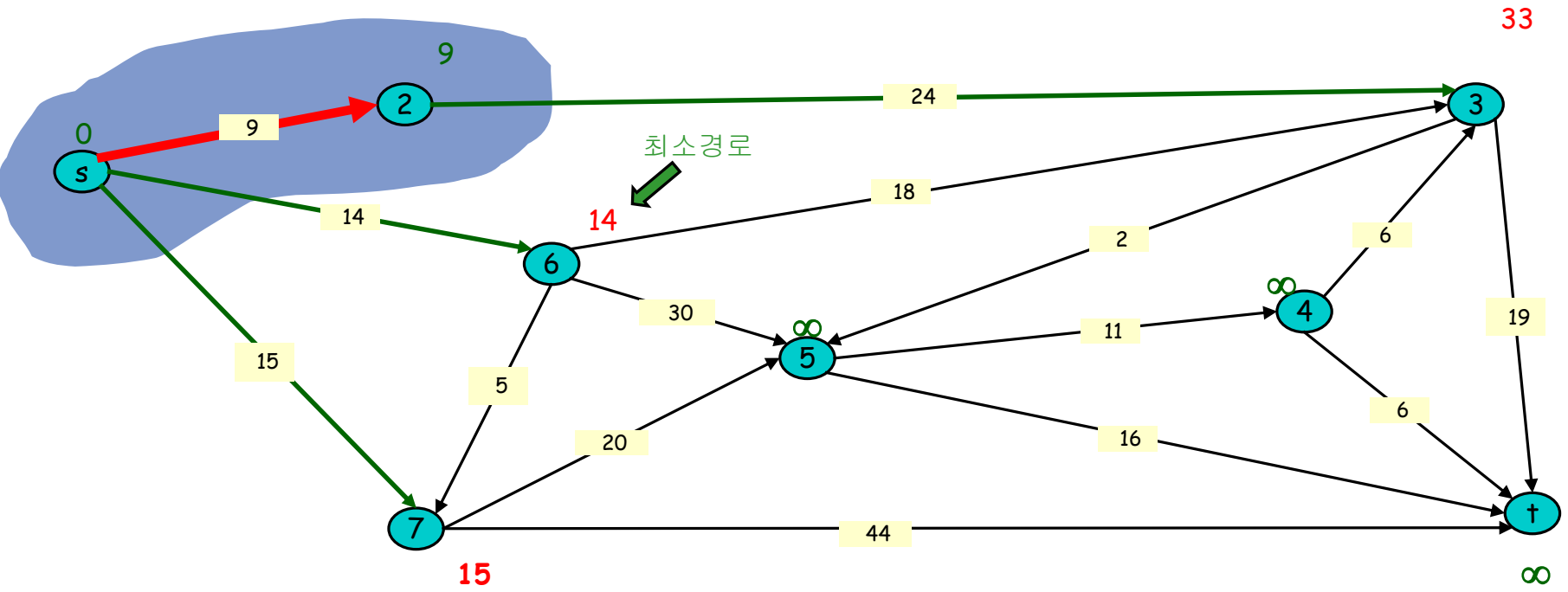


Dijkstra's Shortest Path Algorithm

S	2	3	4	5	6	7	t
0	9	33	∞	∞	14	15	∞

S = { s, 2 }

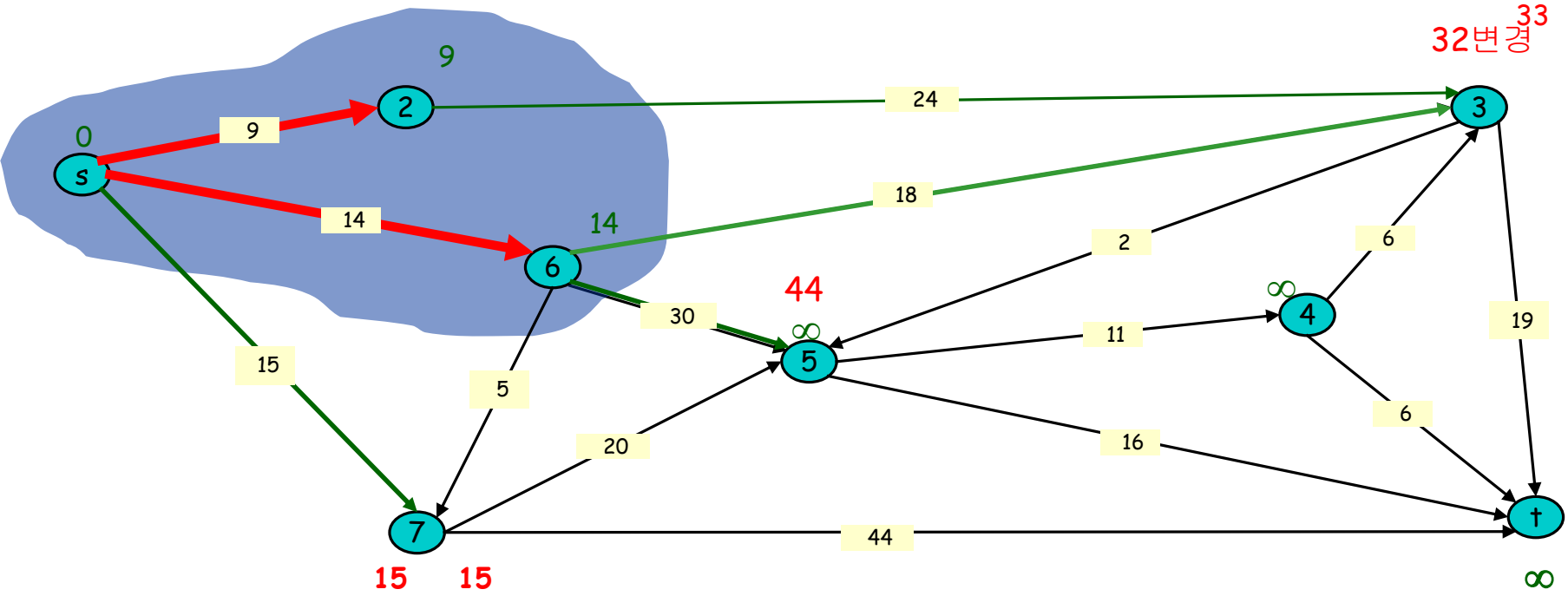
PQ = { 3, 4, 5, 6, 7, t }



Dijkstra's Shortest Path Algorithm

S	2	3	4	5	6	7	t
0	9	32	∞	44	14	15	∞

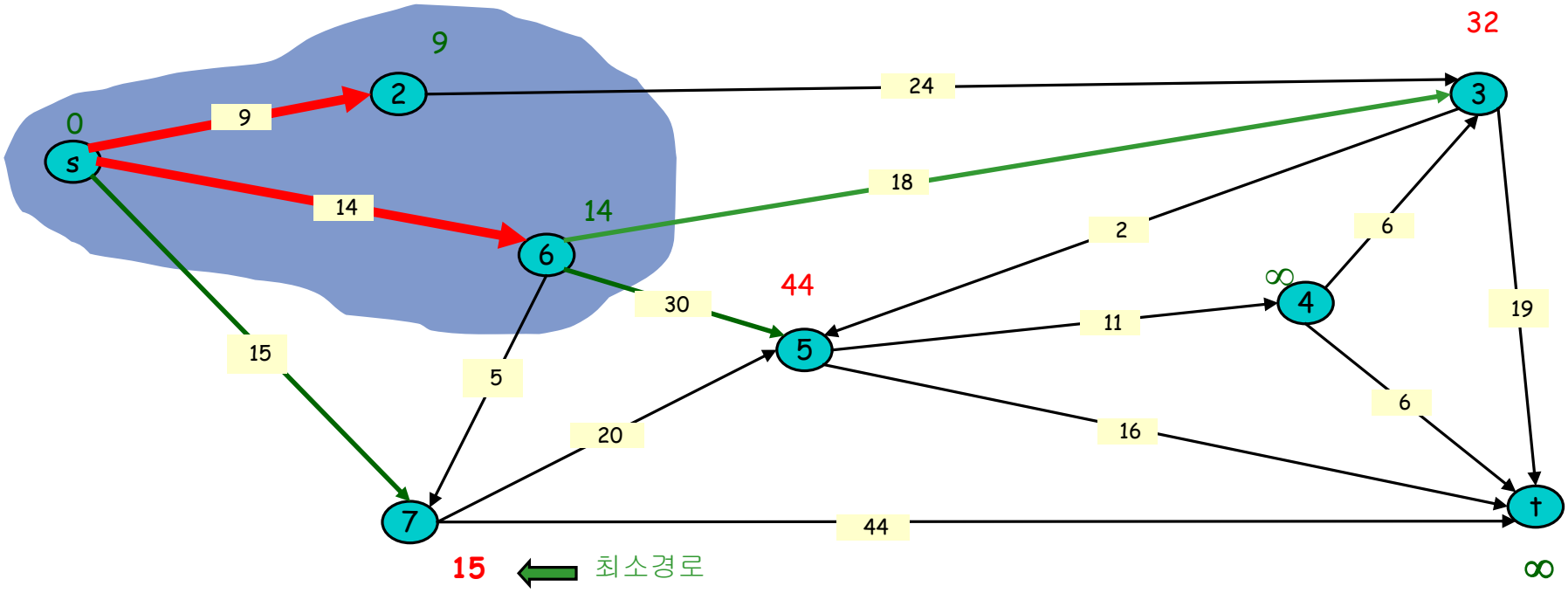
S = { s, 2, 6 }
PQ = { 3, 4, 5, 7, t }



Dijkstra's Shortest Path Algorithm

