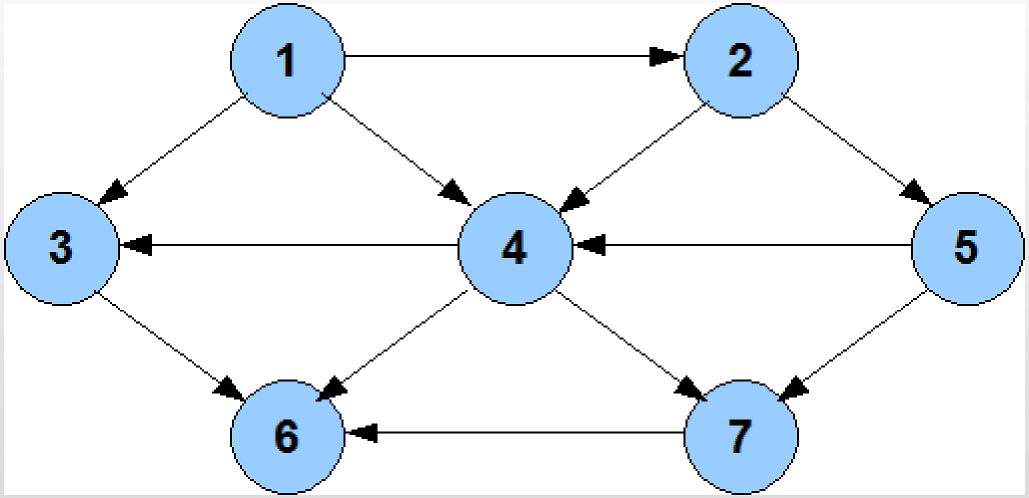
7. Graphs

7.1 Terminology and Representations

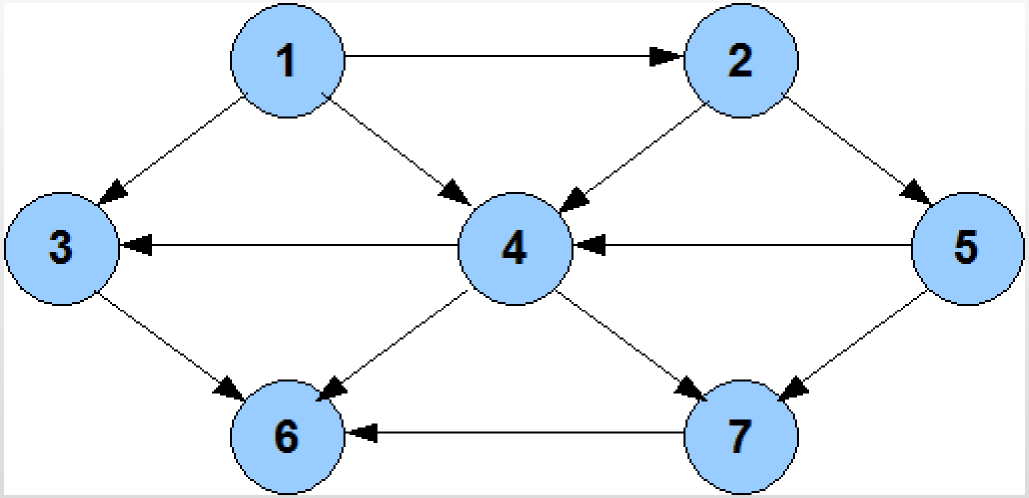
Definitions

- A graph **G = (V, E) consists of a set of *vertices V*, and a set of *edges E*.**



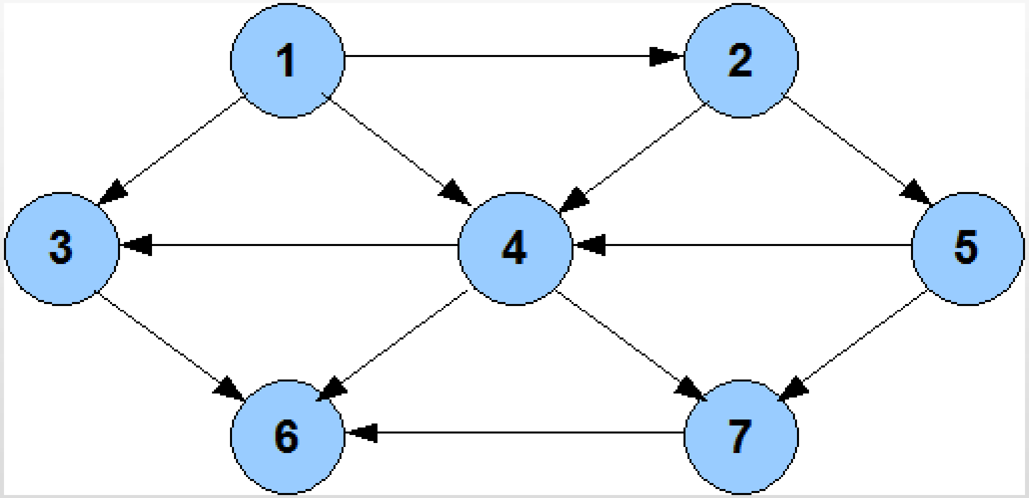
Definitions

- Each edge is a pair (u, v), where u, v are members of V.



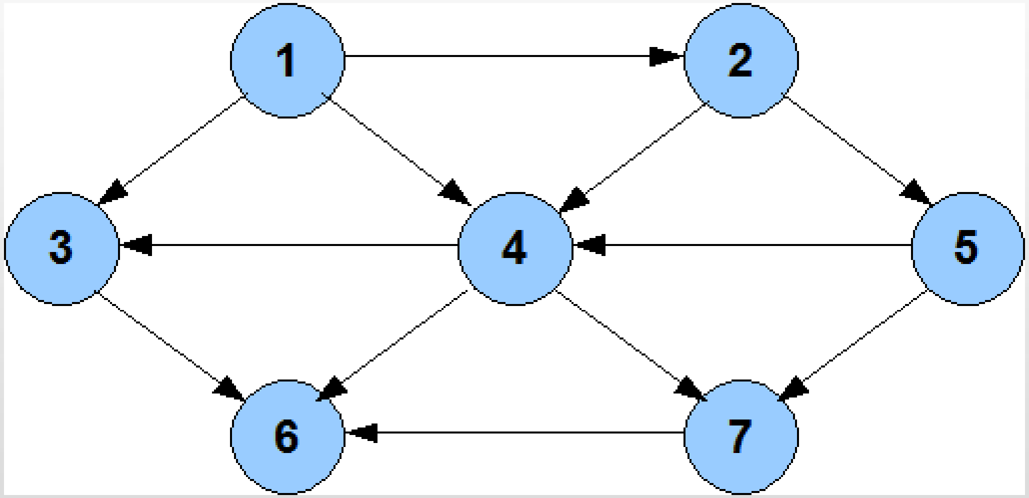
Definitions

- If the pair is ordered(i.e. head/tail), then the graph is *directed(i.e. digraph); undirected, otherwise.*



Definitions

- Vertex v is *adjacent to u if and only if (u, v) is a member of E.*



Definitions

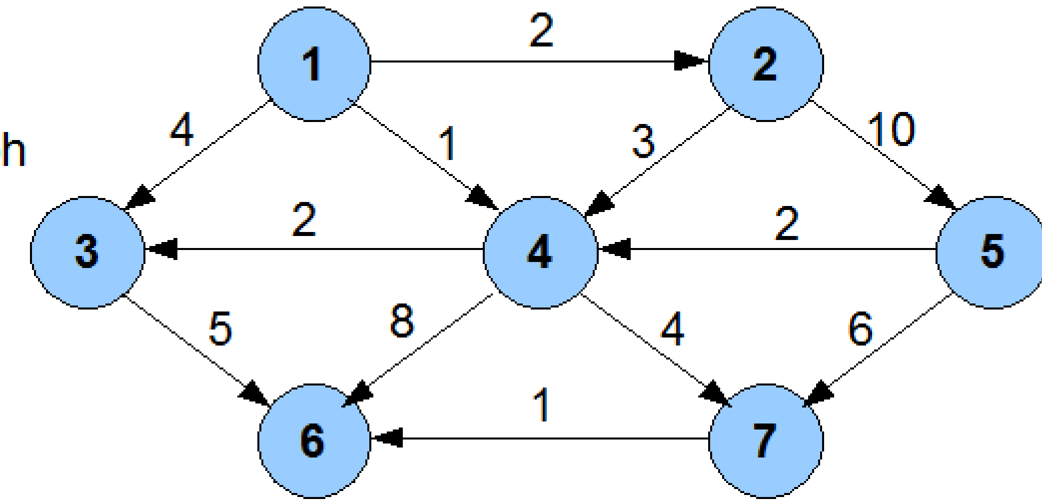
- An edge may have a third component: *weight/cost/label*
- A *path* in a graph is a sequence of *vertices v1*, v2,…, *vn* such that *(vi, vi+1)* is a valid edge, for *1* <= *i* < *n*.
- The *length of the path* is the number of edges on the path.

Definitions

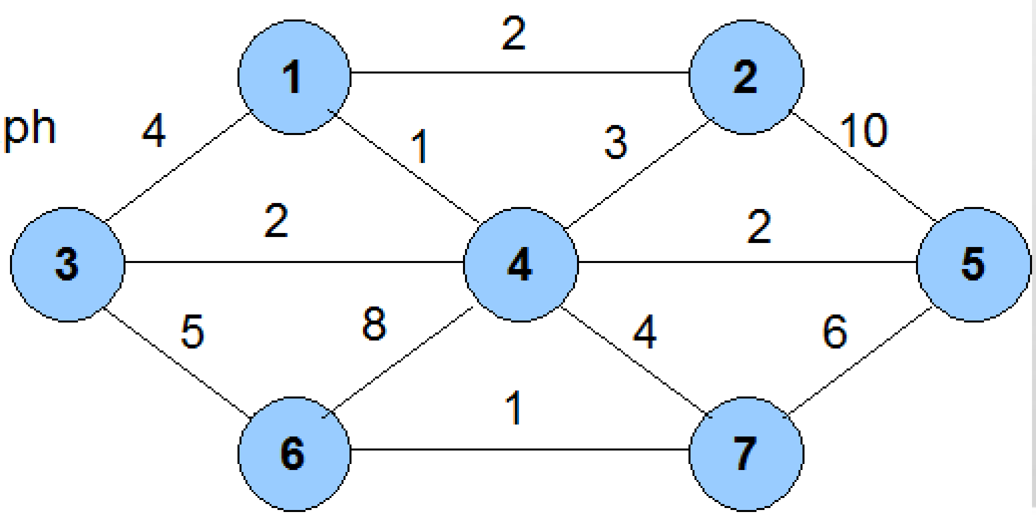
- *Simple path* is a path such that all vertices are distinct, except the first and the last could be the same.
- An undirected graph is *connected if there is a path from every vertex to every other vertex*
- *A complete graph is a graph in which there is an edge between every pair of vertices.*

Examples

Weighted directed graph

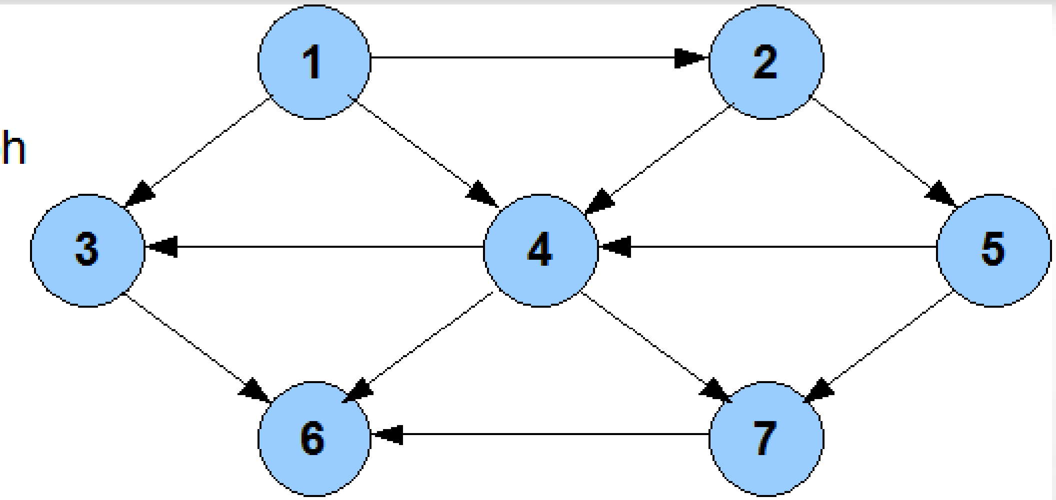


Weighted undirected graph

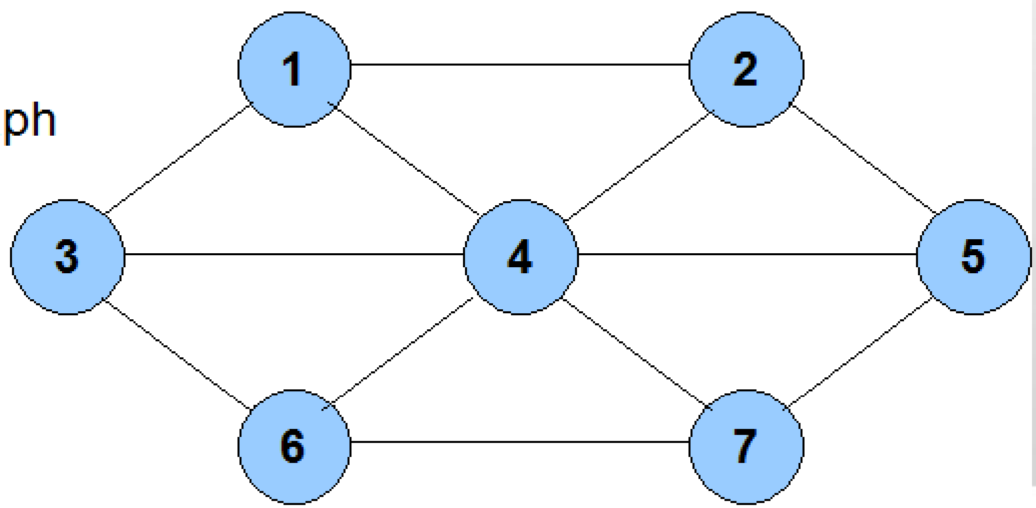


Examples

unweighted
directed graph



unweighted
undirected graph

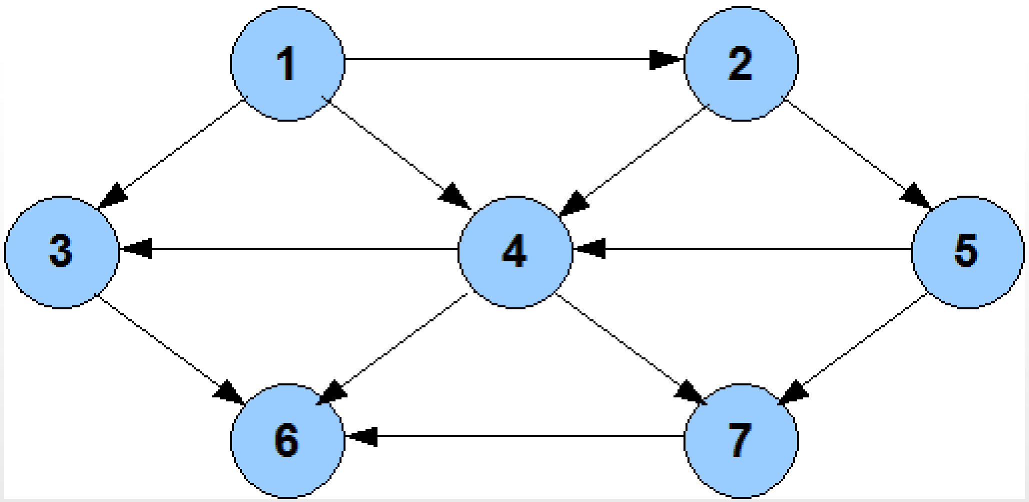


Representation

- **Fundamental operation**
  - **if a vertex is *adjacent to another;***

- Adjacency **matrix**
- Adjacency **list**

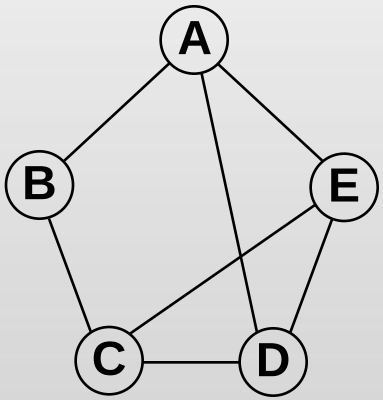
Adjacency Matrix



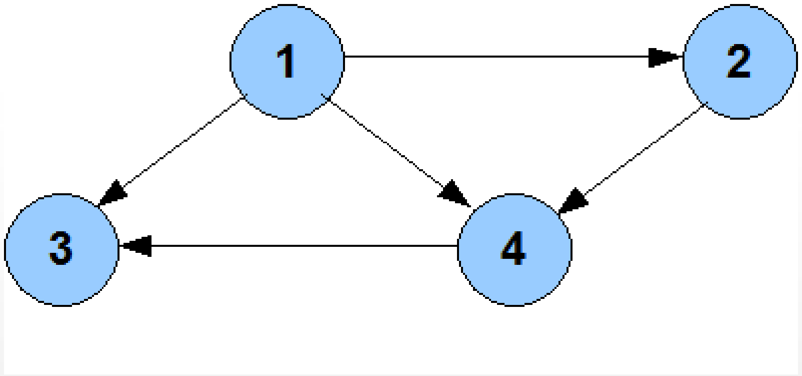
	1	2	3	4	5	6	7
1	0	1	1	1	0	0	0
2	0	0	0	1	1	0	0
3	0	0	0	0	0	1	0
4	0	0	1	0	0	1	1
5	0	0	0	1	0	0	1
6	0	0	0	0	0	0	0
7	0	0	0	0	0	1	0

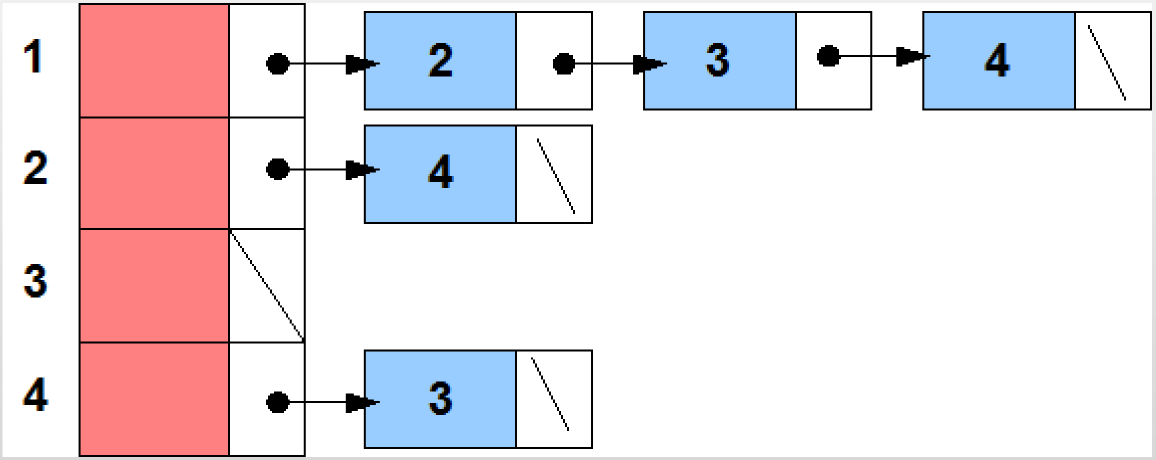
Adjacency Matrix

	A	B	C	D	E
A	0	1	0	1	1
B	1	0	1	0	0
C	0	1	0	1	1
D	1	0	1	0	1
E	1	0	1	1	0

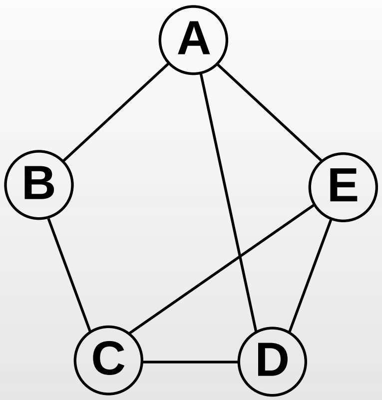


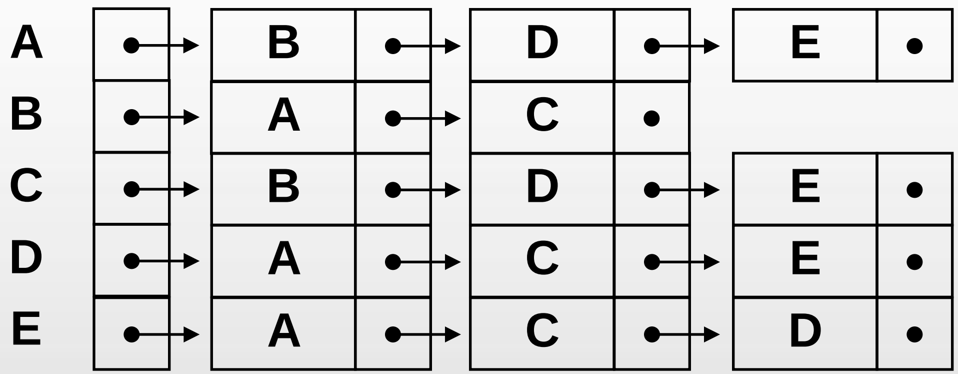
Adjacency List





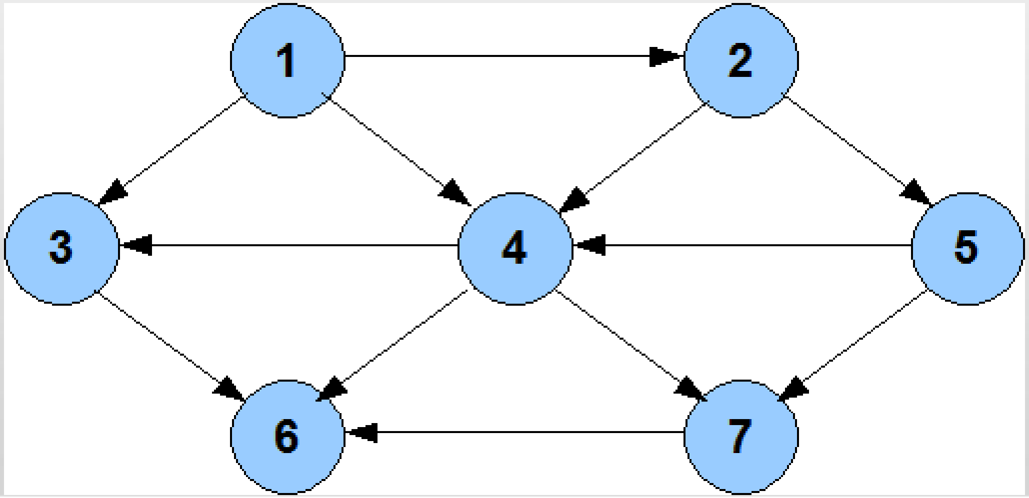
Adjacency List



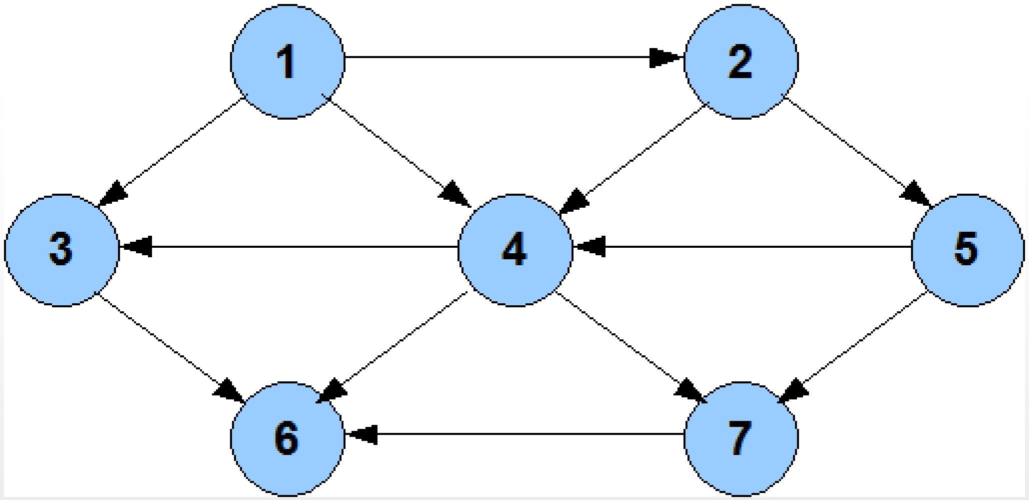


Topological Sort

- An ordering of vertices in a DAG, such that if there is a path from vi to vj, then vj appears after vi in the ordering
- Sample application: course prerequisite structure...



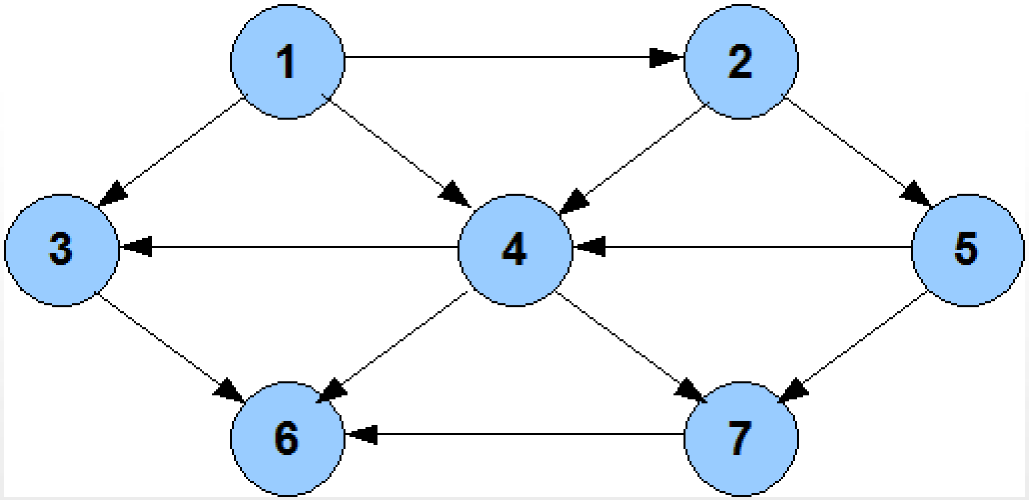
Topological Sort



Topological Sort (Algo)

- Find any vertex with no incoming edges(i.e. indegree of vertex = 0)
- NOTE: indegree of vertex v = # of edges (u, v)
- Print this vertex, and remove it, along with its edges, from the graph
- Repeat steps above

Topological Sort



- Solution A: 1, 2, 5, 4, 3, 7, 6
- Solution B: 1, 2, 5, 4, 7, 3, 6

Graph Representations

Properties/Routines	Adjacency Matrix	Adjacency List
Check if vertex x is adjacent to vertex y	O(1)	O(|V|)
List all adjacent vertices to vertex x	O(|V|)	O(|V|)
List all edges	O(|V|^2)	O(|V|+|E|)

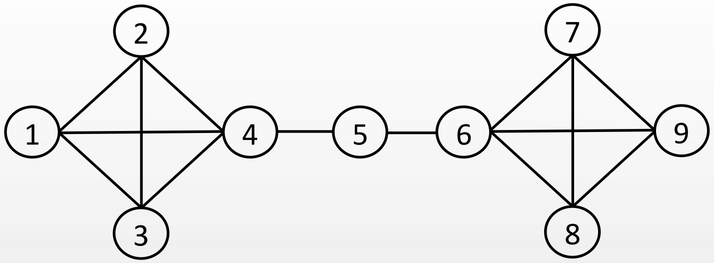
7. Graphs

7.2 Graph Traversals

Graph Searching

- Depth-first Search
  - Pre-order traversal of tree
  - Uses STACK
- Breadth-first Search
  - Level-order traversal of tree
  - Uses QUEUE

Depth-first Search

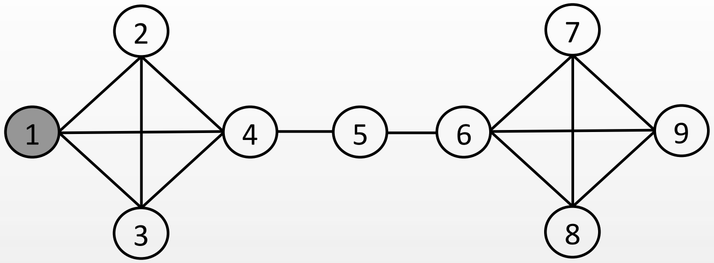


TOS

DFS Stack: 1

Output:

Depth-first Search

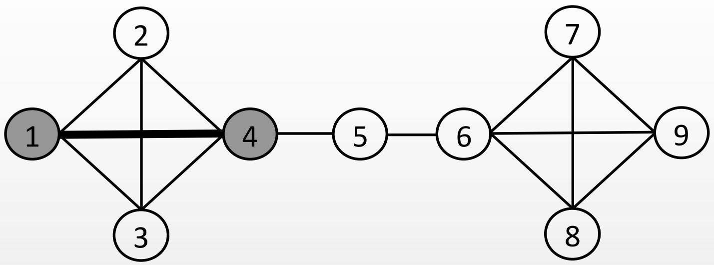


TOS

DFS Stack: 2 3 4

Output: 1

Depth-first Search

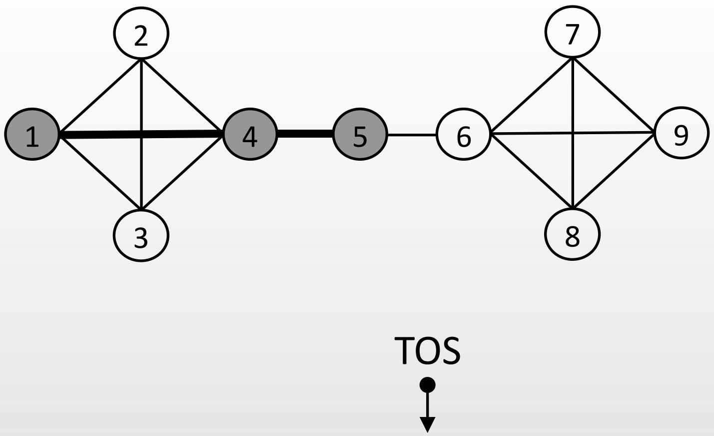


TOS

DFS Stack: 2 3 2 3 5

Output: 1 4

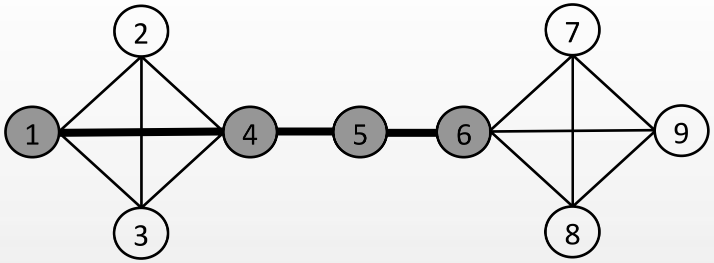
Depth-first Search



DFS Stack: 2 3 2 3 6

Output: 1 4 5

Depth-first Search

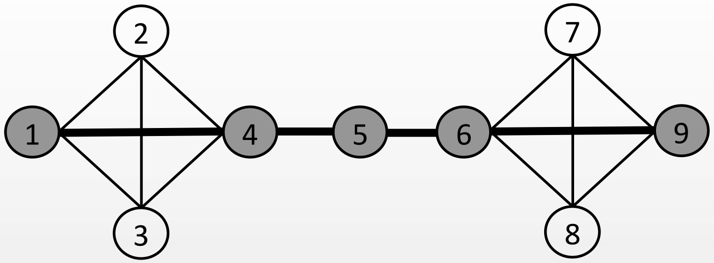


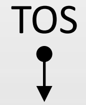
TOS

DFS Stack: 2 3 2 3 7 8 9

Output: 1 4 5 6

Depth-first Search

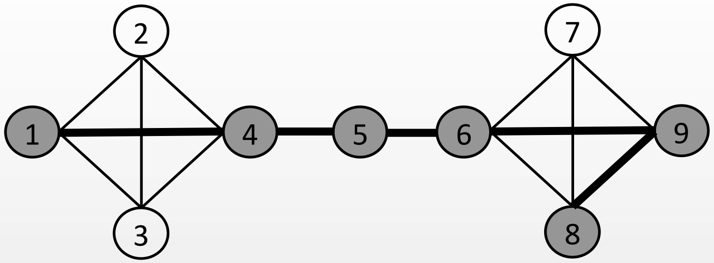




DFS Stack: 2 3 2 3 7 8 7 8

Output: 1 4 5 6 9

Depth-first Search

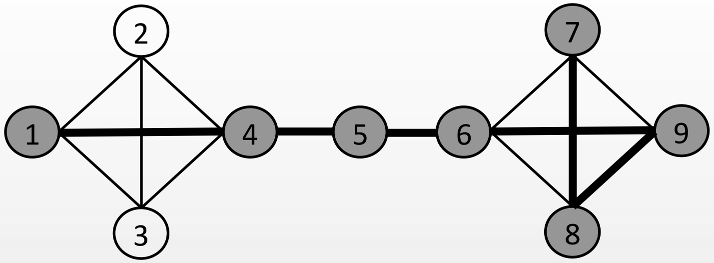


TOS

DFS Stack: 2 3 2 3 7 8 7 7

Output: 1 4 5 6 9 8

Depth-first Search

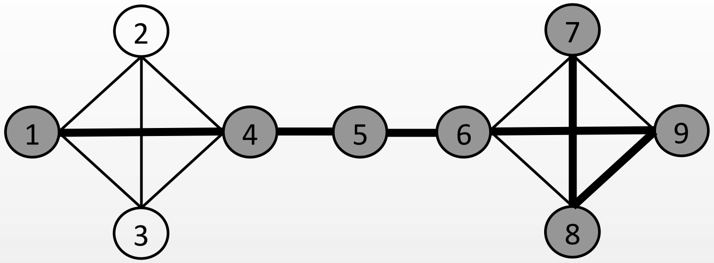


TOS

DFS Stack: 2 3 2 3 7 8 7

Output: 1 4 5 6 9 8 7

Depth-first Search

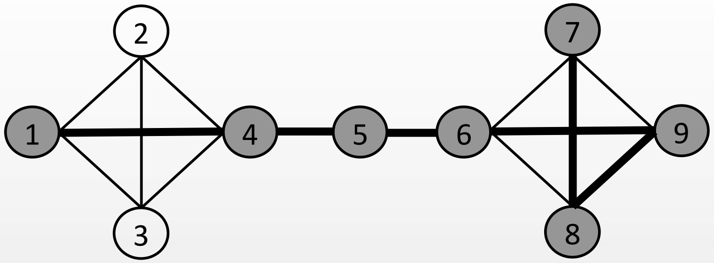


TOS

DFS Stack: 2 3 2 3 7 8

Output: 1 4 5 6 9 8 7

Depth-first Search

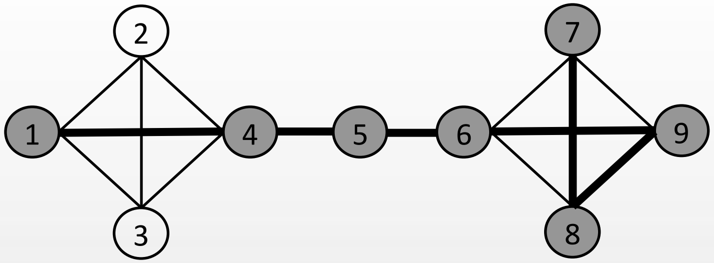


TOS

DFS Stack: 2 3 2 3 7

Output: 1 4 5 6 9 8 7

Depth-first Search

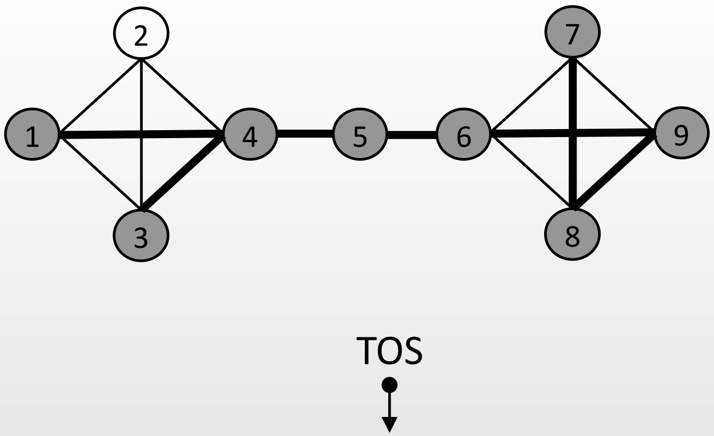


TOS

DFS Stack: 2 3 2 3

Output: 1 4 5 6 9 8 7

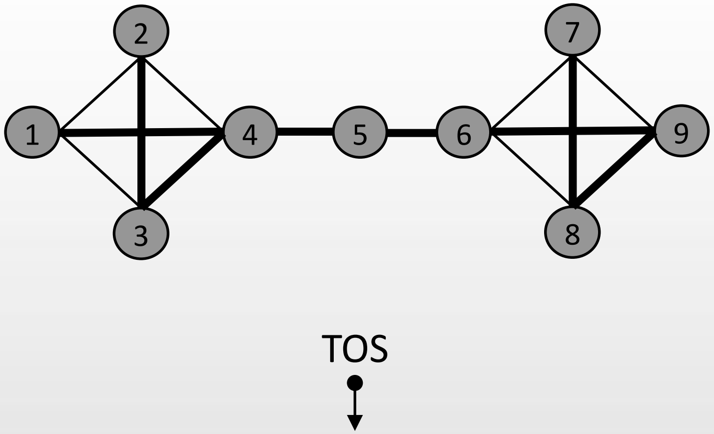
Depth-first Search



DFS Stack: 2 3 2 2

Output: 1 4 5 6 9 8 7 3

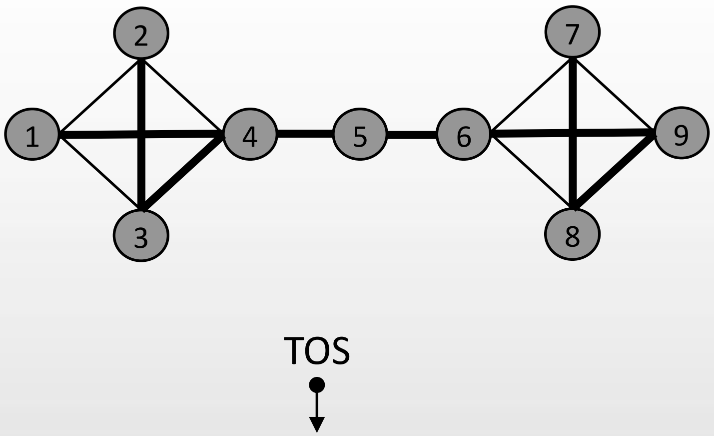
Depth-first Search



DFS Stack: 2 3 2

Output: 1 4 5 6 9 8 7 3 2

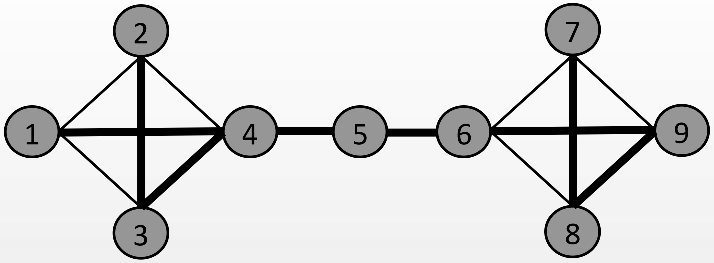
Depth-first Search



DFS Stack: 2 3

Output: 1 4 5 6 9 8 7 3 2

Depth-first Search

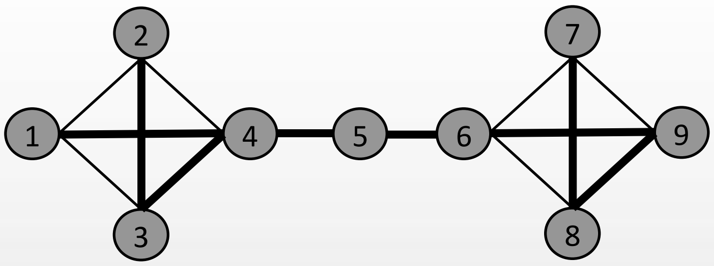


TOS

DFS Stack: 2

Output: 1 4 5 6 9 8 7 3 2

Depth-first Search



TOS

DFS Stack:

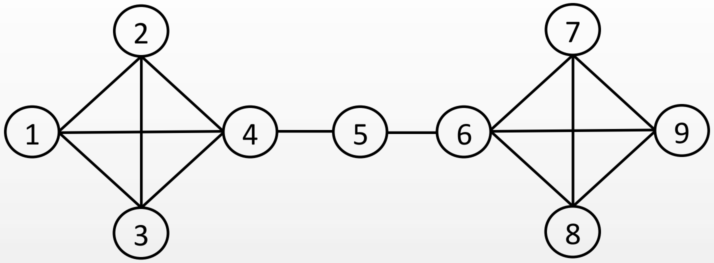
Output: 1 4 5 6 9 8 7 3 2

Depth-First Search (DFS)

```
dfs(int s, int graphsize) {
        int j,v, visited[MAXGRAPHSIZE]; stack dfs_stack;

        for (j=0; j<graphsize; j++) visited[j]=FALSE;
        push(s,dfs_stack);
        do  {
                v=pop(dfs_stack);
                if (!visited[v])  {
                   visited[v]=TRUE;
                   printf("%d\n",v);
                   for each vertex j adjacent to v
                      if (!visited[j]) push(j,dfs_stack);
                }
        } while(!is_empty_stack(dfs_stack)
}
```

Breadth-first Search

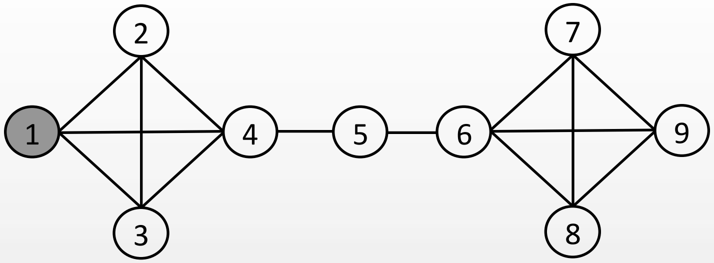


Front

BFS Queue: 1

Output:

Breadth-first Search

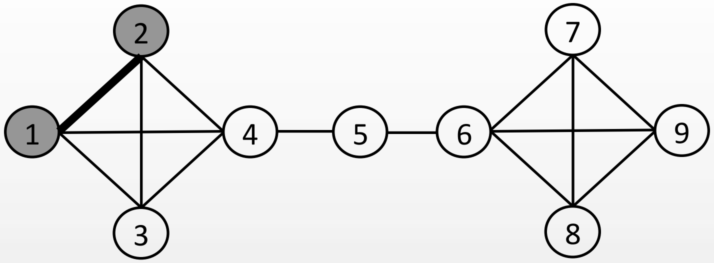


Front

BFS Queue: 2 3 4

Output: 1

Breadth-first Search

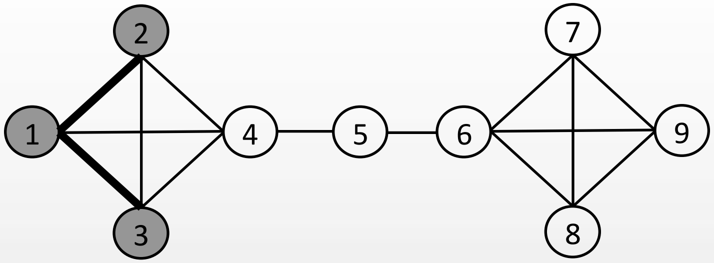


Front

BFS Queue: 3 4 3 4

Output: 1 2

Breadth-first Search

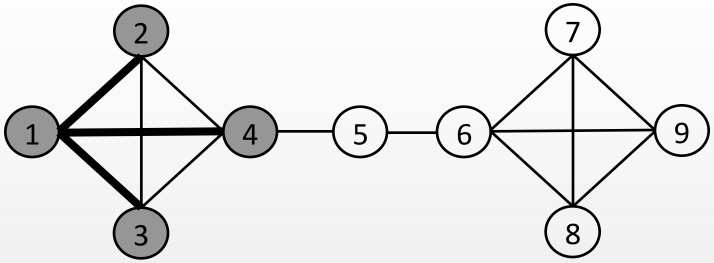


Front

BFS Queue: 4 3 4 4

Output: 1 2 3

Breadth-first Search

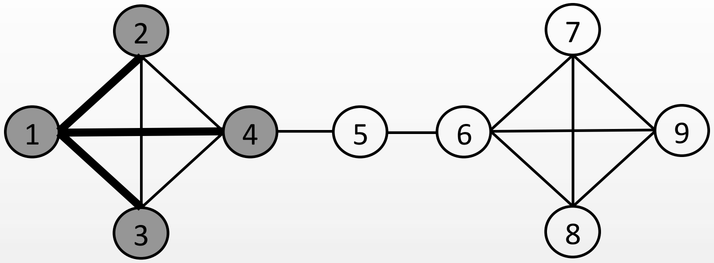


Front

BFS Queue: 3 4 4 5

Output: 1 2 3 4

Breadth-first Search

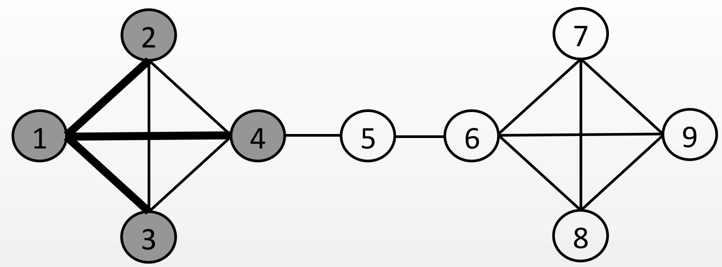


Front

BFS Queue: 4 4 5

Output: 1 2 3 4

Breadth-first Search

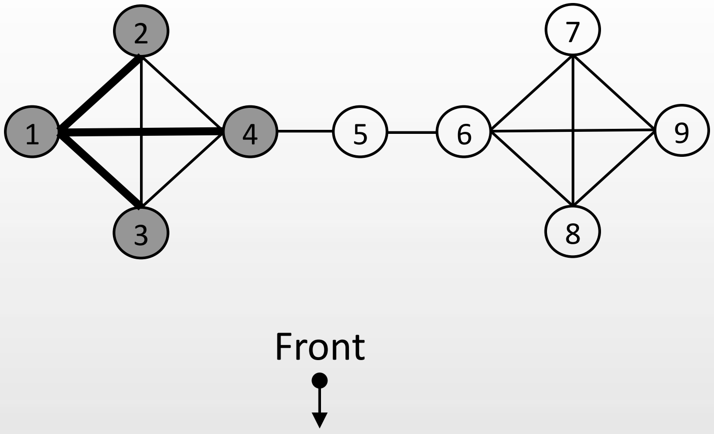


Front

BFS Queue: 4 5

Output: 1 2 3 4

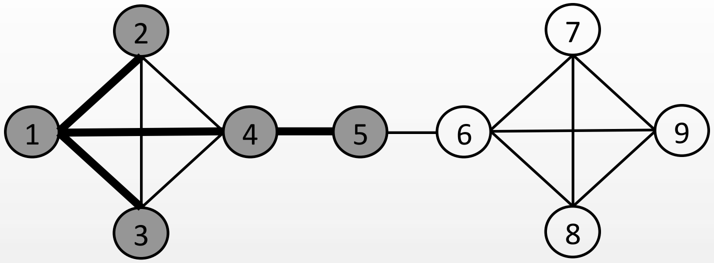
Breadth-first Search



BFS Queue: 5

Output: 1 2 3 4

Breadth-first Search

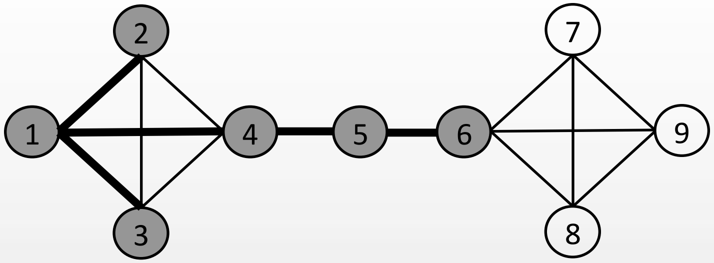


Front

BFS Queue: 6

Output: 1 2 3 4 5

Breadth-first Search

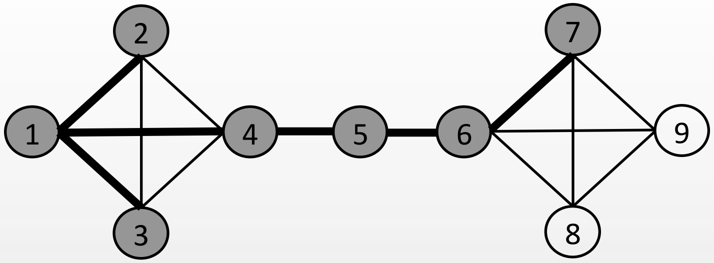


Front

BFS Queue: 7 8 9

Output: 1 2 3 4 5 6

Breadth-first Search

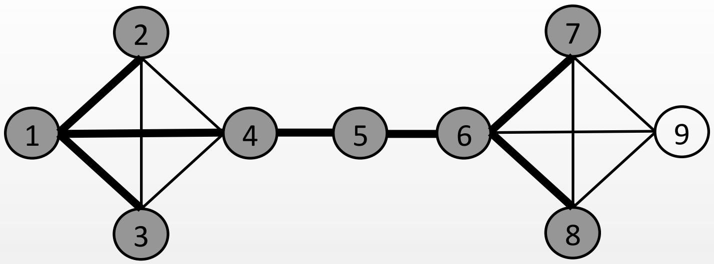


Front

BFS Queue: 8 9 8 9

Output: 1 2 3 4 5 6 7

Breadth-first Search

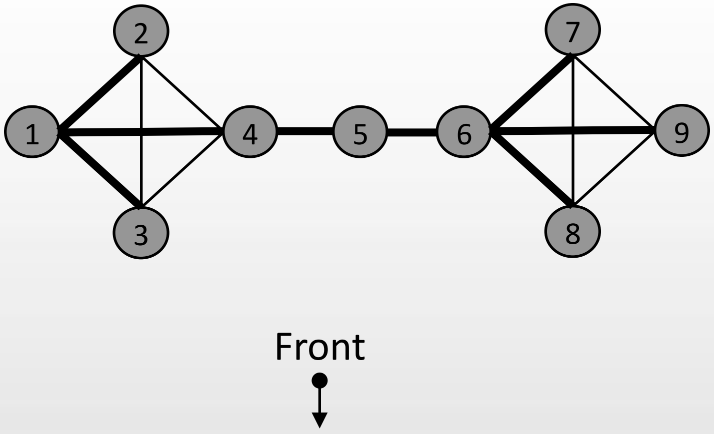


Front

BFS Queue: 9 8 9 9

Output: 1 2 3 4 5 6 7 8

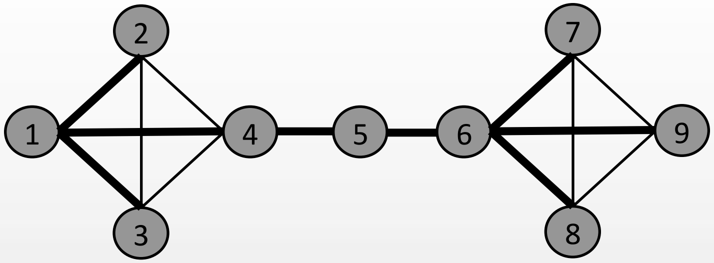
Breadth-first Search



BFS Queue: 8 9 9

Output: 1 2 3 4 5 6 7 8 9

Breadth-first Search

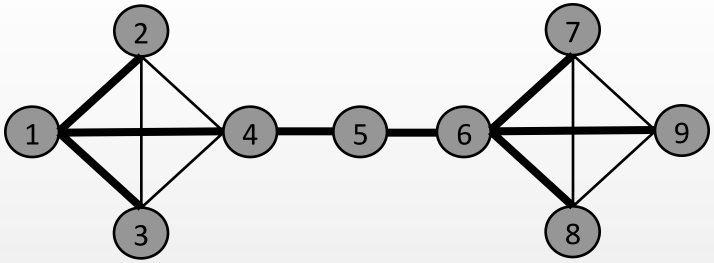


Front

BFS Queue: 9 9

Output: 1 2 3 4 5 6 7

Breadth-first Search

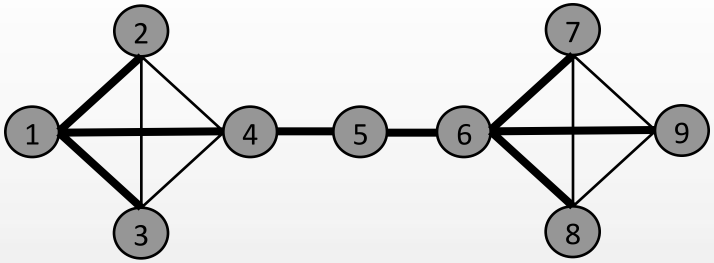


Front

BFS Queue: 9

Output: 1 2 3 4 5 6 7 8 9

Breadth-first Search



Front

BFS Queue:

Output: 1 2 3 4 5 6 7 8 9

Breadth-First Search (BFS)

```
bfs(int s, int graphsize) {
    int j,v, visited[MAXGRAPHSIZE]; queue bfs_queue;

    for (j=0; j<graphsize; j++) visited[j]=FALSE;
    enqueue(s,bfs_queue);
    do  {
        v=dequeue(bfs_queue);
        if (!visited[v])  {
          visited[v]=TRUE;
          printf("%d\n",v);
          for each vertex j adjacent to v
            if (!visited[j]) enqueue(j,bfs_queue);
        }
    } while(!is_empty_queue(bfs_queue)
}
```

7. Graphs

7.3 Shortest Path Problem

Single Source Shortest Path Problem

Given

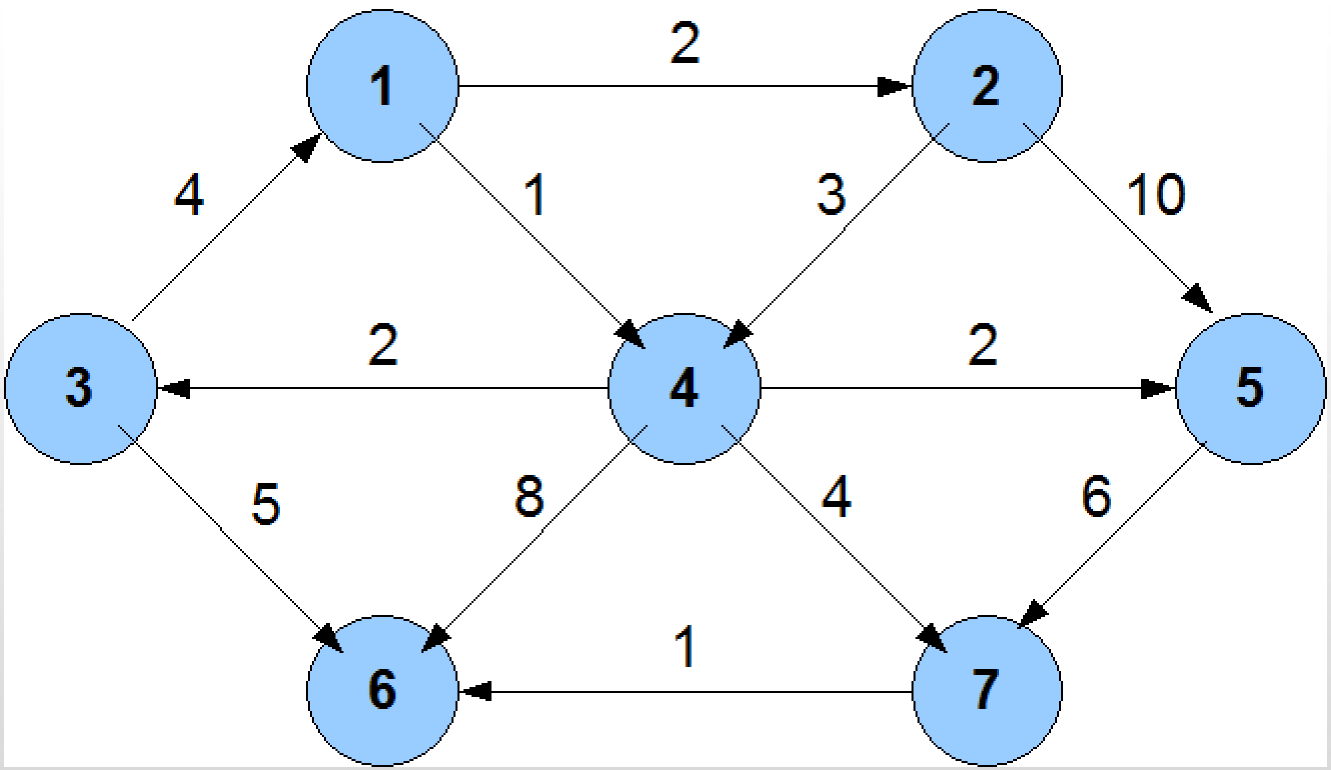
- A directed graph $G = (V, E)$ where
- edges or arcs are assigned *nonnegative* costs or weights
- one vertex *s* specified as the *source* vertex

Single Source Shortest Path Problem

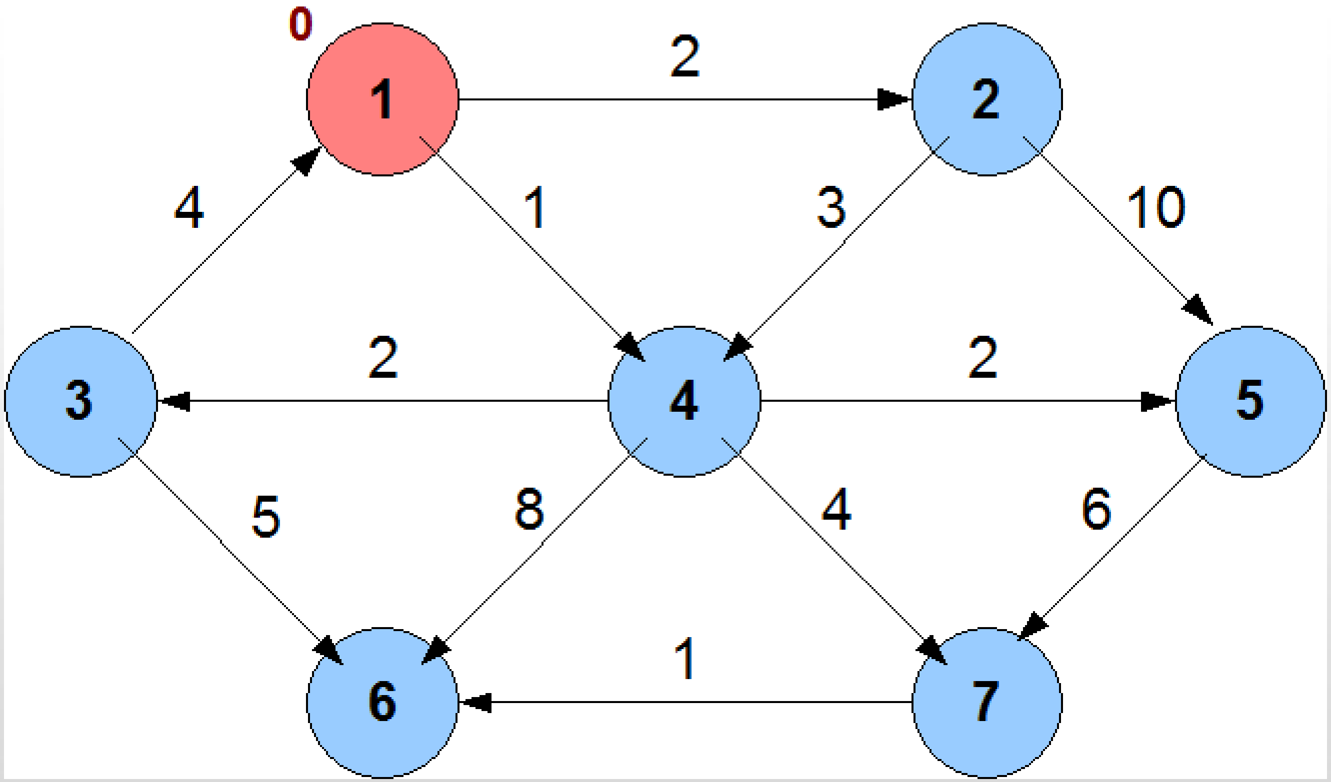
Objective

- Determine the cost of the shortest path (in terms of the costs assigned to the directed edges) from the source vertex to *every other* vertex in V.

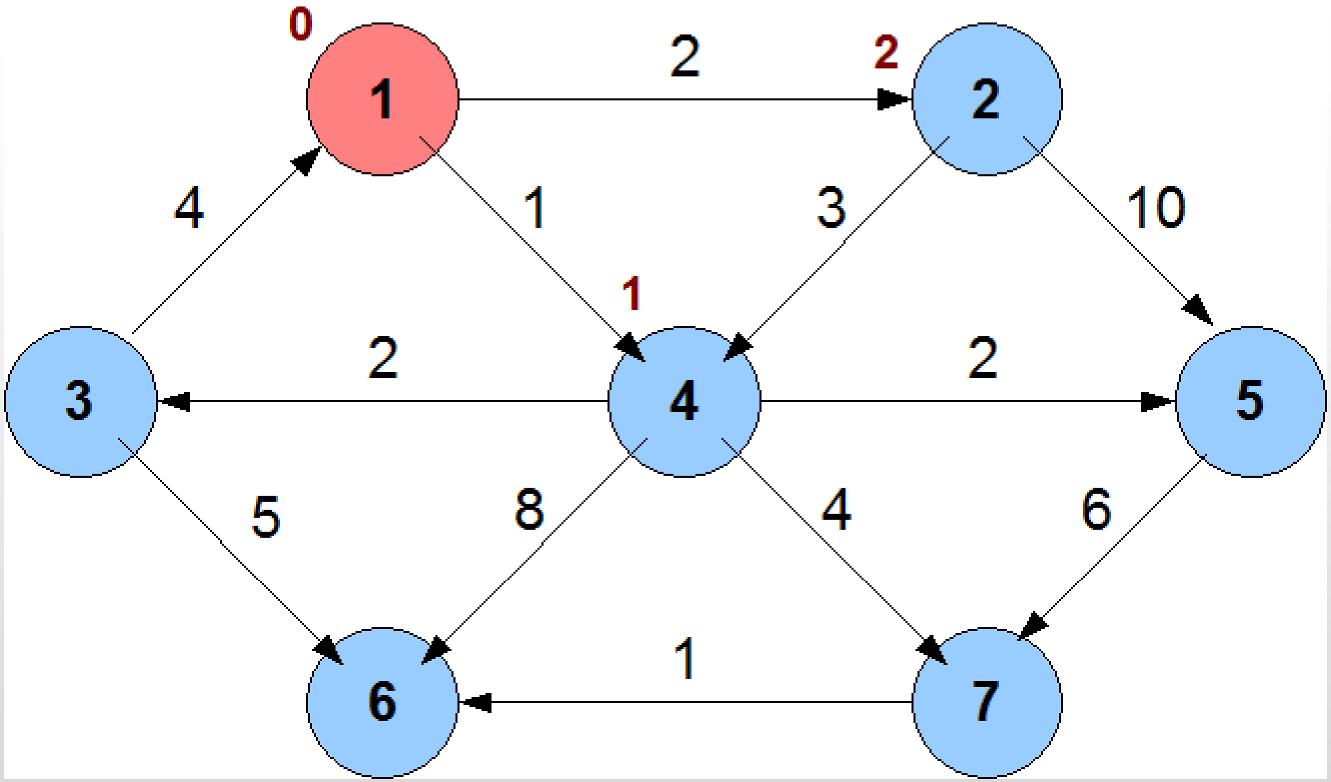
Single Source Shortest Path Problem



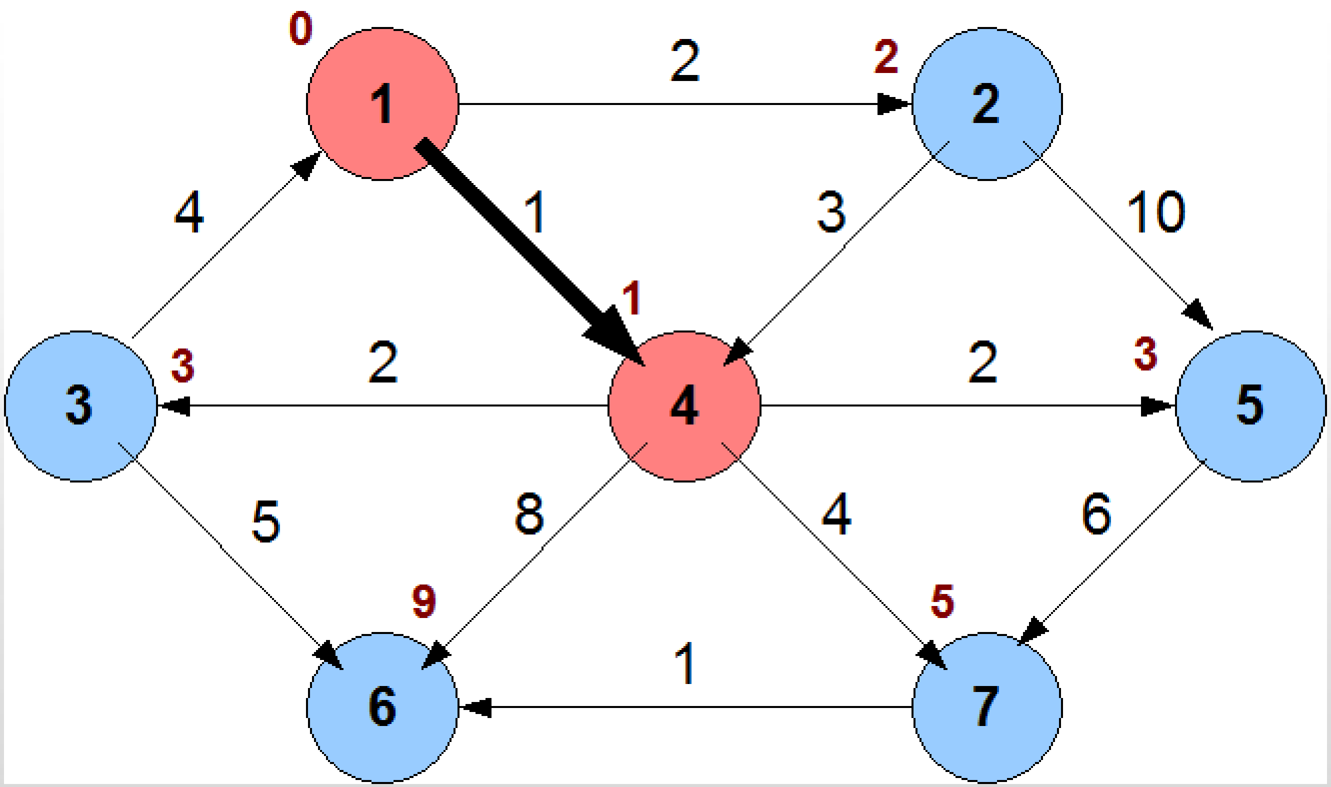
Single Source Shortest Path Problem



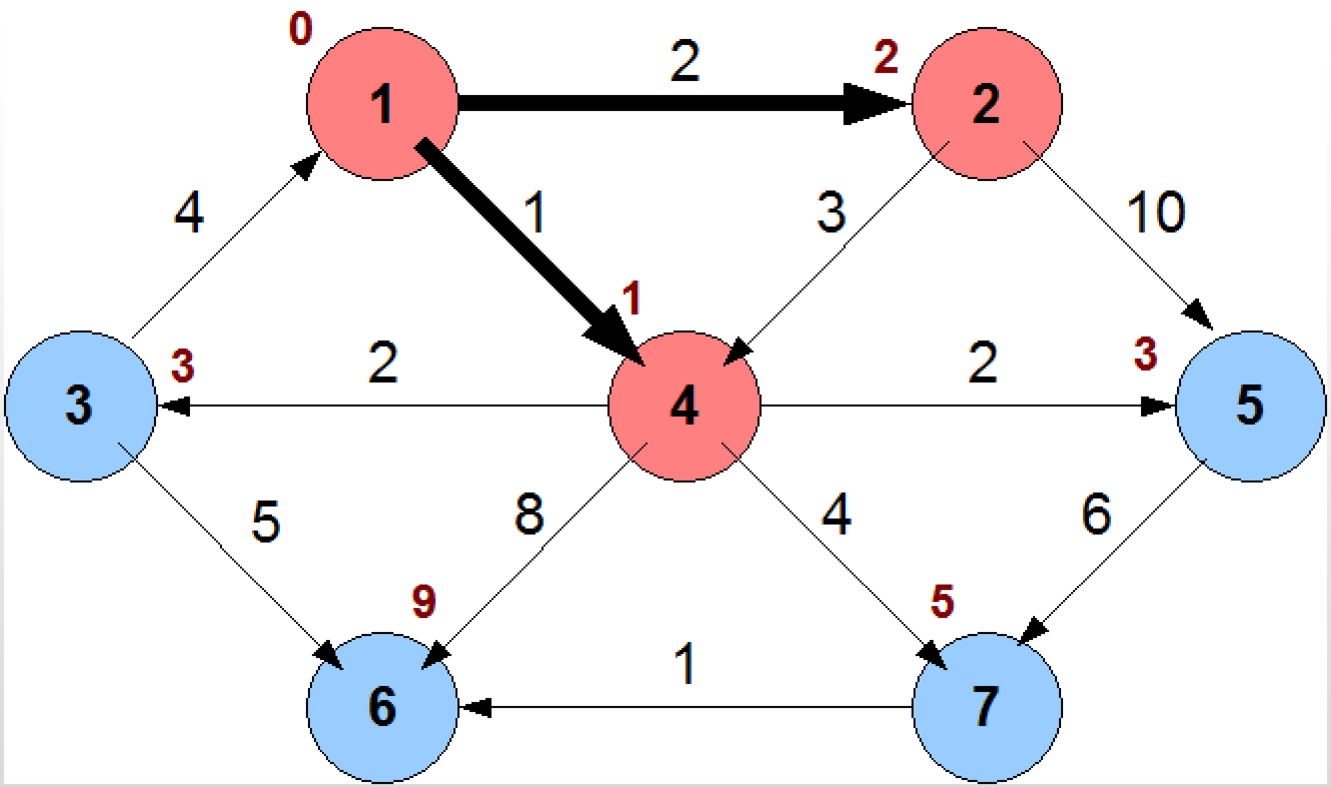
Single Source Shortest Path Problem



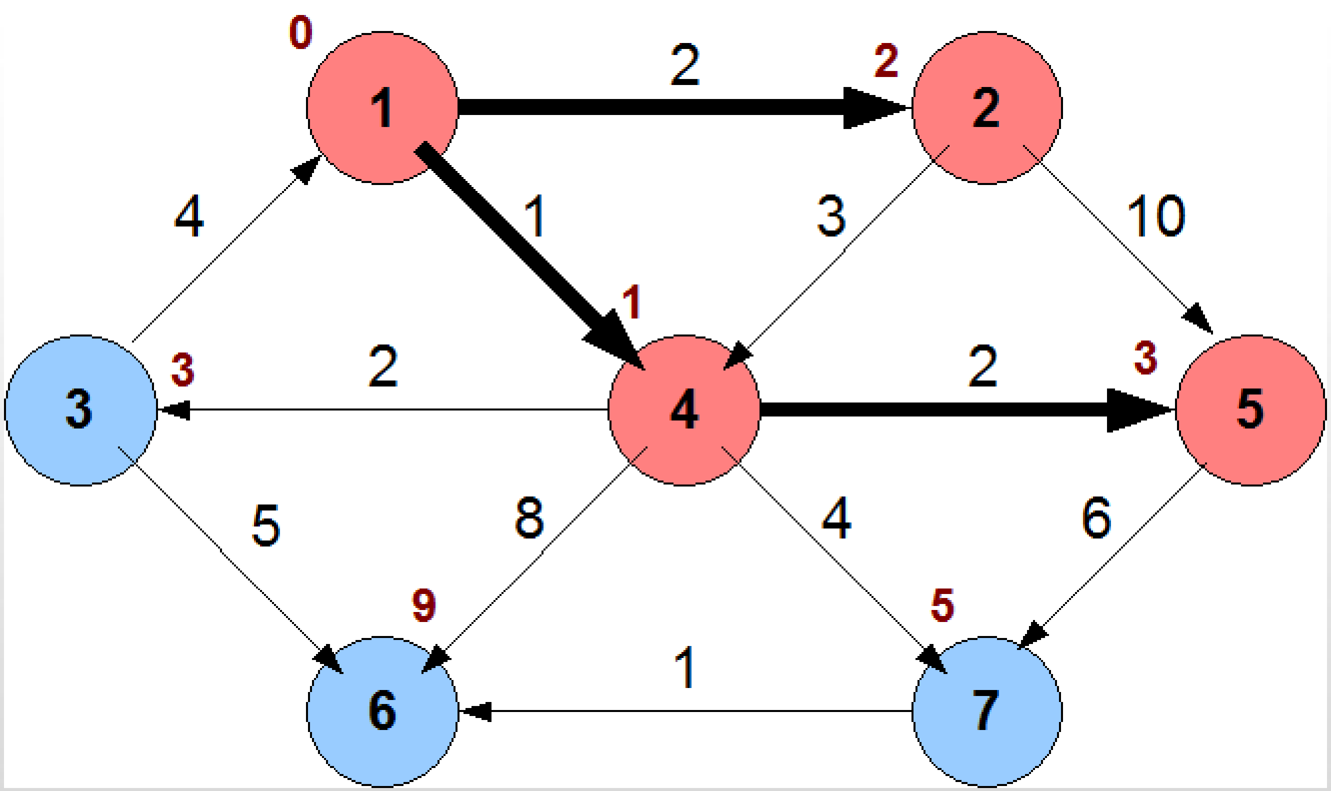
Single Source Shortest Path Problem



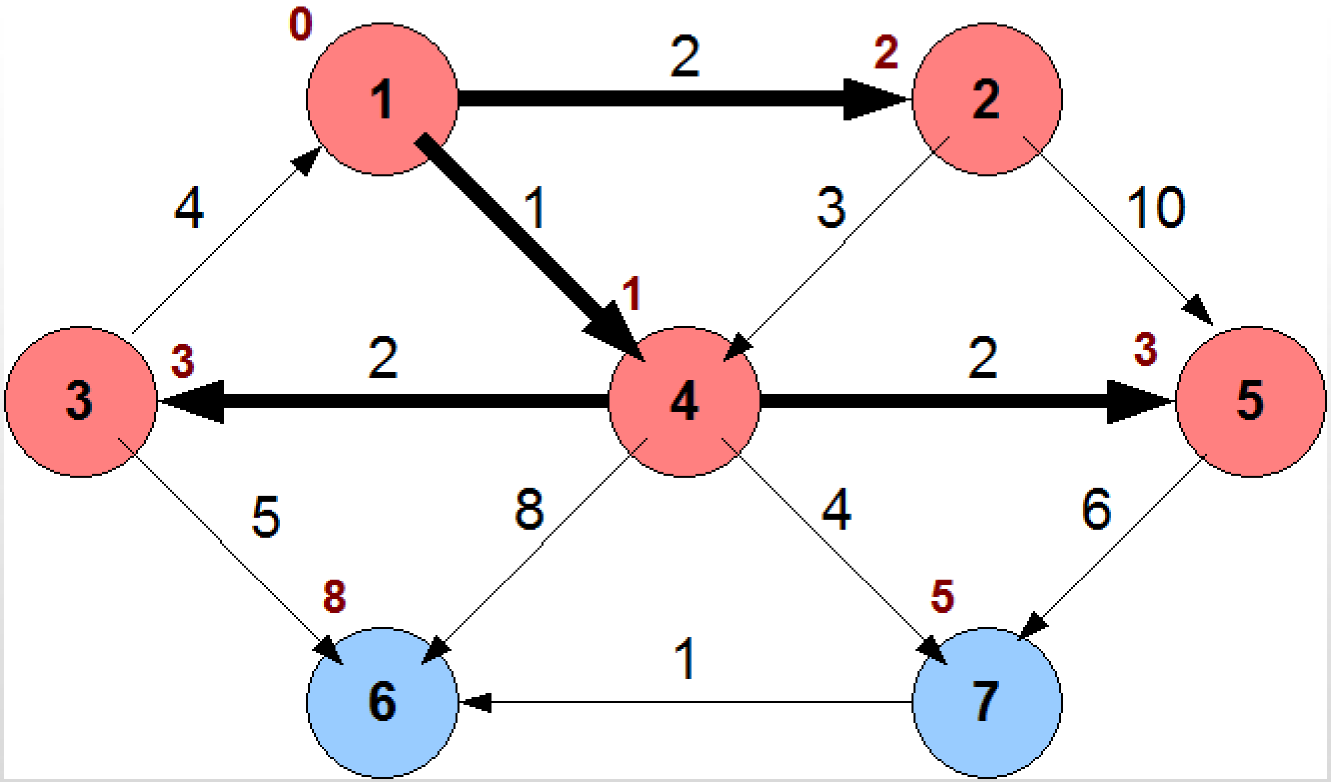
Single Source Shortest Path Problem



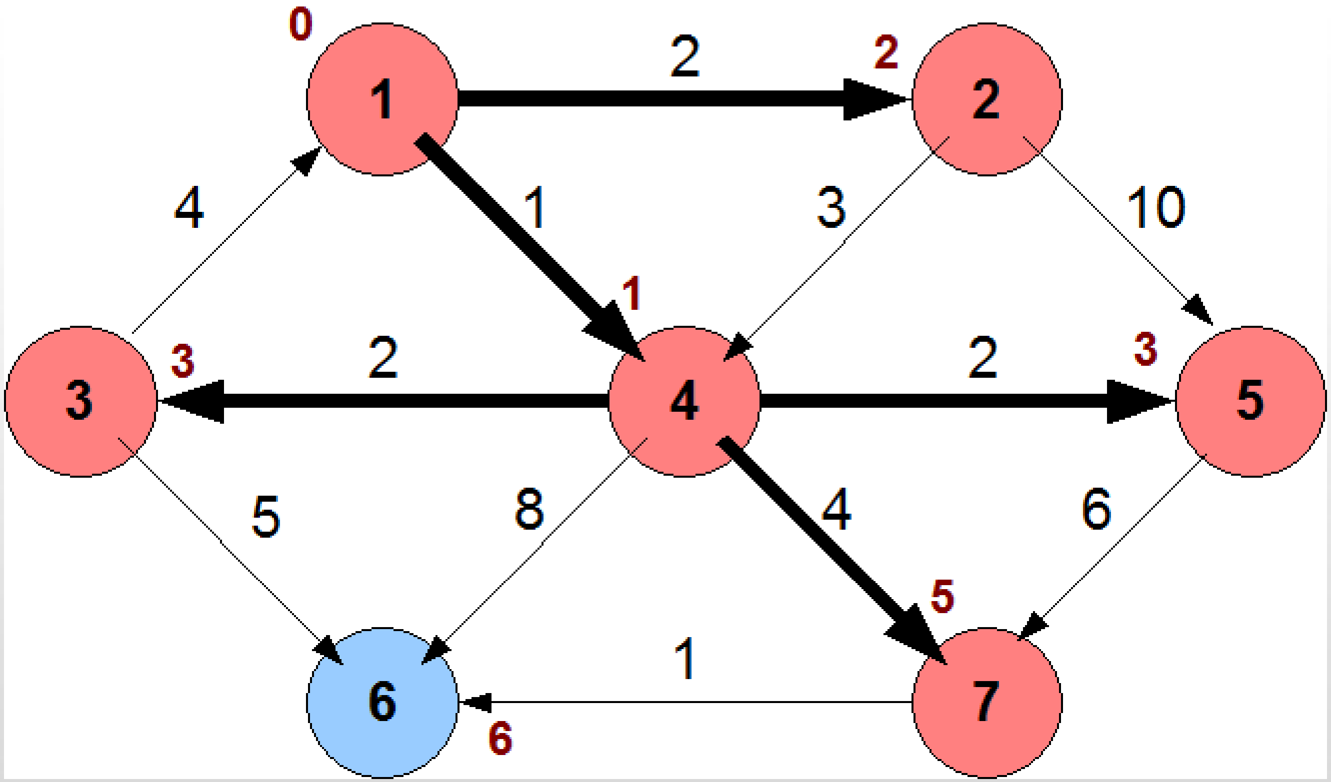
Single Source Shortest Path Problem



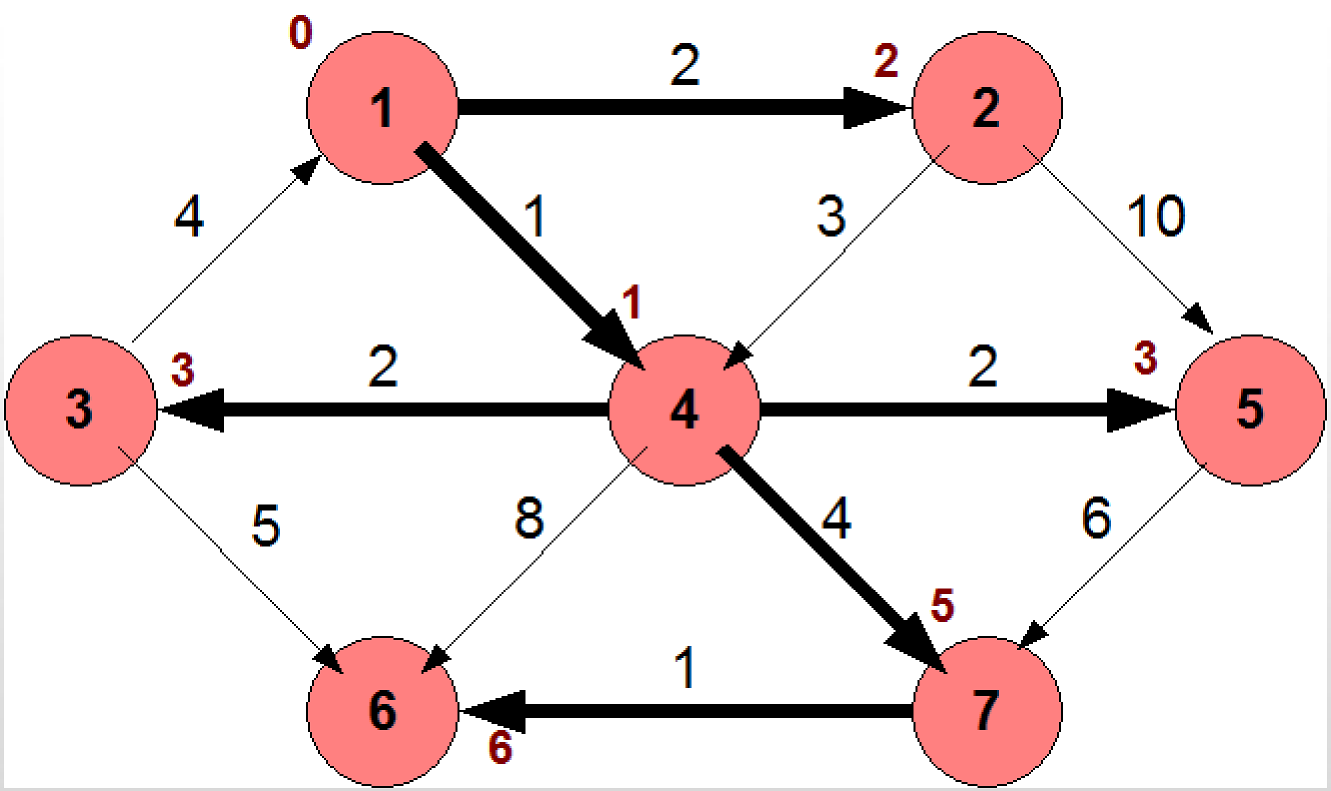
Single Source Shortest Path Problem



Single Source Shortest Path Problem



Single Source Shortest Path Problem



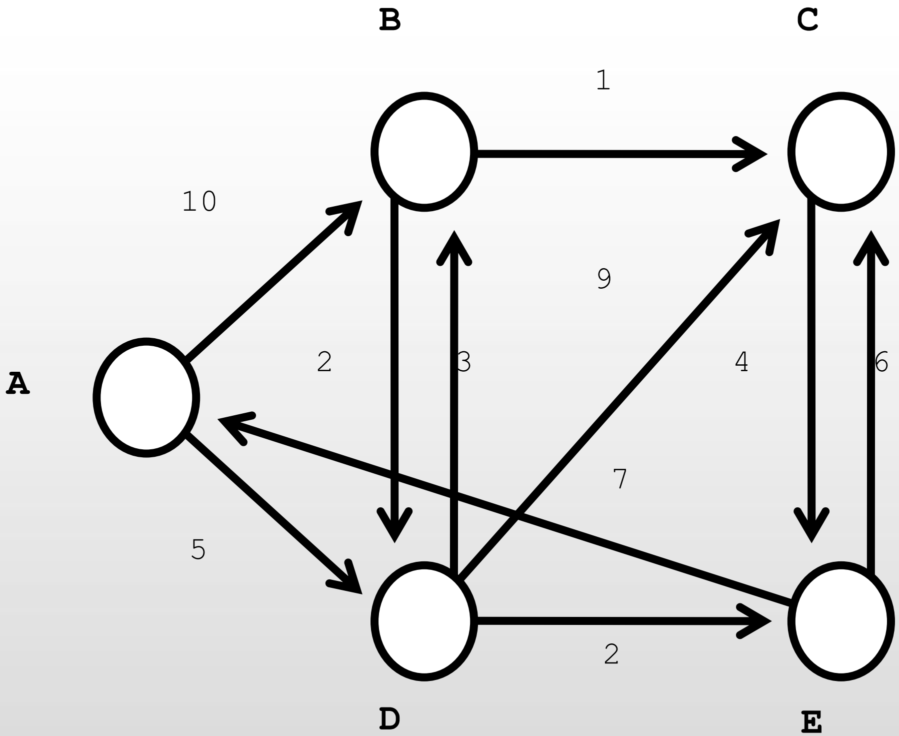
Solution: Dijkstra's Algorithm

Outline:

- Let the set *S* initially contain *s*.
- Let the set of vertices Q initially contain vertices in *V*.
- While Q is not empty
  - Choose a vertex *u* from set Q such that vertex *u* has the cheapest path to the source *s*. This cheapest path should include only those vertices *currently* in set *S*.
  - Add vertex *u* to the set *S*.
  - Remove vertex *u* from the set Q.