S	2	3	4	5	6	7	t
0	9	32	∞	44	14	15	∞

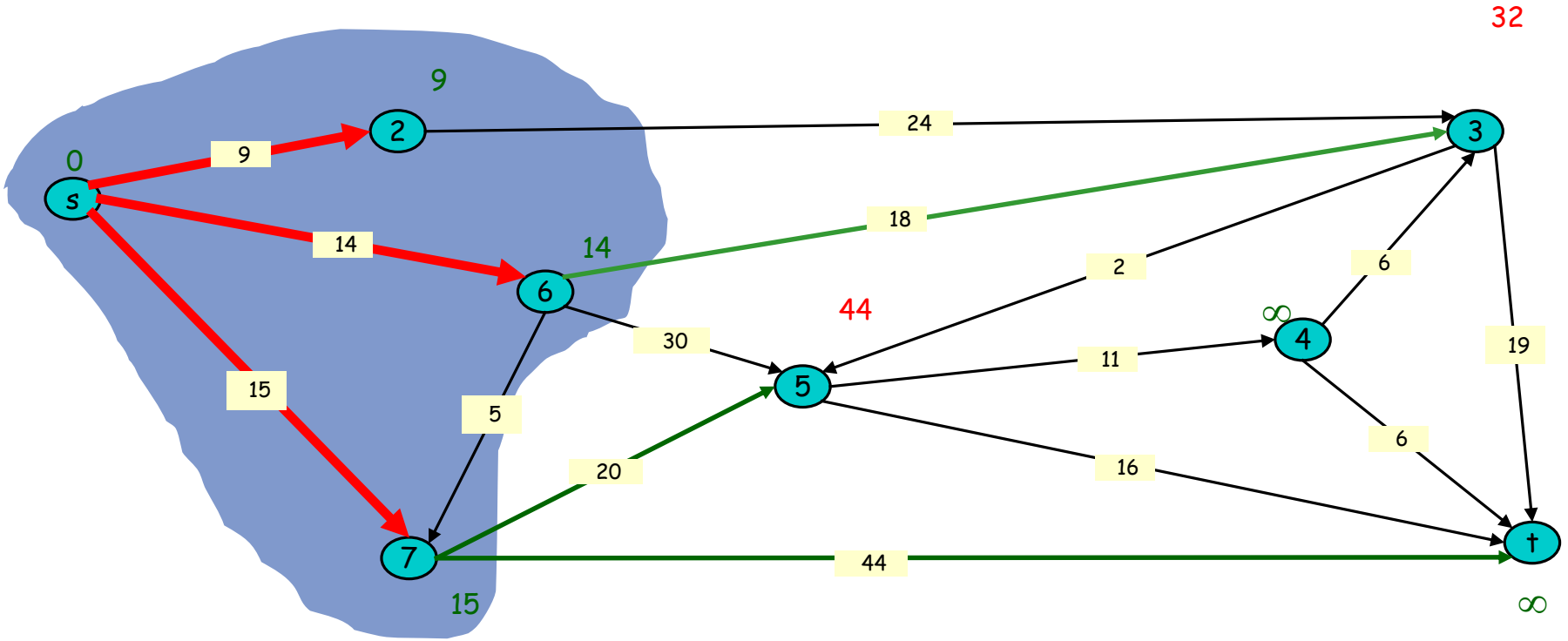
S = { s, 2, 6 }
PQ = { 3, 4, 5, 7, t }



Dijkstra's Shortest Path Algorithm

S	2	3	4	5	6	7	t
0	9	32	∞	35	14	15	59

S = { s, 2, 6, 7 }
PQ = { 3, 4, 5, t }

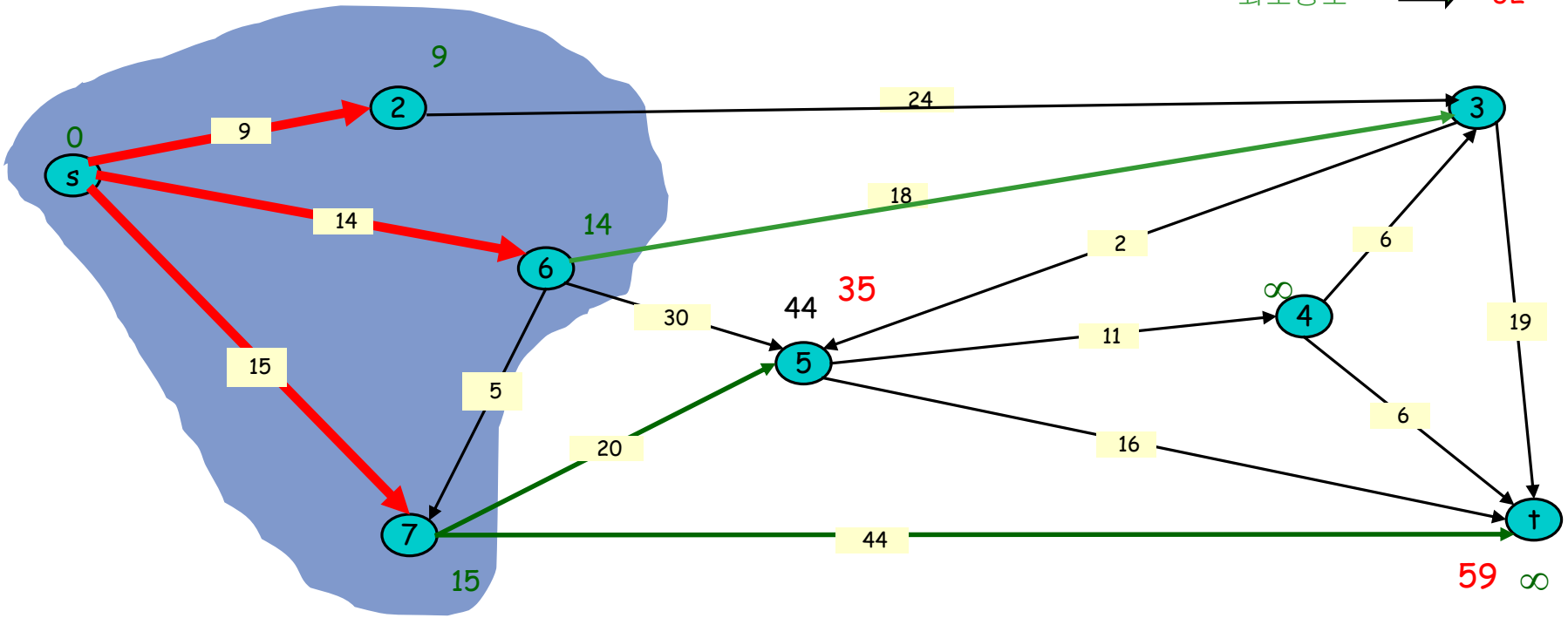


Dijkstra's Shortest Path Algorithm

S	2	3	4	5	6	7	t
0	9	32	∞	35	14	15	59

S = { s, 2, 6, 7 }
PQ = { 3, 4, 5, t }

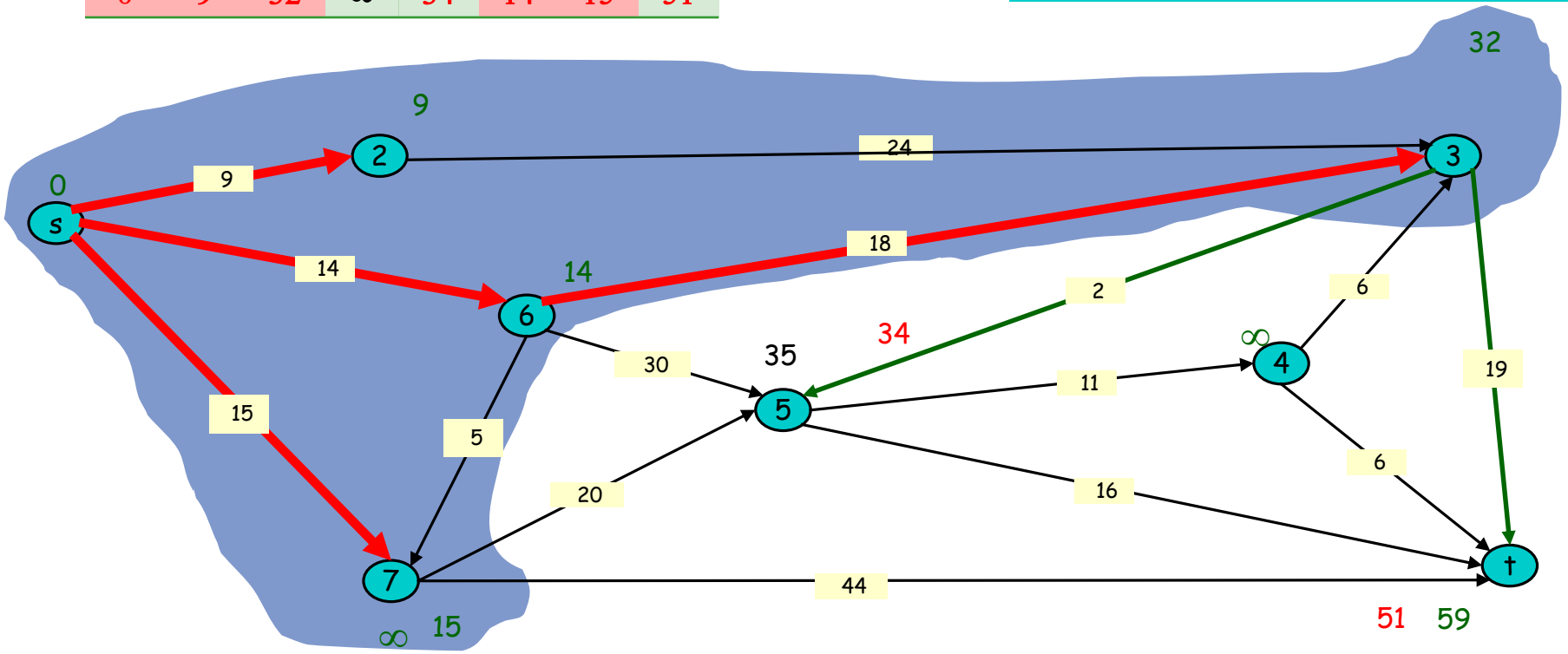
최소경로 ➡ 32



Dijkstra's Shortest Path Algorithm

S	2	3	4	5	6	7	t
0	9	32	∞	34	14	15	51

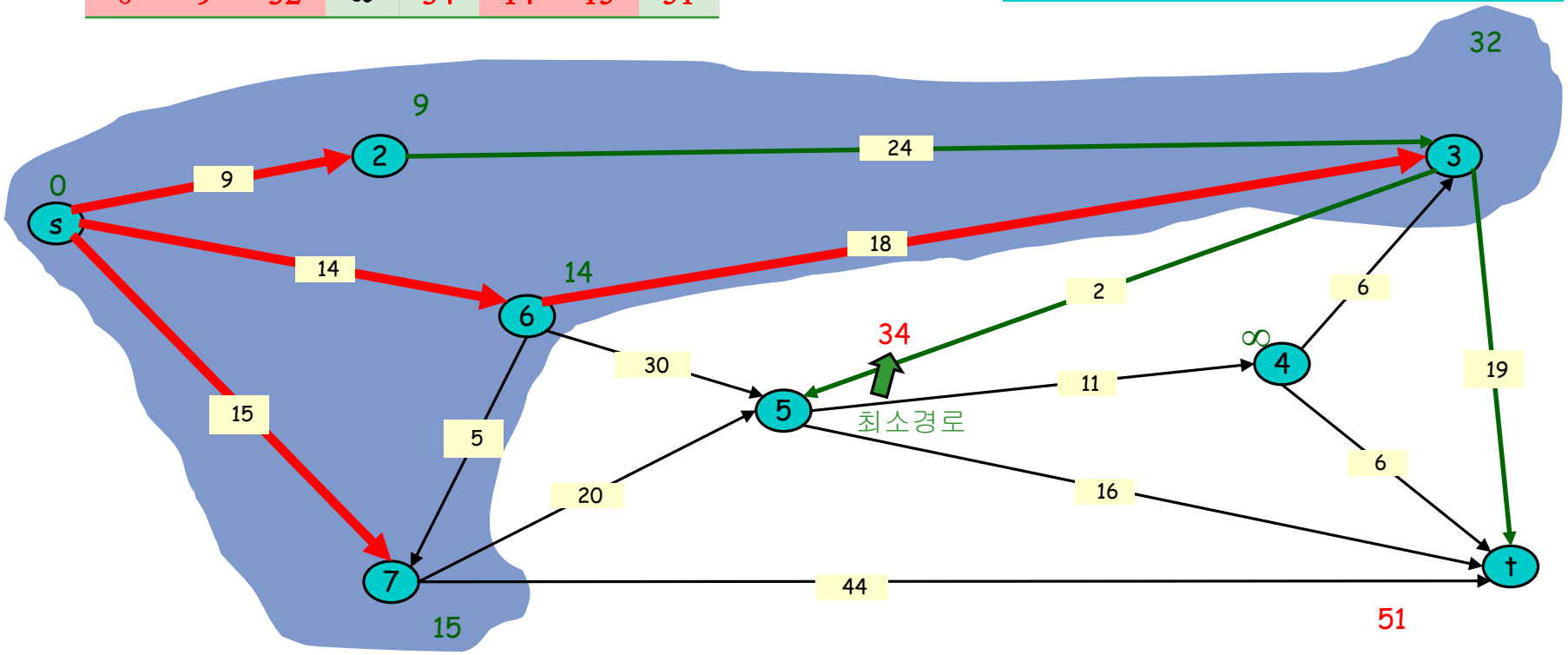
S = { s, 2, 3, 6, 7 }
PQ = { 4, 5, t }



Dijkstra's Shortest Path Algorithm

S	2	3	4	5	6	7	t
0	9	32	∞	34	14	15	51

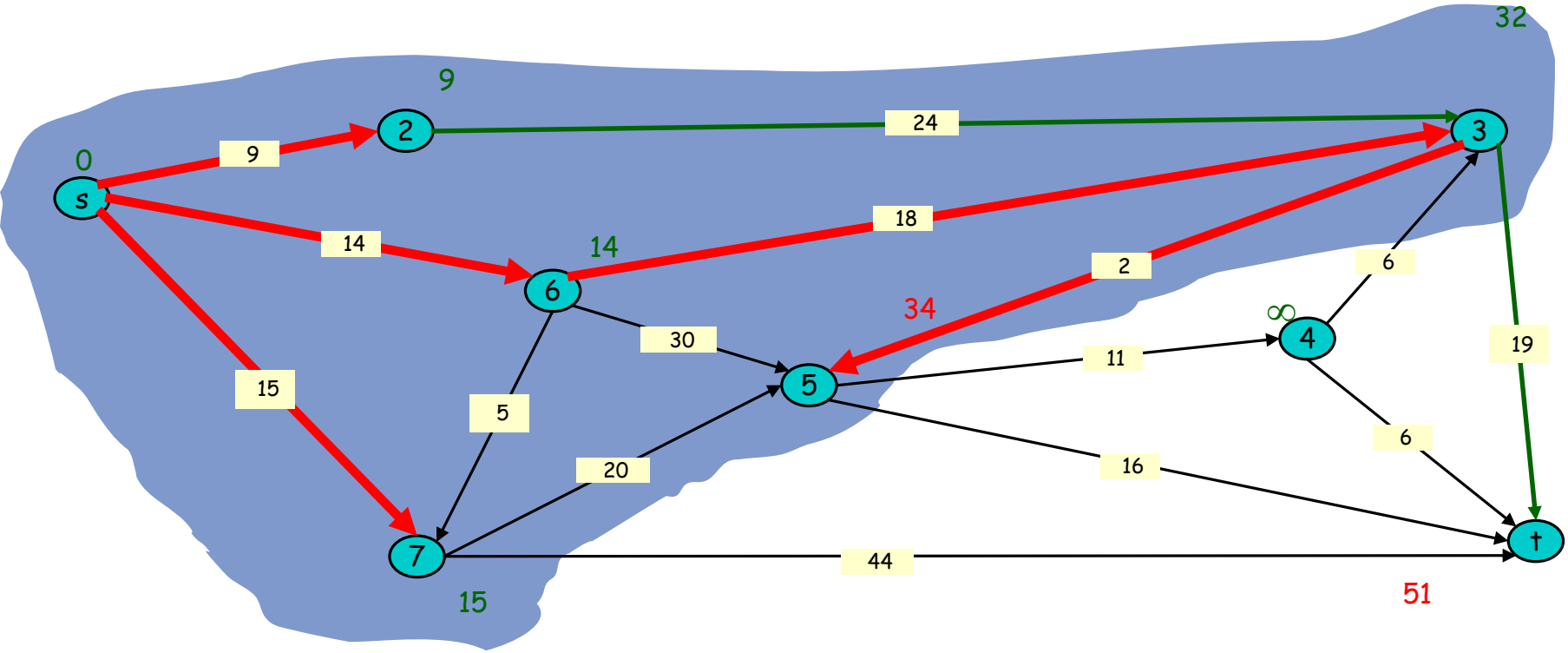
S = { s, 2, 3, 6, 7 }