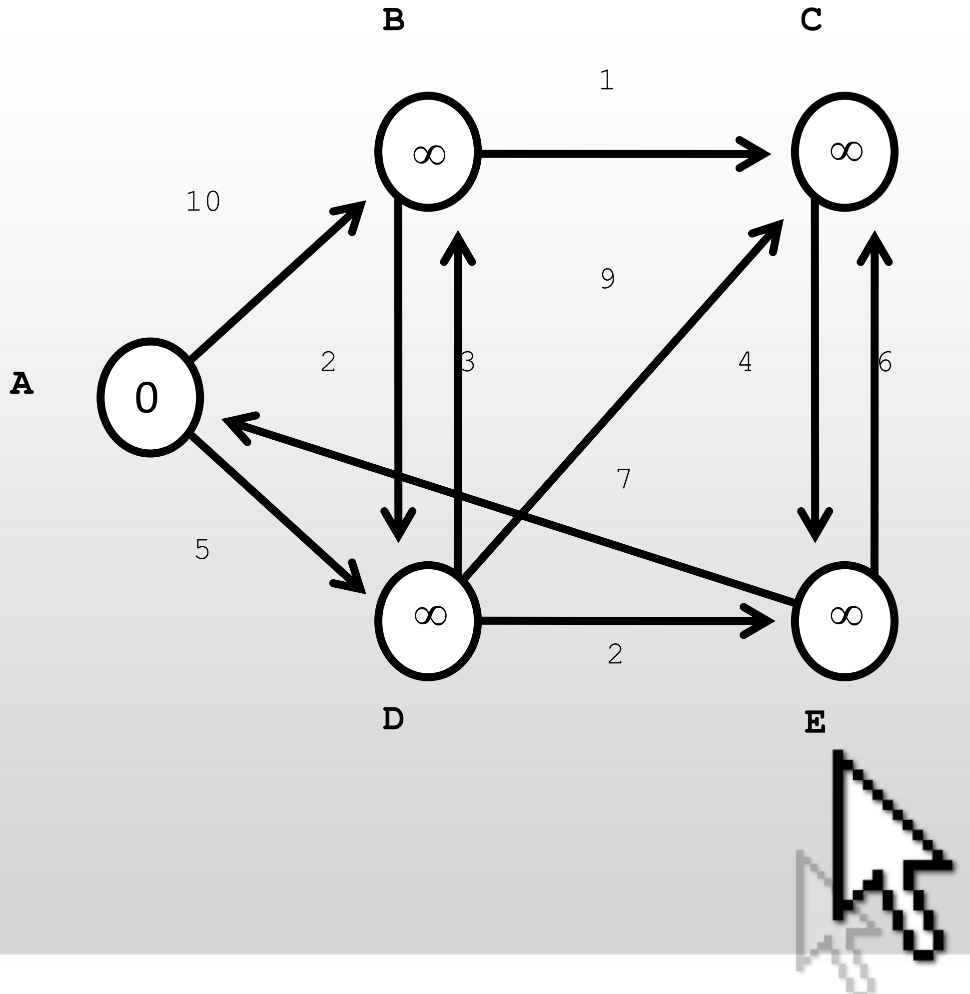
Example (starting at A)

Cost Matrix					
	A	**B**	**C**	**D**	**E**
A	**0**	**10**	∞	**5**	∞
B	∞	**0**	**1**	**2**	∞
C	∞	∞	**0**	∞	**4**
D	∞	**3**	**9**	**0**	**2**
E	**7**	∞	**6**	∞	**0**



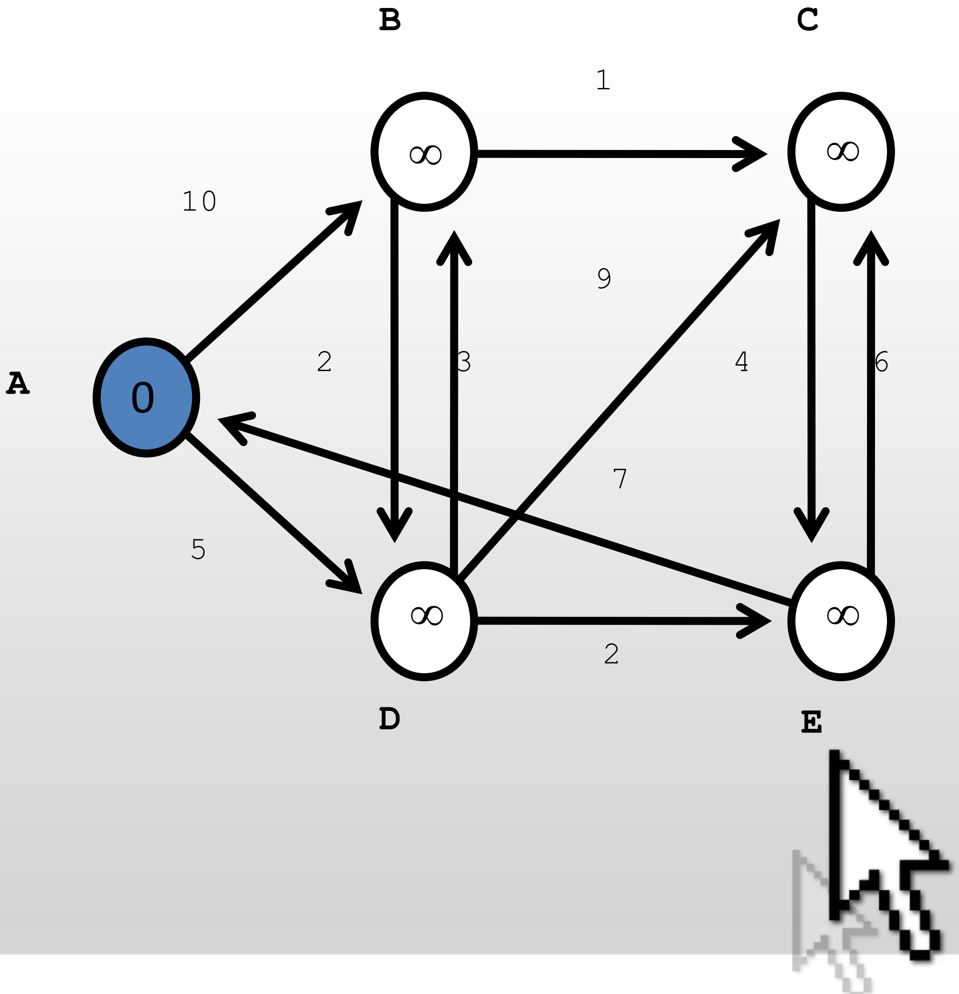
Initialization : d[v] = ∞ and d[s] = 0
S = {} and Q ={A,B,C,D,E}

Cost Matrix					
	A	**B**	**C**	**D**	**E**
A	**0**	**10**	∞	**5**	∞
B	∞	**0**	**1**	**2**	∞
C	∞	∞	**0**	∞	**4**
D	∞	**3**	**9**	**0**	**2**
E	**7**	∞	**6**	∞	**0**



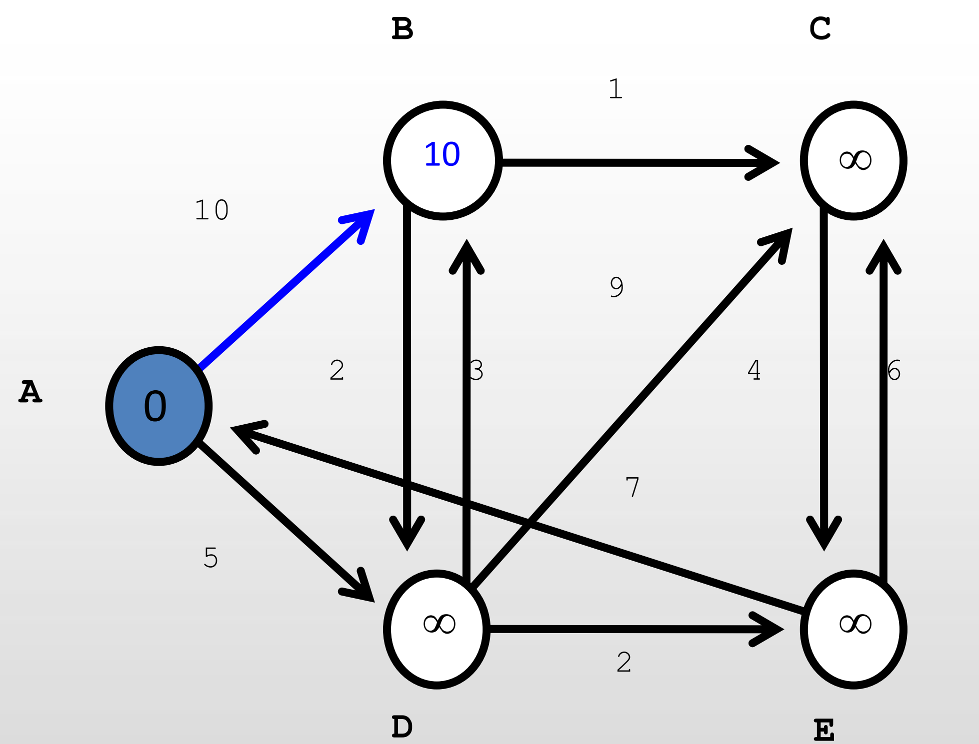
Iter 1: u = A
S = {A} and Q ={B,C,D,E}

Cost Matrix					
	A	B	C	D	E
A	0	10	∞	5	∞
B	∞	0	1	2	∞
C	∞	∞	0	∞	4
D	∞	3	9	0	2
E	7	∞	6	∞	0



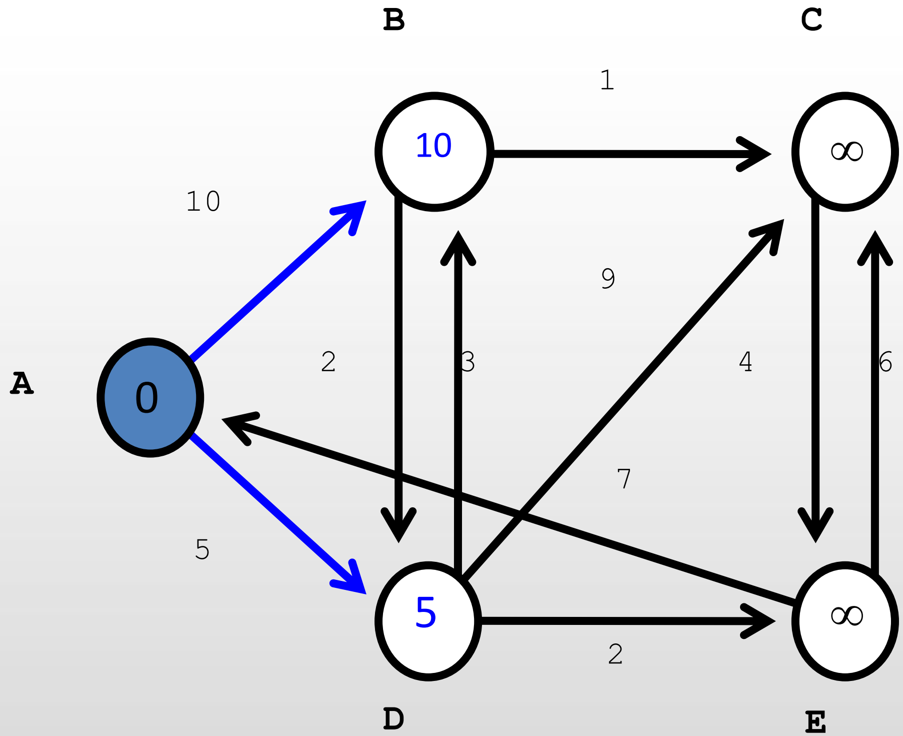
Iter 1: u = A
S = {A} and Q ={B,C,D,E}

Cost Matrix					
	A	**B**	**C**	**D**	**E**
A	**0**	**10**	∞	**5**	∞
B	∞	**0**	**1**	**2**	∞
C	∞	∞	**0**	∞	**4**
D	∞	**3**	**9**	**0**	**2**
E	**7**	∞	**6**	∞	**0**



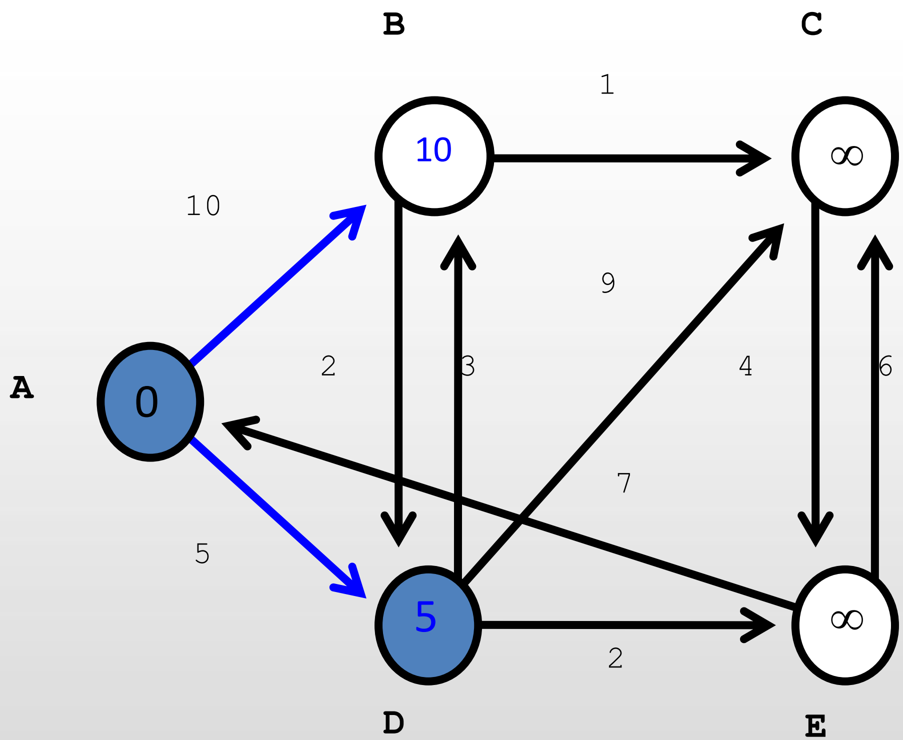
Iter 1: u = A
S = {A} and Q ={B,C,D,E}

Cost Matrix					
	A	**B**	**C**	**D**	**E**
A	**0**	**10**	∞	**5**	∞
B	∞	**0**	**1**	**2**	∞
C	∞	∞	**0**	∞	**4**
D	∞	**3**	**9**	**0**	**2**
E	**7**	∞	**6**	∞	**0**



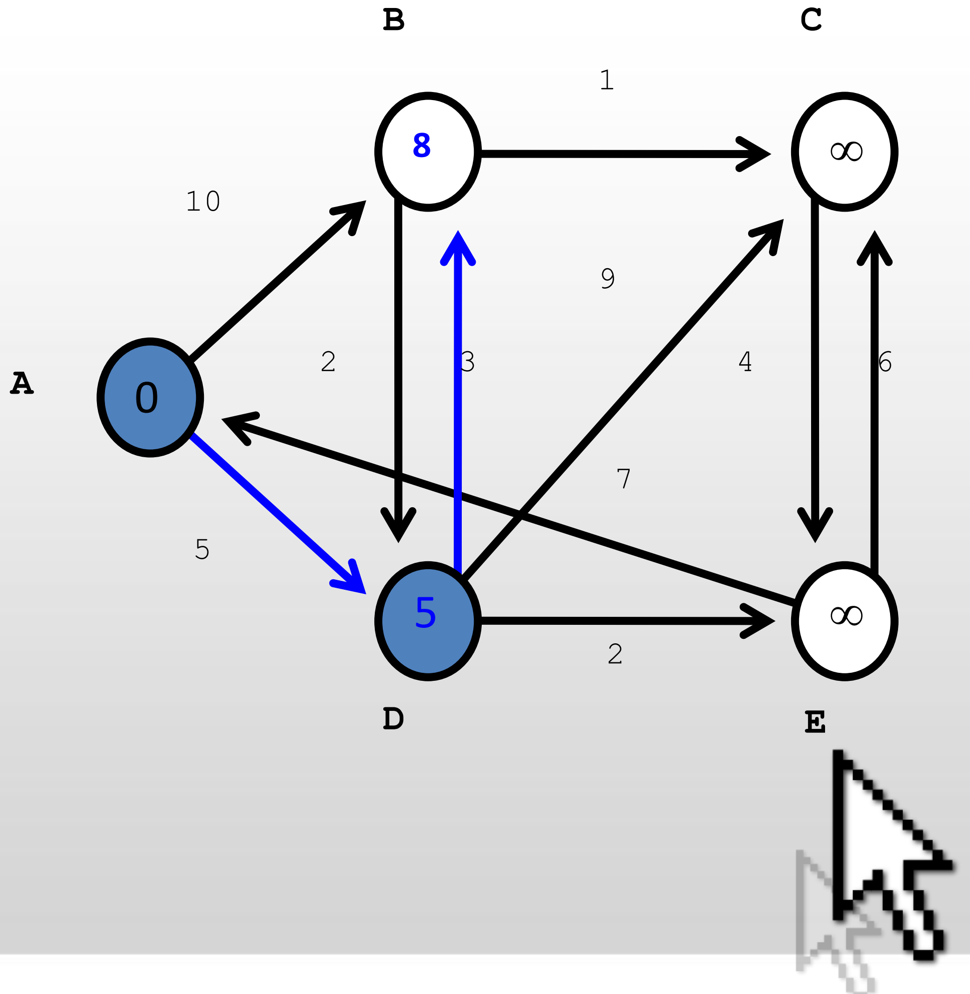
Iter 2: u = D
S = {A, D} and Q ={B,C,E}

Cost Matrix					
	A	**B**	**C**	**D**	**E**
A	**0**	**10**	∞	**5**	∞
B	∞	**0**	**1**	**2**	∞
C	∞	∞	**0**	∞	**4**
D	∞	**3**	**9**	**0**	**2**
E	**7**	∞	**6**	∞	**0**



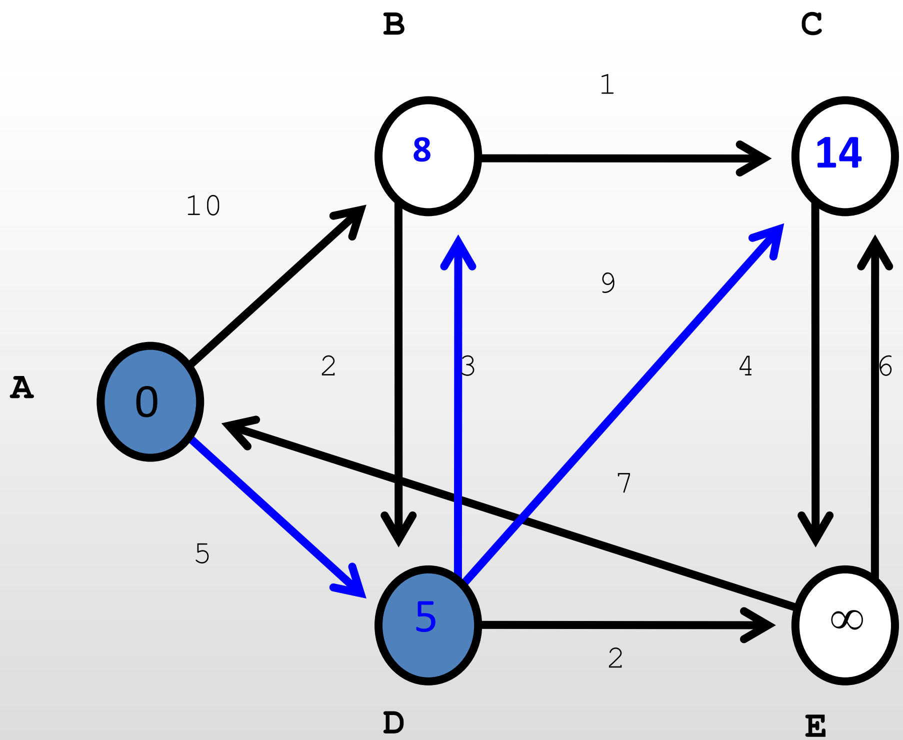
Iter 2: u = D
S = {A, D} and Q ={B,C,E}

Cost Matrix					
	A	B	C	D	E
A	0	10	∞	5	∞
B	∞	0	1	2	∞
C	∞	∞	0	∞	4
D	∞	3	9	0	2
E	7	∞	6	∞	0



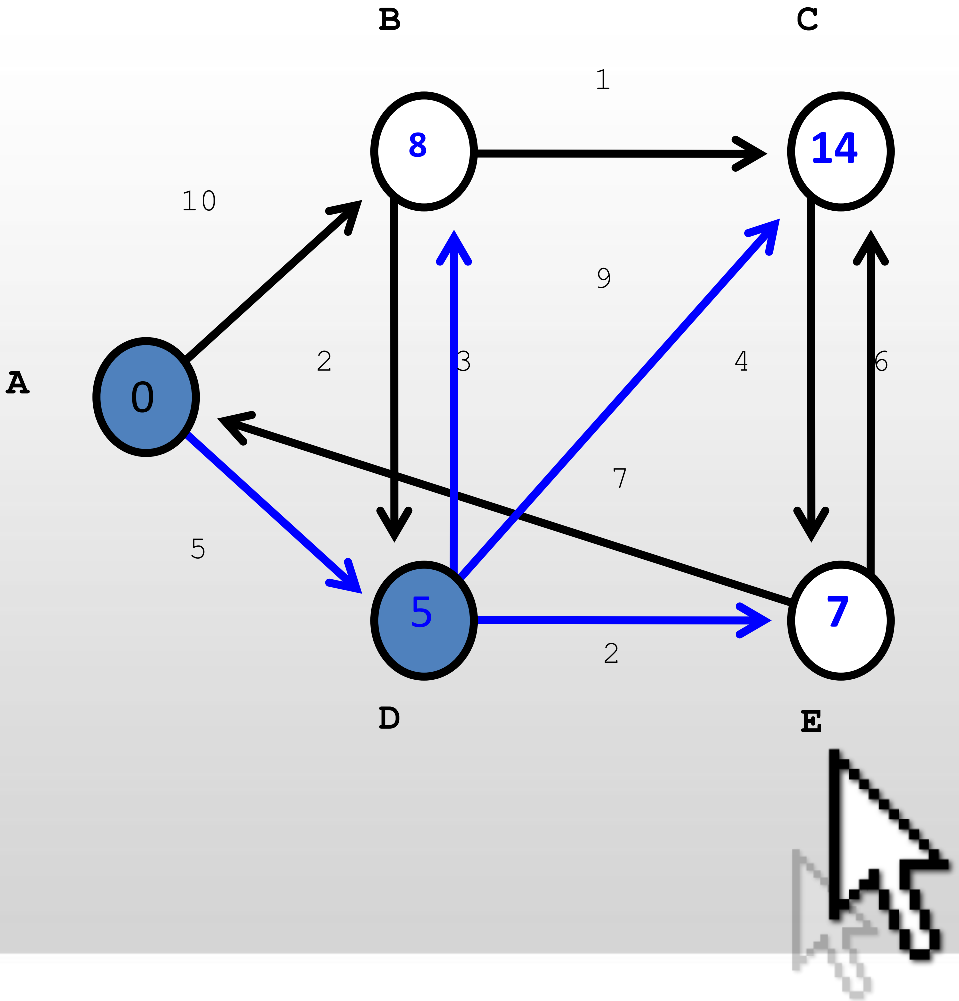
Iter 2: u = D
S = {A, D} and Q ={B,C,E}

Cost Matrix					
	A	B	C	D	E
A	0	10	∞	5	∞
B	∞	0	1	2	∞
C	∞	∞	0	∞	4
D	∞	3	9	0	2
E	7	∞	6	∞	0



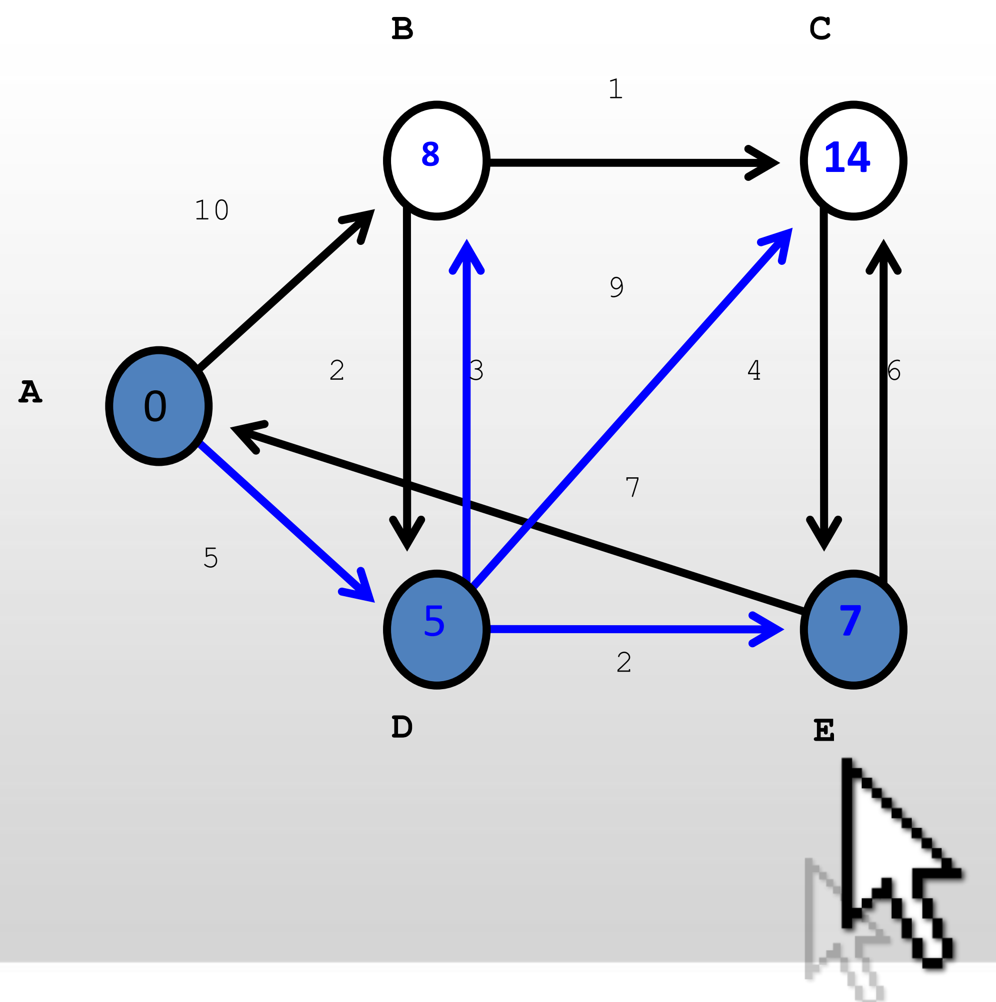
Iter 2: u = D
S = {A, D} and Q ={B,C,E}

Cost Matrix					
	A	**B**	**C**	**D**	**E**
A	0	10	∞	5	∞
B	∞	0	1	2	∞
C	∞	∞	0	∞	4
D	∞	3	9	0	2
E	7	∞	6	∞	0



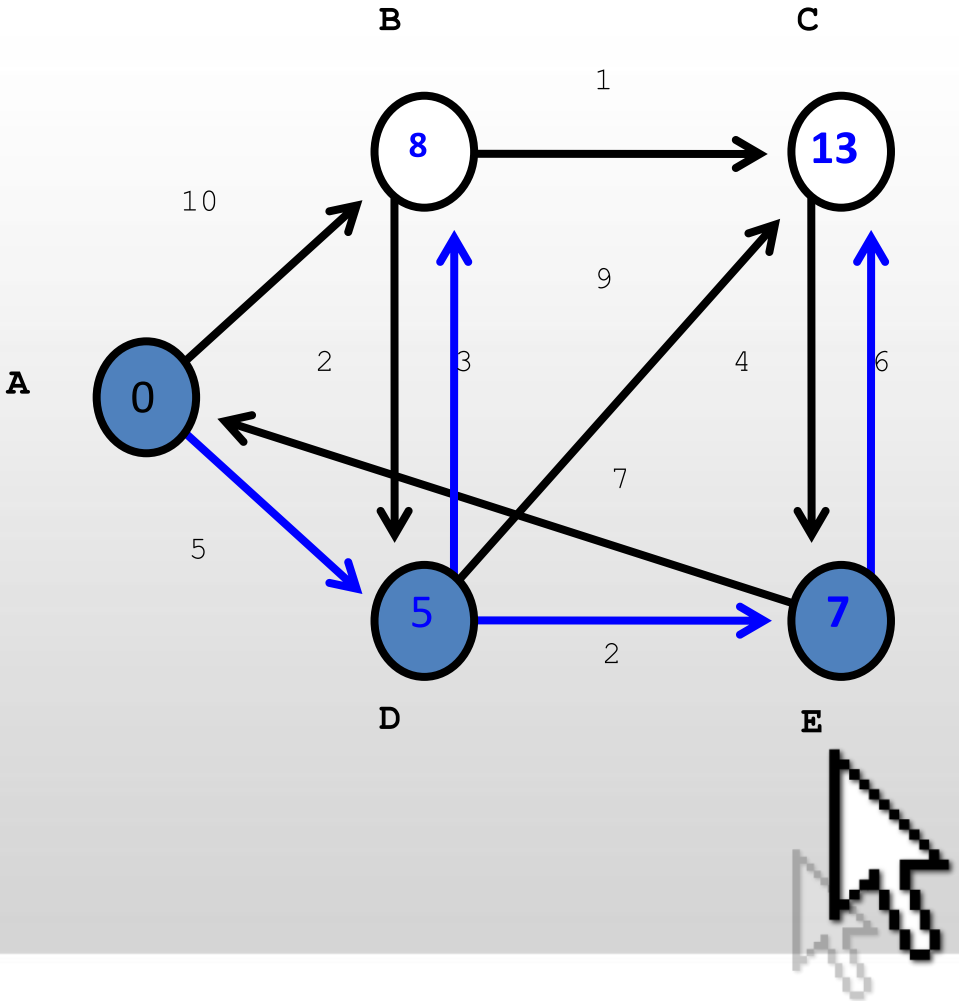
Iter 3: u = E
S = {A, D, E} and Q ={B,C}

Cost Matrix					
	A	**B**	**C**	**D**	**E**
A	**0**	**10**	∞	**5**	∞
B	∞	**0**	**1**	**2**	∞
C	∞	∞	**0**	∞	**4**
D	∞	**3**	**9**	**0**	**2**
E	**7**	∞	**6**	∞	**0**



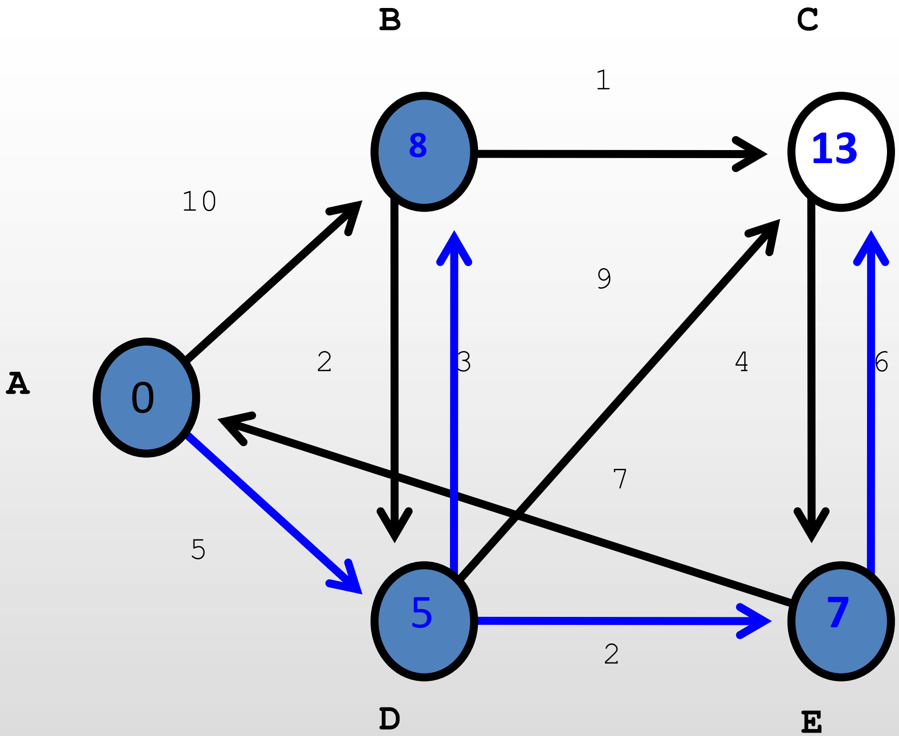
Iter 3: u = E
S = {A, D, E} and Q ={B,C}

Cost Matrix					
	A	B	C	D	E
A	0	10	∞	5	∞
B	∞	0	1	2	∞
C	∞	∞	0	∞	4
D	∞	3	9	0	2
E	7	∞	6	∞	0



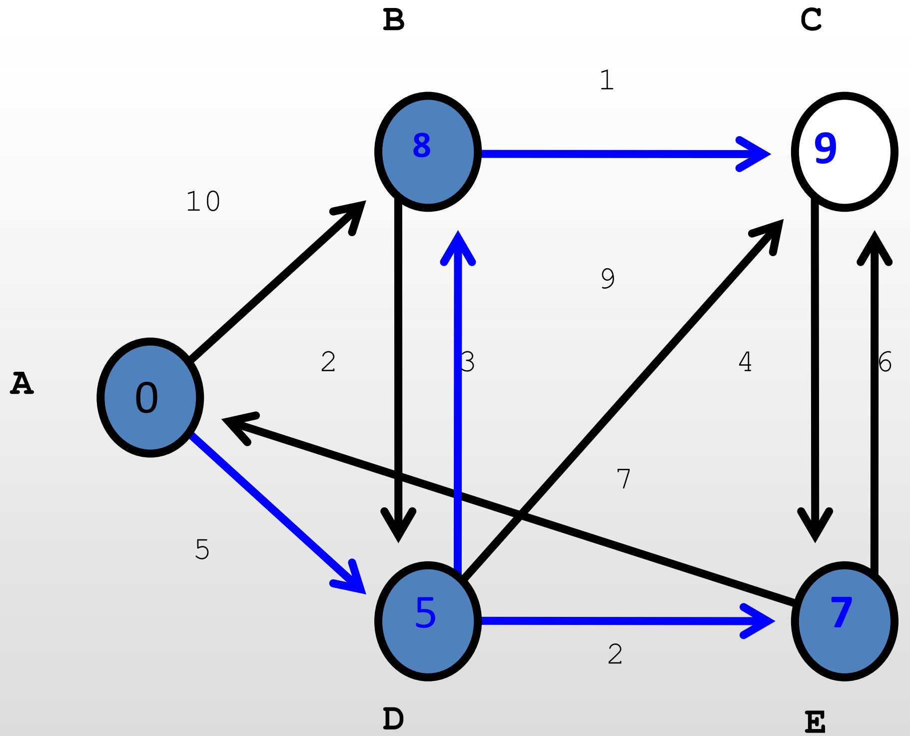
Iter 4: u = B
S = {A, D, E, B} and Q ={C}

Cost Matrix					
	A	**B**	**C**	**D**	**E**
A	0	10	∞	5	∞
B	∞	0	1	2	∞
C	∞	∞	0	∞	4
D	∞	3	9	0	2
E	7	∞	6	∞	0



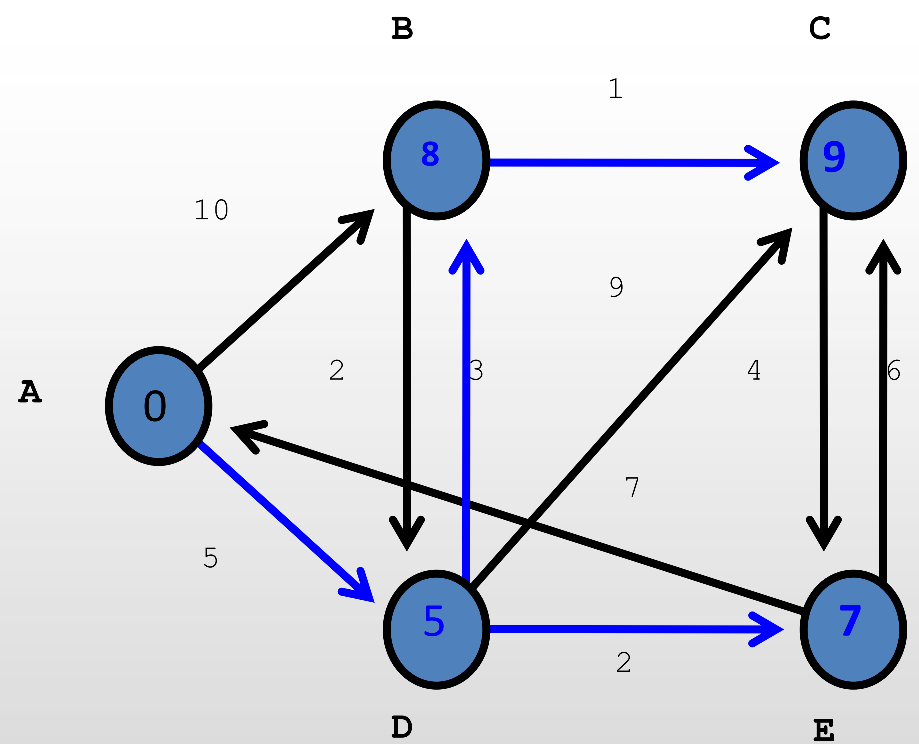
Iter 4: u = B
S = {A, D, E, B} and Q ={C}

Cost Matrix					
	A	B	C	D	E
A	0	10	∞	5	∞
B	∞	0	1	2	∞
C	∞	∞	0	∞	4
D	∞	3	9	0	2
E	7	∞	6	∞	0



Iter 5: u = C
S = {A, D, E, B, C} and Q ={}

Cost Matrix					
	A	**B**	**C**	**D**	**E**
A	0	10	∞	5	∞
B	∞	0	1	2	∞
C	∞	∞	0	∞	4
D	∞	3	9	0	2
E	7	∞	6	∞	0



All Pairs Shortest Path

Given

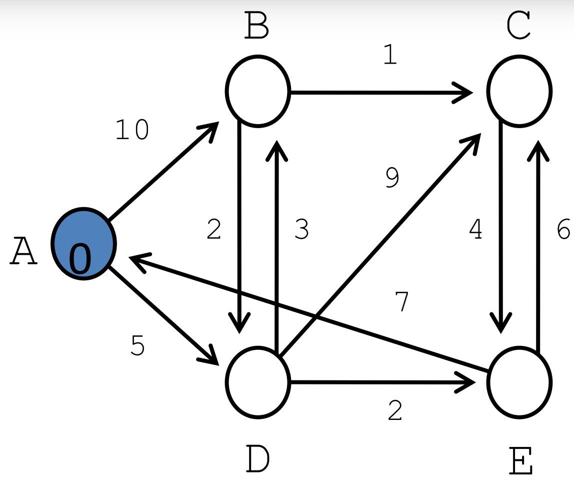
- A directed graph $G = (V, E)$ where
- edges or arcs are assigned *nonnegative* costs or weights

Objective

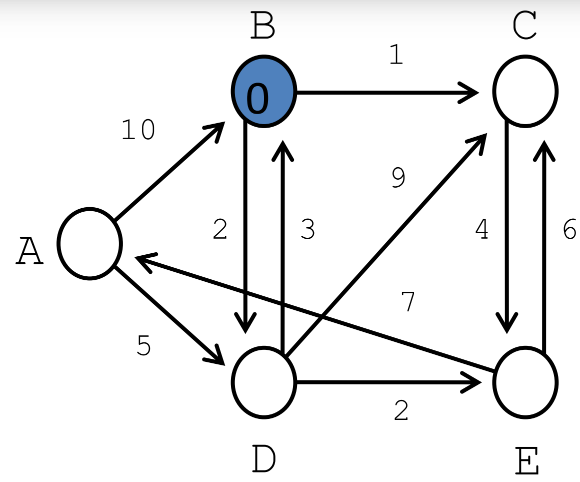
- Find the shortest distance (in terms of the costs assigned to the directed edges) from any pair of nodes in the graph G.

Solution: Floyd's Algorithm

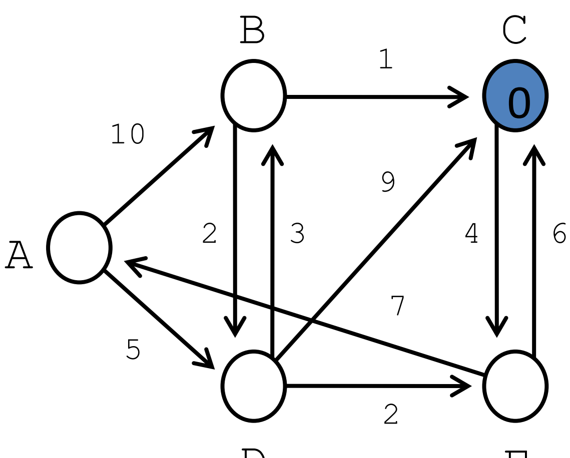
	A	B	C	D	E
A	0	10	∞	5	∞
B	∞	0	1	2	∞
C	∞	∞	0	∞	4
D	∞	3	9	0	2
E	7	∞	6	∞	0



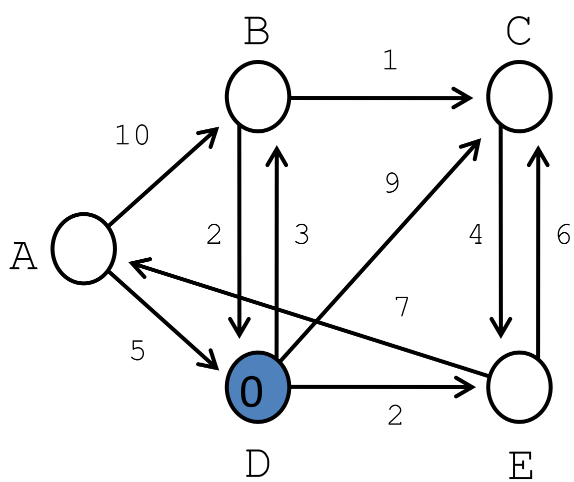
i=A



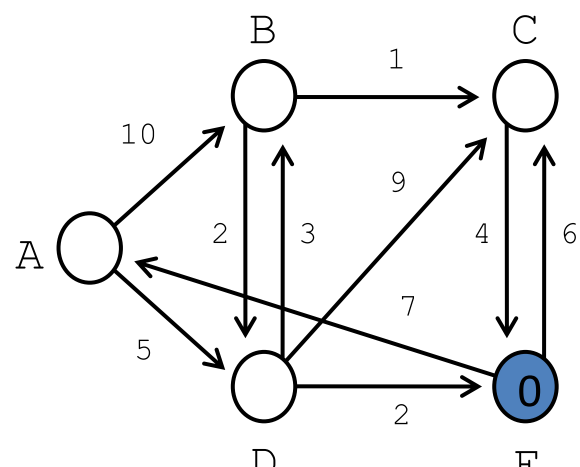
i=B



i=C



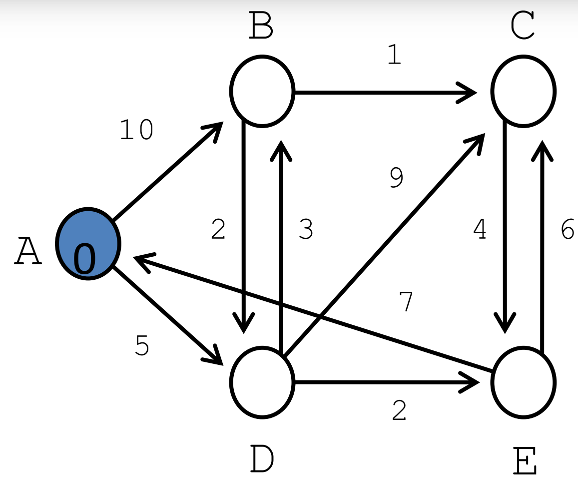
i=D



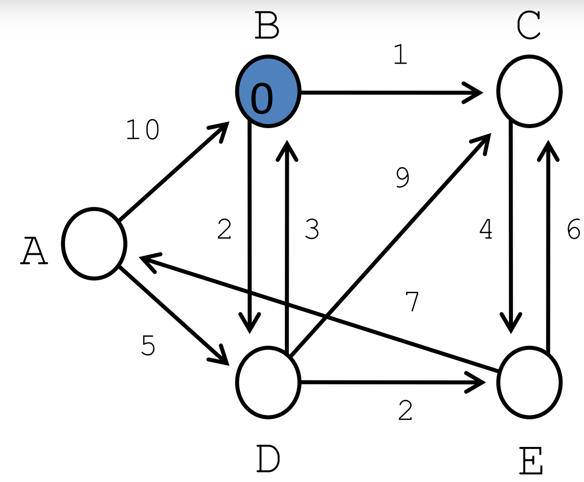
i=E

Solution: Floyd's Algorithm (k= A)

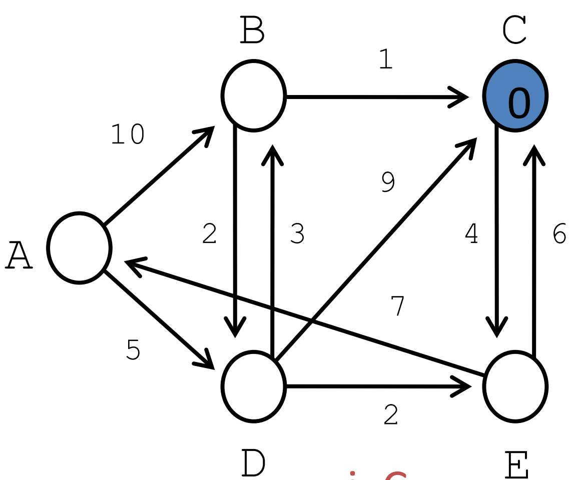
	A	B	C	D	E
A	0	10	∞	5	∞
B	∞	0	1	2	∞
C	∞	∞	0	∞	4
D	∞	3	9	0	2
E	7	∞	6	∞	0



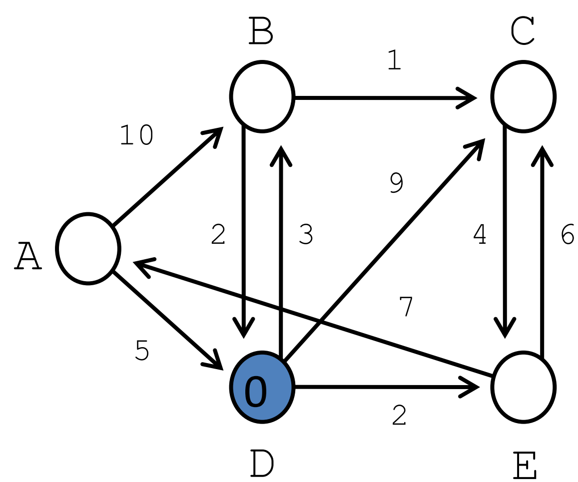
i=A



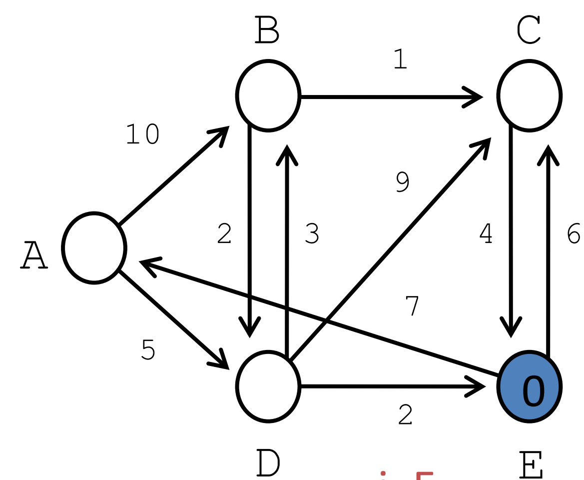
i=B



i=C



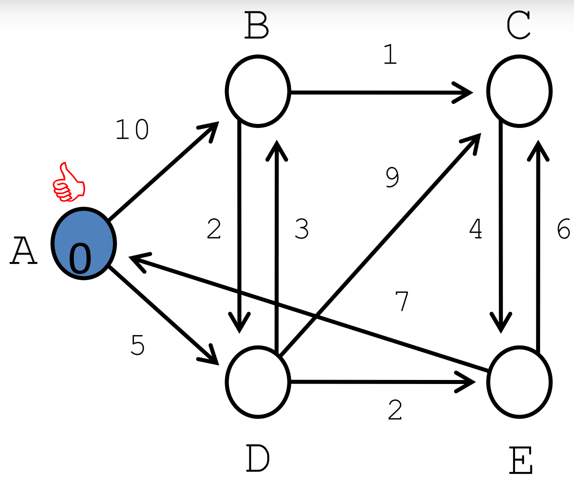
i=D



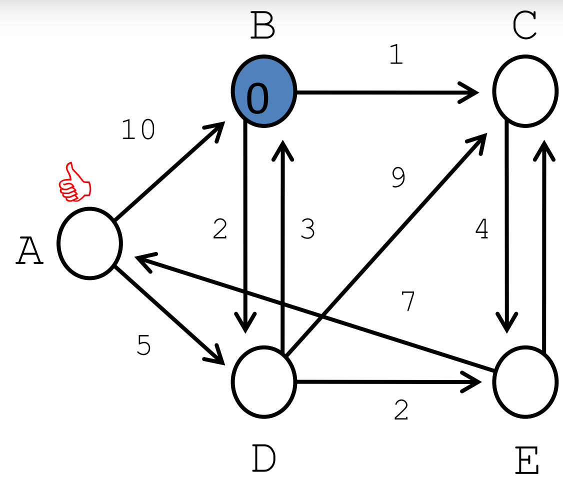
i=E

Solution: Floyd's Algorithm (k= A)

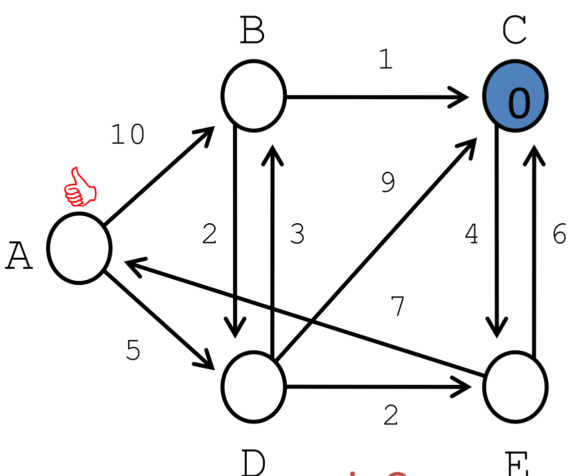
	A	B	C	D	E
A	0	10	∞	5	∞
B	∞	0	1	2	∞
C	∞	∞	0	∞	4
D	∞	3	9	0	2
E	7	∞	6	∞	0