PQ = { 4, 5, t }



Dijkstra's Shortest Path Algorithm

S	2	3	4	5	6	7	t
0	9	32	∞	34	14	15	51

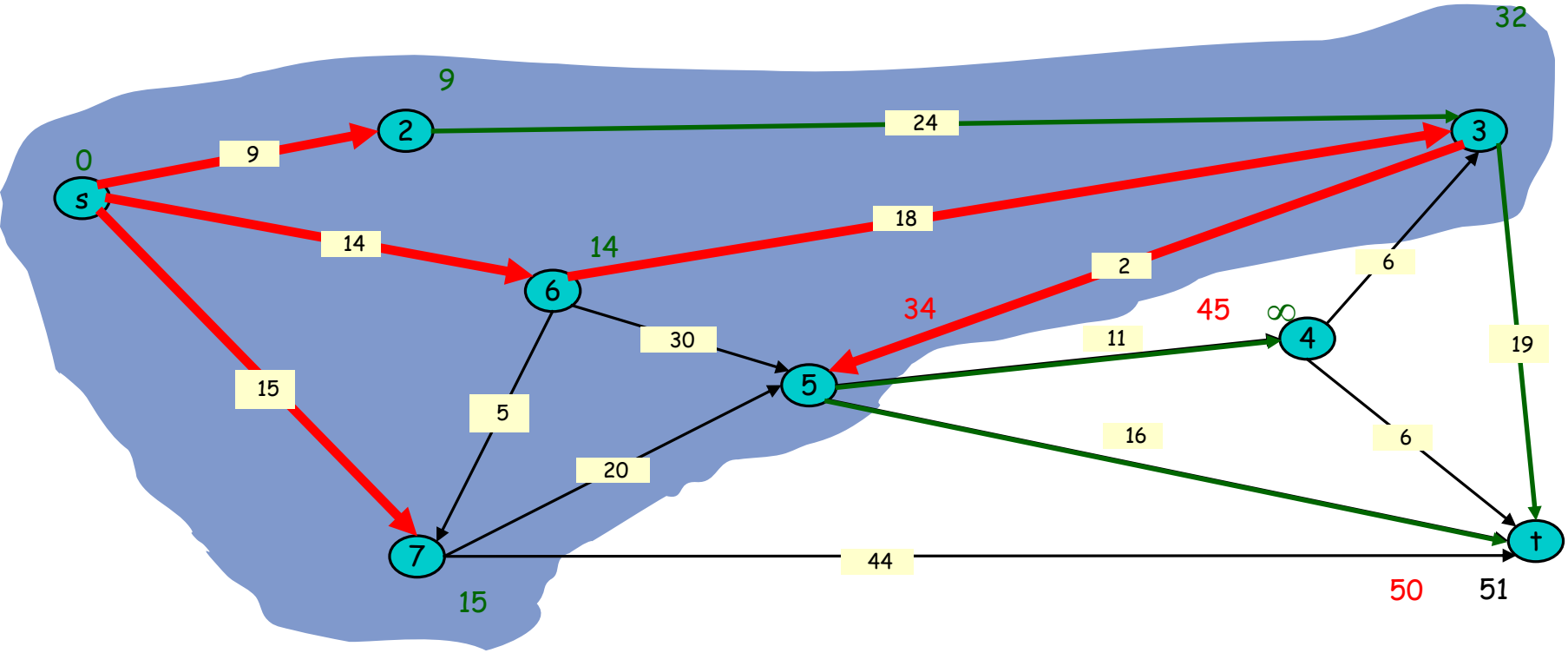
S = { s, 2, 3, 5, 6, 7 }
PQ = { 4, t }



Dijkstra's Shortest Path Algorithm

S	2	3	4	5	6	7	t
0	9	32	∞	34	14	15	51

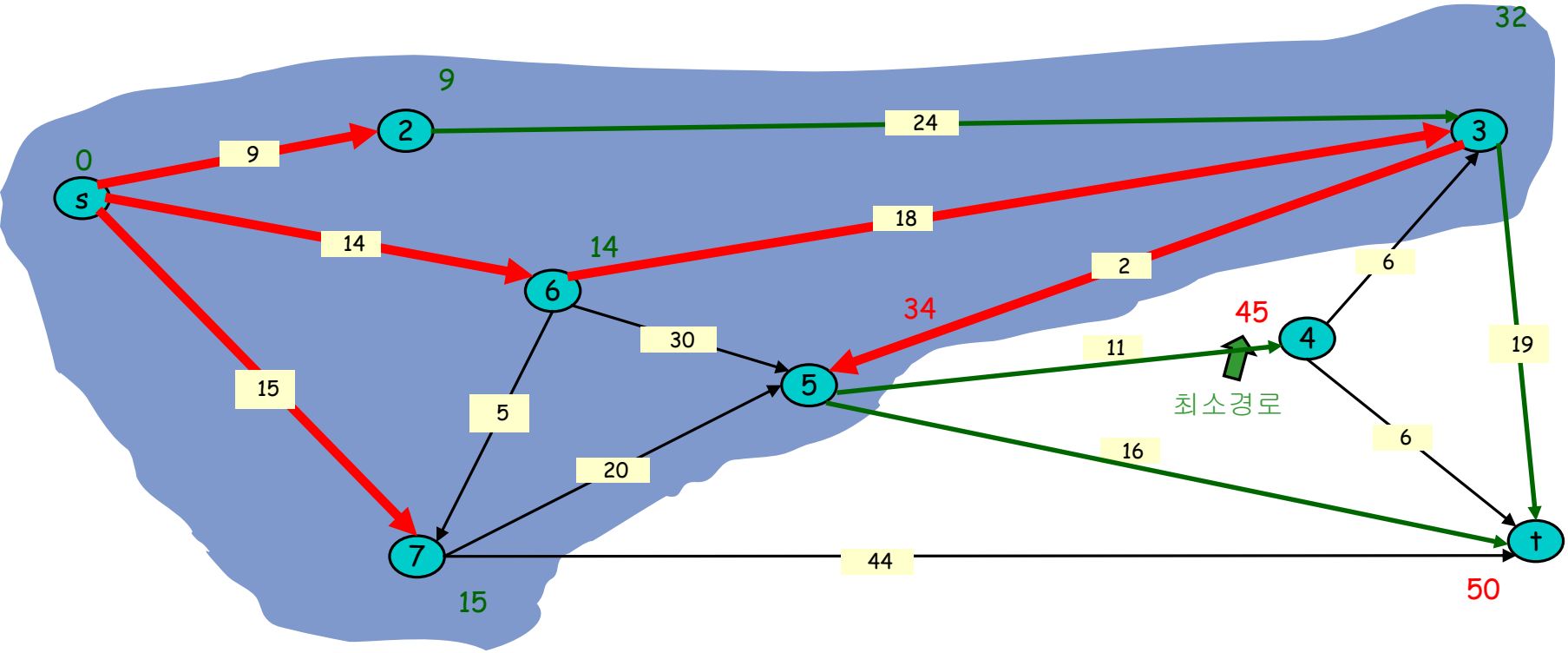
S = { s, 2, 3, 5, 6, 7 }
PQ = { 4, t }



Dijkstra's Shortest Path Algorithm

S	2	3	4	5	6	7	t
0	9	32	∞	34	14	15	51

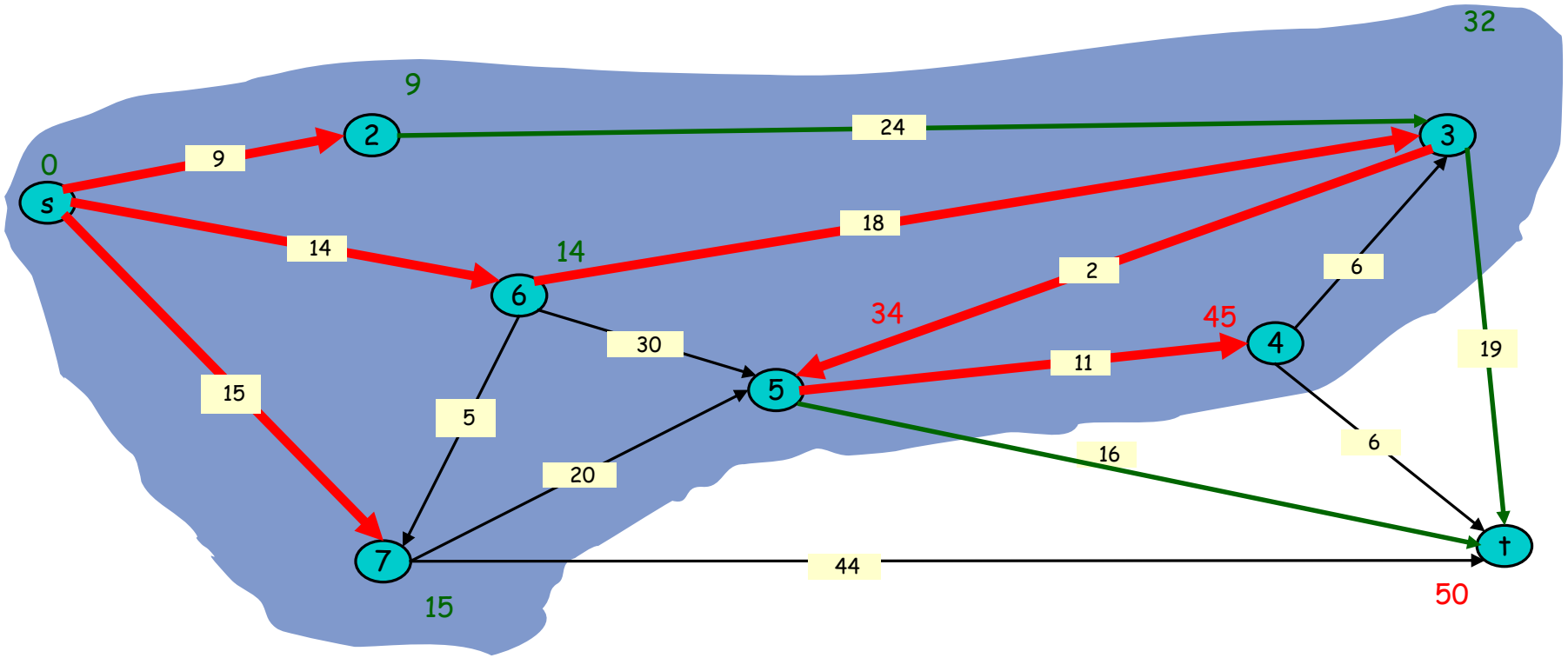
S = { s, 2, 3, 5, 6, 7 }
PQ = { 4, t }



Dijkstra's Shortest Path Algorithm

S	2	3	4	5	6	7	t
0	9	32	45	34	14	15	51

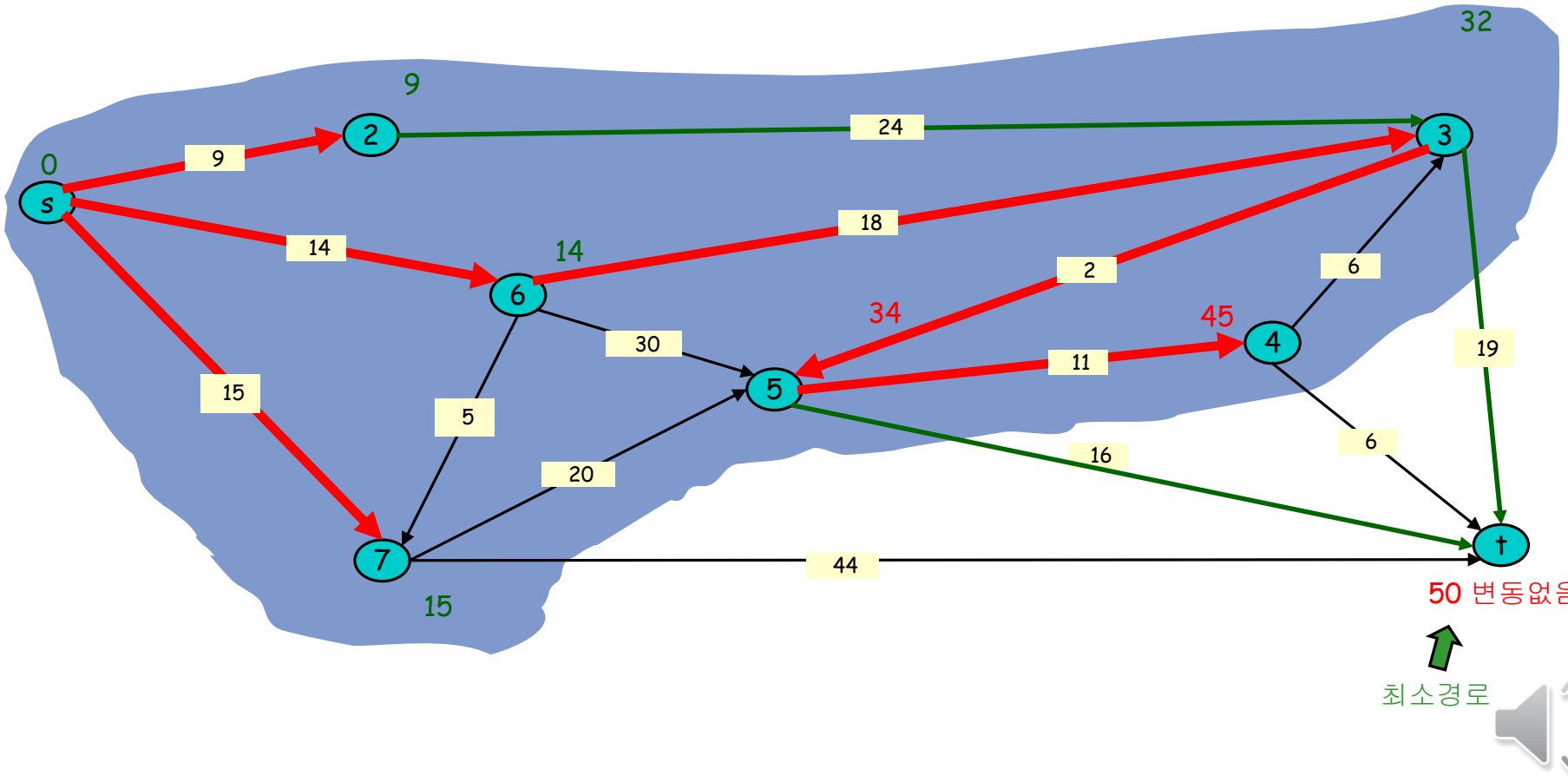
S = { s, 2, 3, 4, 5, 6, 7 }
PQ = {t }



Dijkstra's Shortest Path Algorithm

S	2	3	4	5	6	7	t
0	9	32	45	34	14	15	51

S = { s, 2, 3, 4, 5, 6, 7 }
PQ = {t }