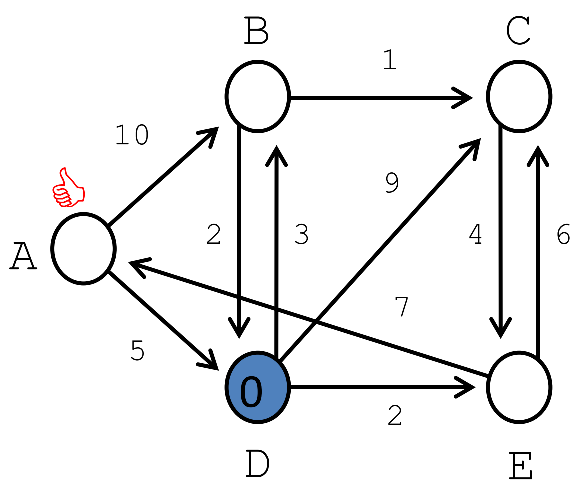
i=A



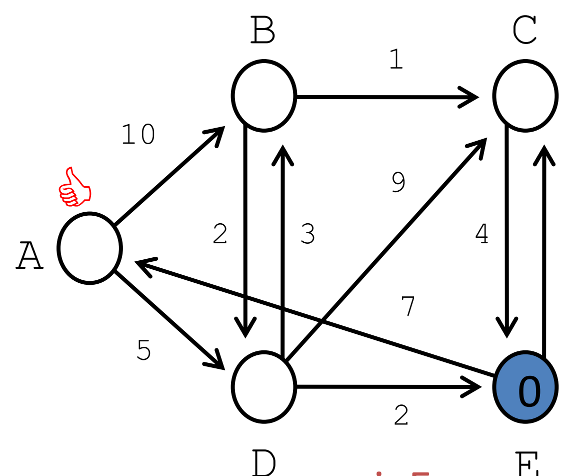
i=B



i=C



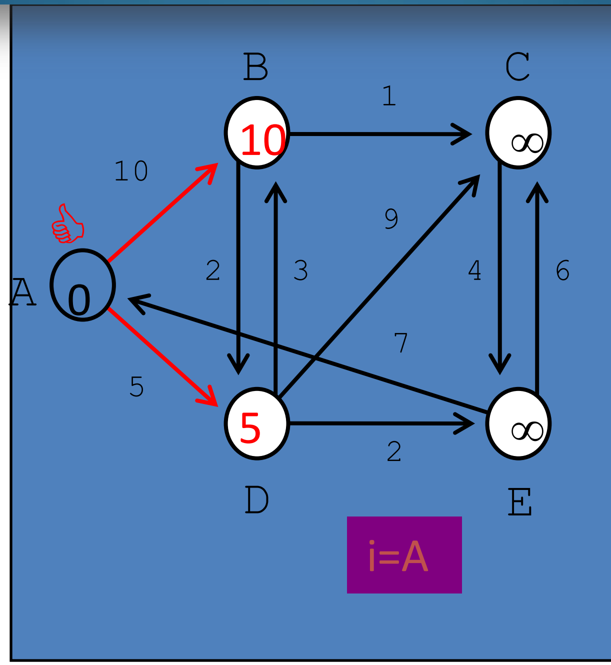
i=D

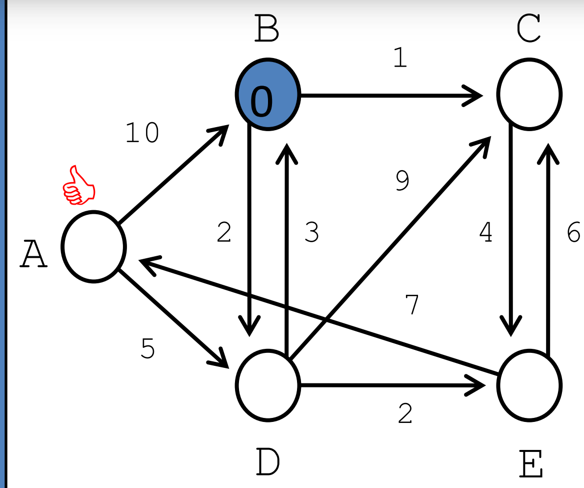


i=E

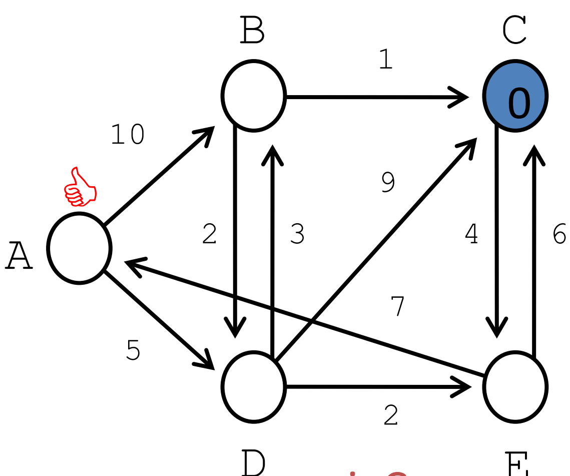
Solution: Floyd's Algorithm (k= A)

	A	B	C	D	E
A	0	10	∞	5	∞
B	∞	0	1	2	∞
C	∞	∞	0	∞	4
D	∞	3	9	0	2
E	7	∞	6	∞	0

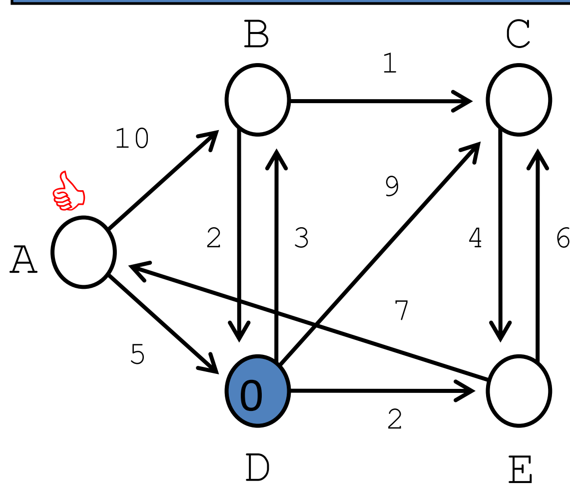




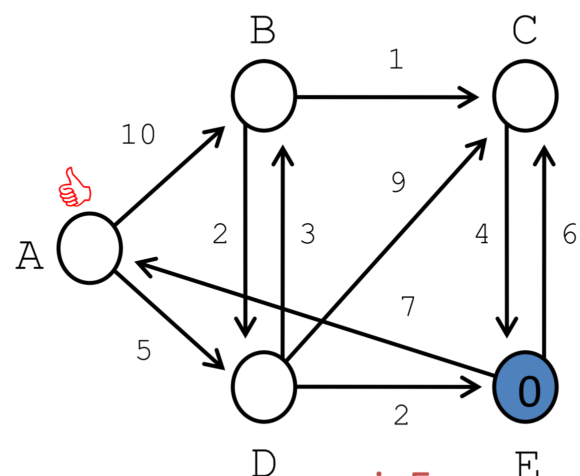
i=B



i=C



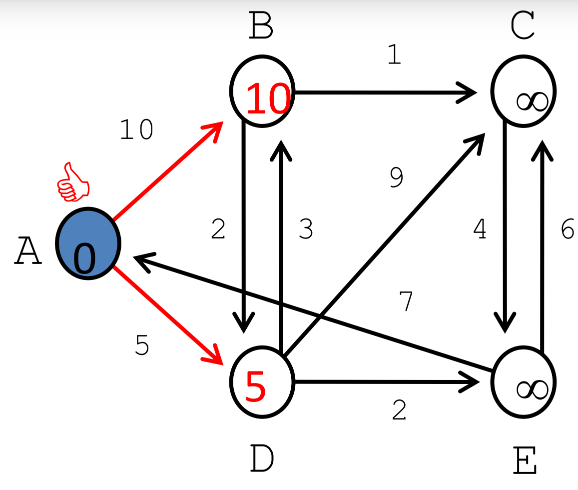
i=D



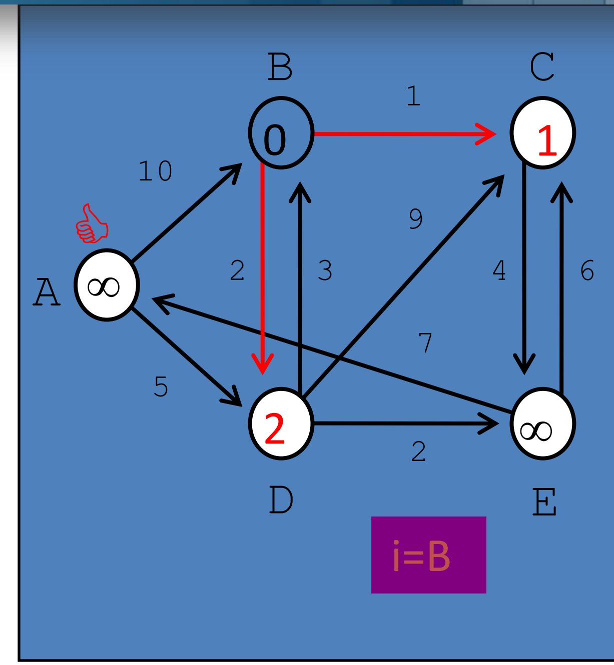
i=E

Solution: Floyd's Algorithm (k= A)

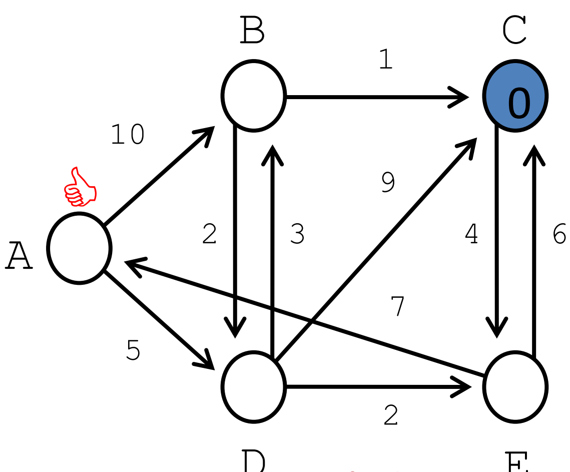
	A	B	C	D	E
A	0	10	∞	5	∞
B	∞	0	1	2	∞
C	∞	∞	0	∞	4
D	∞	3	9	0	2
E	7	∞	6	∞	0



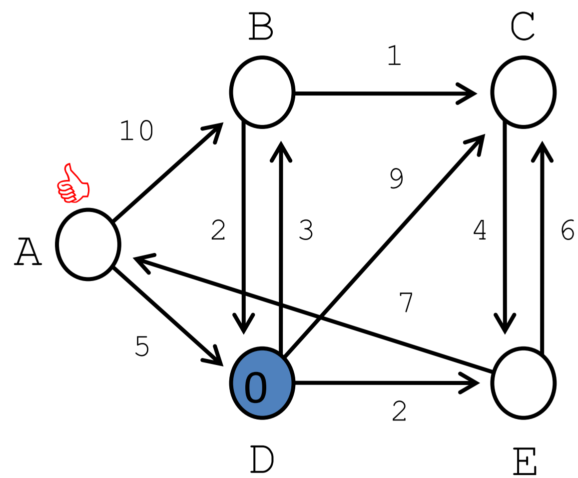
i=A



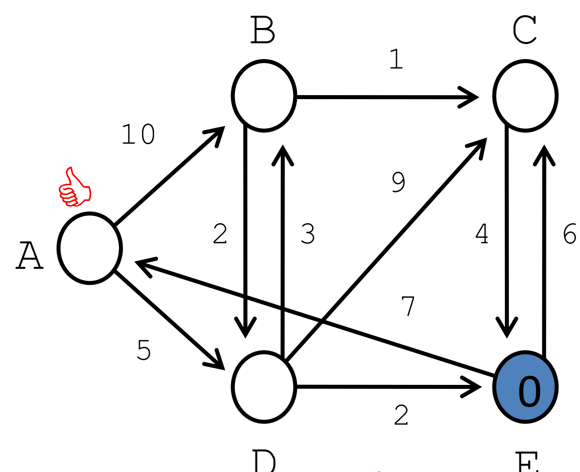
i=B



i=C



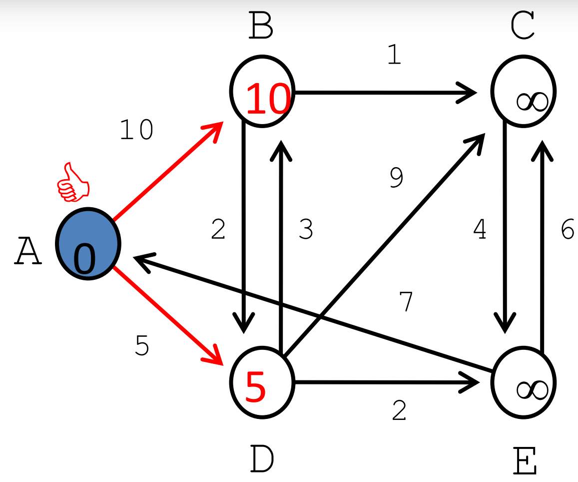
i=D



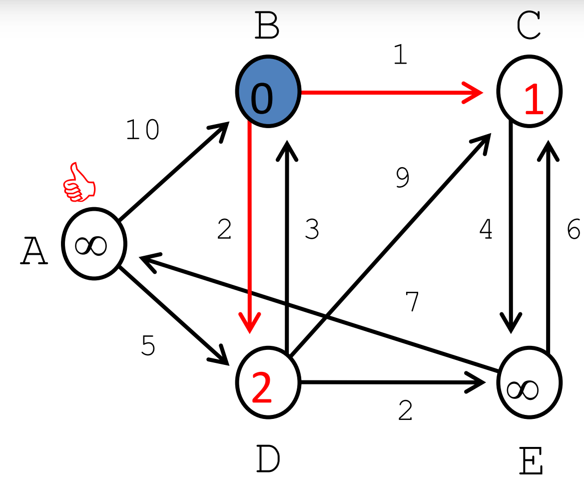
i=E

Solution: Floyd's Algorithm (k= A)

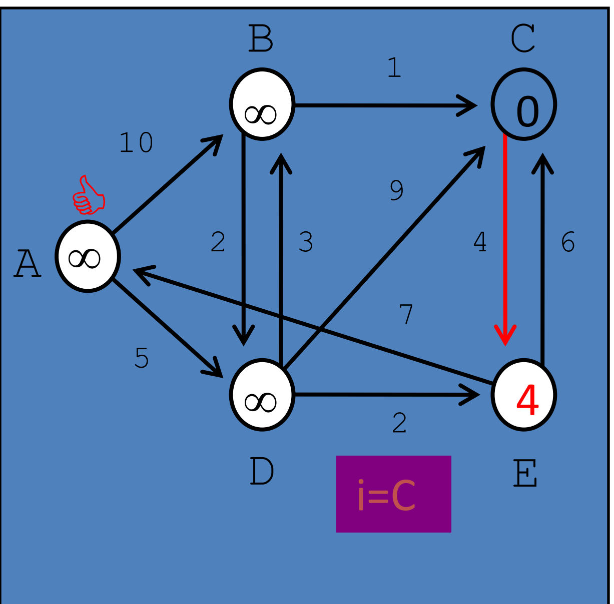
	A	B	C	D	E
A	0	10	∞	5	∞
B	∞	0	1	2	∞
C	∞	∞	0	∞	4
D	∞	3	9	0	2
E	7	∞	6	∞	0

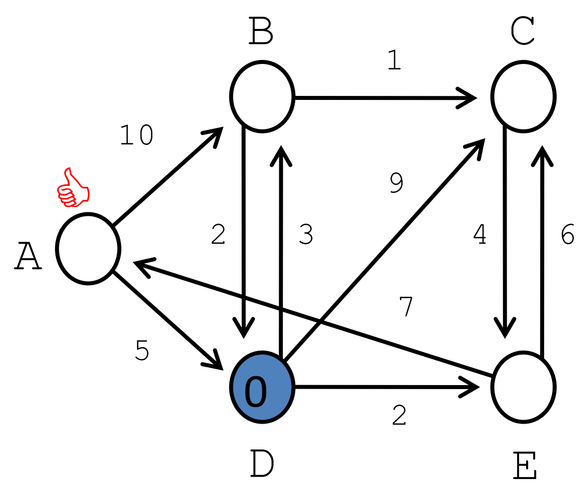


i=A

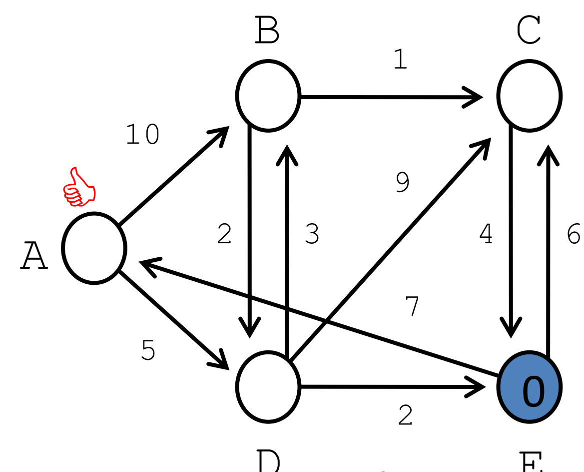


i=B





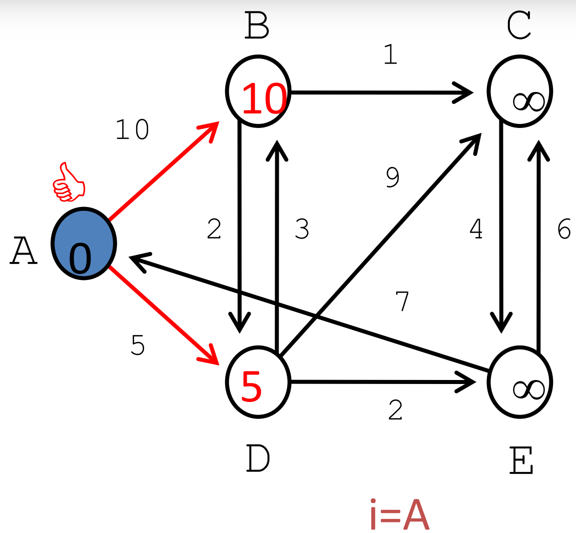
i=D

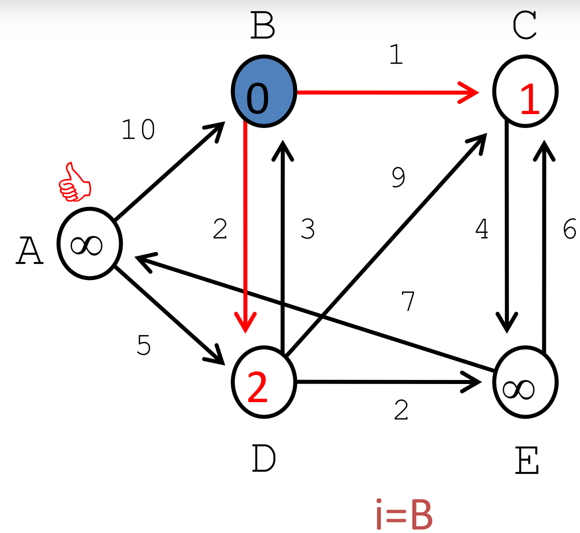


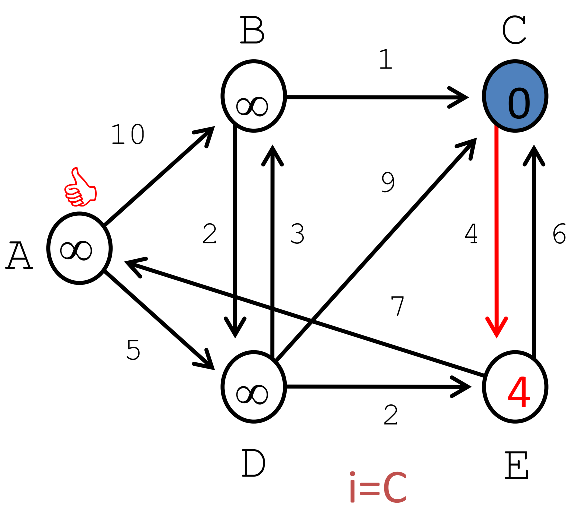
i=E

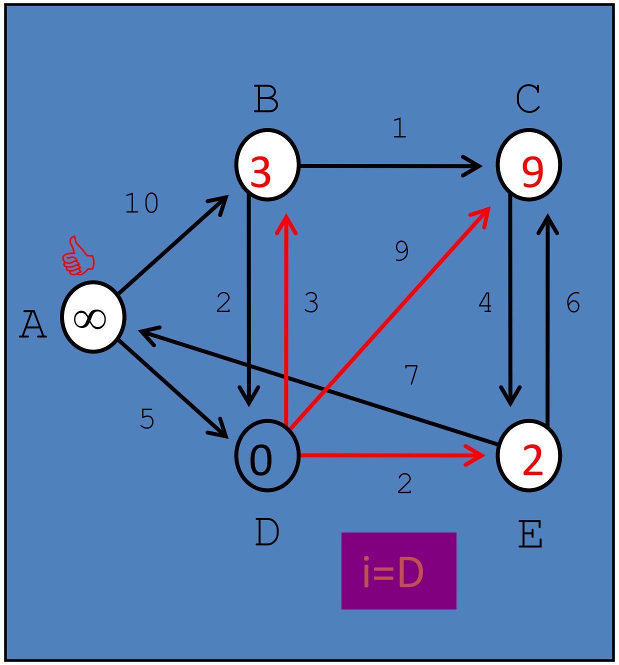
Solution: Floyd's Algorithm (k= A)

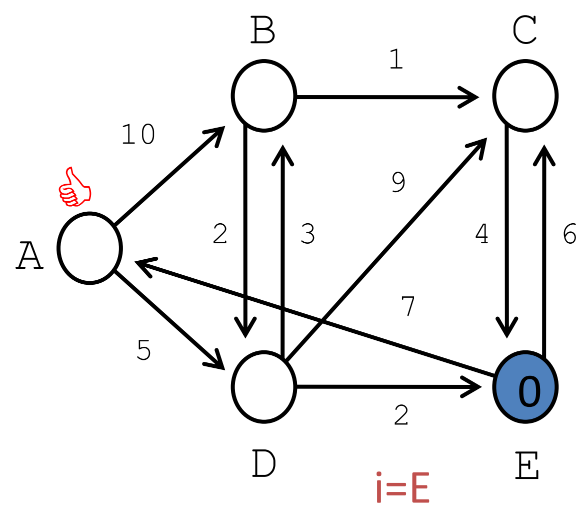
	A	B	C	D	E
A	0	10	∞	5	∞
B	∞	0	1	2	∞
C	∞	∞	0	∞	4
D	∞	3	9	0	2
E	7	∞	6	∞	0





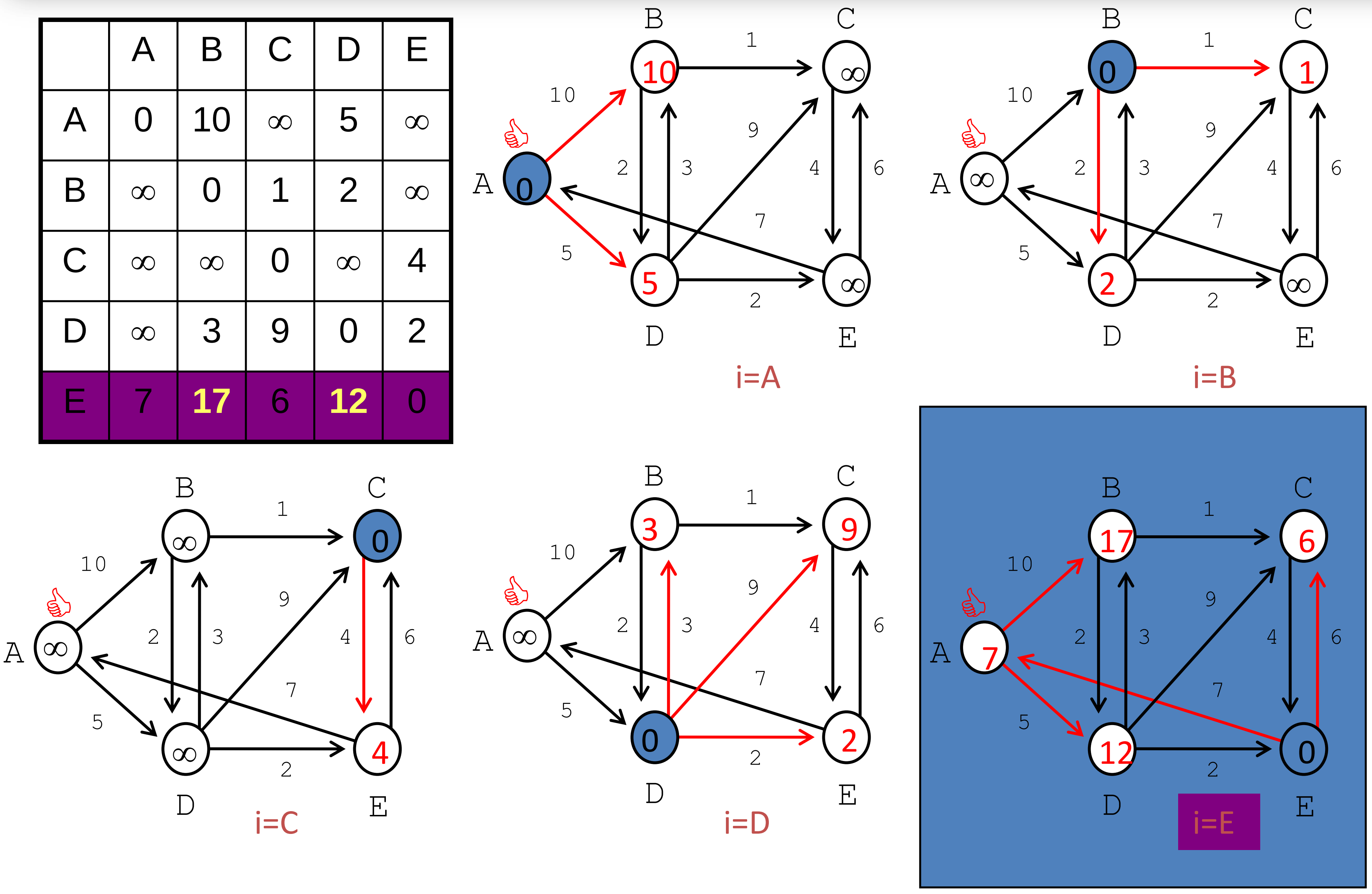






Solution: Floyd's Algorithm (k= A)

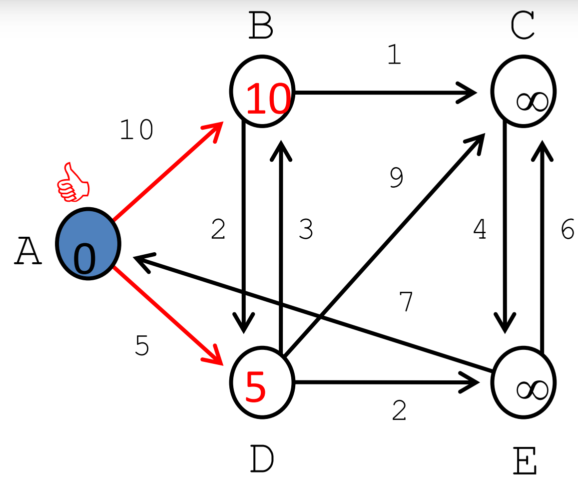
	A	B	C	D	E
A	0	10	∞	5	∞
B	∞	0	1	2	∞
C	∞	∞	0	∞	4
D	∞	3	9	0	2
E	7	17	6	12	0



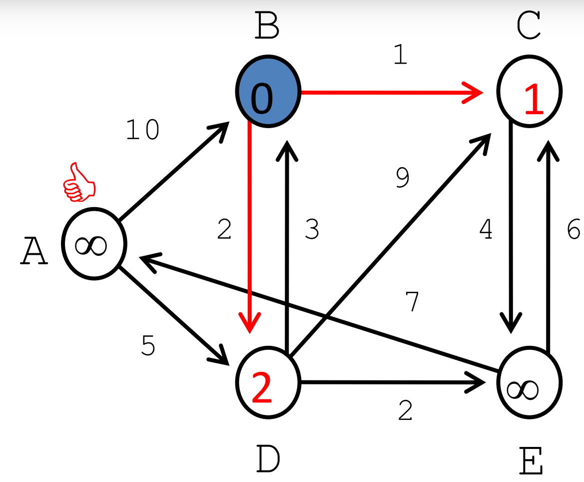
Solution: Floyd's Algorithm

Resulting Table (k= A)

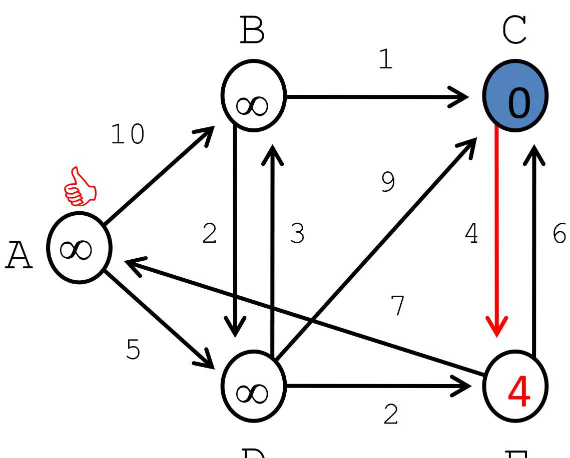
	A	B	C	D	E
A	0	10	∞	5	∞
B	∞	0	1	2	∞
C	∞	∞	0	∞	4
D	∞	3	9	0	2
E	7	17	6	12	0



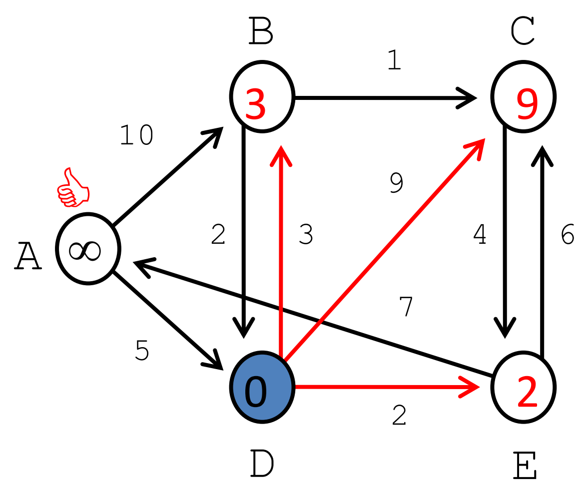
i=A



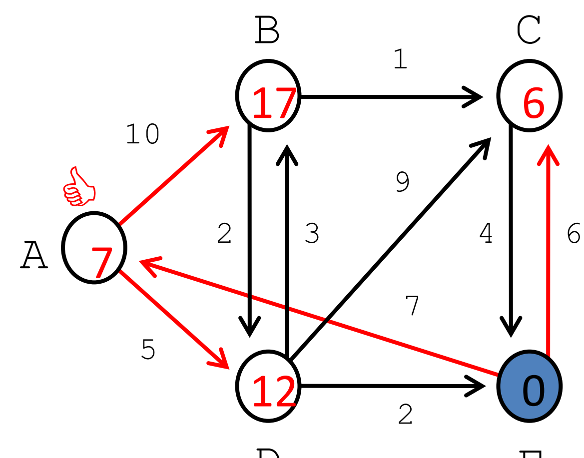
i=B



i=C



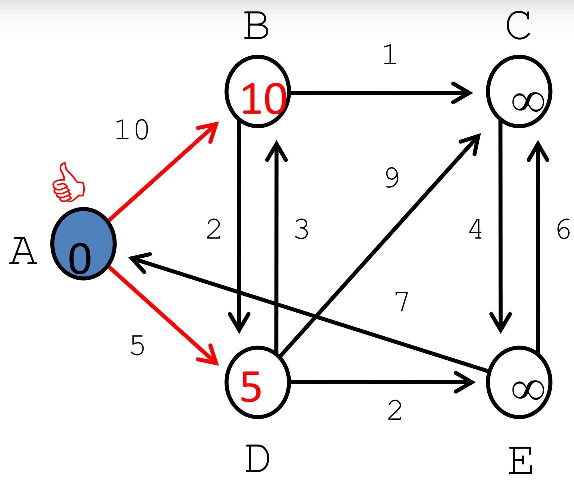
i=D



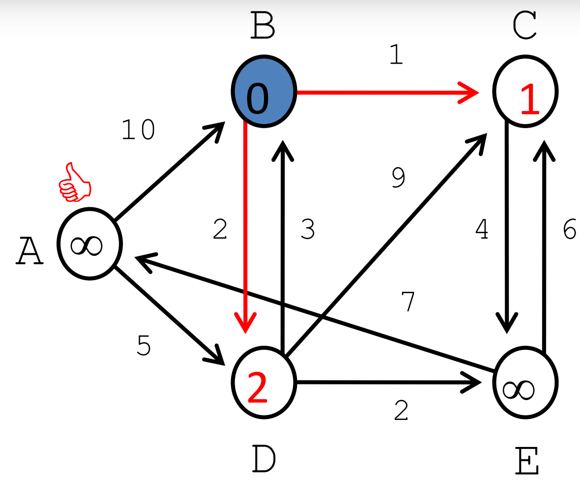
i=E

Solution: Floyd's Algorithm (k = B)

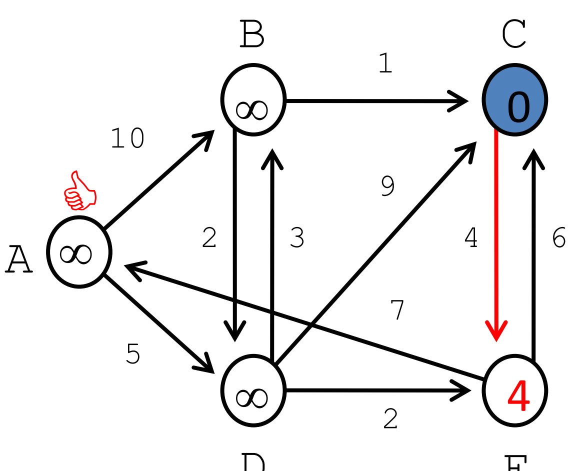
	A	B	C	D	E
A	0	10	∞	5	∞
B	∞	0	1	2	∞
C	∞	∞	0	∞	4
D	∞	3	9	0	2
E	7	17	6	12	0



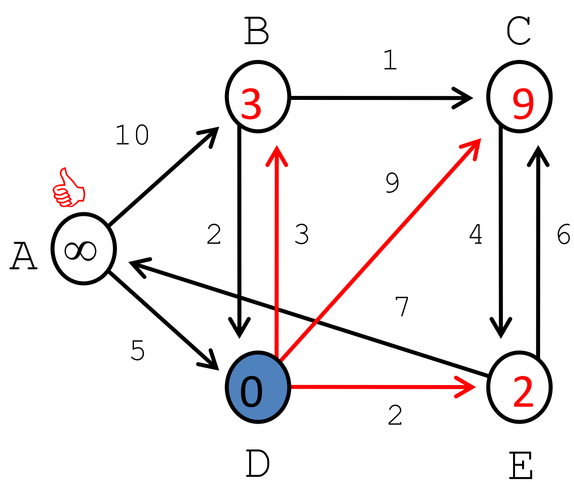
i=A



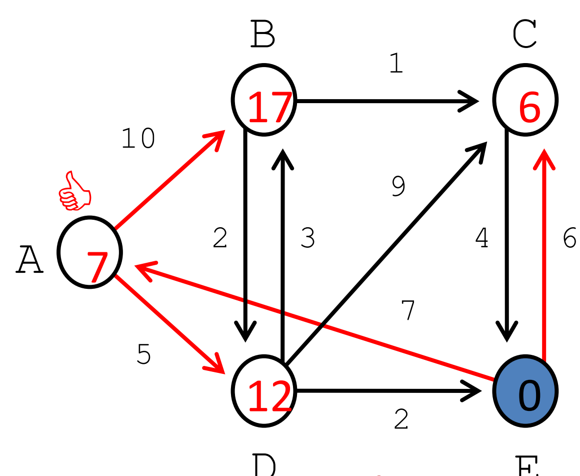
i=B



i=C



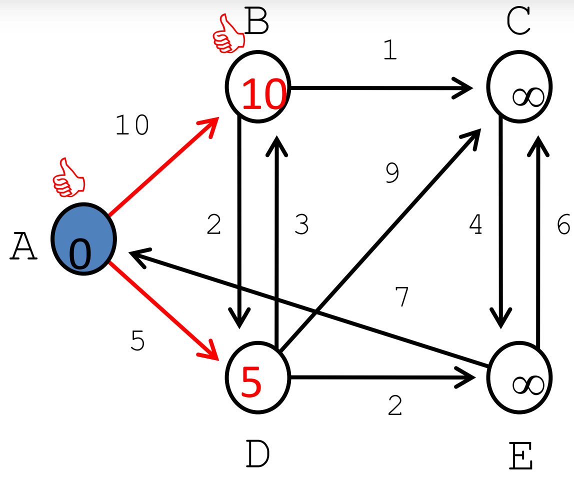
i=D



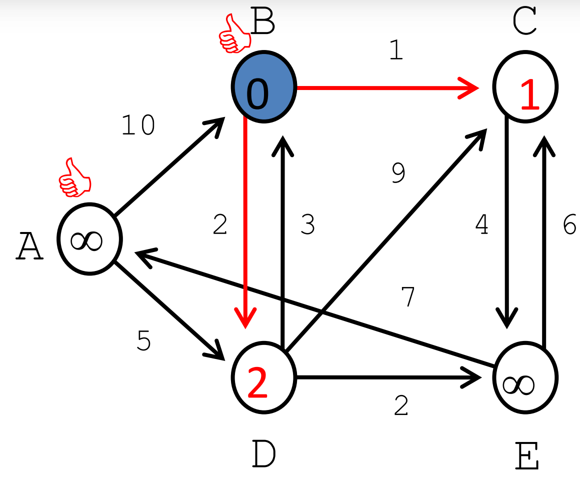
i=E

Solution: Floyd's Algorithm (k = B)

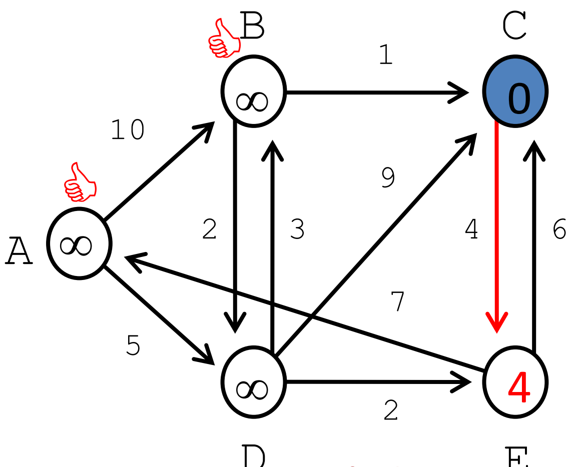
	A	B	C	D	E
A	0	10	∞	5	∞
B	∞	0	1	2	∞
C	∞	∞	0	∞	4
D	∞	3	9	0	2
E	7	17	6	∞	0



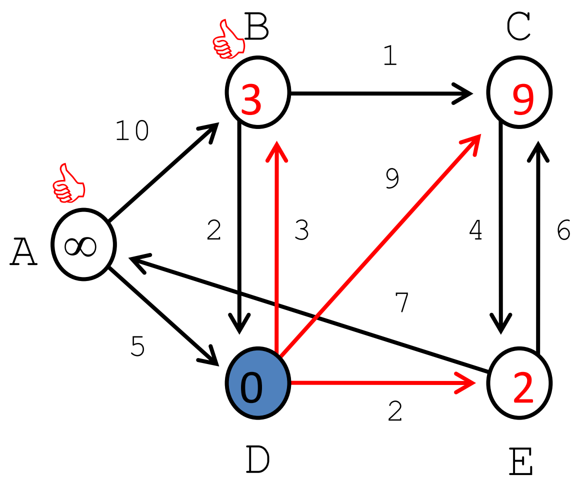
i=A



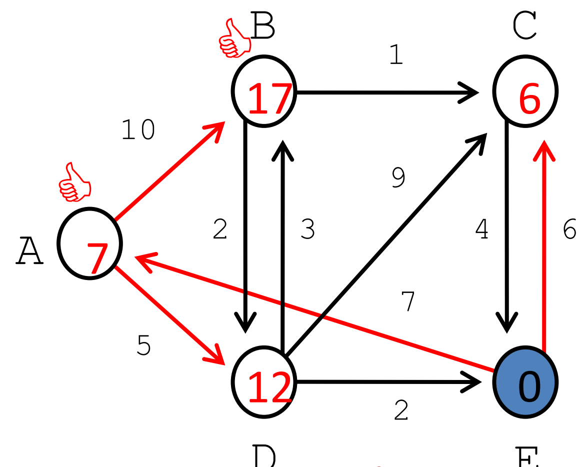
i=B



i=C



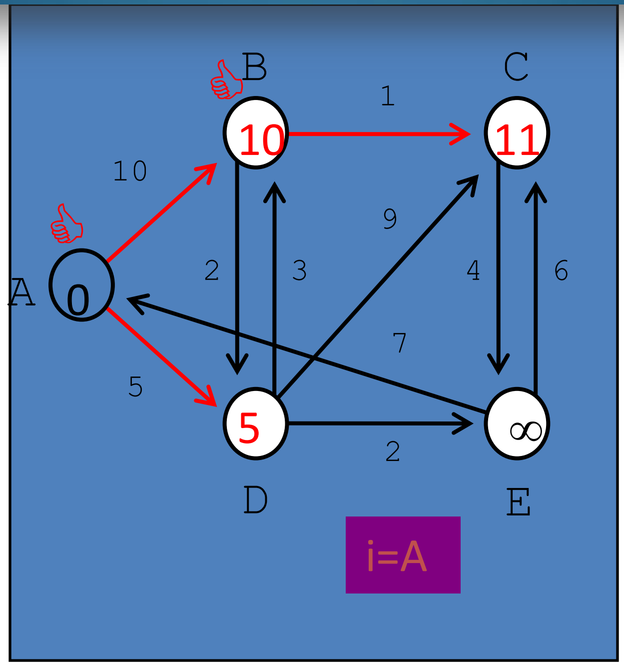
i=D

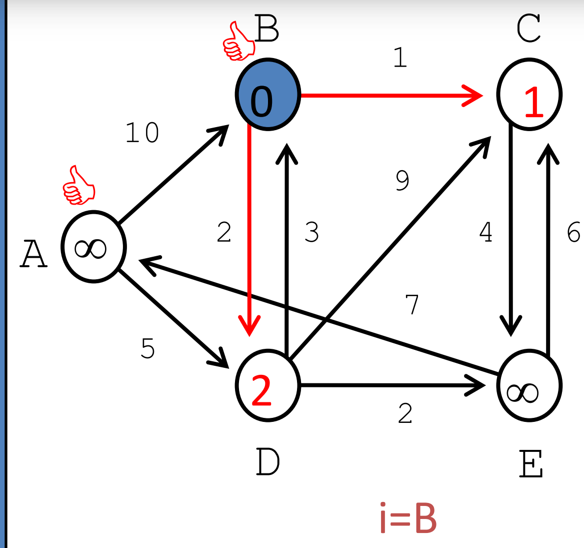


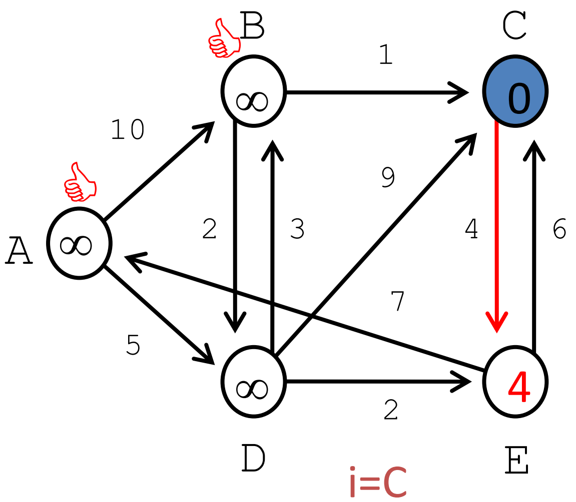
i=E

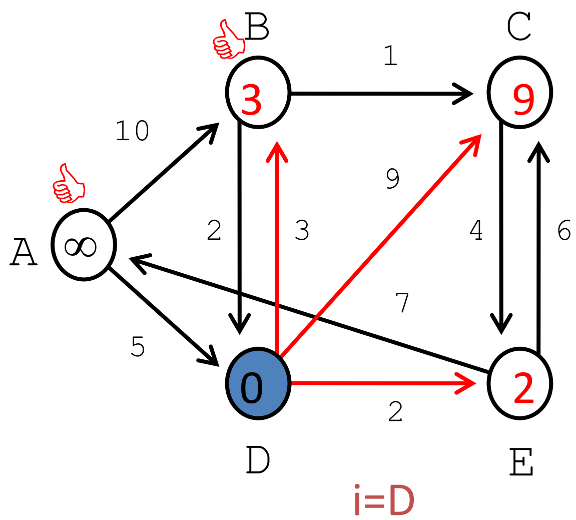
Solution: Floyd's Algorithm (k = B)

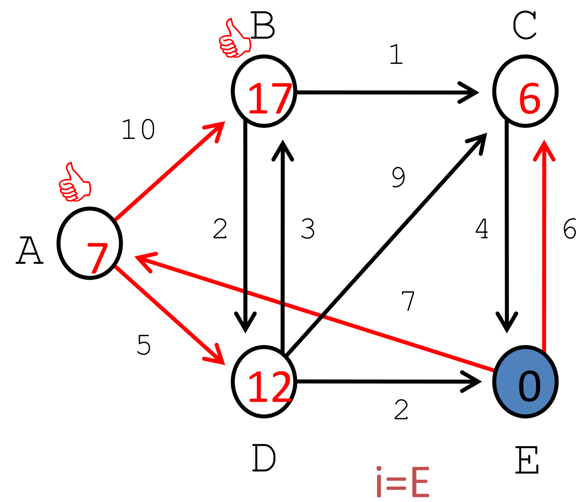
	A	B	C	D	E
A	0	10	11	5	∞
B	∞	0	1	2	∞
C	∞	∞	0	∞	4
D	∞	3	9	0	2
E	7	17	6	12	0