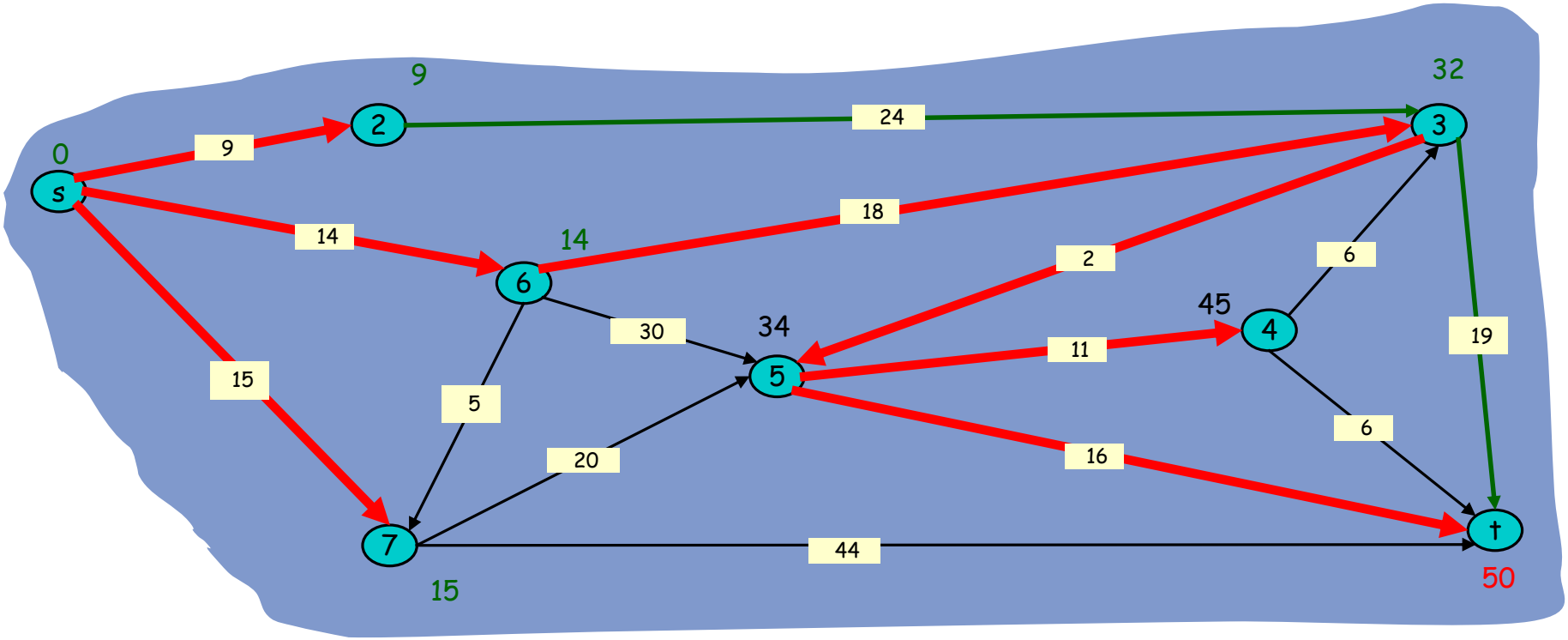


Dijkstra's Shortest Path Algorithm

S	2	3	4	5	6	7	t
0	9	32	45	34	14	15	51

S = { s, 2, 3, 4, 5, 6, 7, t }

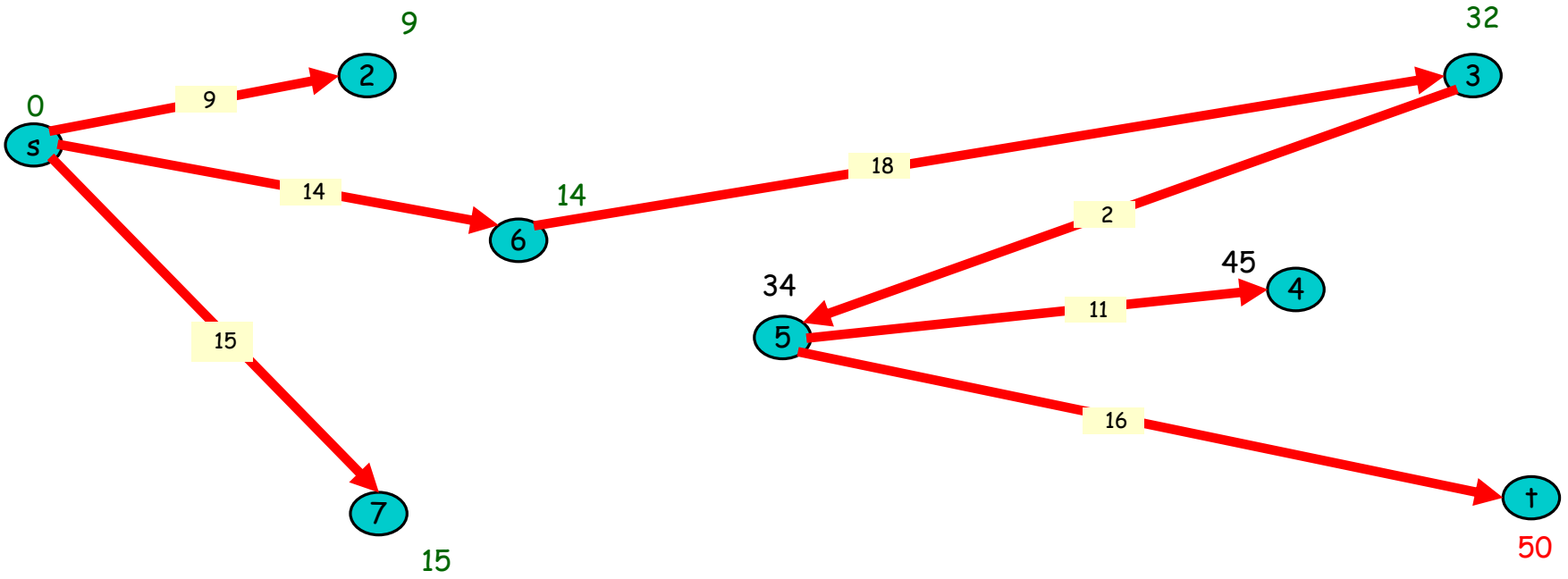
PQ = {}



Dijkstra's Shortest Path Algorithm

S	2	3	4	5	6	7	t
0	9	32	45	34	14	15	51

S = { s, 2, 3, 4, 5, 6, 7, t }
PQ = {}



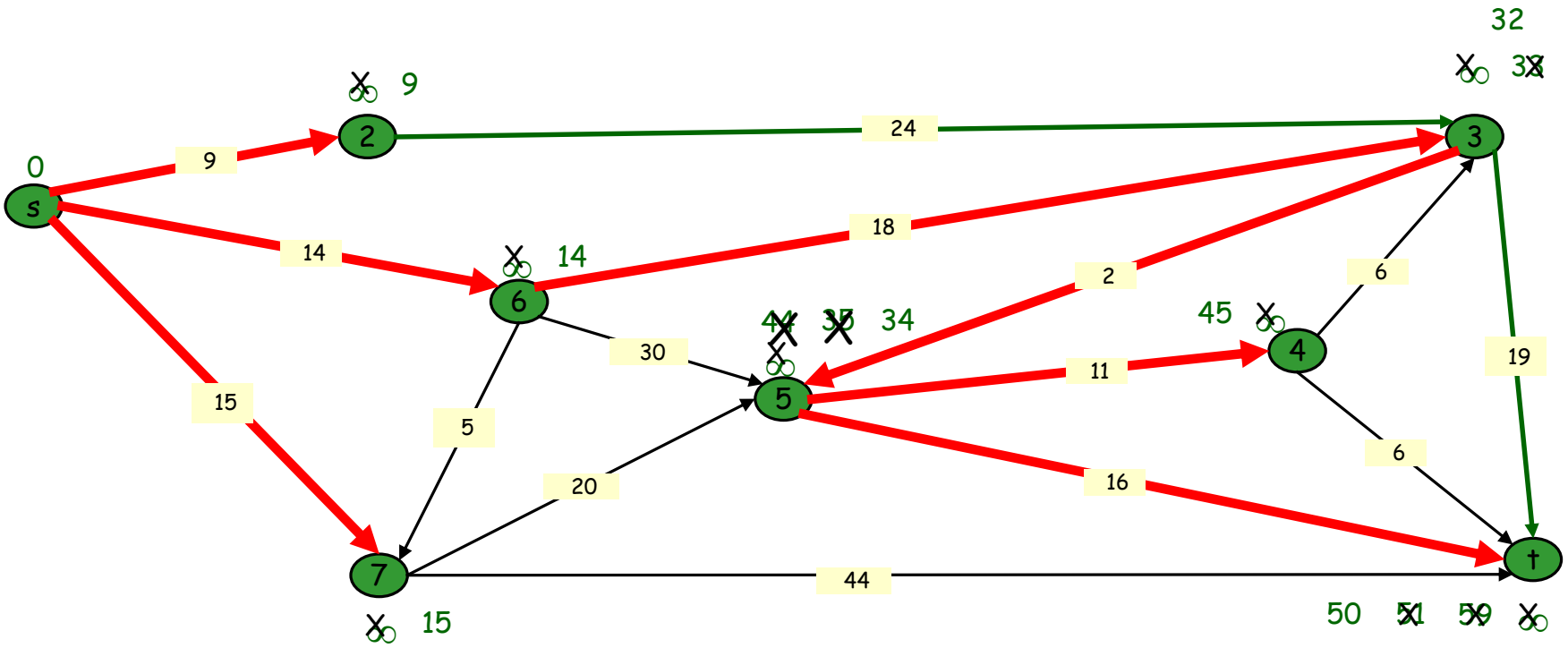
Spanning Tree (간선트리)

Dijkstra's Shortest Path Algorithm

S	2	3	4	5	6	7	t
0	9	32	45	34	14	15	50

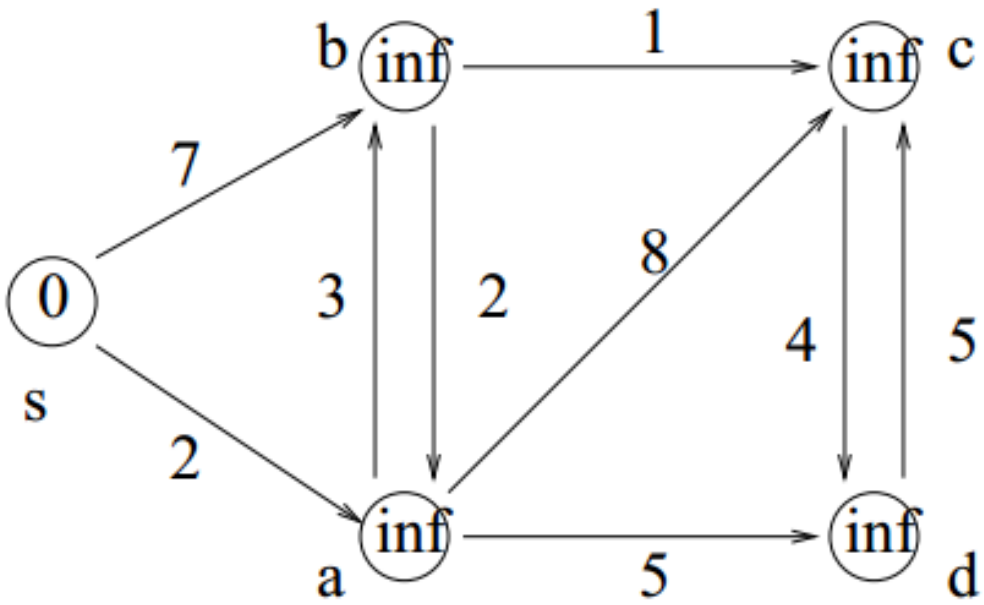
S = { s, 2, 3, 4, 5, 6, 7, t }

PQ = { }

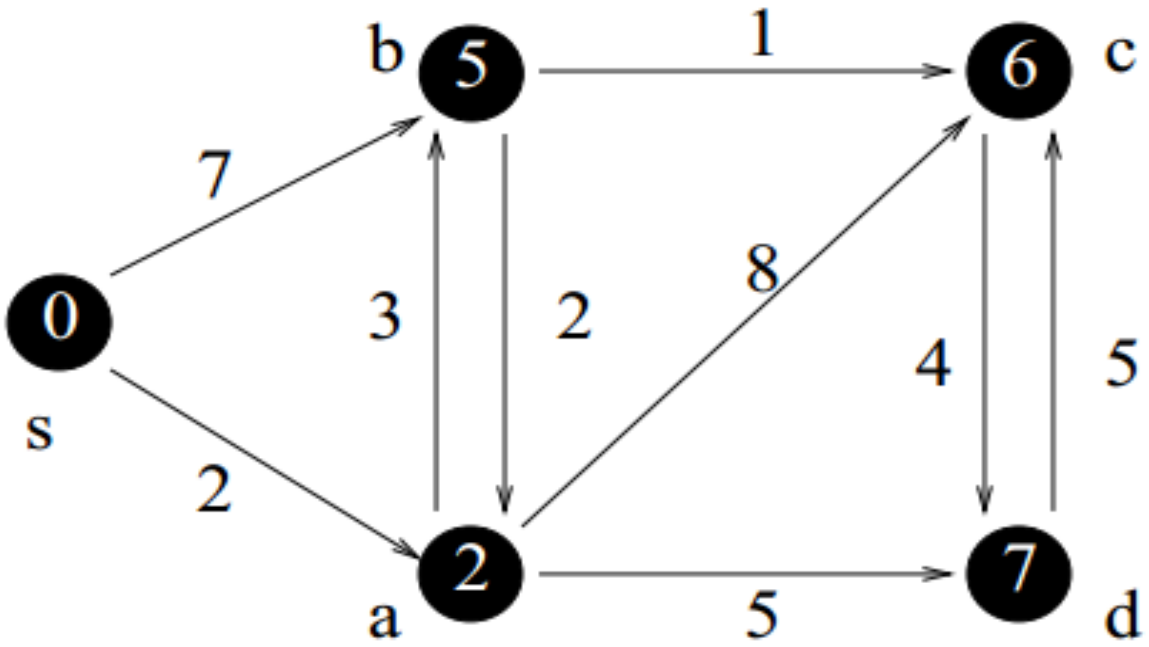


EXERCISE 1

Find the shortest path from 0 to every vertices on this graph



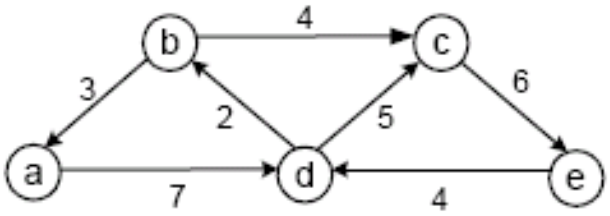
Solution 1



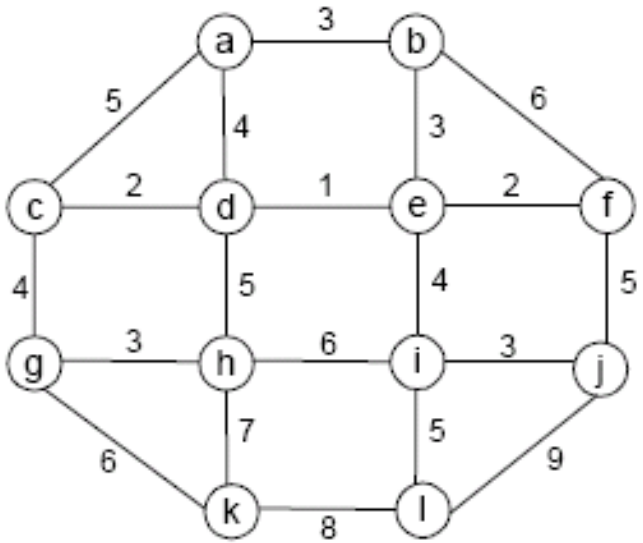
v	a	b	c	d
	2	5	6	7

2. Solve the following instances of the single-source shortest-paths problem with vertex a as the source:

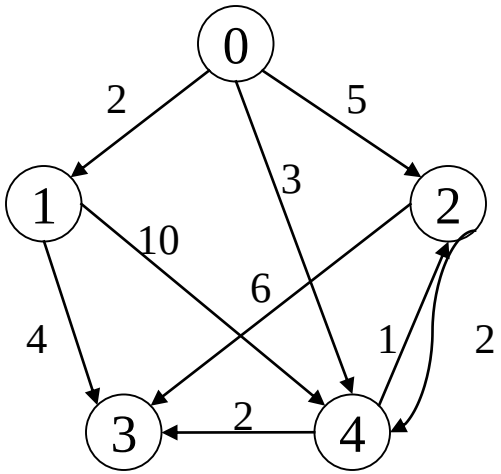
a.



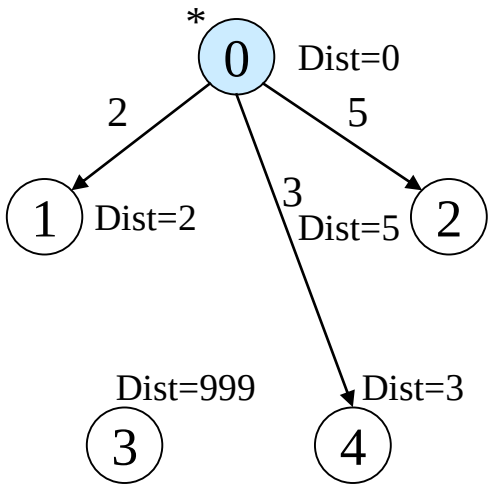
b.