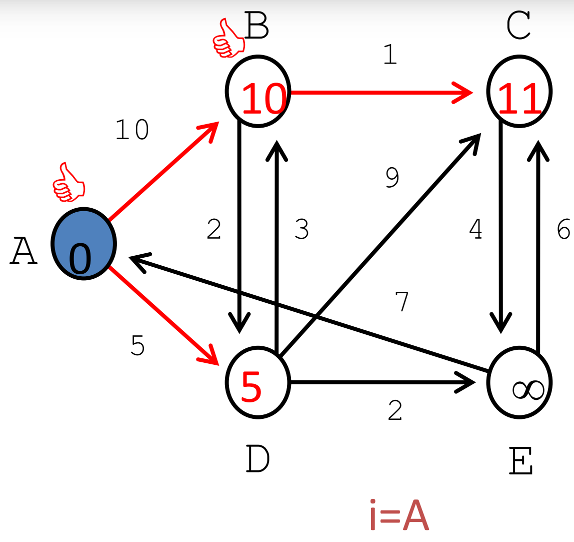


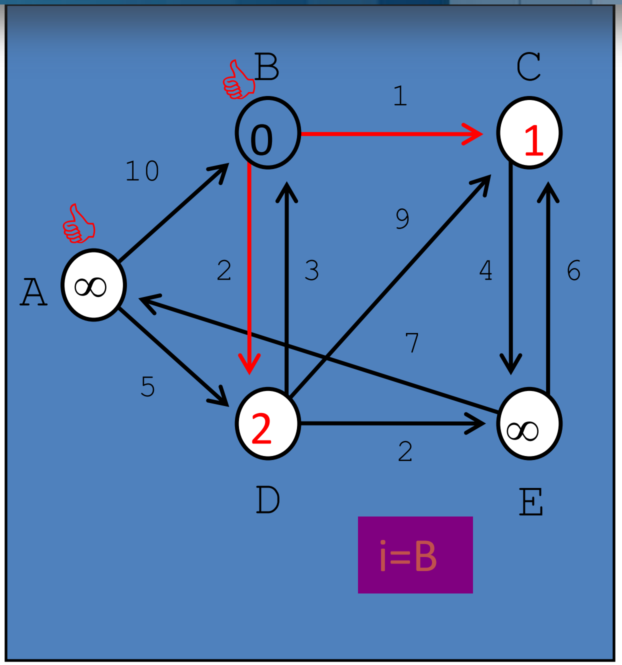


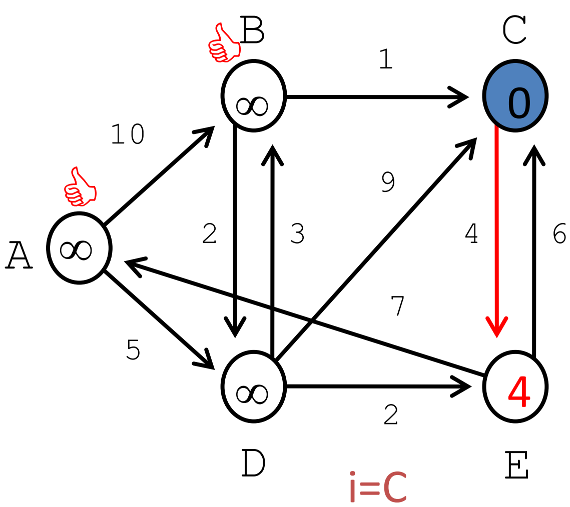


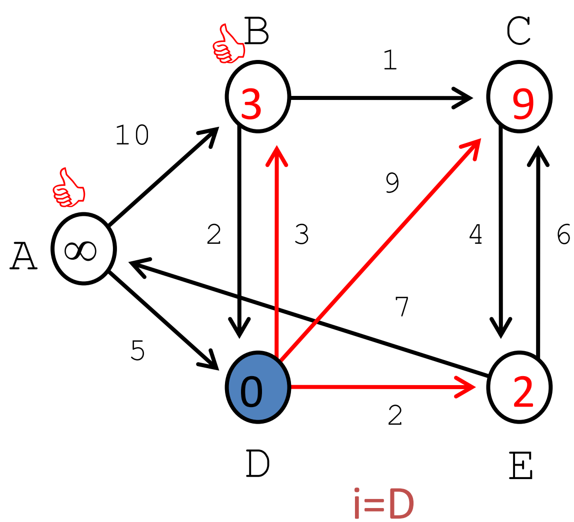
Solution: Floyd's Algorithm (k = B)

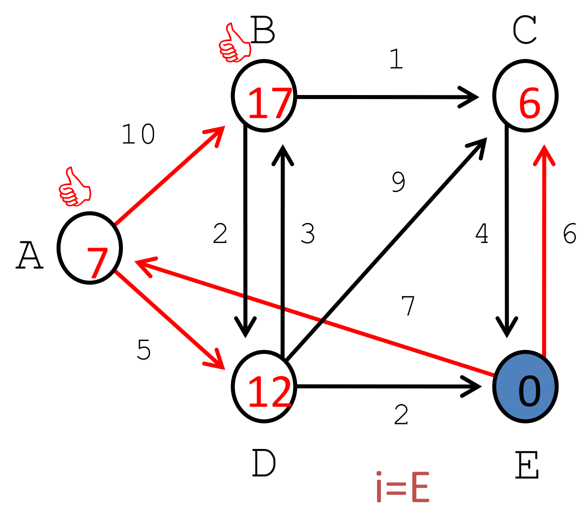
	A	B	C	D	E
A	0	10	11	5	∞
B	∞	0	1	2	∞
C	∞	∞	0	∞	4
D	∞	3	9	0	2
E	7	17	6	12	0





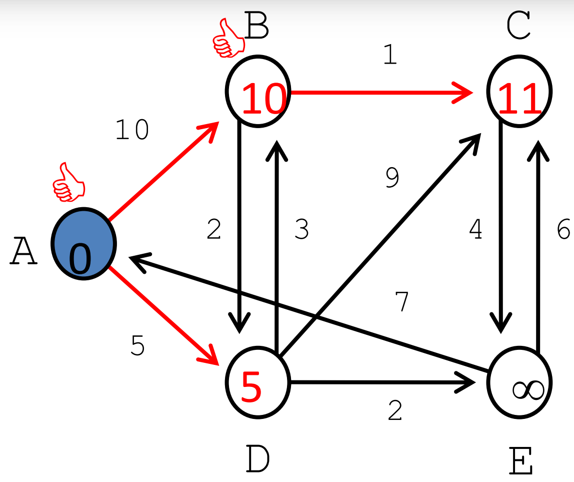




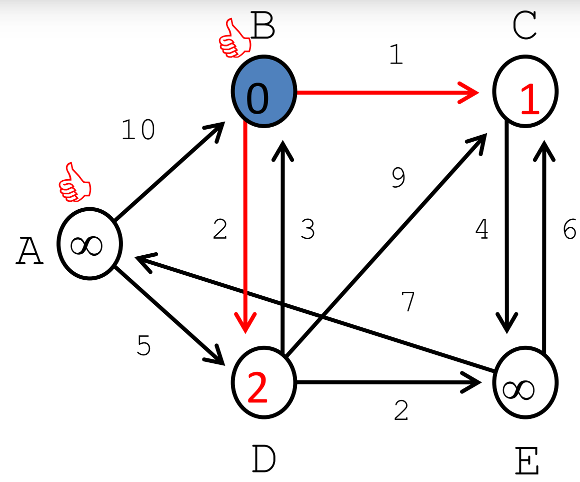


Solution: Floyd's Algorithm (k = B)

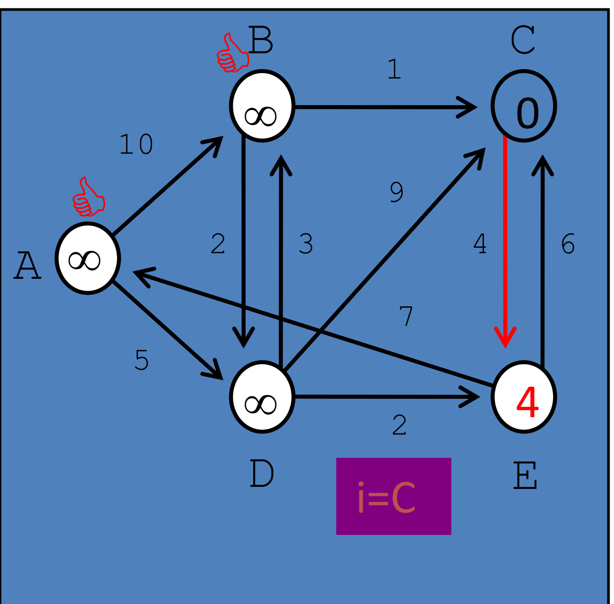
	A	B	C	D	E
A	0	10	11	5	∞
B	∞	0	1	2	∞
C	∞	∞	0	∞	4
D	∞	3	9	0	2
E	7	17	6	12	0

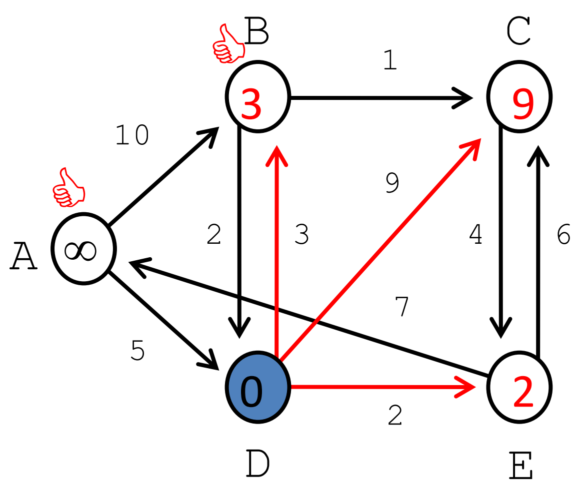


i=A

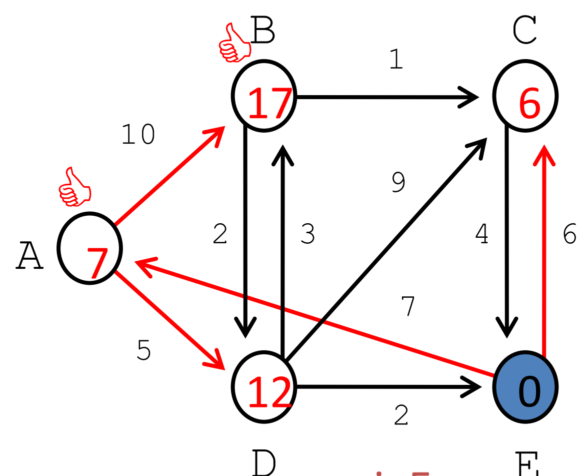


i=B





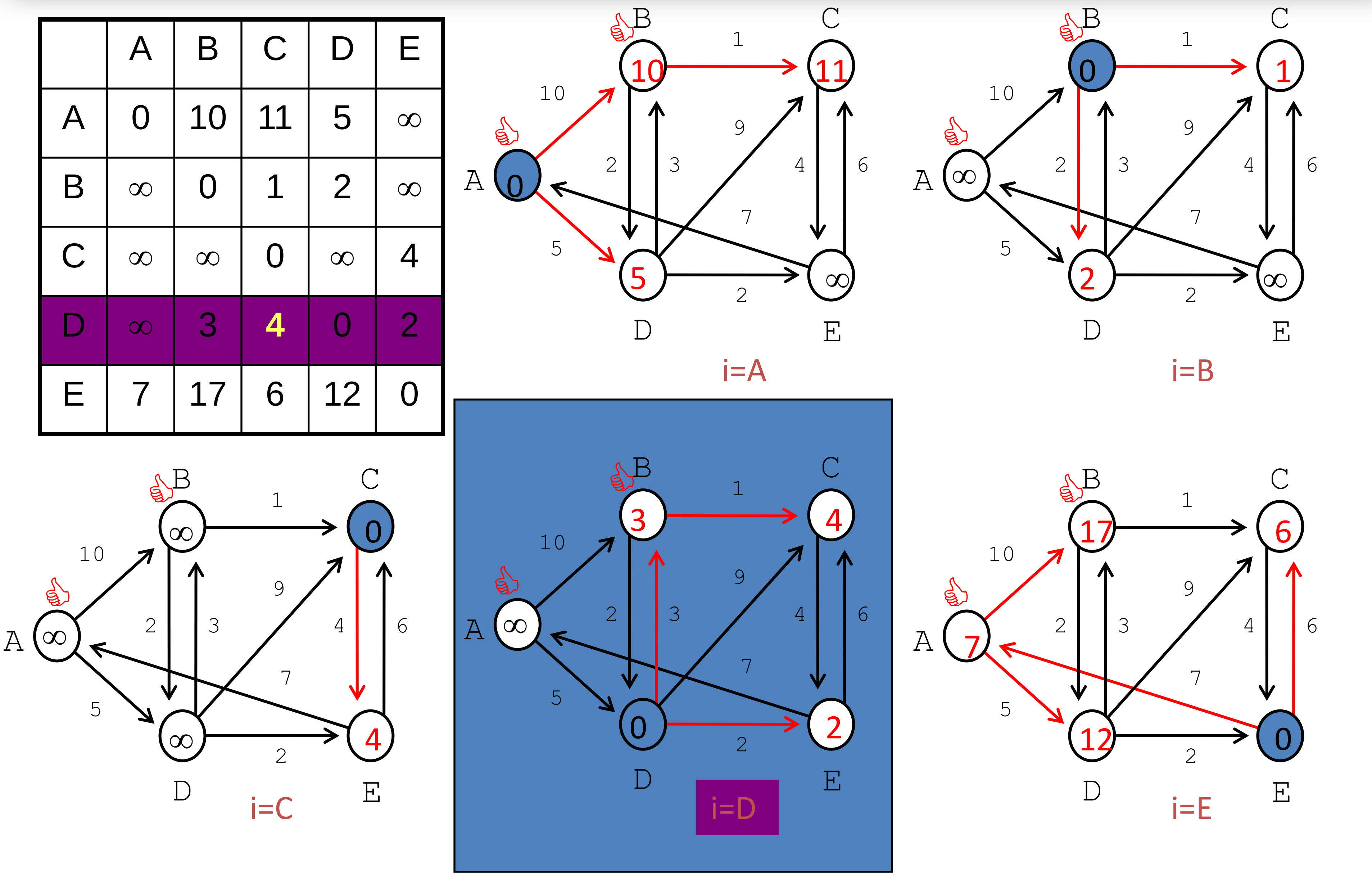
i=D



i=E

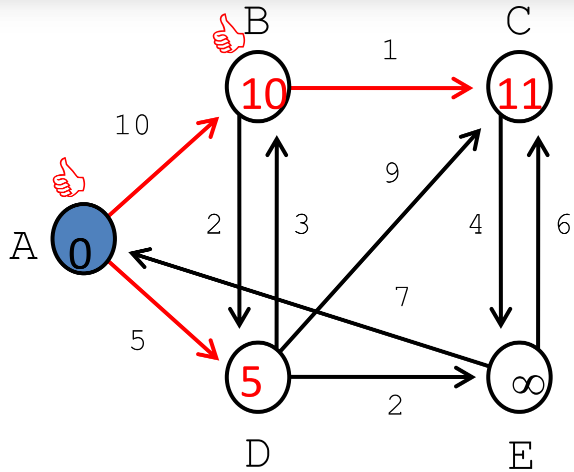
Solution: Floyd's Algorithm (k = B)

	A	B	C	D	E
A	0	10	11	5	∞
B	∞	0	1	2	∞
C	∞	∞	0	∞	4
D	∞	3	4	0	2
E	7	17	6	12	0

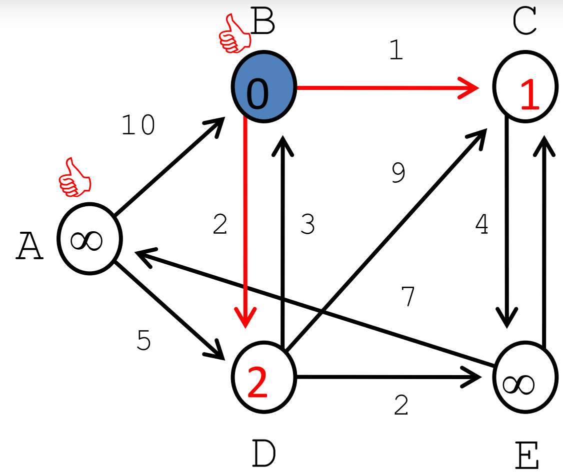


Solution: Floyd's Algorithm (k = B)

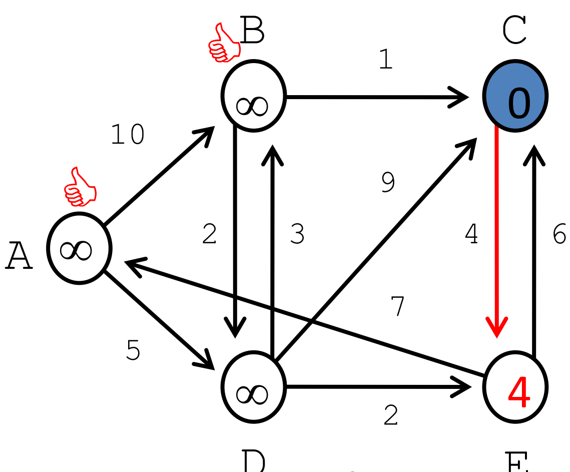
	A	B	C	D	E
A	0	10	11	5	∞
B	∞	0	1	2	∞
C	∞	∞	0	∞	4
D	∞	3	4	0	2
E	7	17	6	12	0



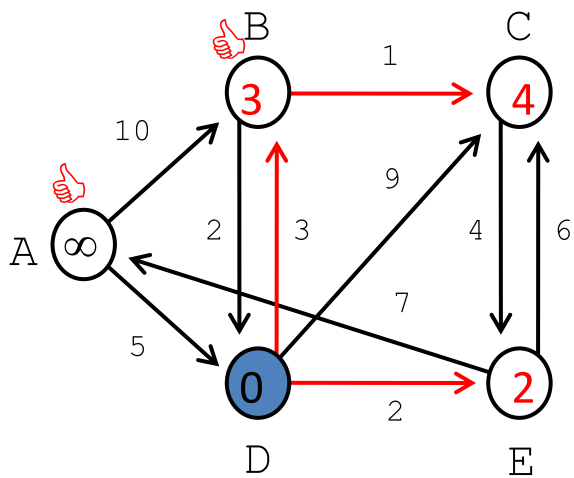
i=A



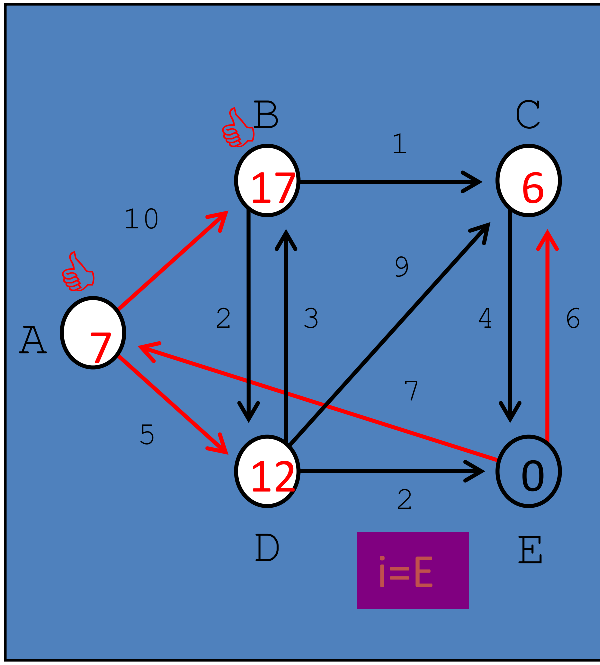
i=B



i=C



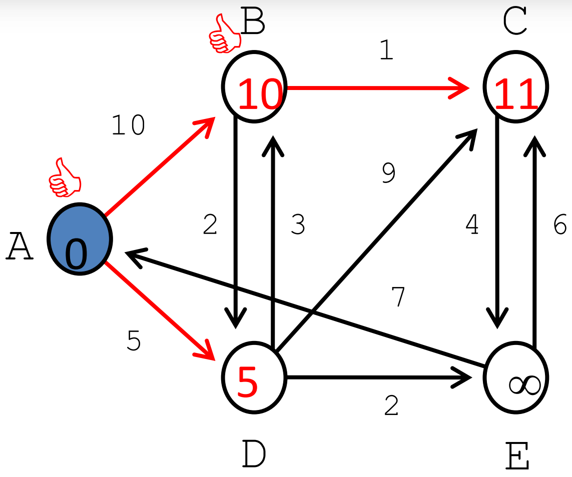
i=D



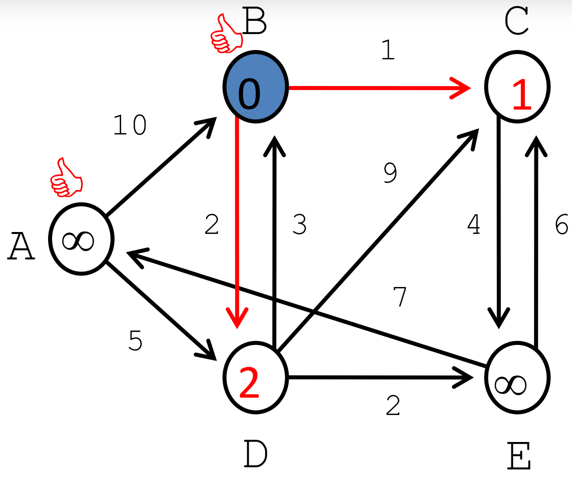
i=E

Solution: Floyd's Algorithm   Resulting table (k = B)

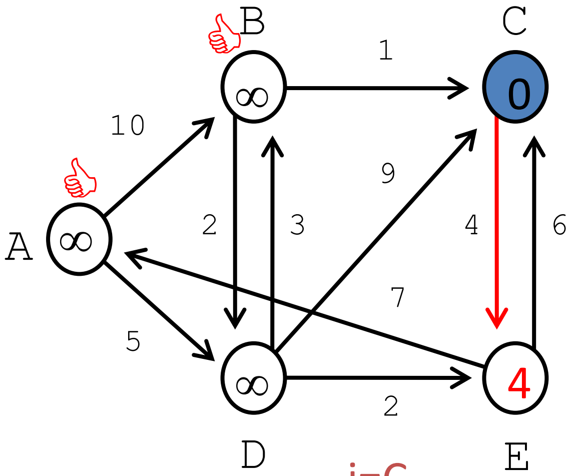
	A	B	C	D	E
A	0	10	11	5	∞
B	∞	0	1	2	∞
C	∞	∞	0	∞	4
D	∞	3	4	0	2
E	7	17	6	12	0



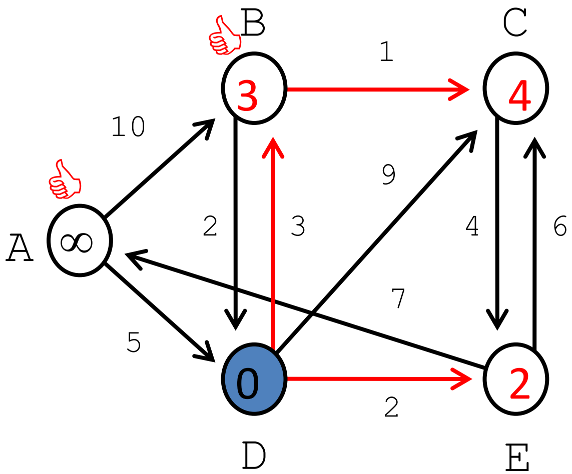
i=A



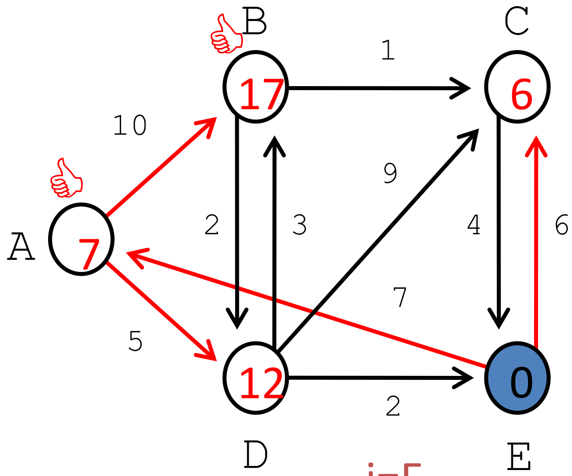
i=B



i=C



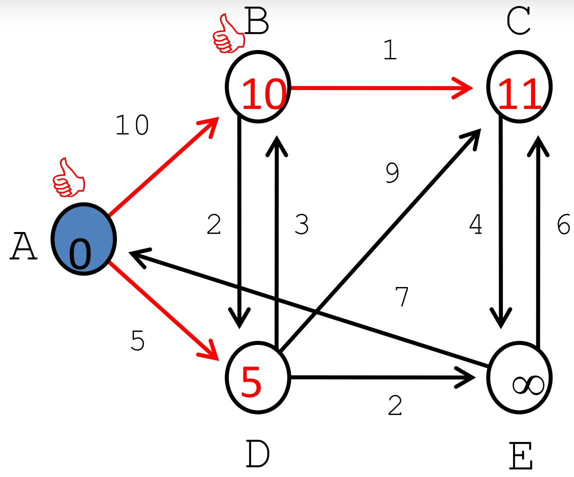
i=D



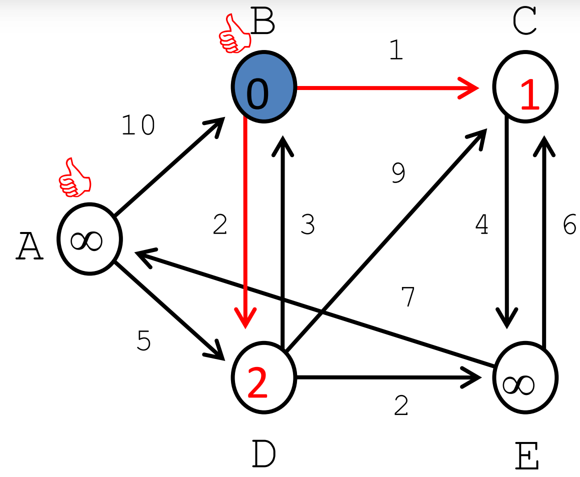
i=E

Solution: Floyd's Algorithm (k = C)

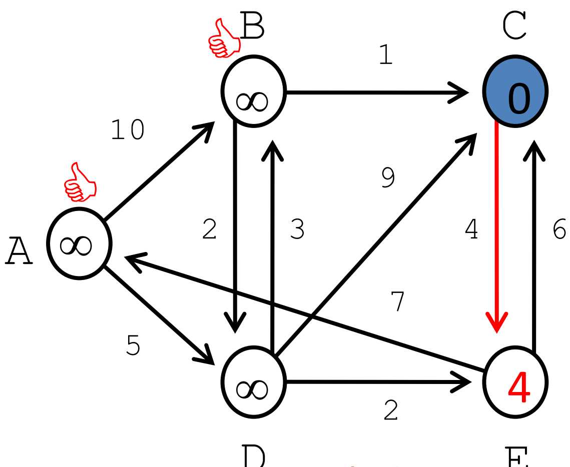
	A	B	C	D	E
A	0	10	11	5	∞
B	∞	0	1	2	∞
C	∞	∞	0	∞	4
D	∞	3	4	0	2
E	7	17	6	12	0



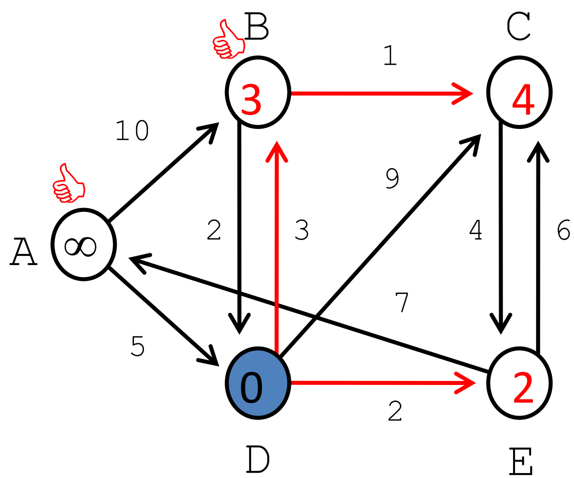
i=A



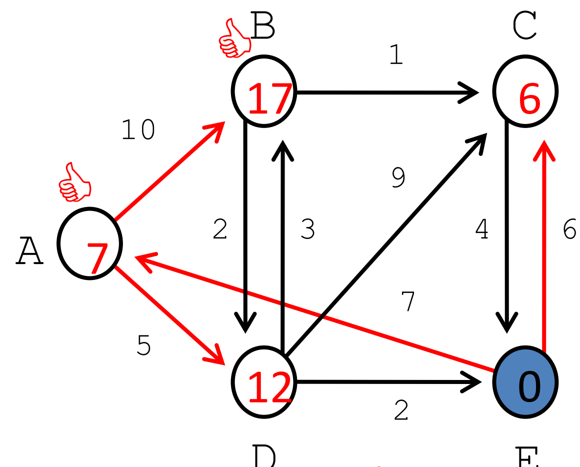
i=B



i=C



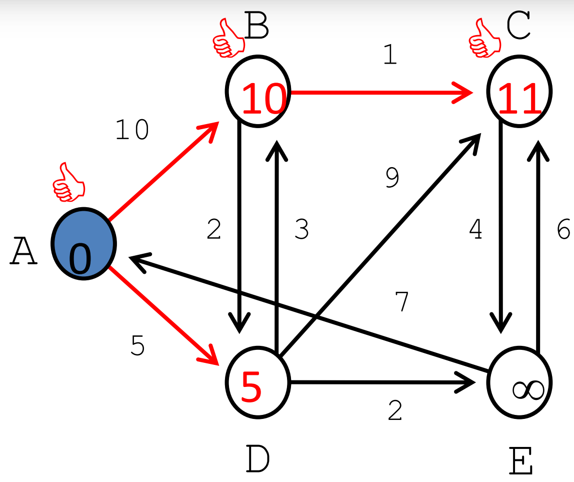
i=D



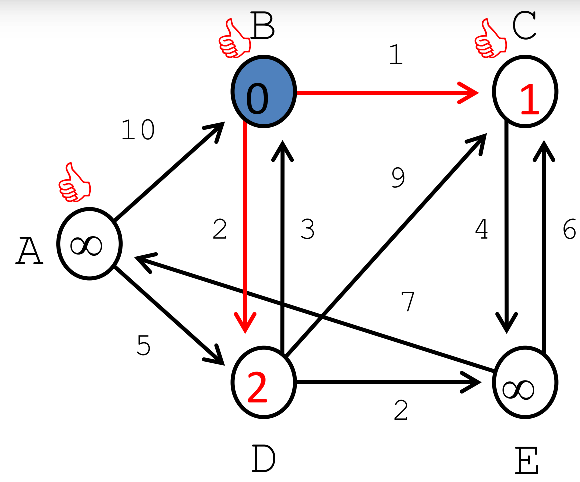
i=E

Solution: Floyd's Algorithm (k = C)

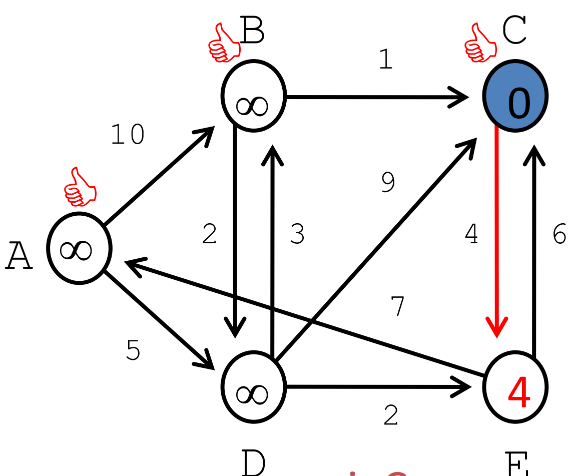
	A	B	C	D	E
A	0	10	11	5	∞
B	∞	0	1	2	∞
C	∞	∞	0	∞	4
D	∞	3	4	0	2
E	7	17	6	12	0



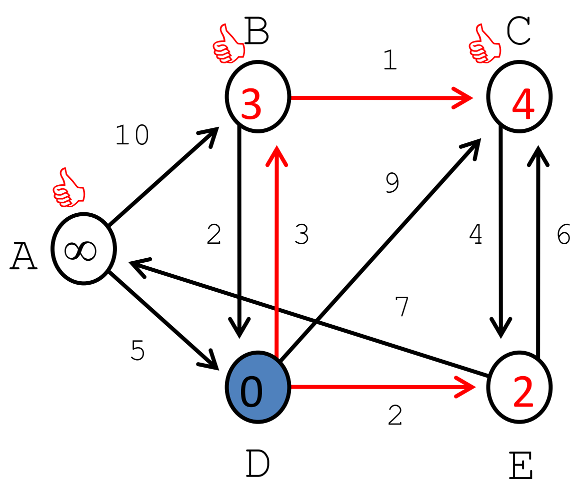
i=A



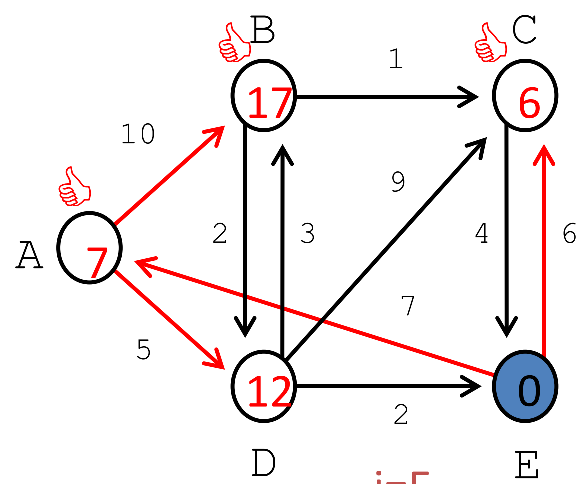
i=B



i=C



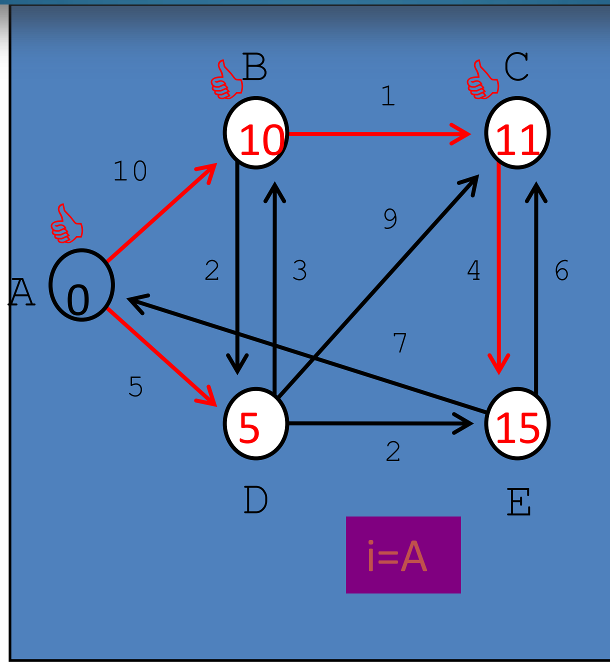
i=D

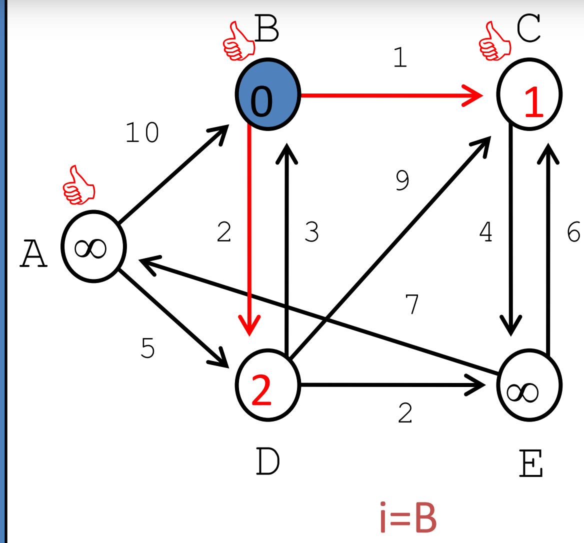


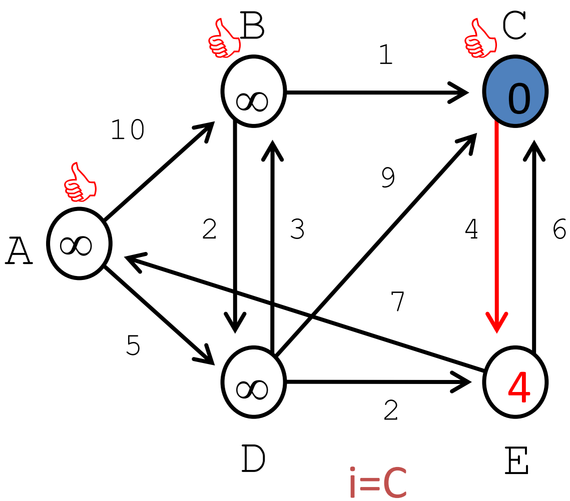
i=E

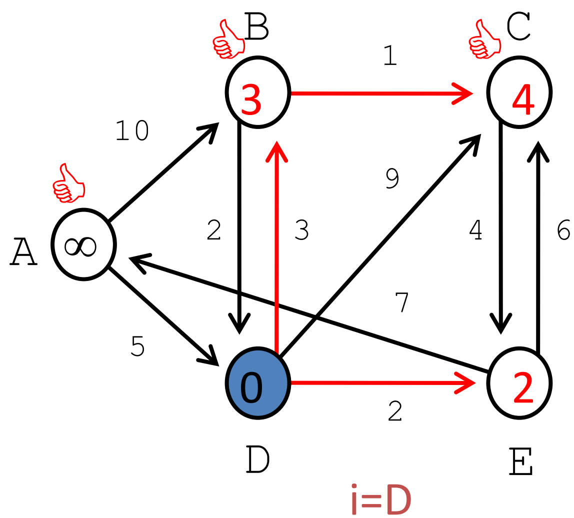
Solution: Floyd's Algorithm (k = C)

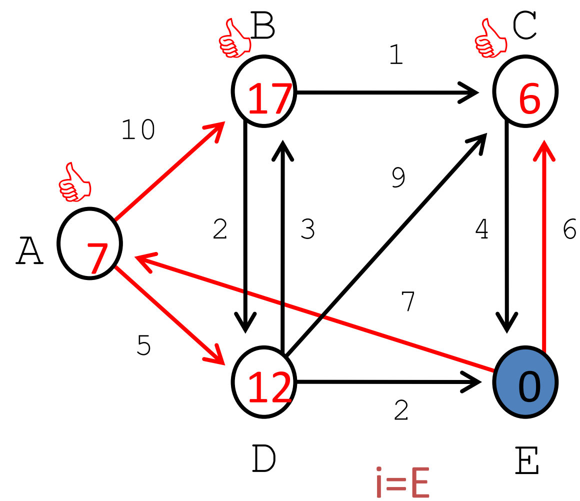
	A	B	C	D	E
A	0	10	11	5	**15**
B	∞	0	1	2	∞
C	∞	∞	0	∞	4
D	∞	3	4	0	2
E	7	17	6	12	0





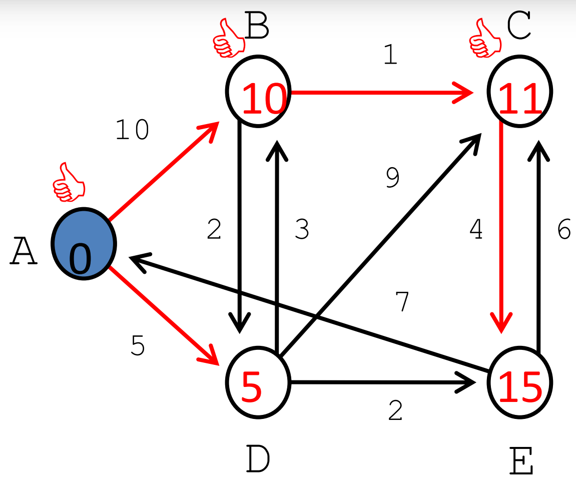




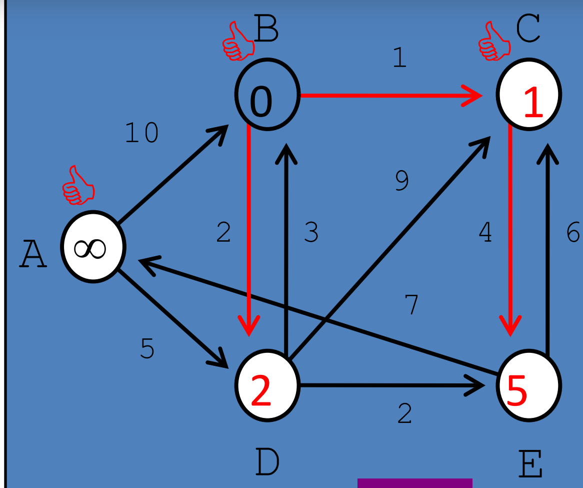


Solution: Floyd's Algorithm (k = C)

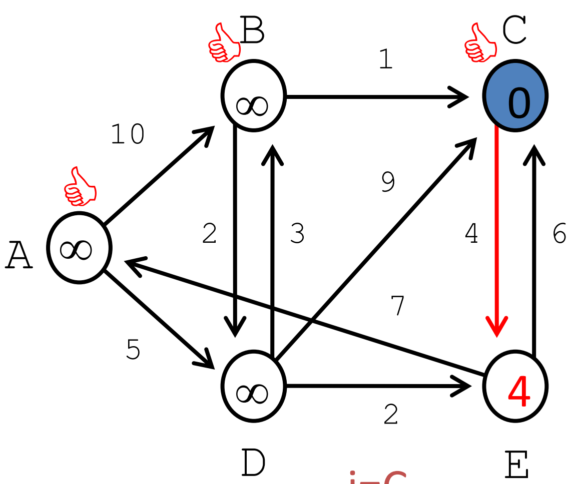
	A	B	C	D	E
A	0	10	11	5	15
B	∞	0	1	2	5
C	∞	∞	0	∞	4
D	∞	3	4	0	2
E	7	17	6	12	0



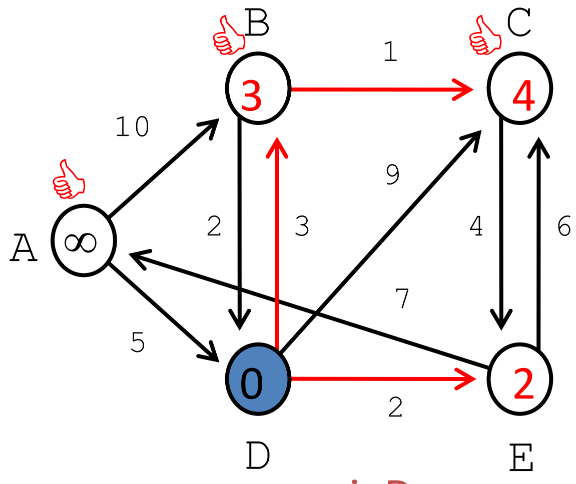
i=A



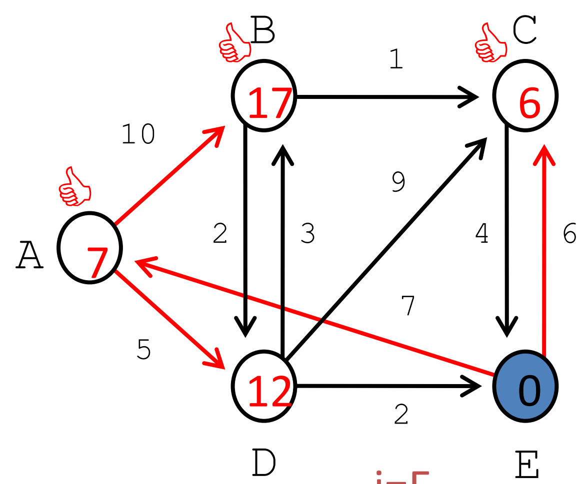
i=B



i=C



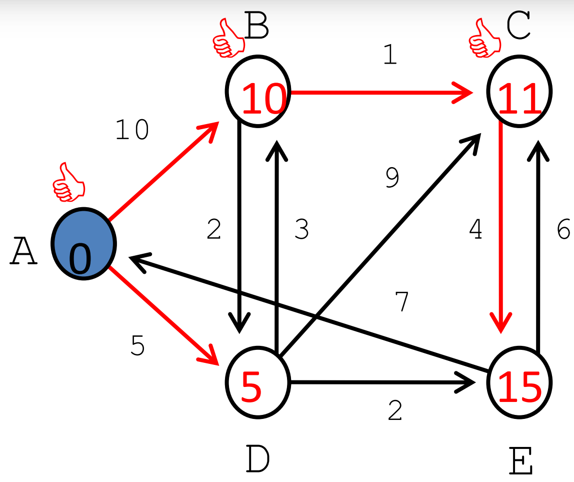
i=D



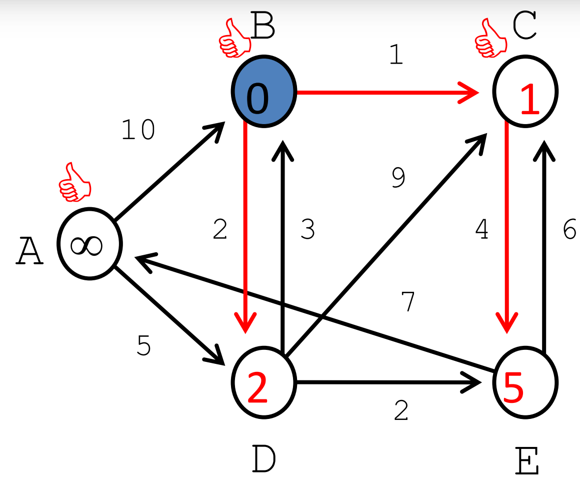
i=E

Solution: Floyd's Algorithm (k = C)

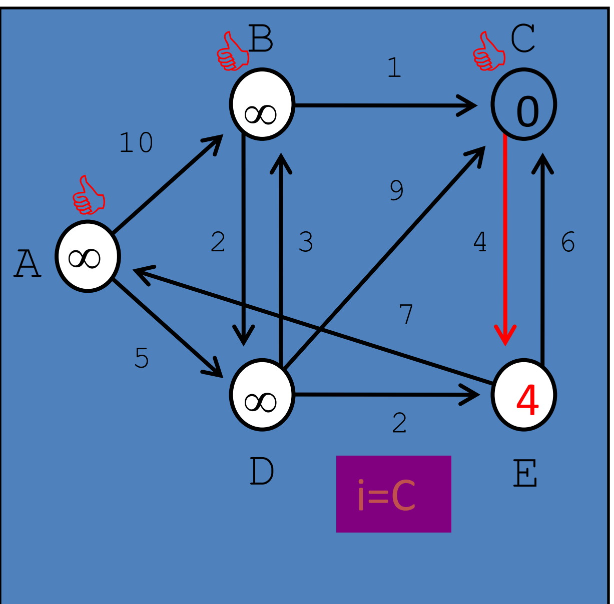
	A	B	C	D	E
A	0	10	11	5	15
B	∞	0	1	2	5
C	∞	∞	0	∞	4
D	∞	3	4	0	2
E	7	17	6	12	0

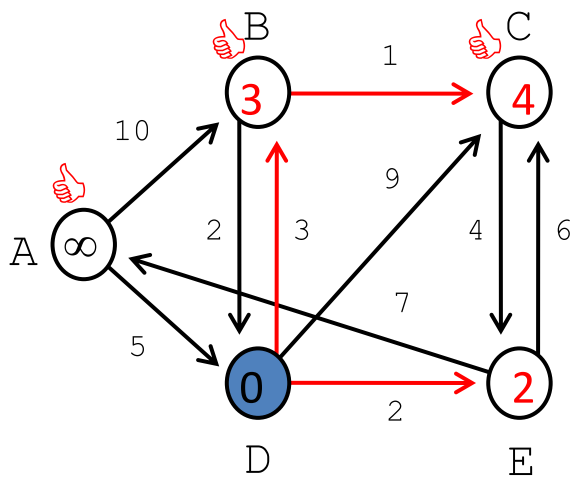


i=A

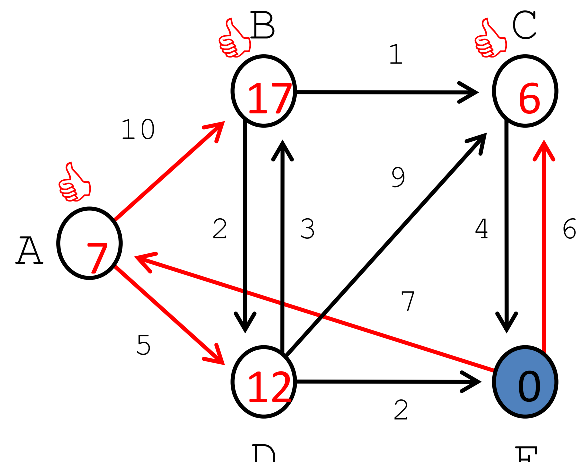


i=B





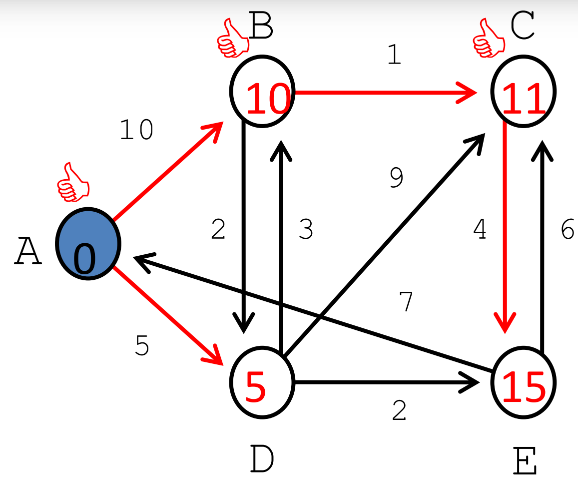
i=D



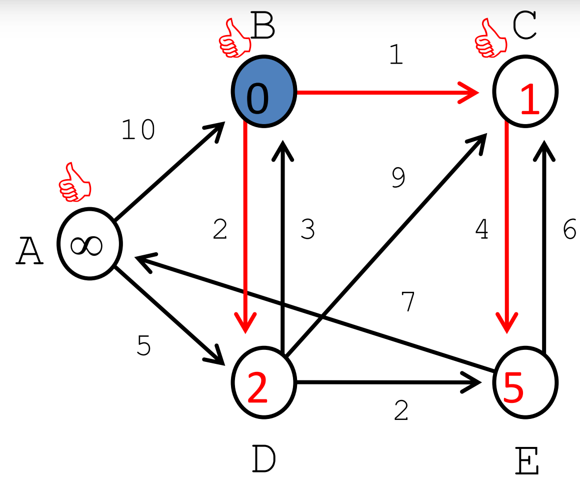
i=E

Solution: Floyd's Algorithm (k = C)

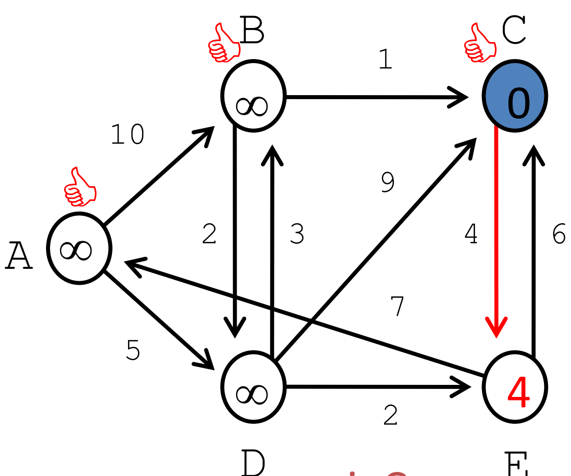
	A	B	C	D	E
A	0	10	11	5	15
B	∞	0	1	2	5
C	∞	∞	0	∞	4
D	∞	3	4	0	2
E	7	17	6	12	0



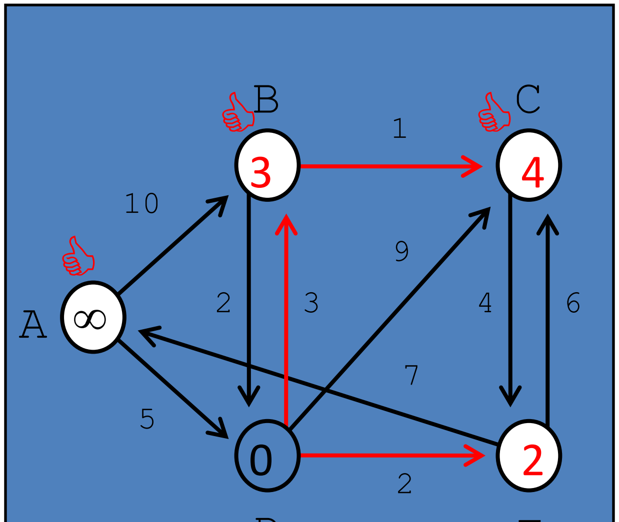
i=A



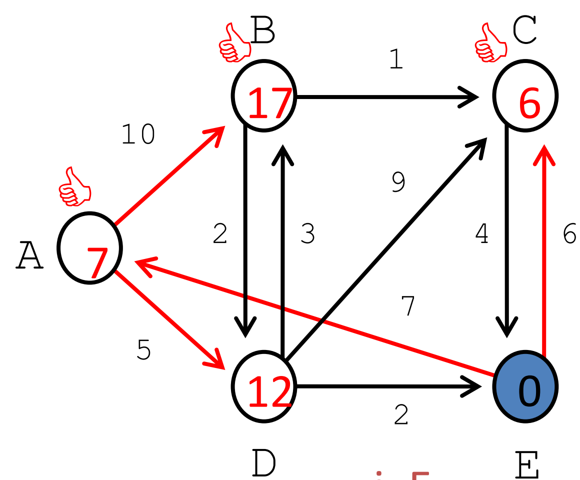
i=B



i=C



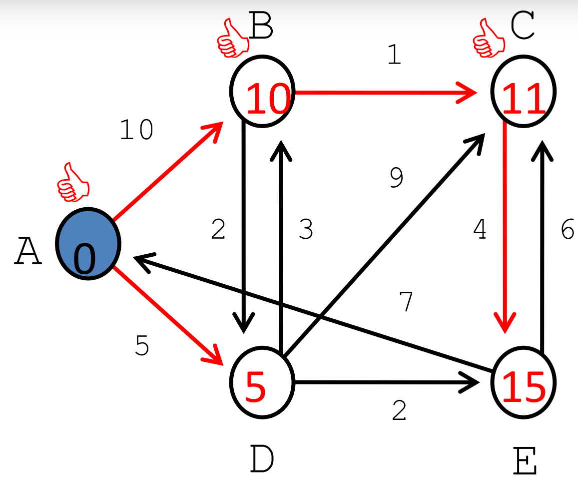
i=D



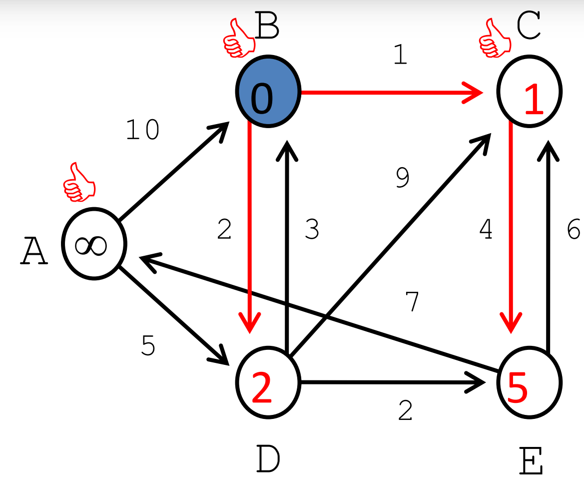
i=E

Solution: Floyd's Algorithm (k = C)

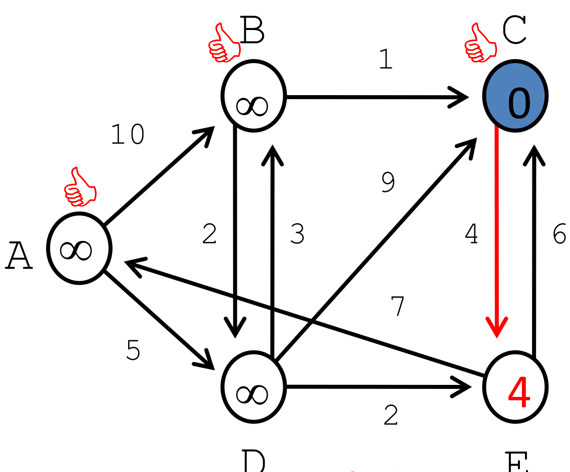
	A	B	C	D	E
A	0	10	11	5	15
B	∞	0	1	2	5
C	∞	∞	0	∞	4
D	∞	3	4	0	2
E	7	17	6	12	0



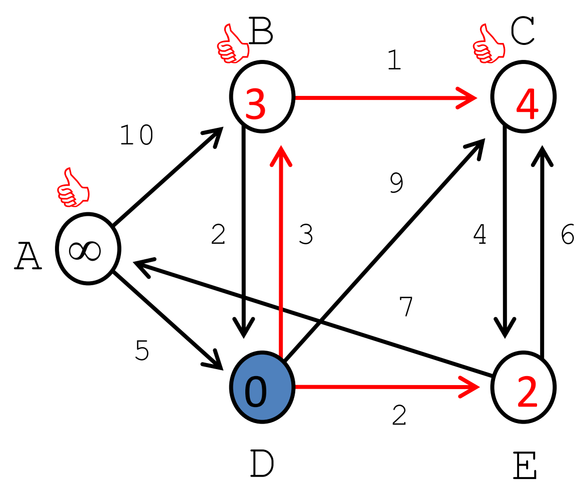
i=A



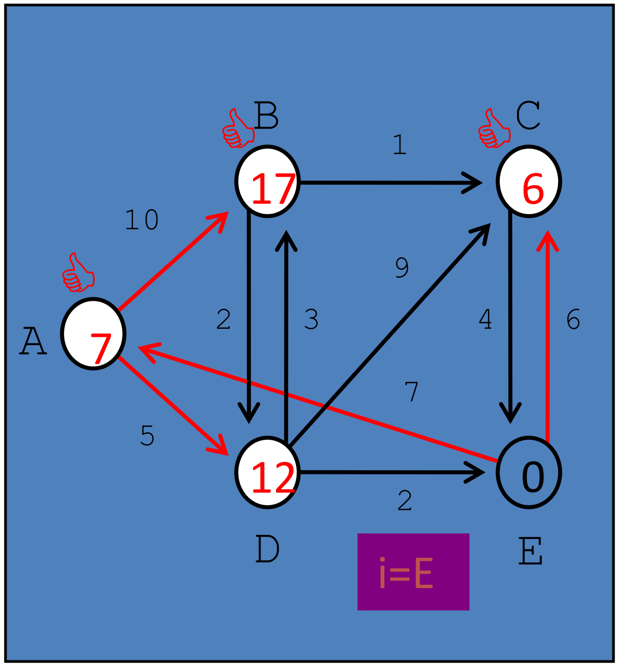
i=B



i=C

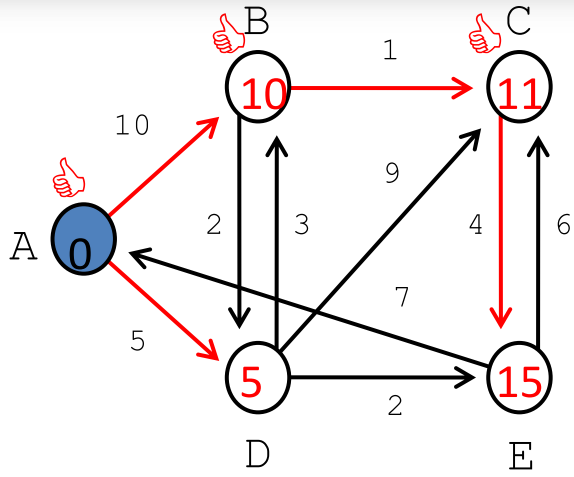


i=D

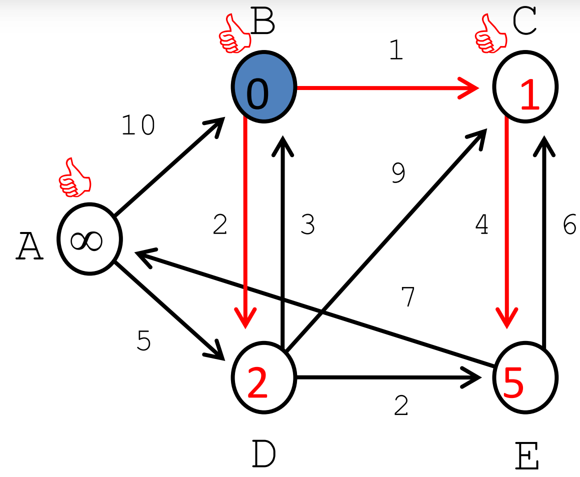


Solution: Floyd's Algorithm Resulting table (k = C)

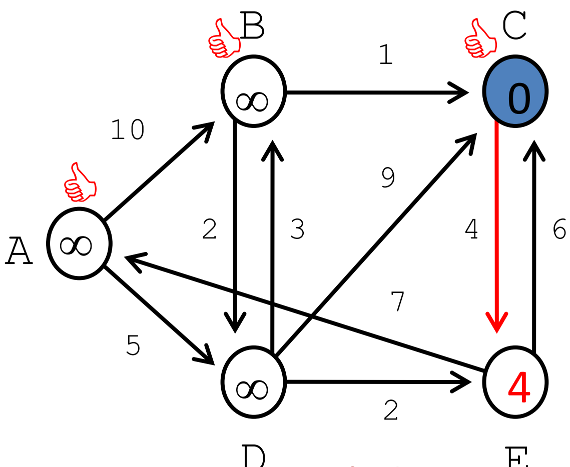
	A	B	C	D	E
A	0	10	11	5	15
B	∞	0	1	2	5
C	∞	∞	0	∞	4
D	∞	3	4	0	2
E	7	17	6	12	0



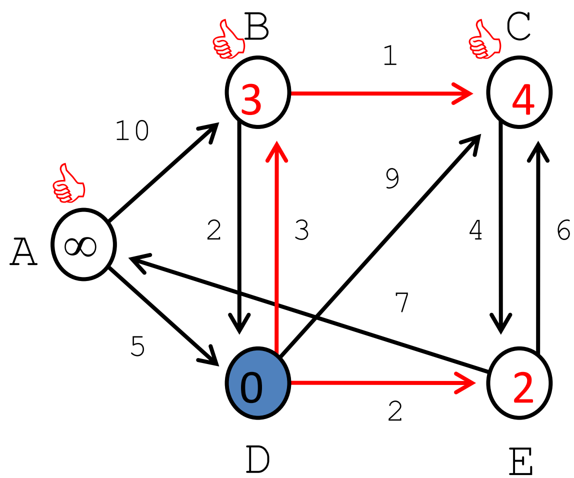
i=A



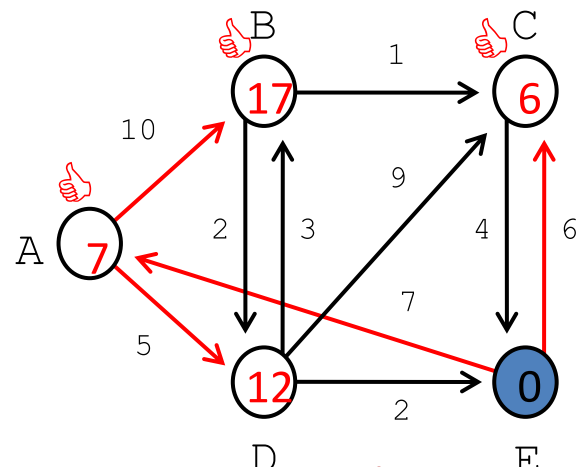
i=B



i=C



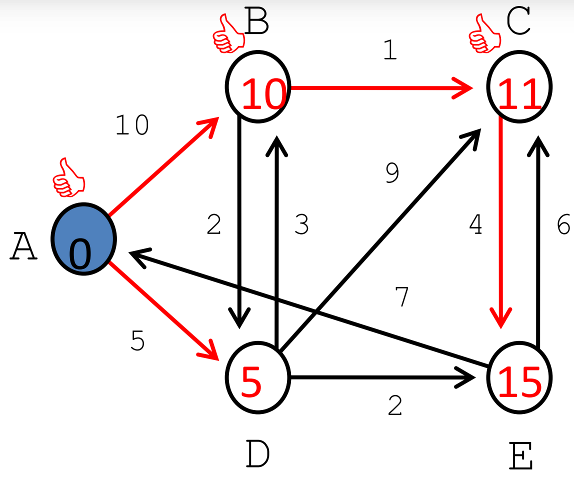
i=D



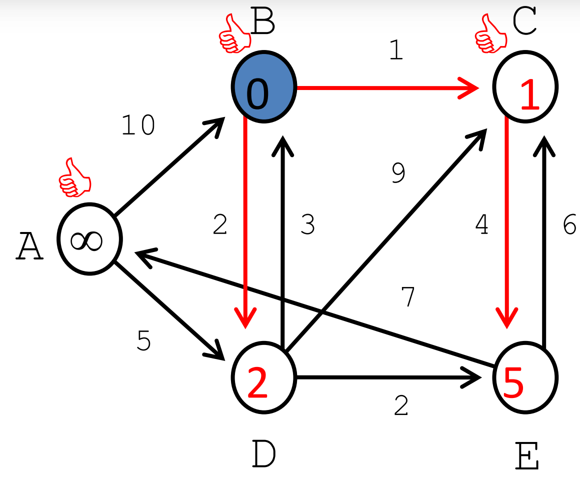
i=E

Solution: Floyd's Algorithm (k = D)

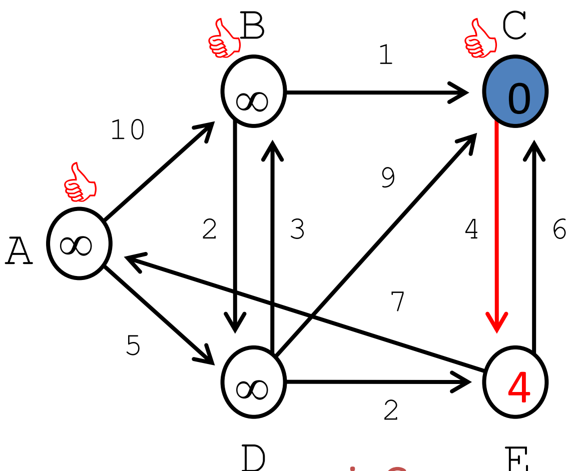
	A	B	C	D	E
A	0	10	11	5	15
B	∞	0	1	2	5
C	∞	∞	0	∞	4
D	∞	3	4	0	2
E	7	17	6	12	0



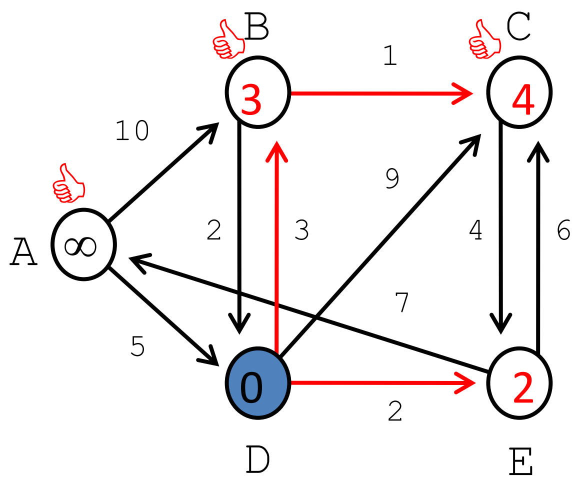
i=A



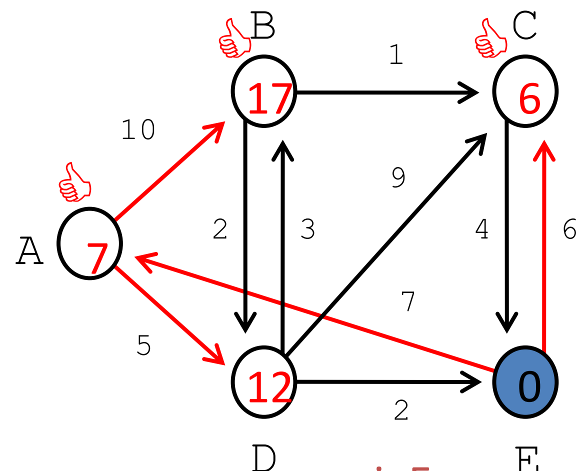
i=B



i=C



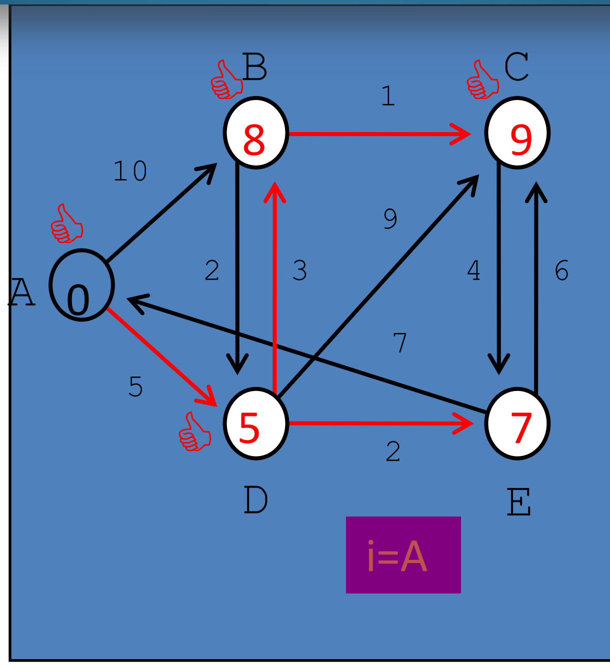
i=D

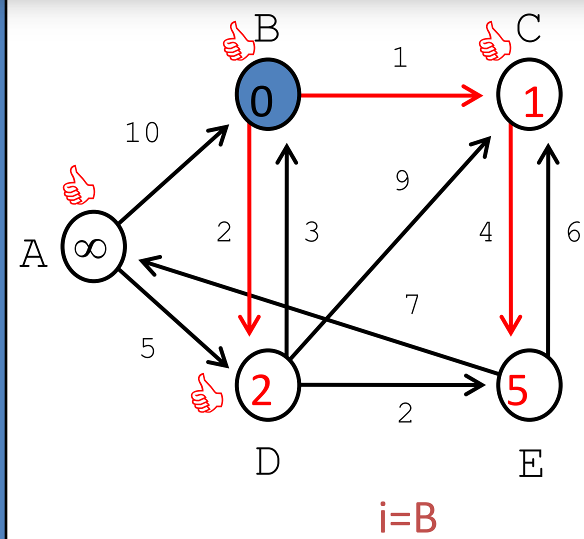


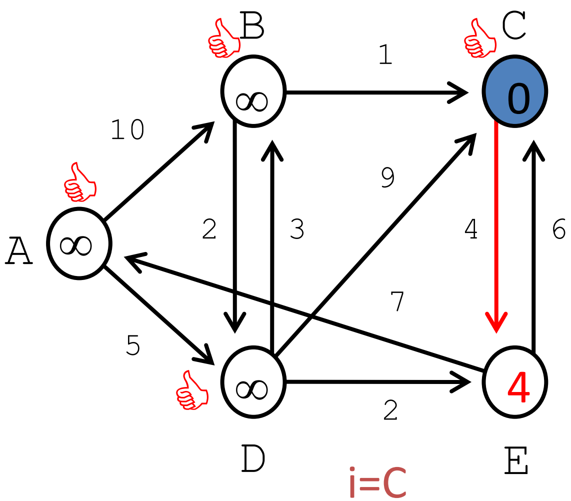
i=E

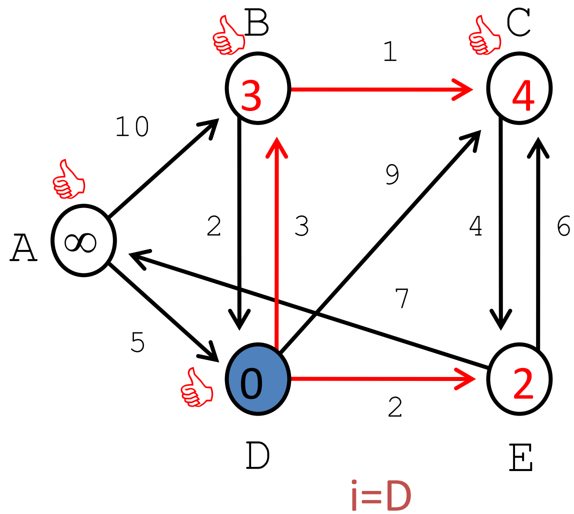
Solution: Floyd's Algorithm (k = D)

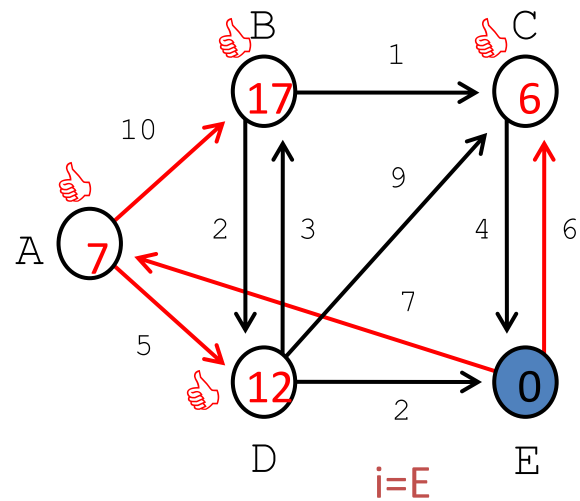
	A	B	C	D	E
A	0	8	9	5	7
B	∞	0	1	2	5
C	∞	∞	0	∞	4
D	∞	3	4	0	2
E	7	17	6	12	0





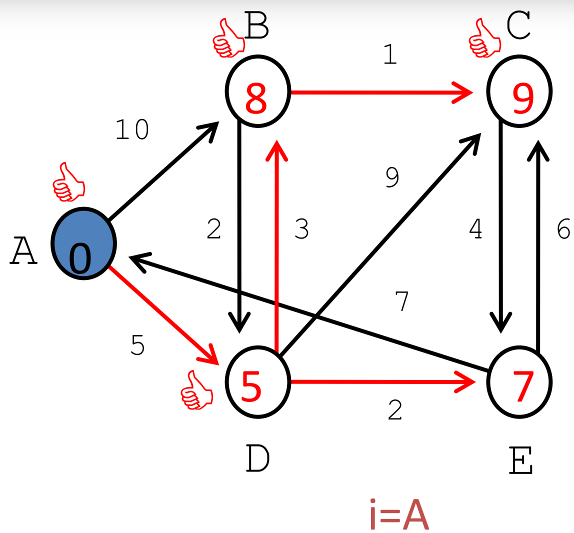


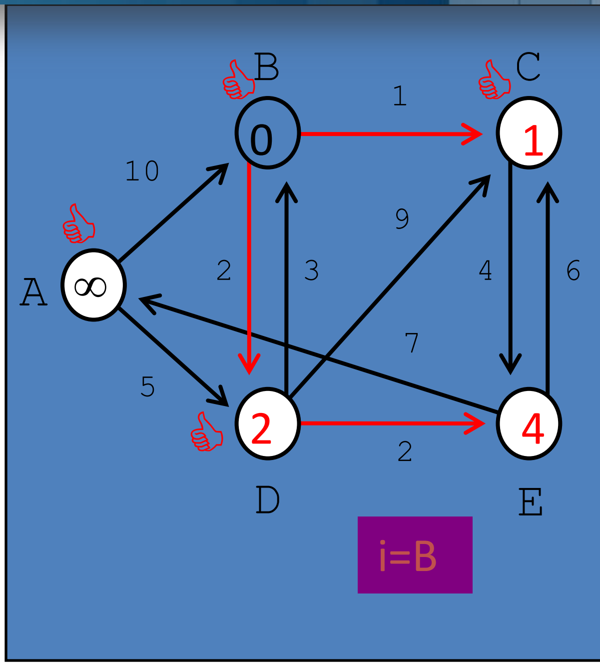


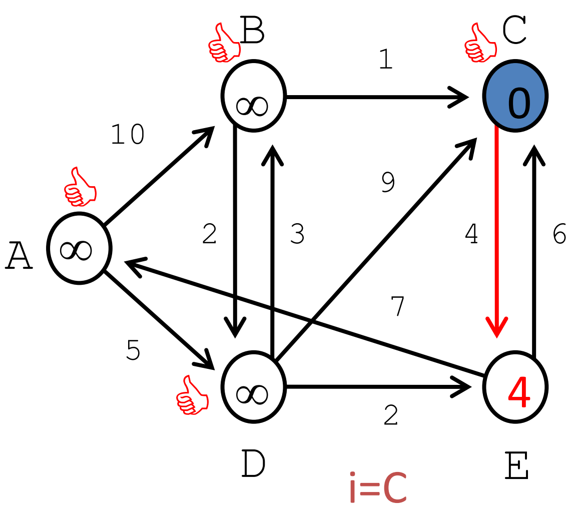


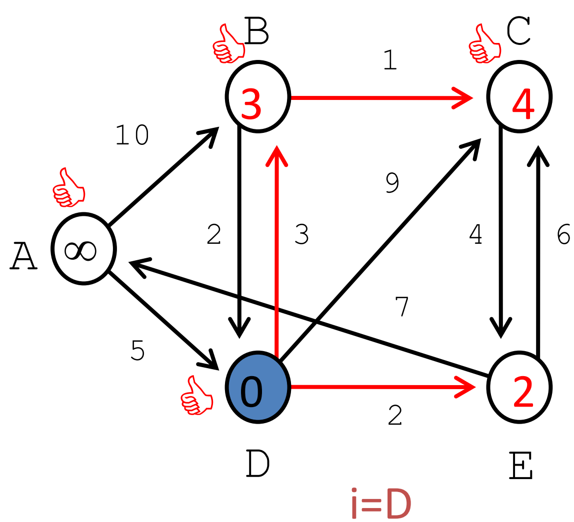
Solution: Floyd's Algorithm (k = D)

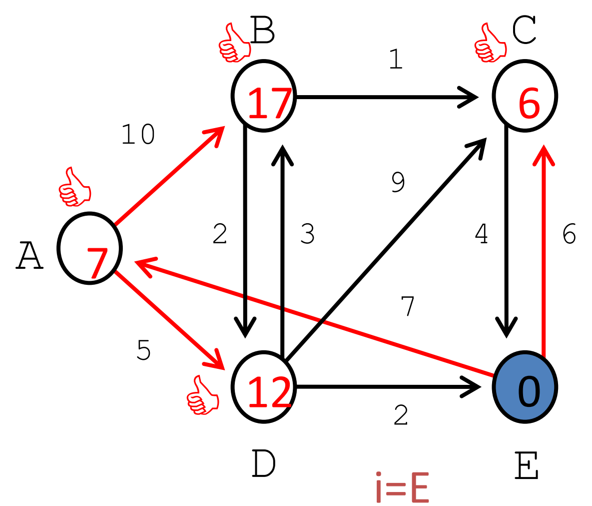
	A	B	C	D	E
A	0	8	9	5	7
B	∞	0	1	2	4
C	∞	∞	0	∞	4
D	∞	3	4	0	2
E	7	17	6	12	0





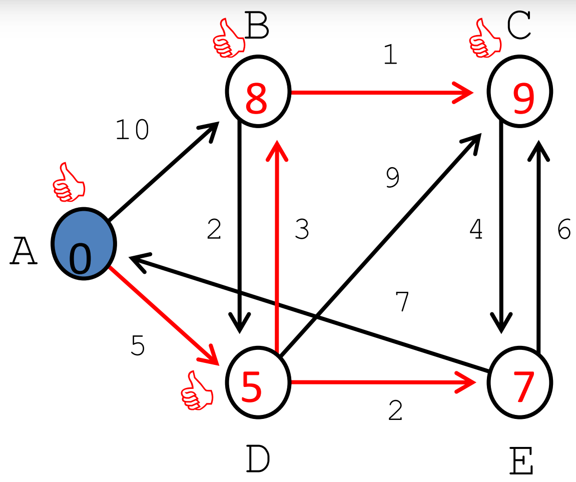




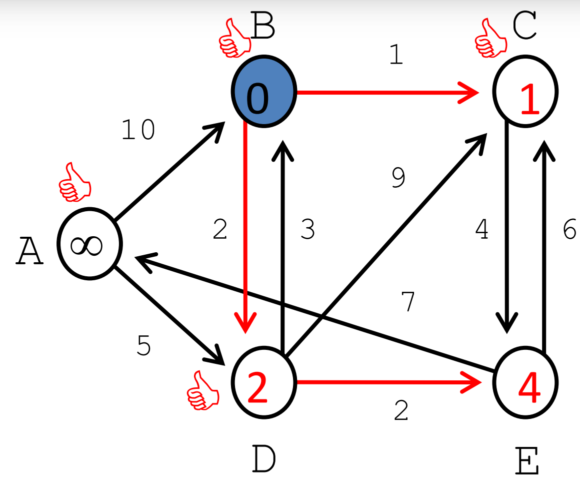


Solution: Floyd's Algorithm (k = D)

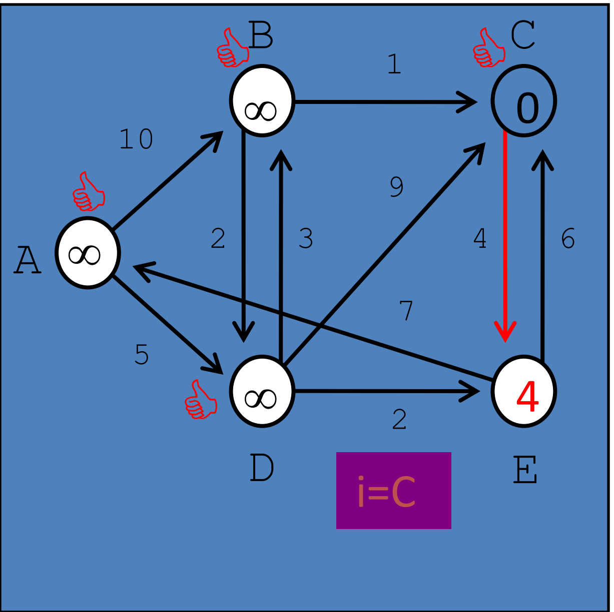
	A	B	C	D	E
A	0	8	9	5	7
B	∞	0	1	2	4
C	∞	∞	0	∞	4
D	∞	3	4	0	2
E	7	17	6	12	0

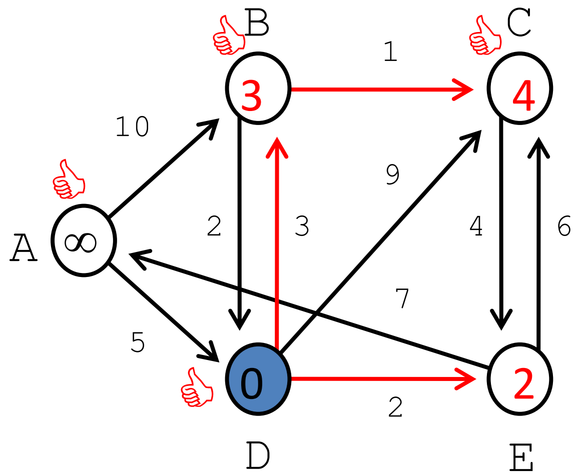


i=A

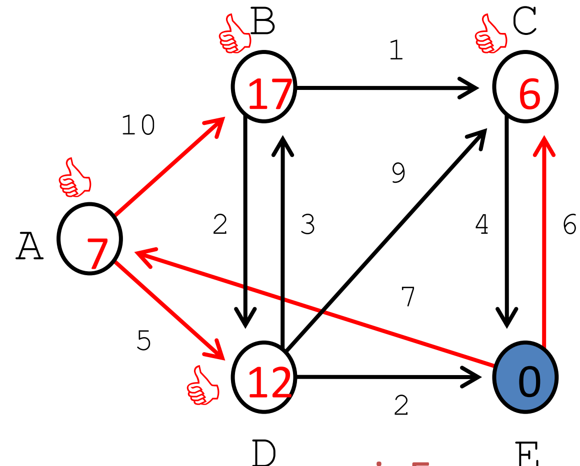


i=B





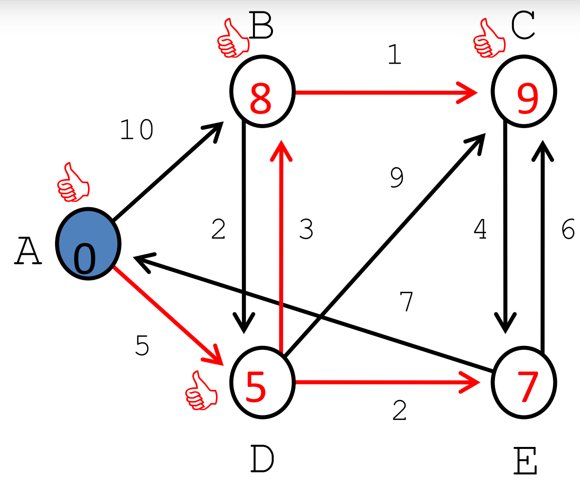
i=D



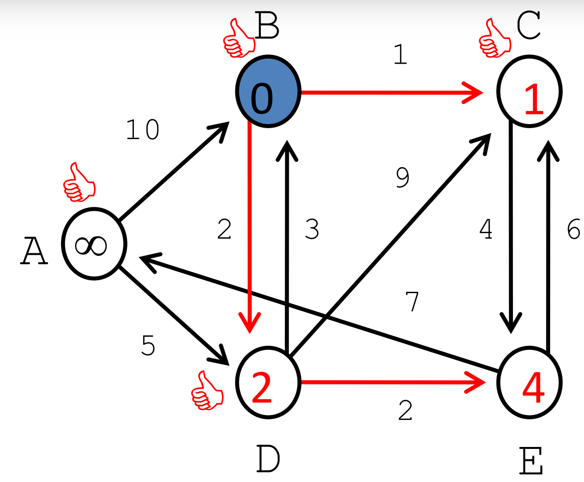
i=E

Solution: Floyd's Algorithm (k = D)

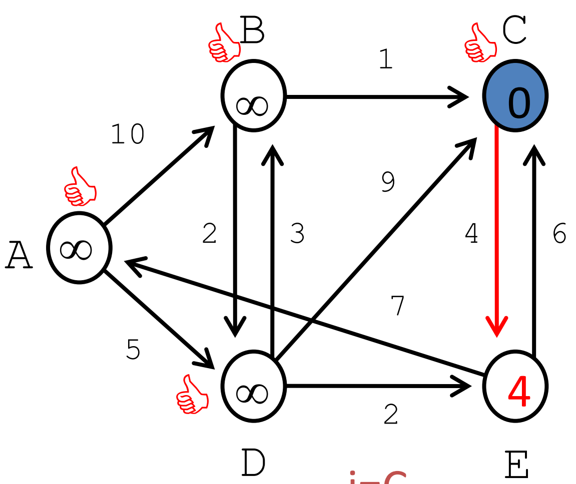
	A	B	C	D	E
A	0	8	9	5	7
B	∞	0	1	2	4
C	∞	∞	0	∞	4
D	∞	3	4	0	2
E	7	17	6	12	0



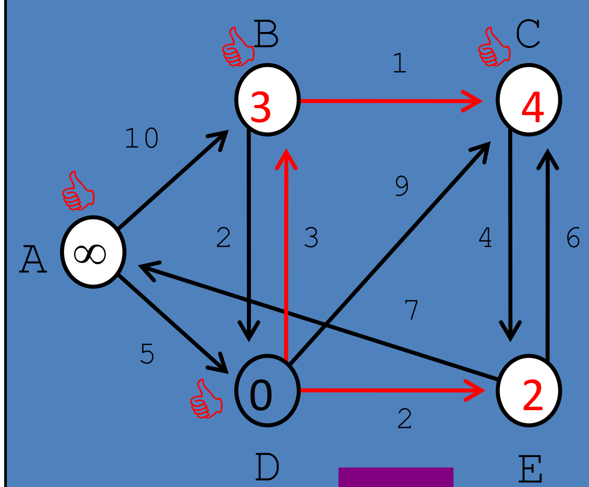
i=A



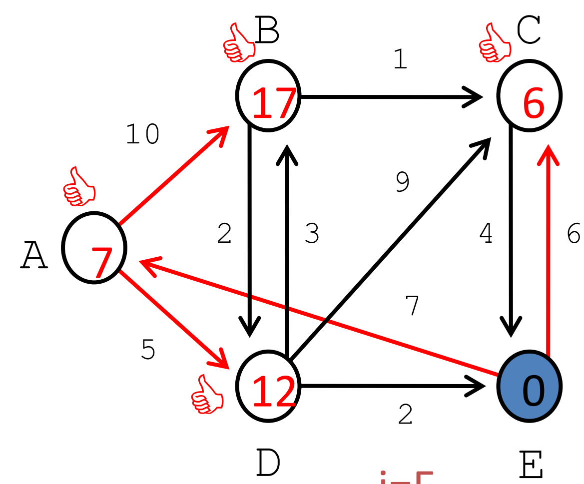
i=B



i=C



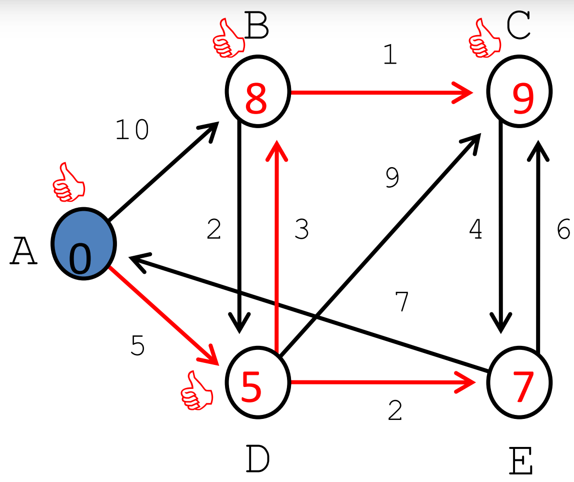
i=D



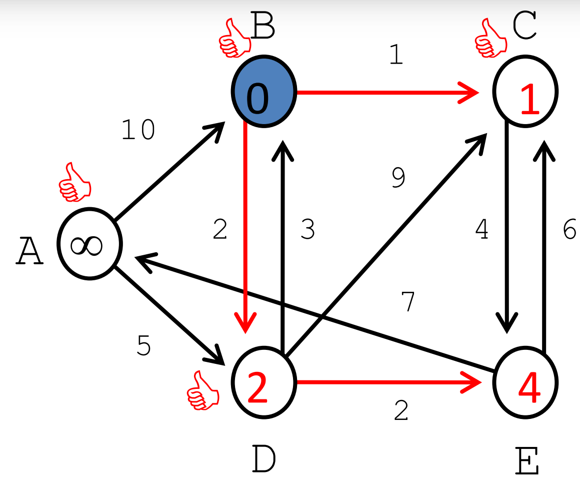
i=E

Solution: Floyd's Algorithm (k = D)