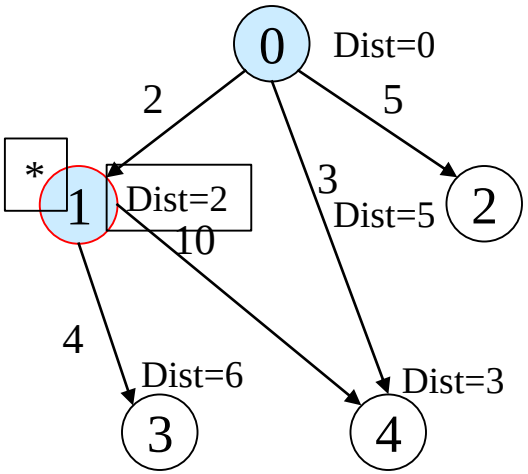
하나의 정점에서 다른 모든 정점까지의 최단경로(3)



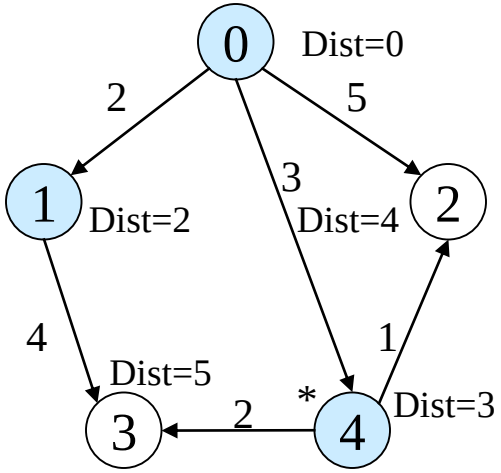
G = (V, E)



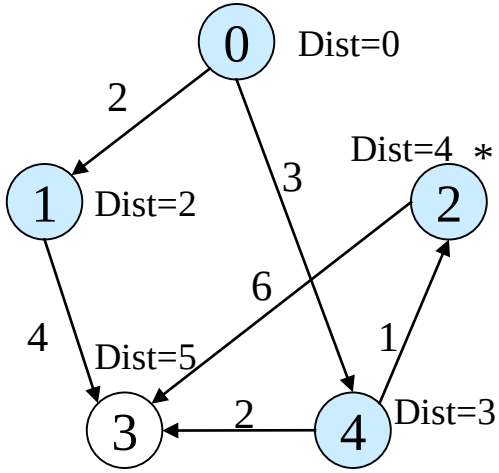
(a) s={0}, 정점 1 선정



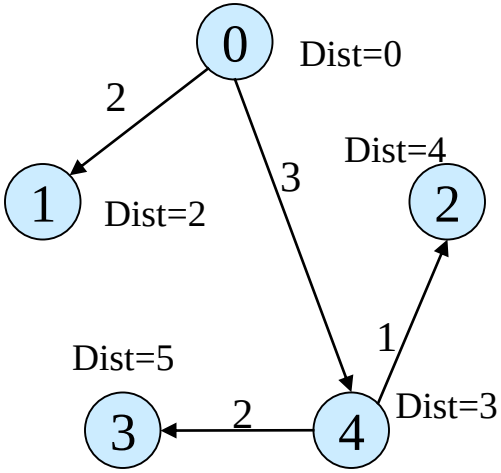
(b) s={0,1}, 정점 4 선정



(c) s={0,1,4}, 정점 2 선정



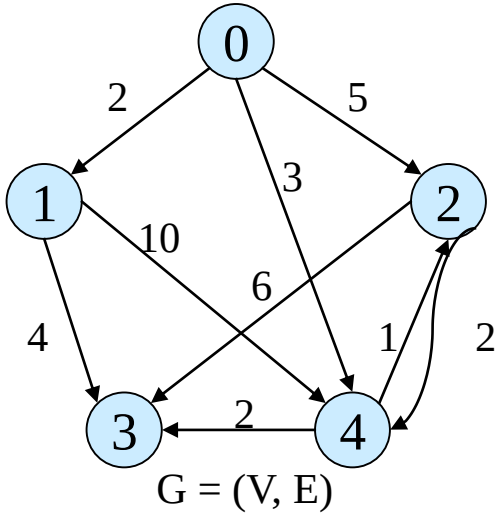
(d) s={0,1,4,2}, 정점 3 선정



(e) s={0,1,4,2,3}

최단 경로 계산의 예

하나의 정점에서 다른 모든 정점까지의 최단경로(6)



G = (V, E)

for 루프	선택된 정점	S=true인 정점	Dist[i]				
			[0]	[1]	[2]	[3]	[4]
초기화		[0]	0	2	5	999	3
1	[1]	[0],[1]	0	2	5	6	3
2	[4]	[0],[1],[4]	0	2	4	5	3
3	[2]	[0],[1],[4],[2]	0	2	4	5	3

그래프 G에 대한 shortestPath 수행 내용

감사합니다.