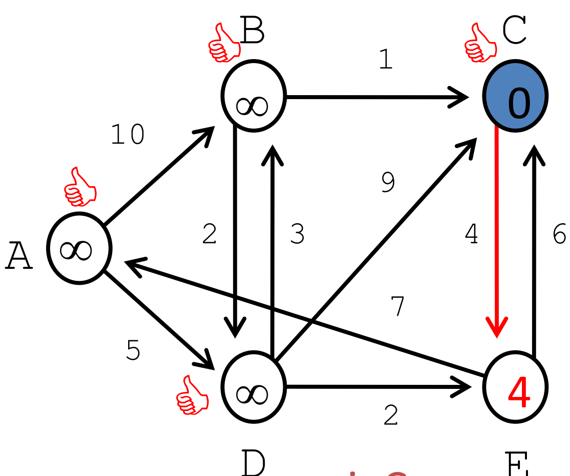
	A	B	C	D	E
A	0	8	9	5	7
B	∞	0	1	2	4
C	∞	∞	0	∞	4
D	∞	3	4	0	2
E	7	**15**	6	12	0



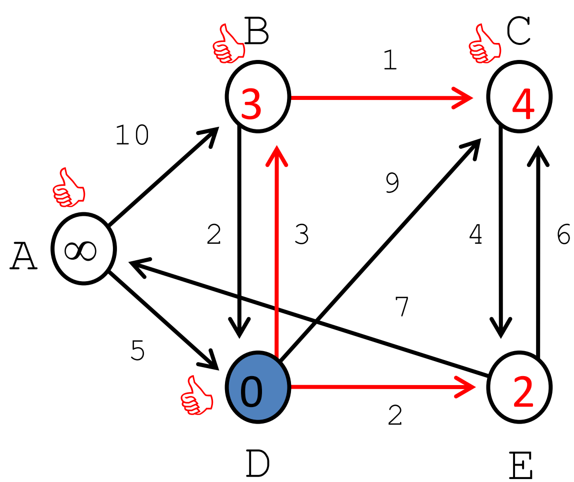
i=A



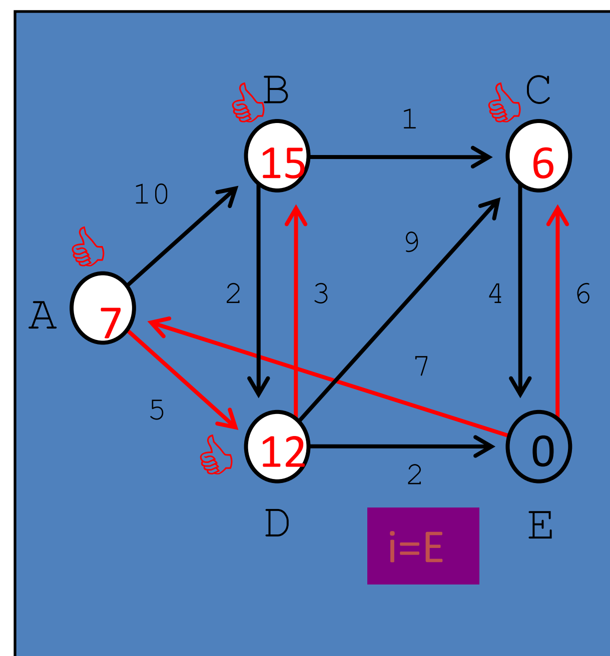
i=B



i=C



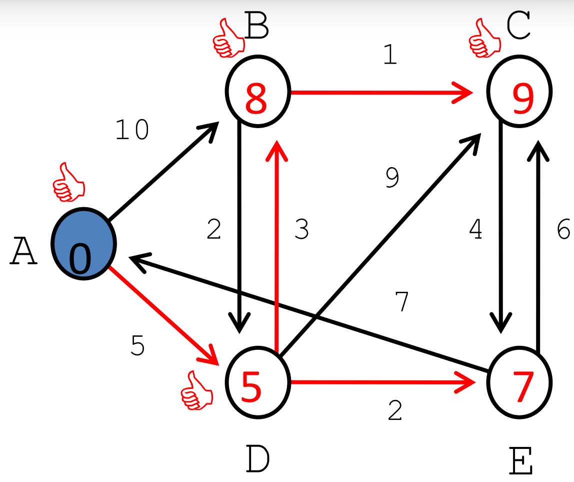
i=D



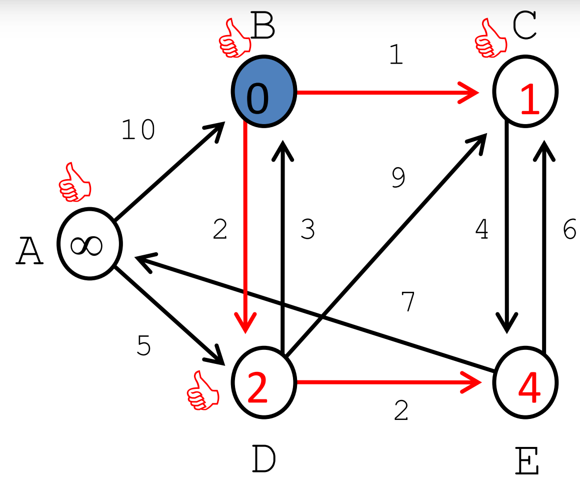
Solution: Floyd's Algorithm

Resulting table (k = D)

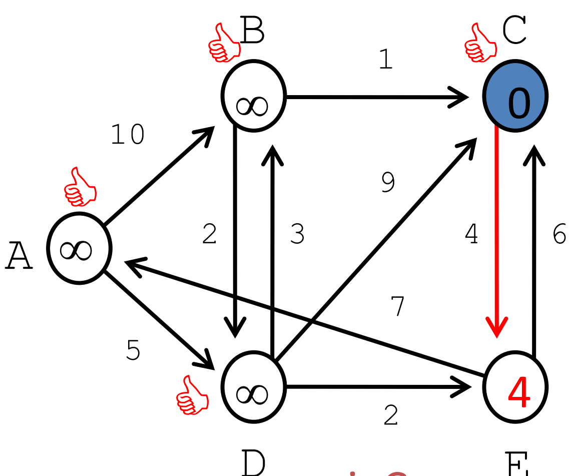
	A	B	C	D	E
A	0	8	9	5	7
B	∞	0	1	2	4
C	∞	∞	0	∞	4
D	∞	3	4	0	2
E	7	15	6	12	0



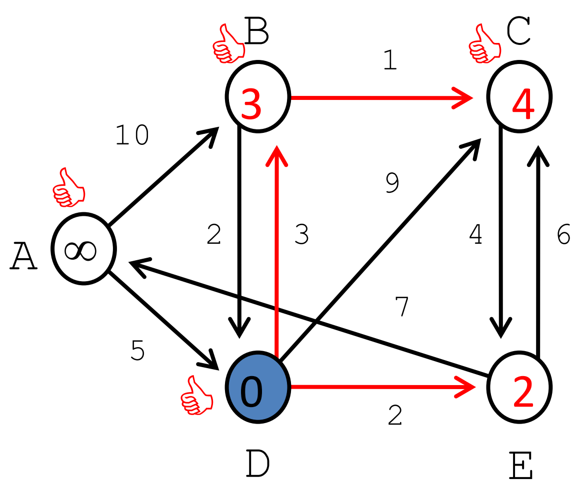
i=A



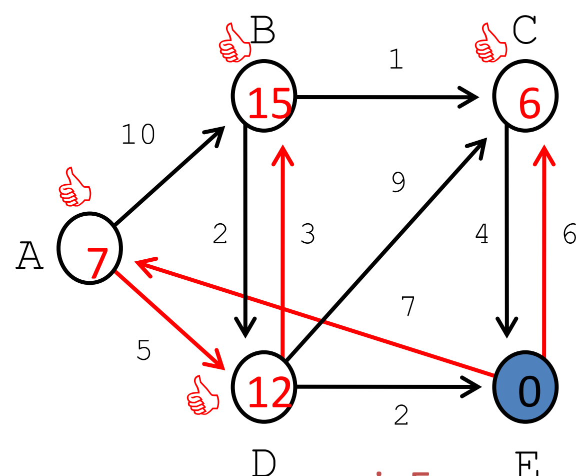
i=B



i=C



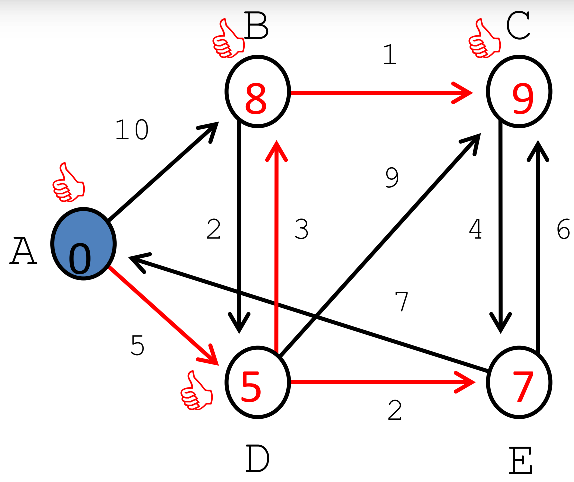
i=D



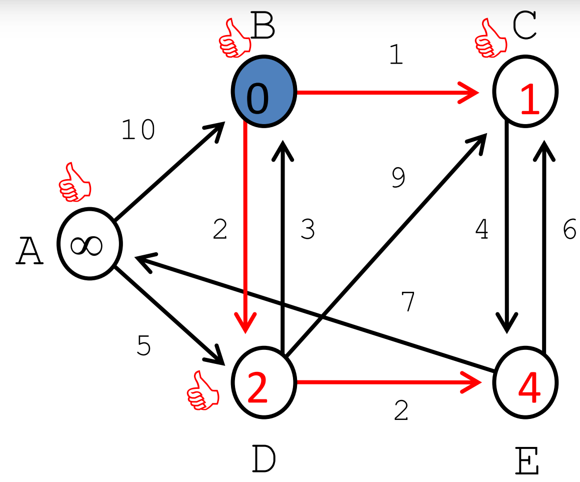
i=E

Solution: Floyd's Algorithm (k = E)

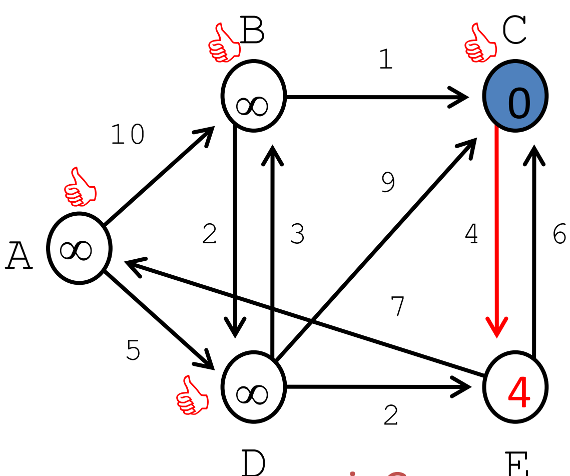
	A	B	C	D	E
A	0	8	9	5	7
B	∞	0	1	2	4
C	∞	∞	0	∞	4
D	∞	3	4	0	2
E	7	15	6	12	0



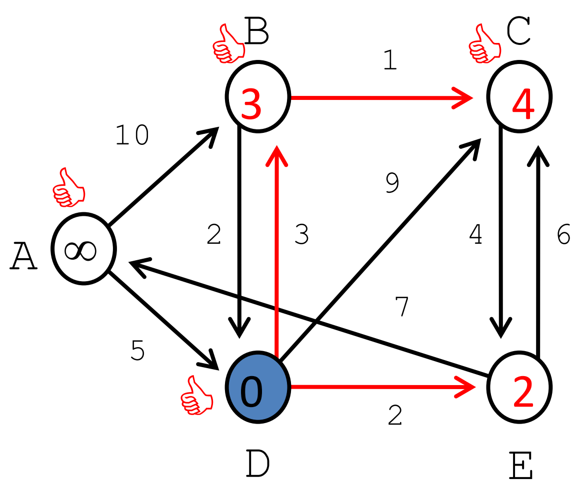
i=A



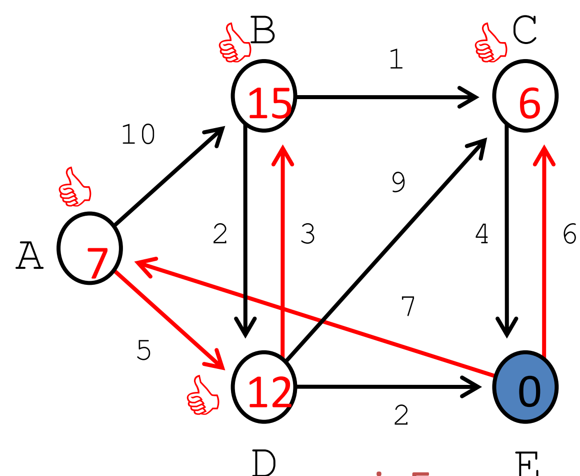
i=B



i=C



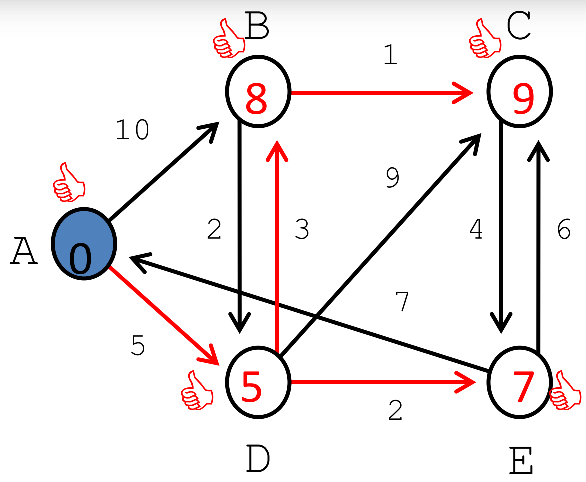
i=D



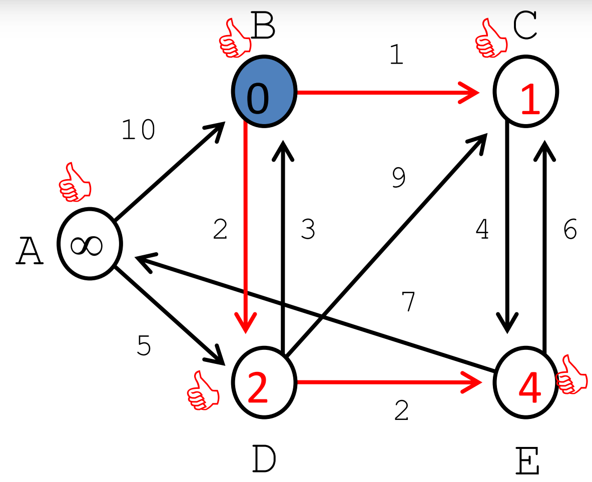
i=E

Solution: Floyd's Algorithm (k = E)

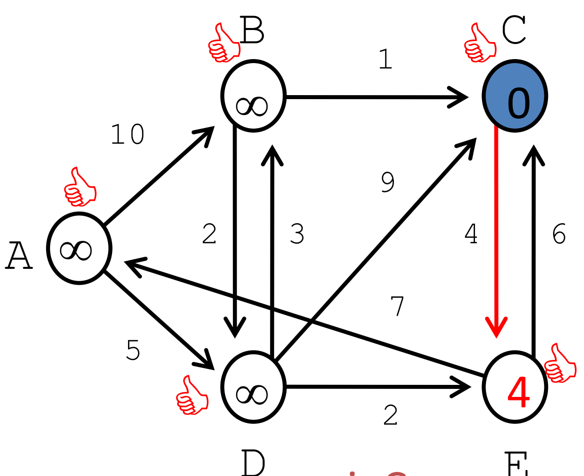
	A	B	C	D	E
A	0	8	9	5	7
B	∞	0	1	2	4
C	∞	∞	0	∞	4
D	∞	3	4	0	2
E	7	15	6	12	0



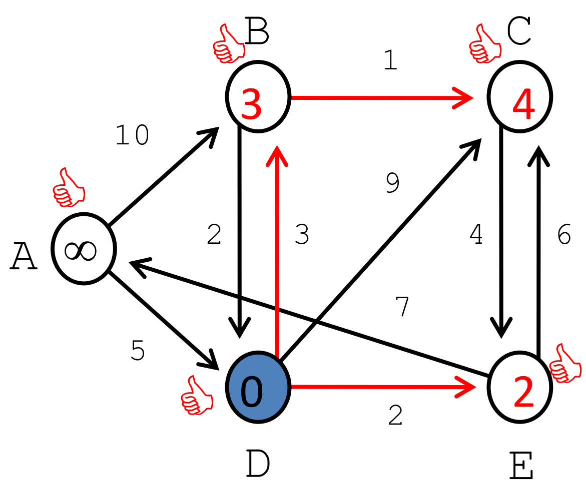
i=A



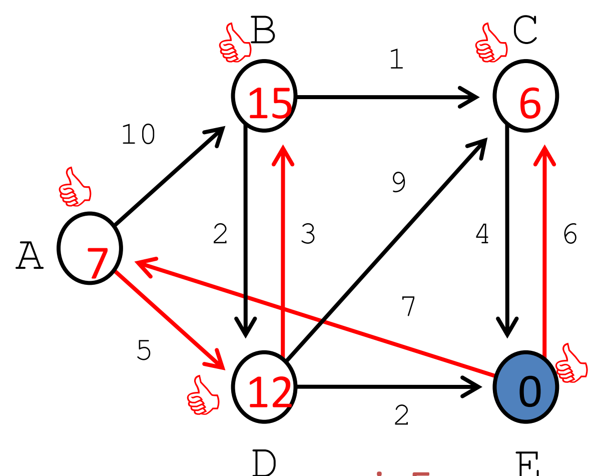
i=B



i=C



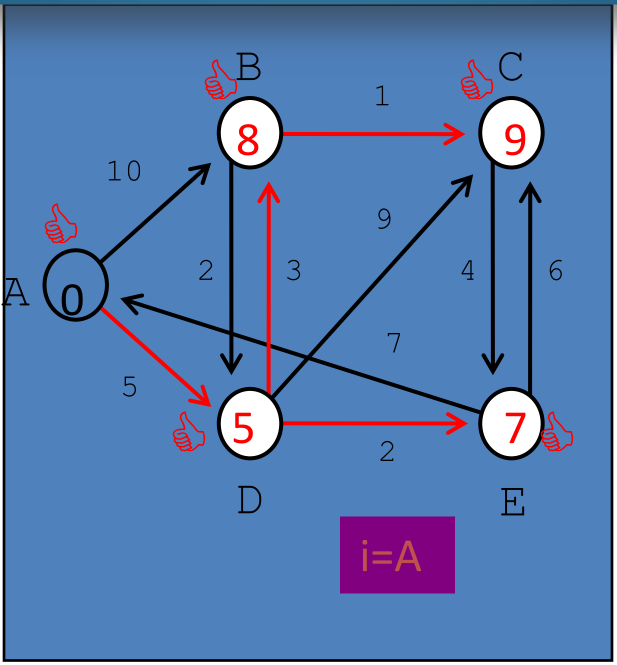
i=D

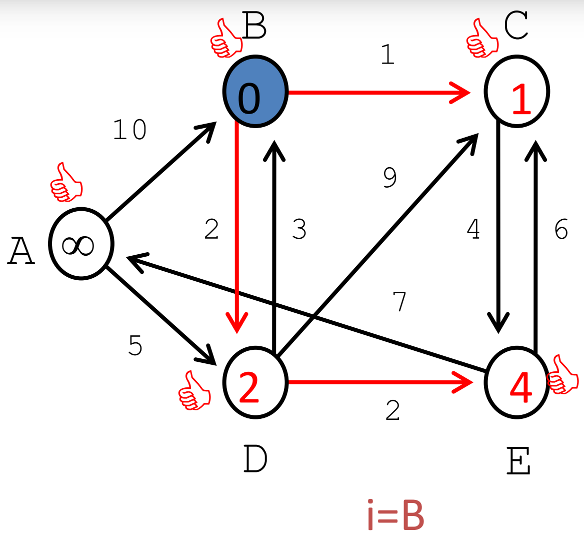


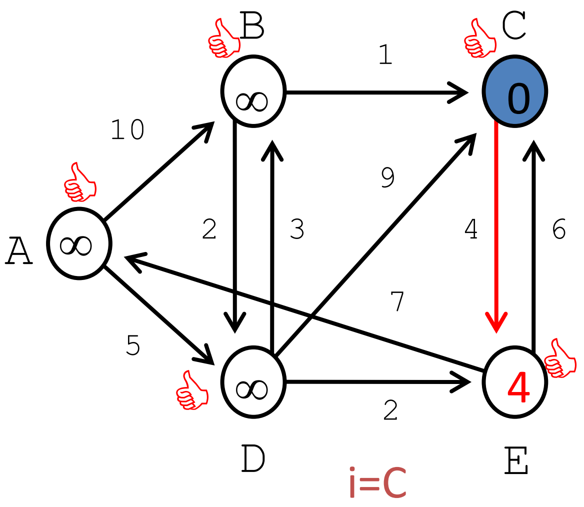
i=E

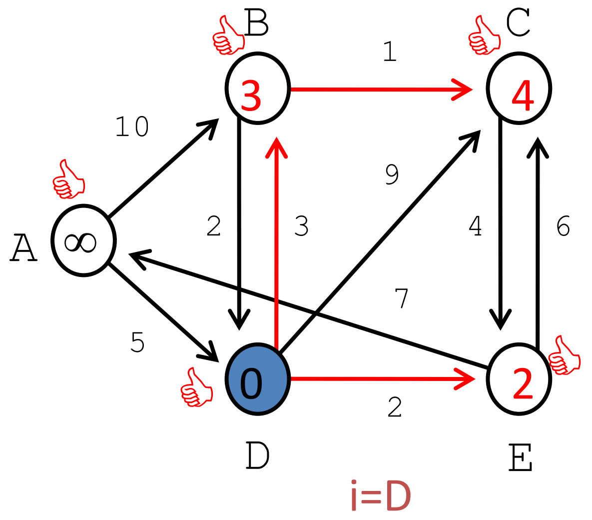
Solution: Floyd's Algorithm (k = E)

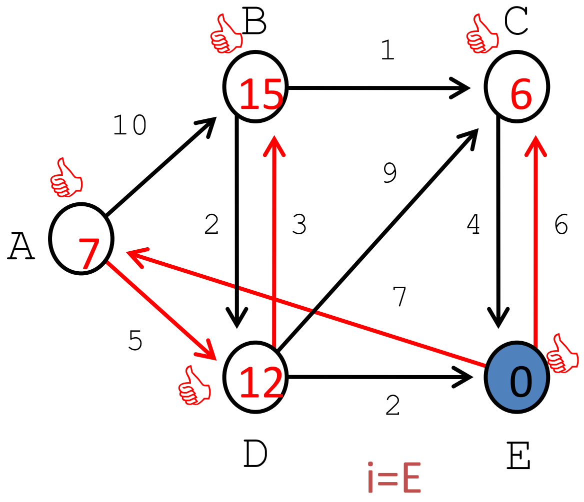
	A	B	C	D	E
A	0	8	9	5	7
B	∞	0	1	2	4
C	∞	∞	0	∞	4
D	∞	3	4	0	2
E	7	15	6	12	0





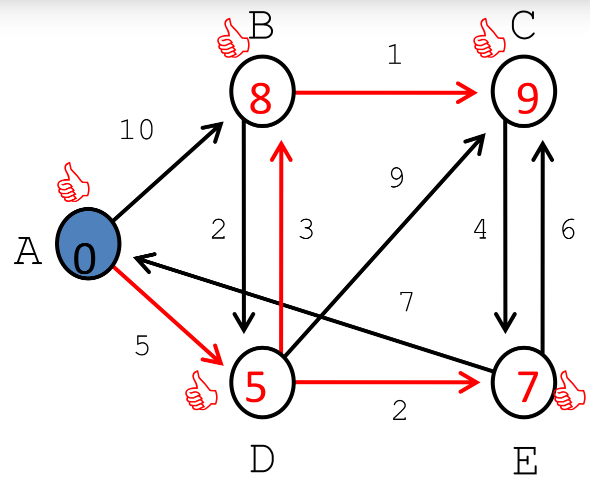




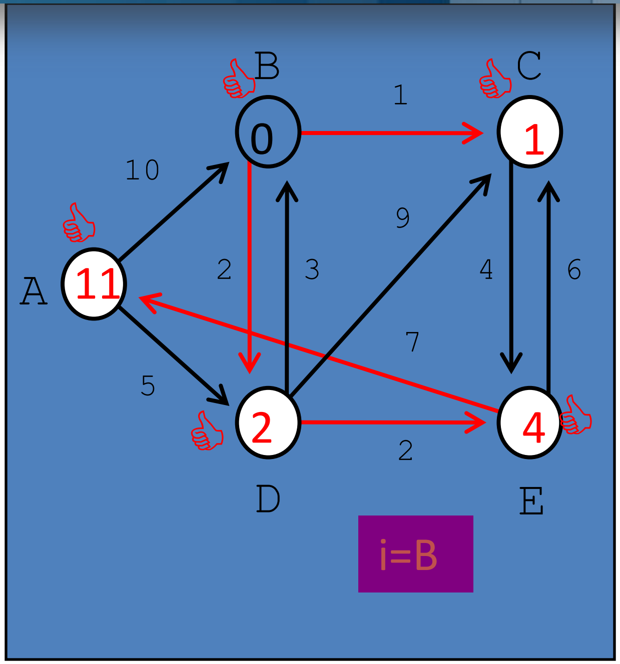


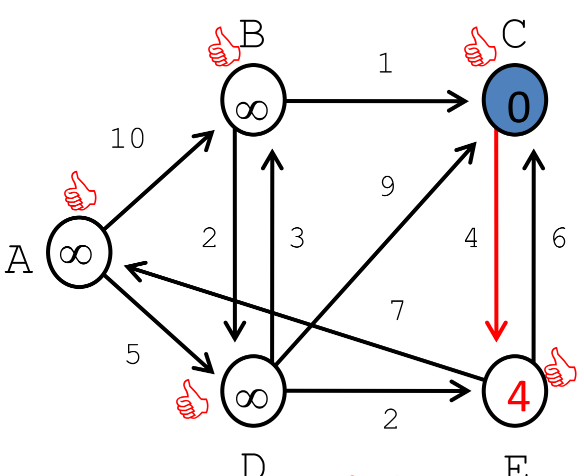
Solution: Floyd's Algorithm (k = E)

	A	B	C	D	E
A	0	8	9	5	7
B	11	0	1	2	4
C	∞	∞	0	∞	4
D	∞	3	4	0	2
E	7	15	6	12	0

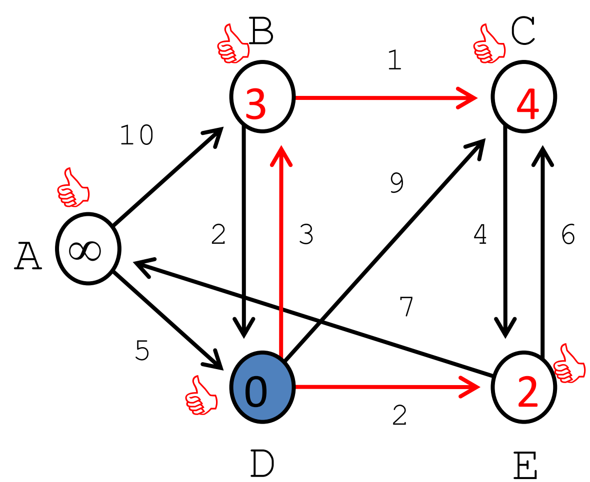


i=A

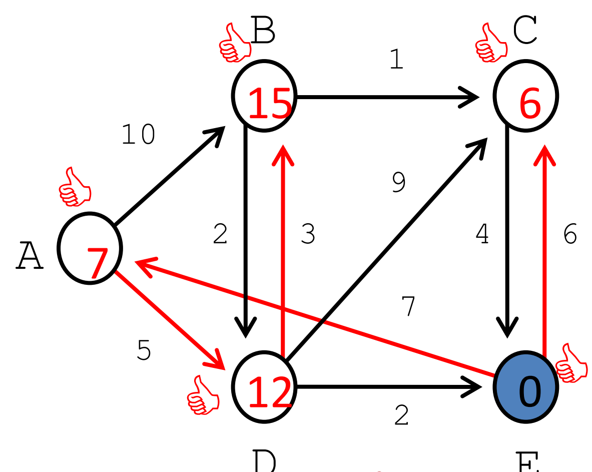




i=C



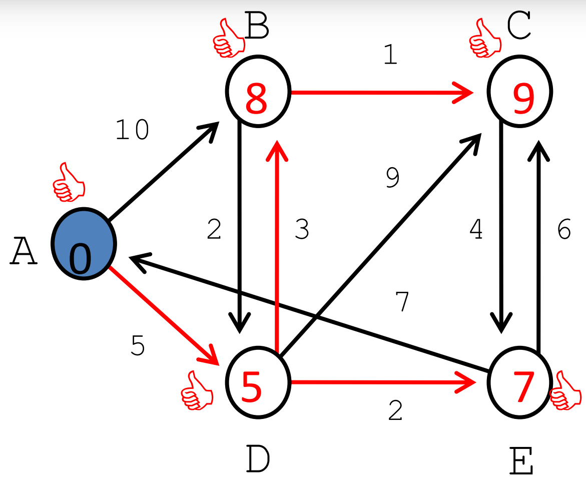
i=D



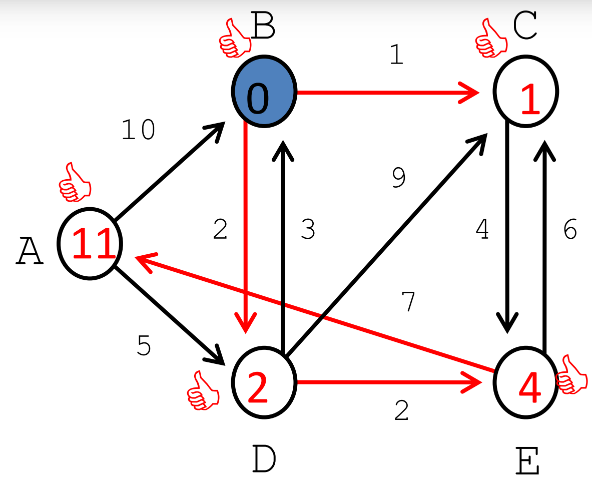
i=E

Solution: Floyd's Algorithm (k = E)

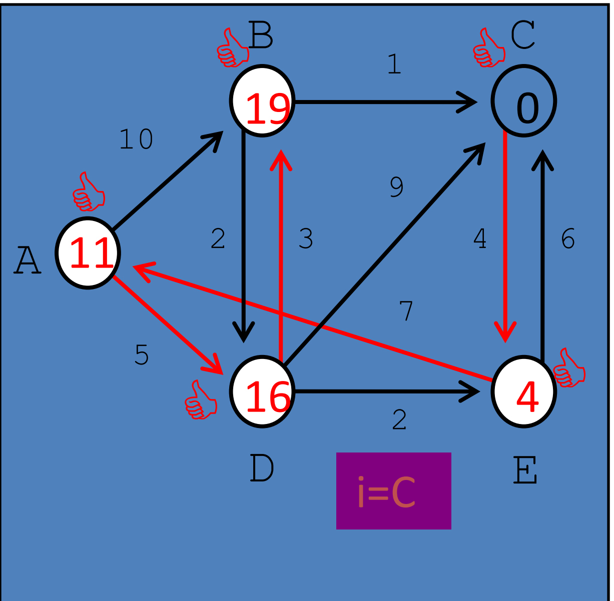
	A	B	C	D	E
A	0	8	9	5	7
B	11	0	1	2	4
C	11	19	0	16	4
D	∞	3	4	0	2
E	7	15	6	12	0

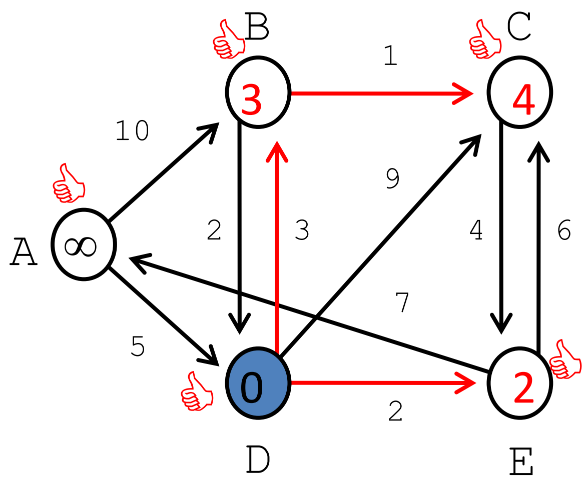


i=A

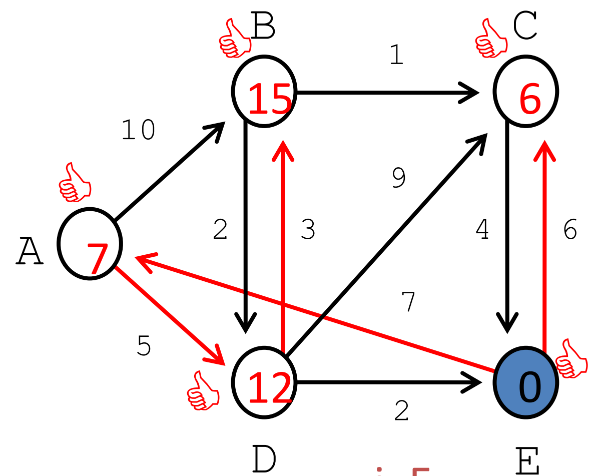


i=B





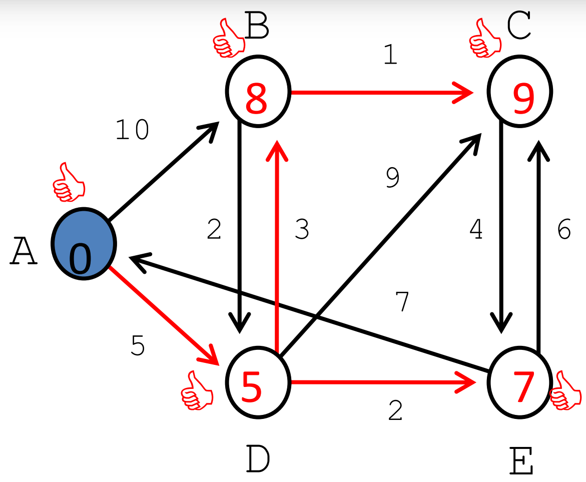
i=D



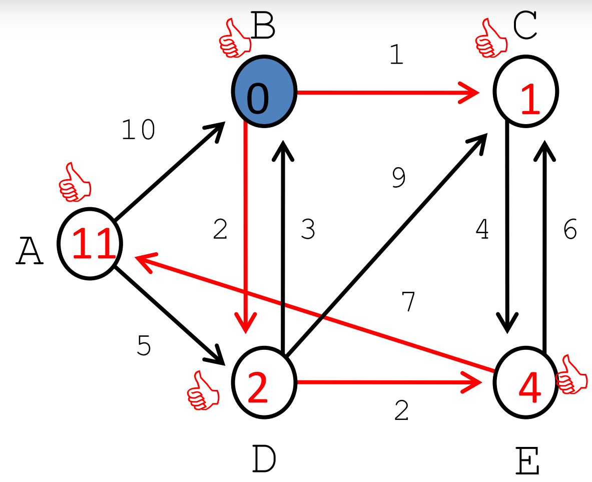
i=E

Solution: Floyd's Algorithm (k = E)

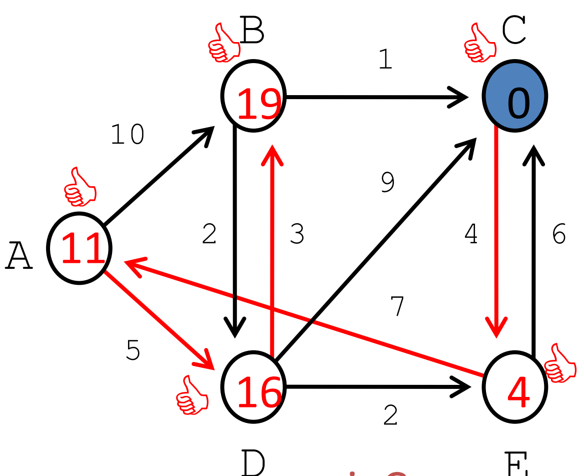
	A	B	C	D	E
A	0	8	9	5	7
B	11	0	1	2	4
C	11	19	0	16	4
D	9	3	4	0	2
E	7	15	6	12	0



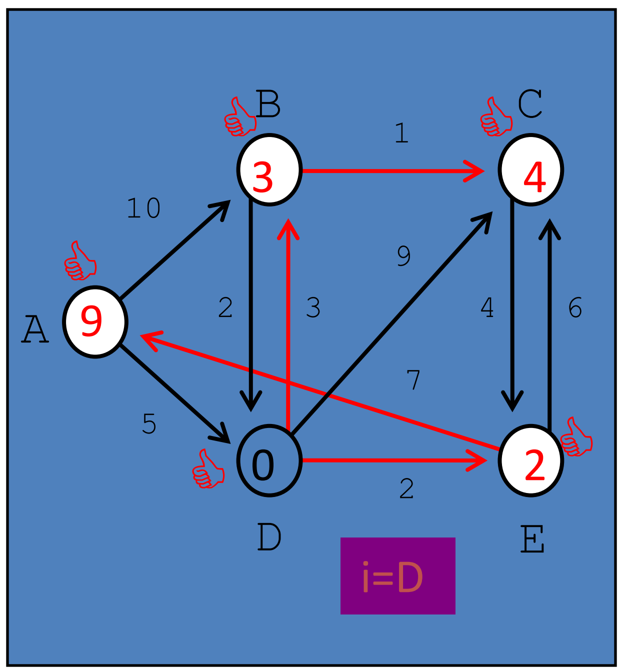
i=A

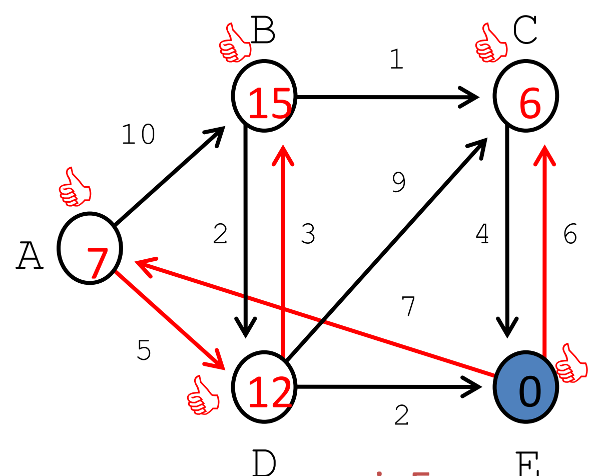


i=B



i=C

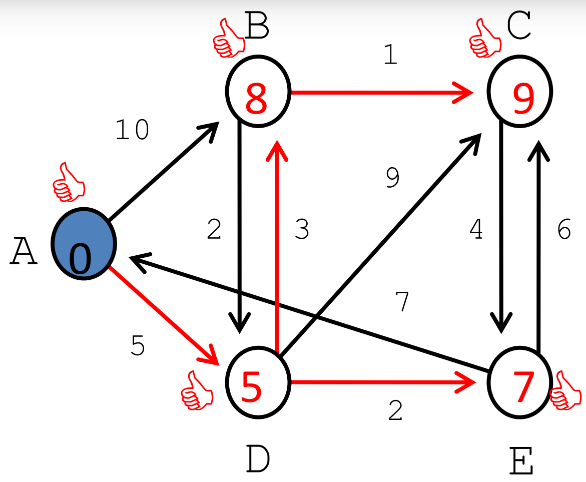




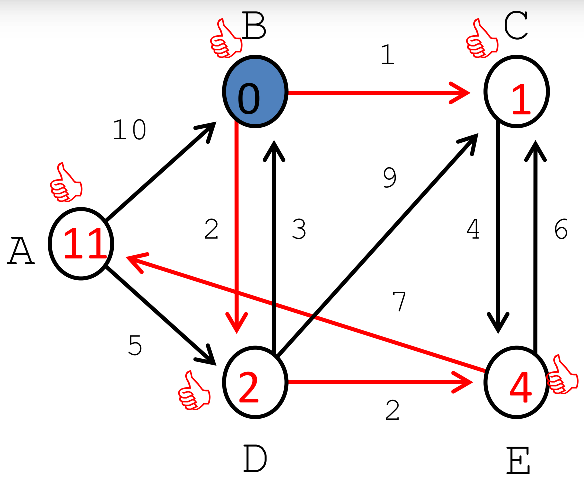
i=E

Solution: Floyd's Algorithm (k = E)

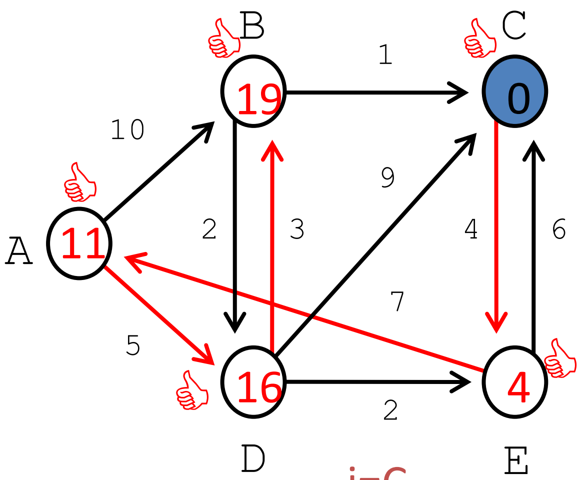
	A	B	C	D	E
A	0	8	9	5	7
B	11	0	1	2	4
C	11	19	0	16	4
D	9	3	4	0	2
E	7	15	6	12	0



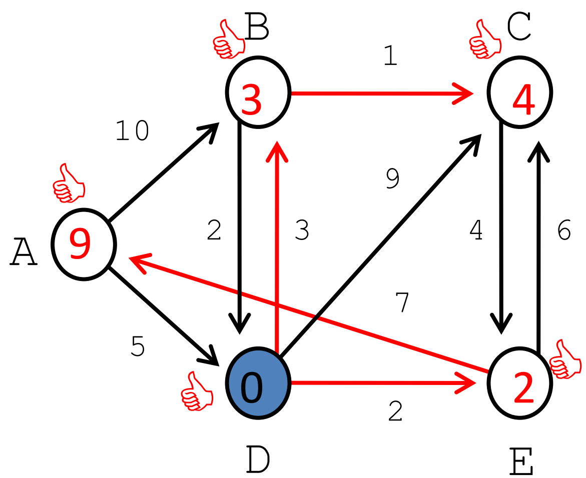
i=A



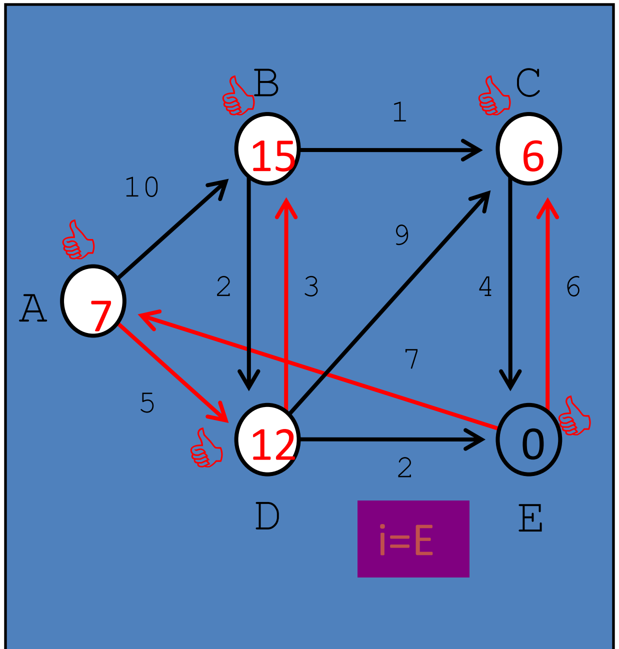
i=B



i=C



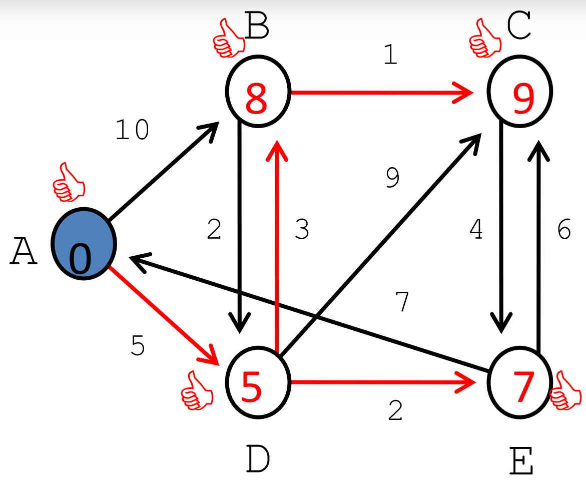
i=D



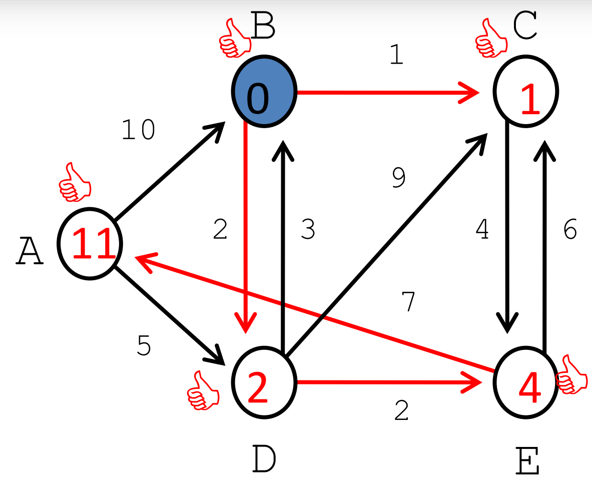
Final Cost Matrix

Resulting table (k = E)

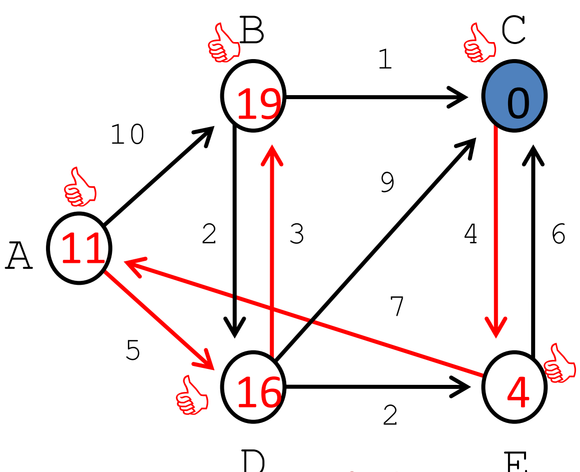
	A	B	C	D	E
A	0	8	9	5	7
B	11	0	1	2	4
C	11	19	0	16	4
D	9	3	4	0	2
E	7	15	6	12	0



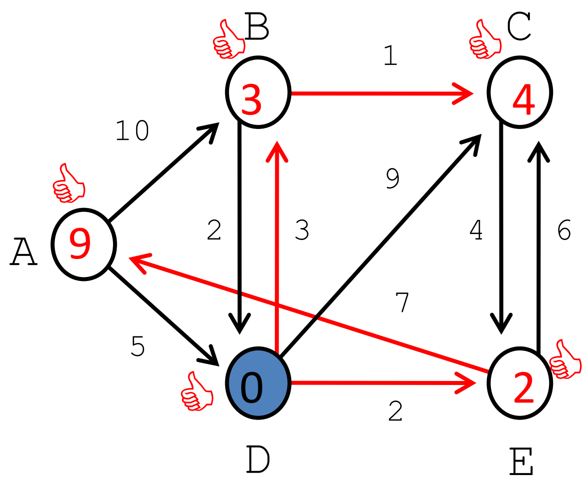
i=A



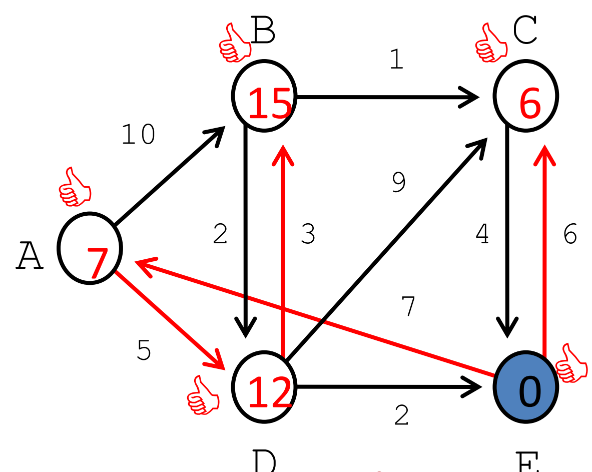
i=B



i=C



i=D



i=E

7. Graphs

7.4 Other Graph Problems

Minimum Spanning Tree

- A *spanning tree* is a tree which includes all the vertices in the graph.

- A *minimum spanning tree* of an undirected graph G is a tree formed from graph edges that connects all the vertices of G at lowest total cost. An MST exists if and only if G is connected.

Problem

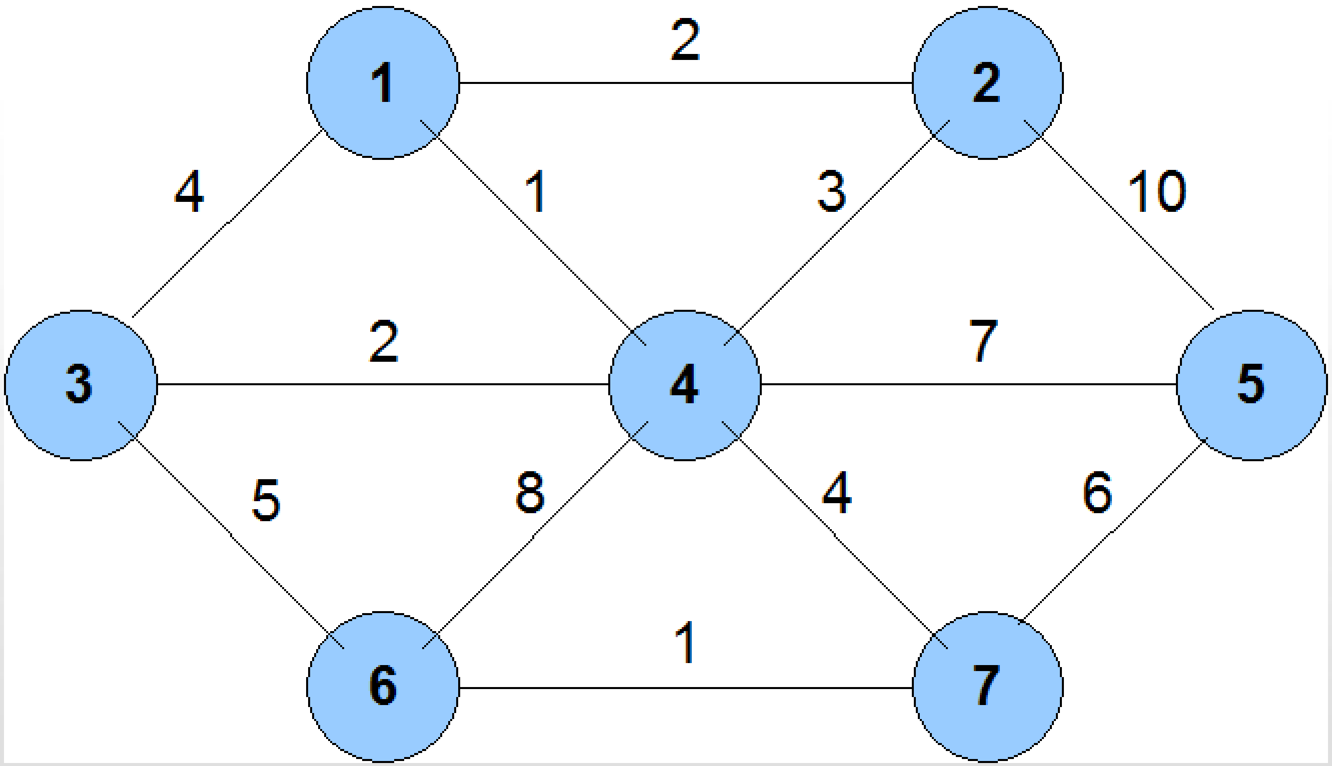
- Given a *weighted* undirected graph $G = (V, E)$ where $V = \{1, 2, 3, \ldots, n\}$, find a spanning tree T such that the sum of the weights of the edges is the smallest possible.

Greedy Algorithms for MST

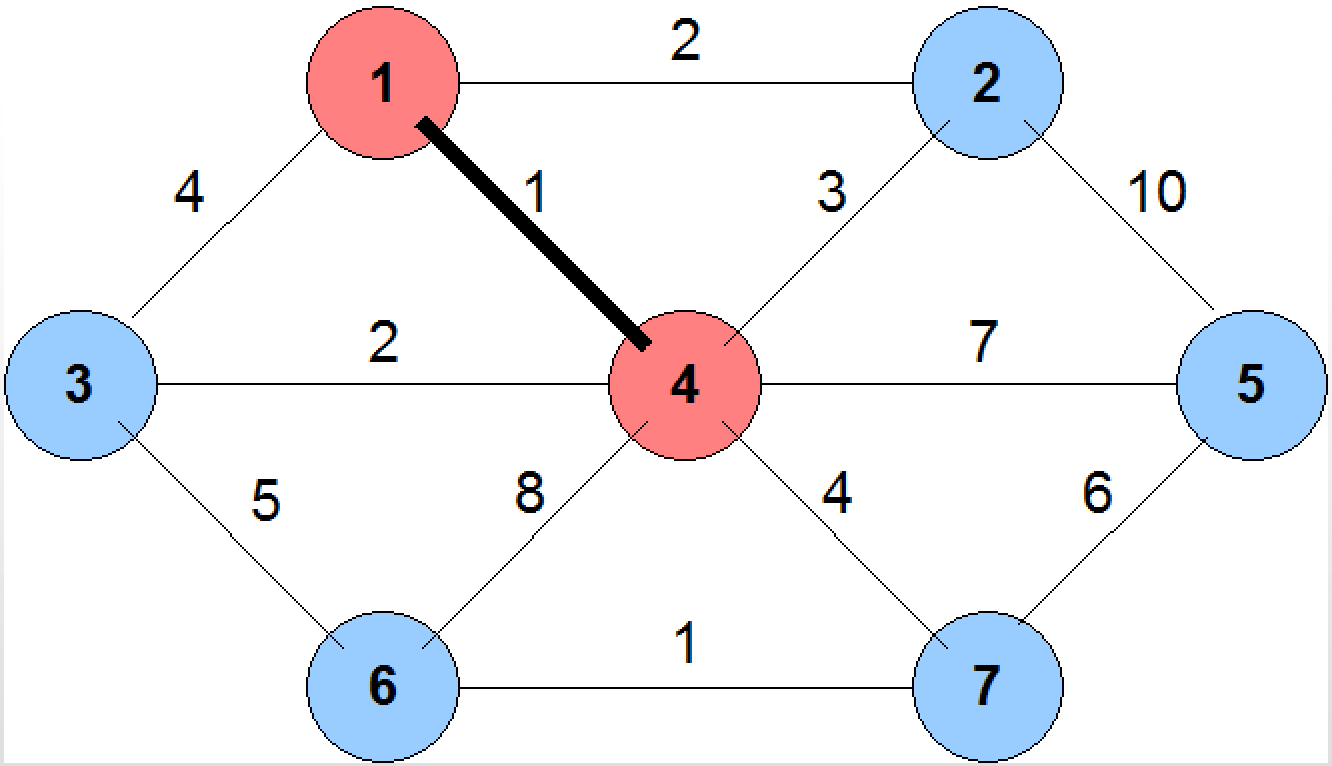
Algo 1: Kruskal’s Algorithm (shortest edge first)

- Sort the edges of G in increasing order.
- Examine the edges of G in increasing order and retain those that do not form a cycle.

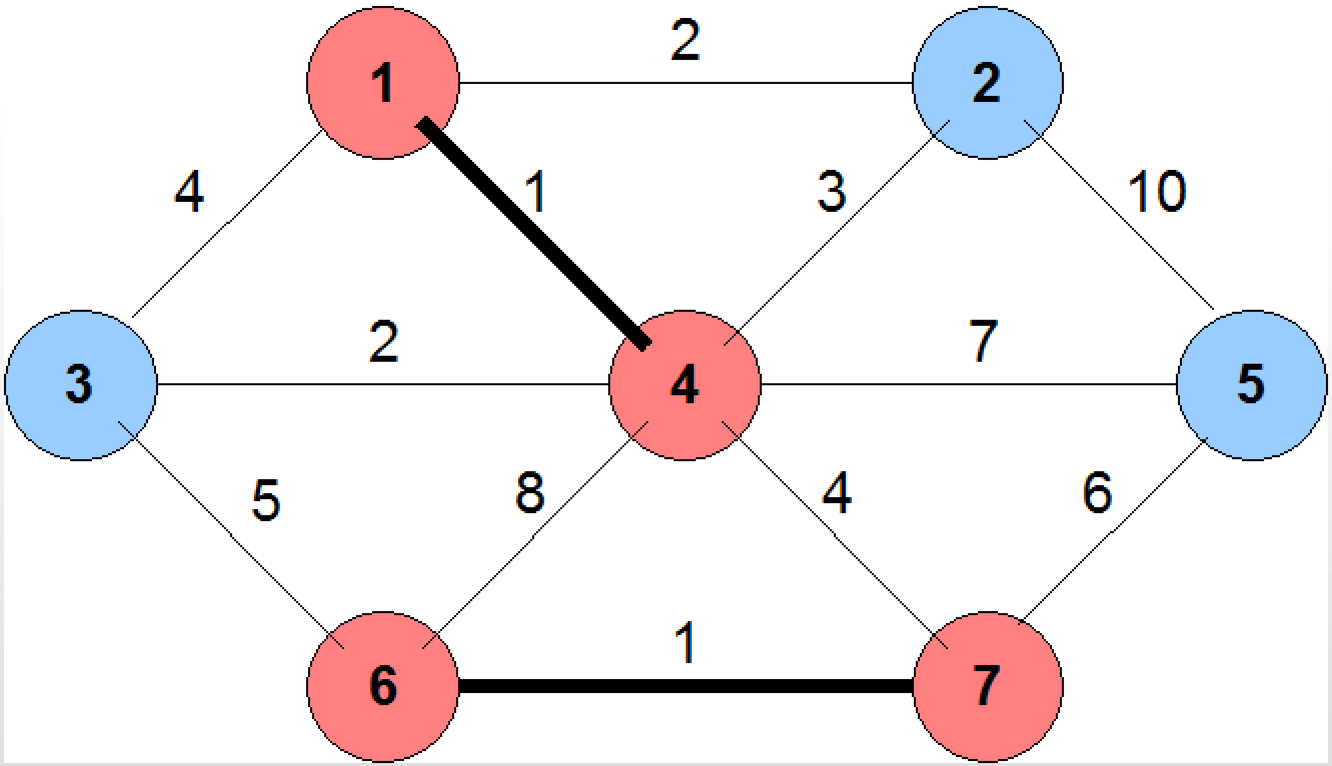
Kruskal's Algorithm



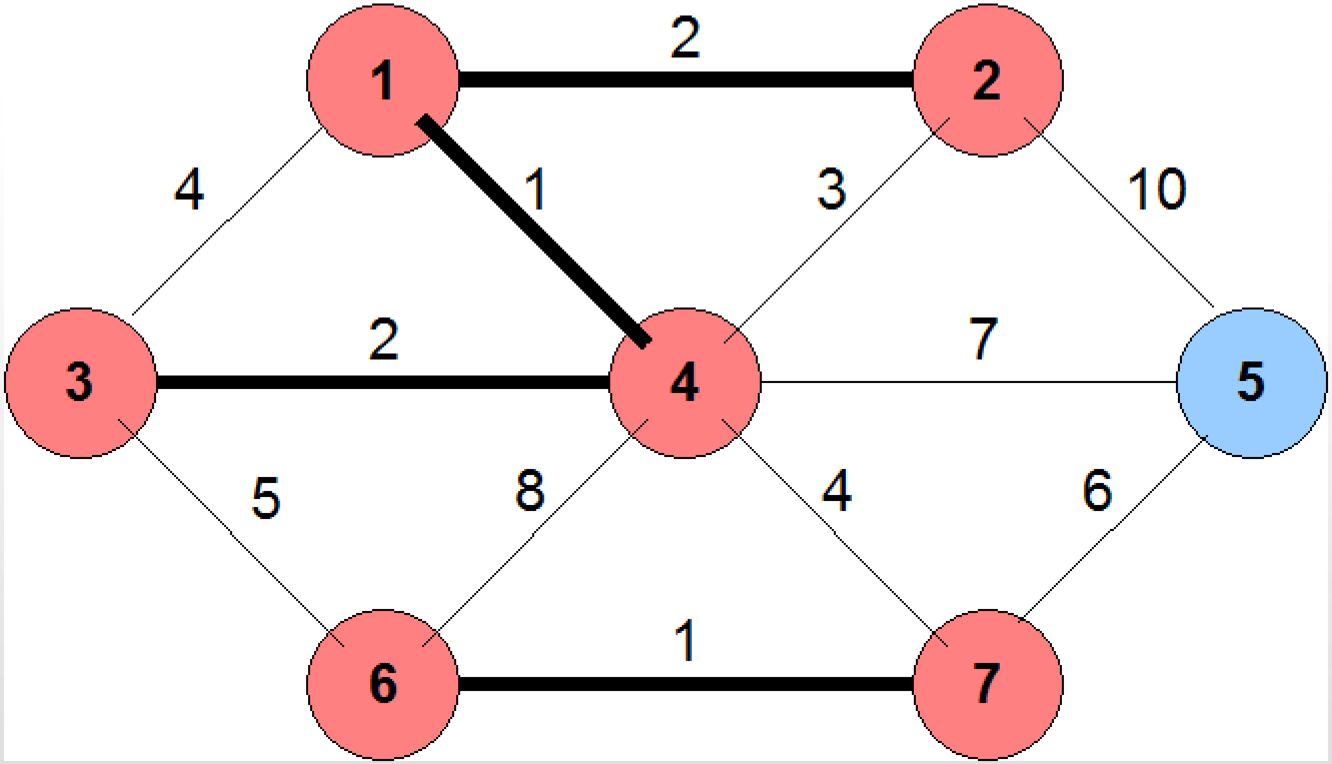
Kruskal's Algorithm



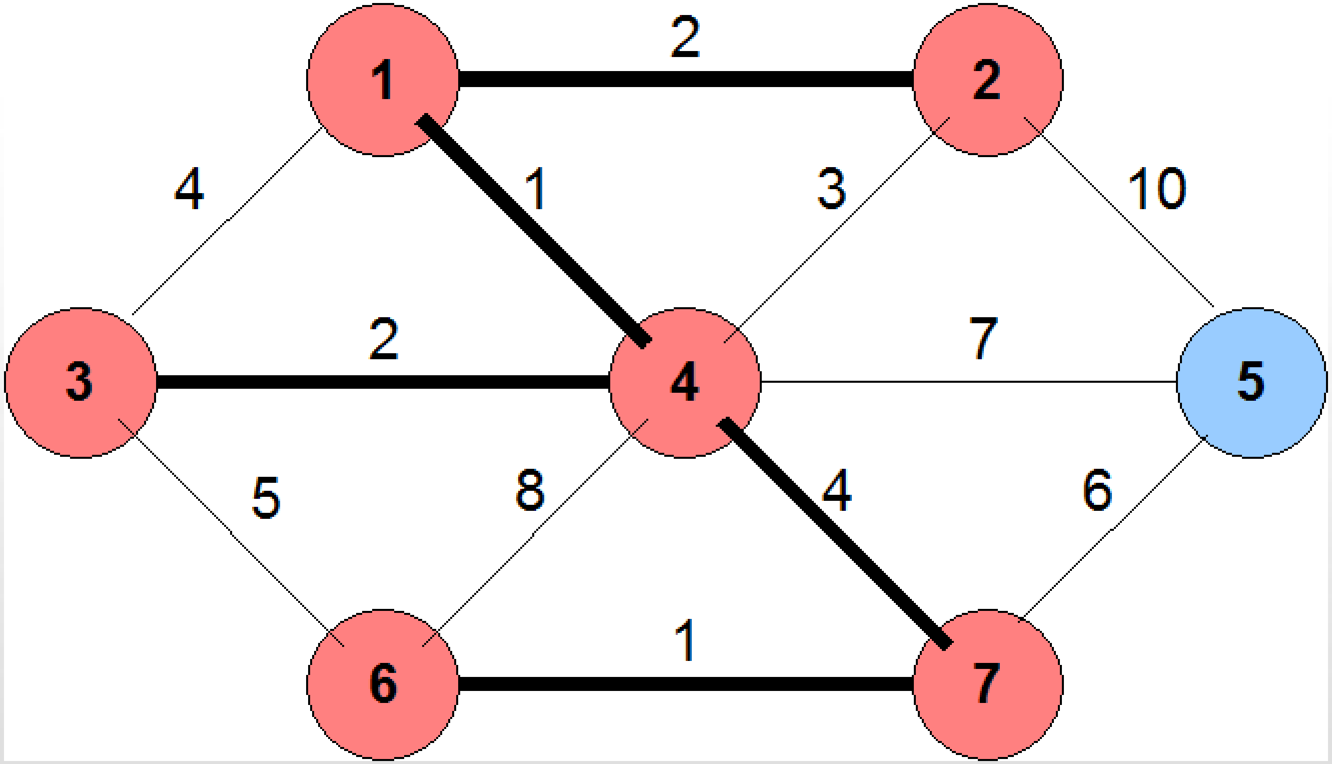
Kruskal's Algorithm



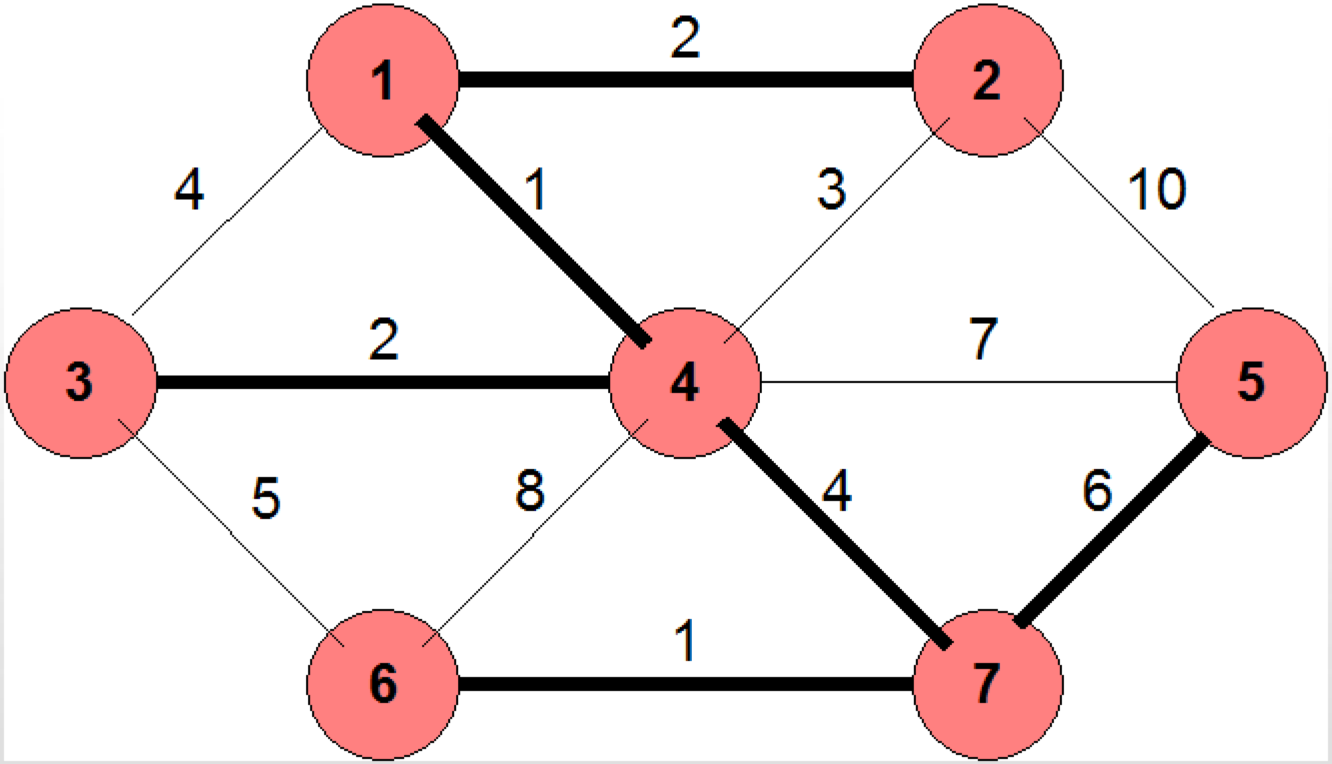
Kruskal’s Algorithm



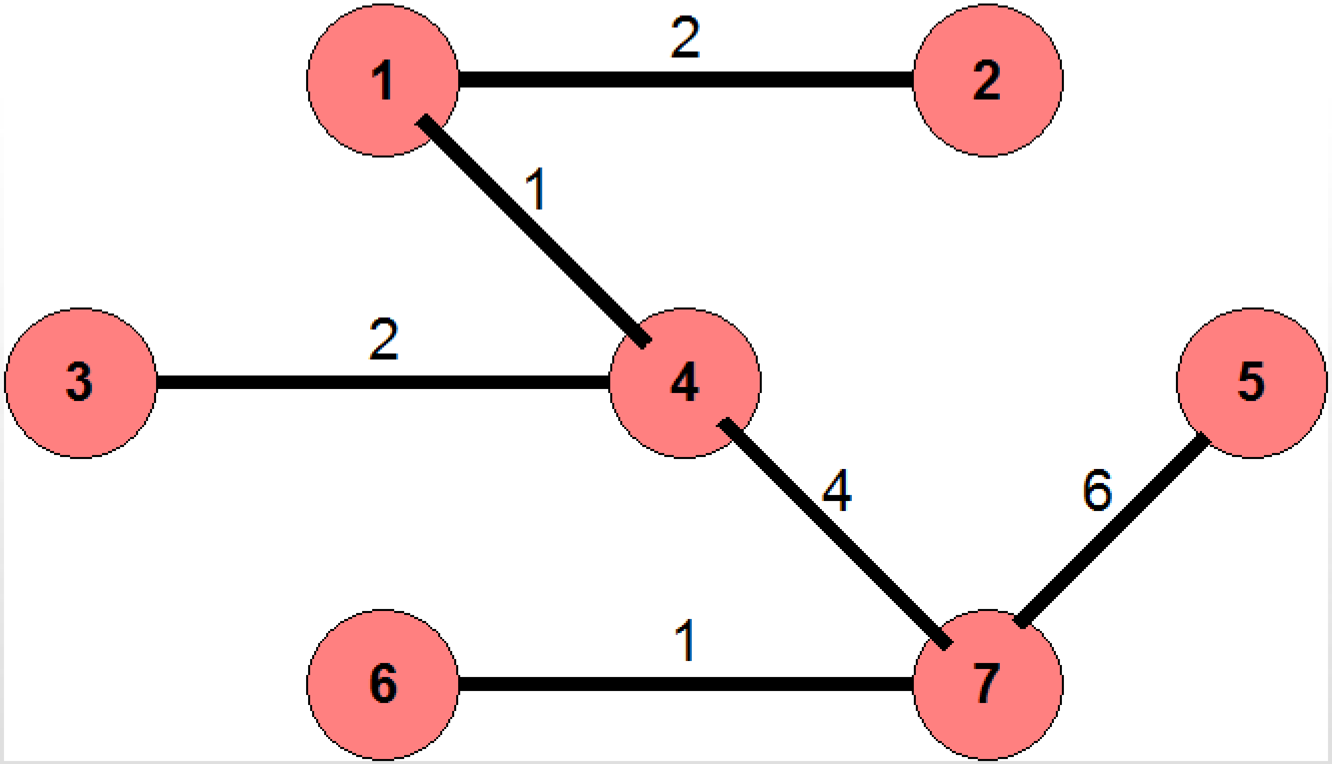
Kruskal's Algorithm



Kruskal's Algorithm



Kruskal's Algorithm

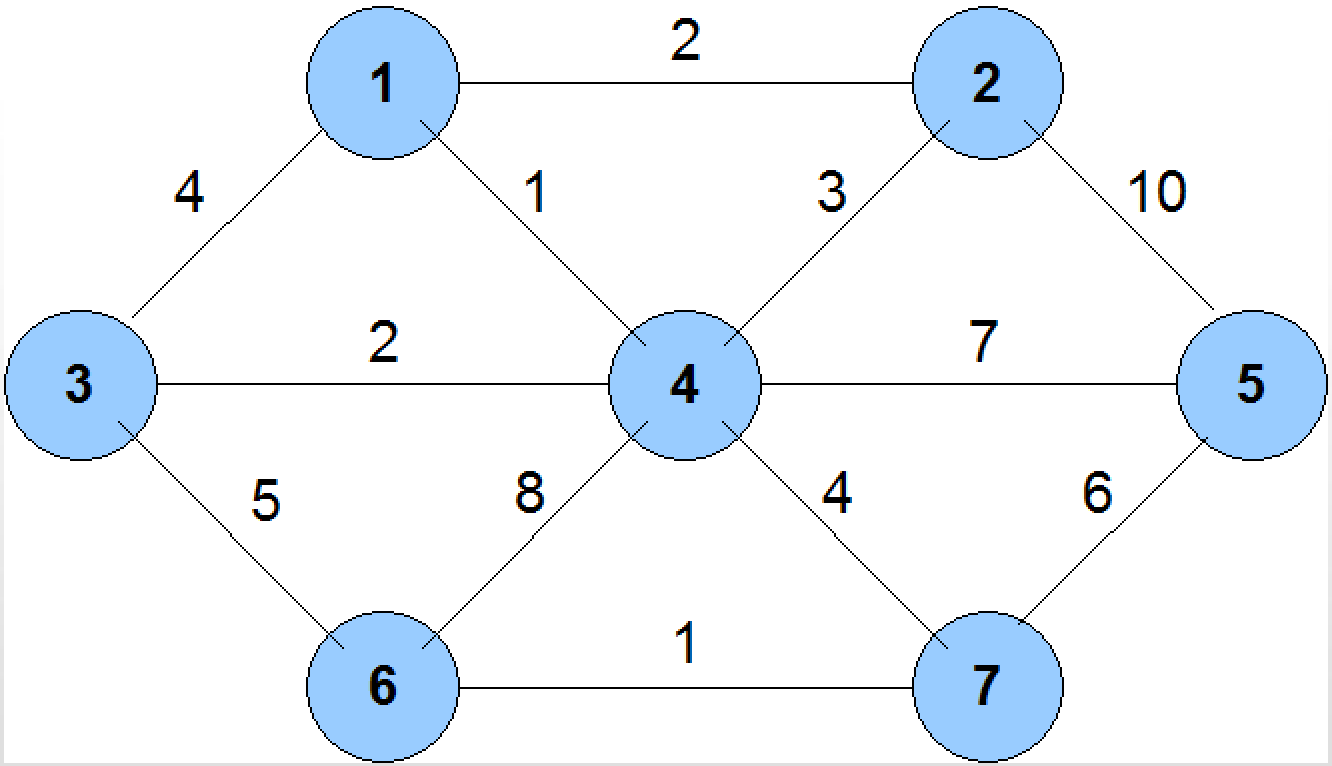


Greedy Algorithms for MST

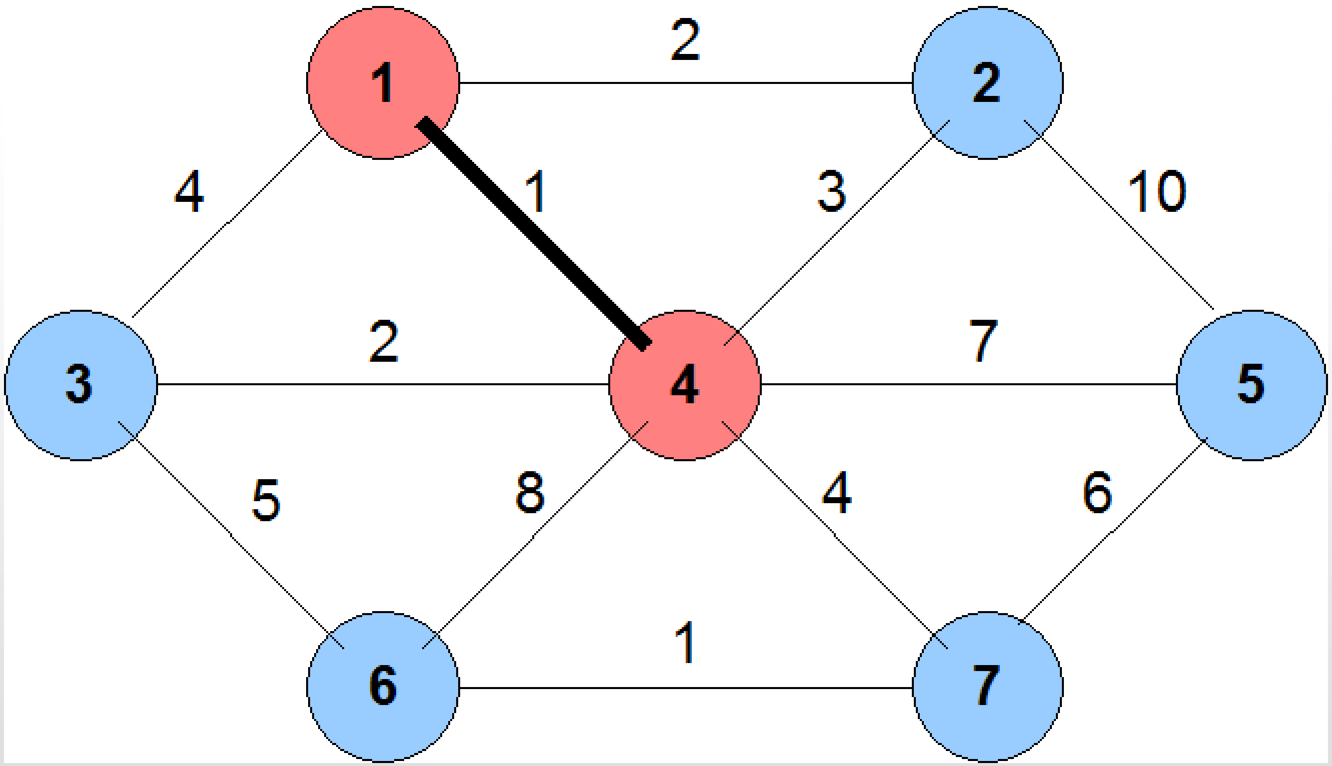
Algo 2: Prim's Algorithm (nearest neighbor)

- Start with some arbitrary vertex *v*.
- Let *(u,v)* be the edge of least weight incident on v; edge *(u,v)* is included in the tree.
- From among the edges incident on either *u* or v, we select the edge with the least weight and include this edge in the partially formed tree.

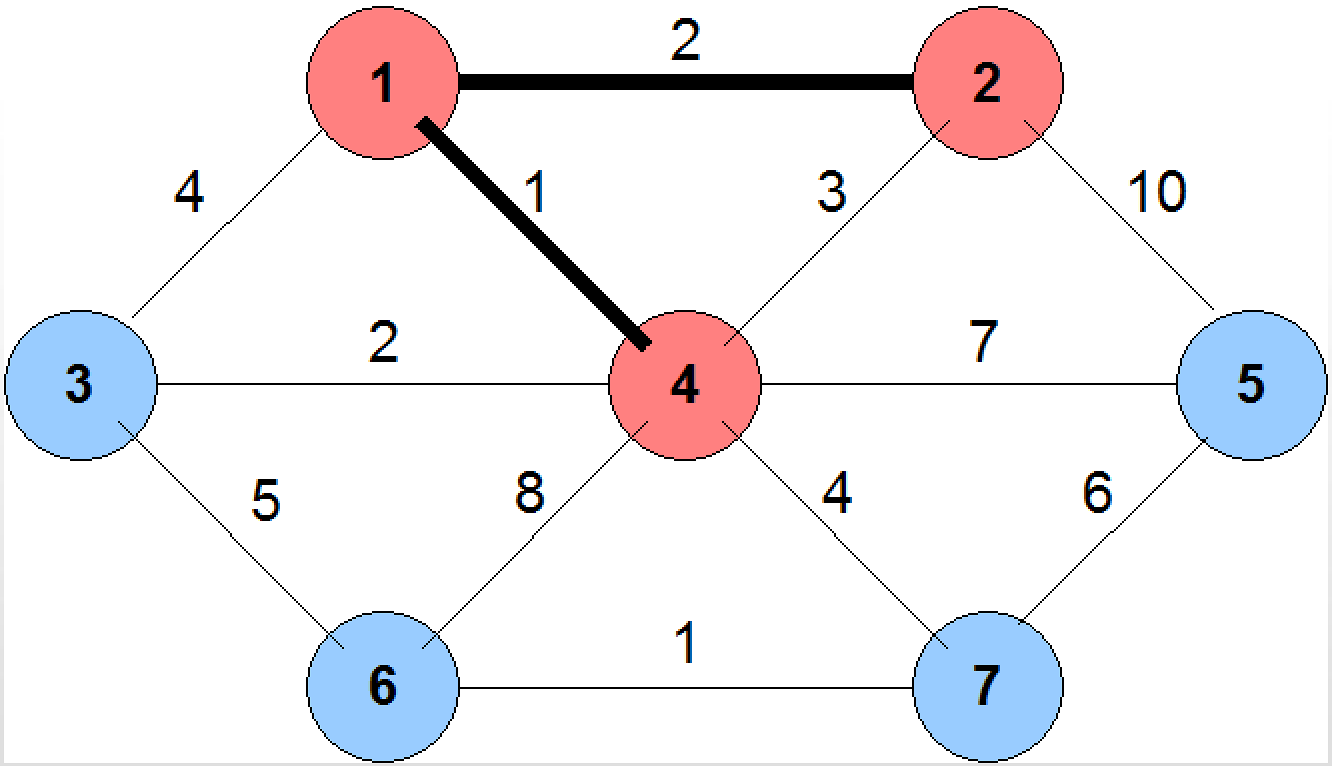
Prim's Algorithm



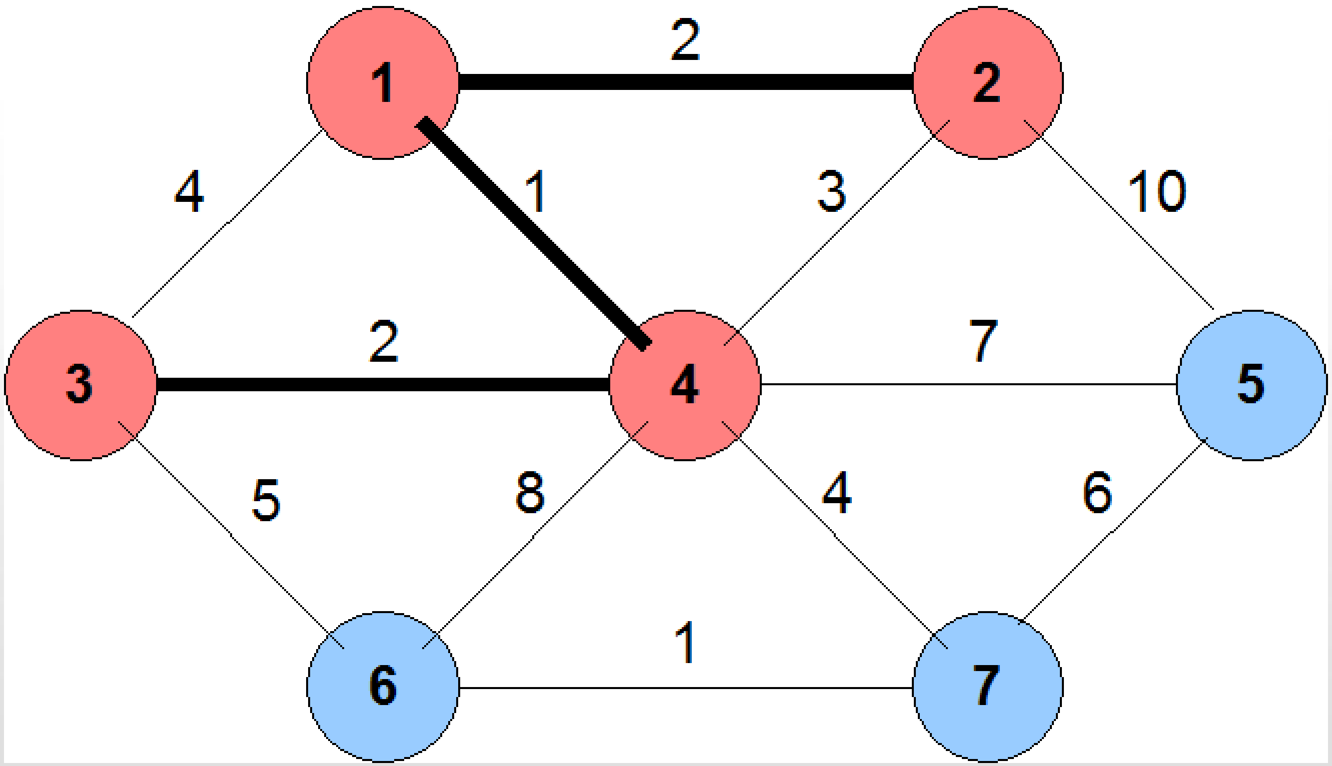
Prim's Algorithm



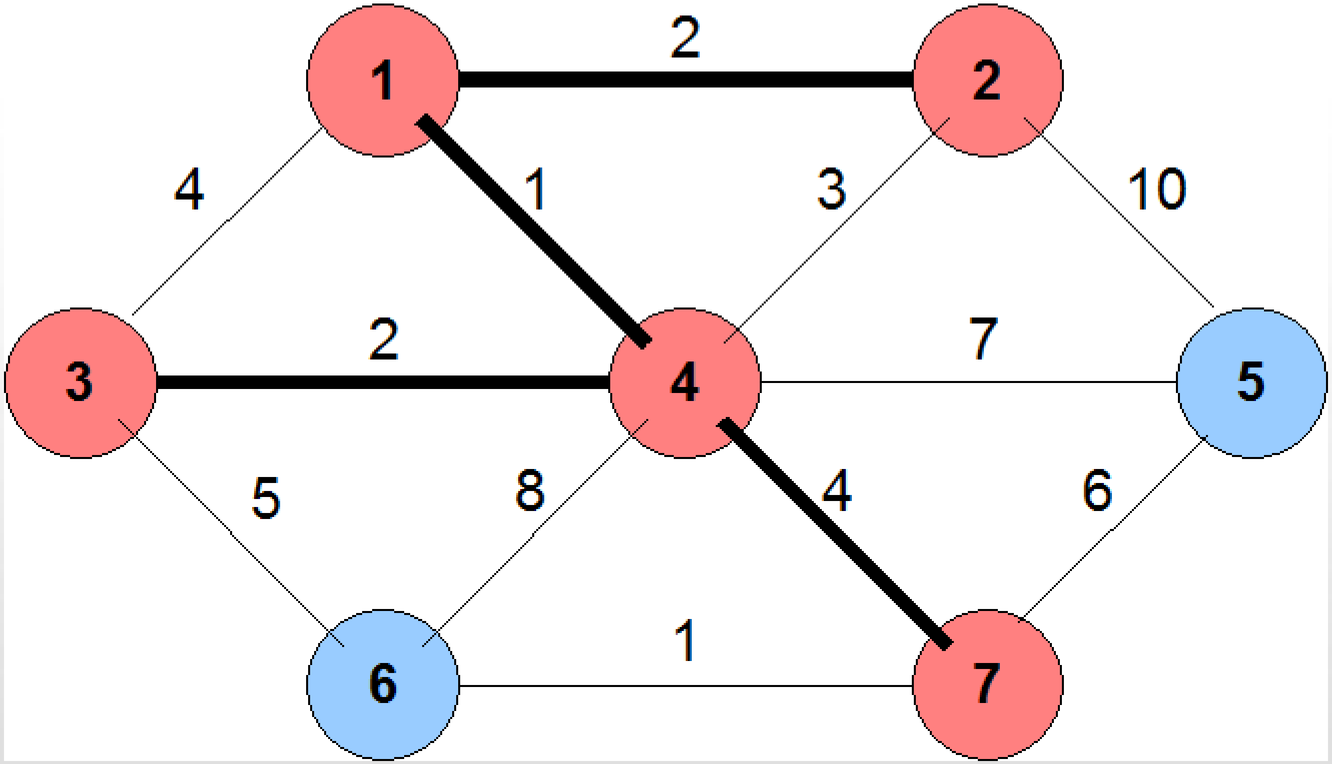
Prim's Algorithm



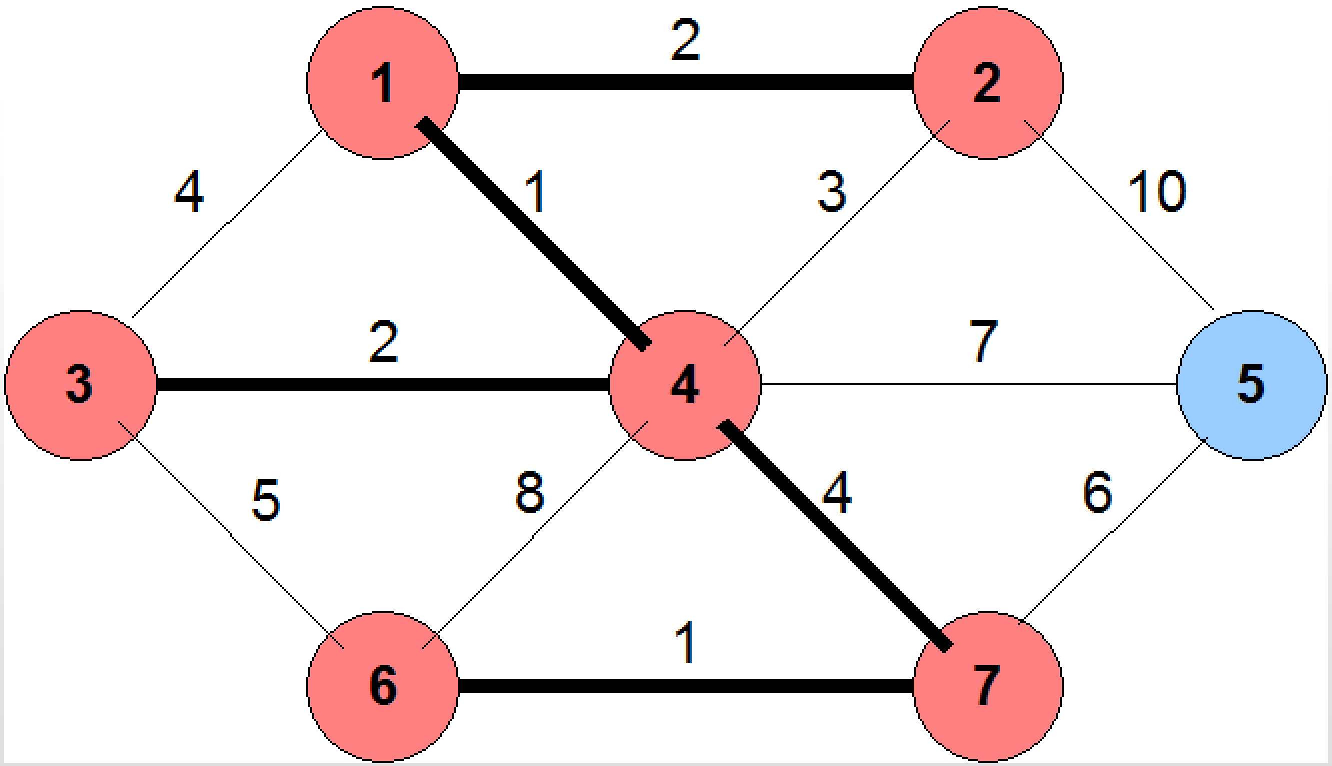
Prim's Algorithm



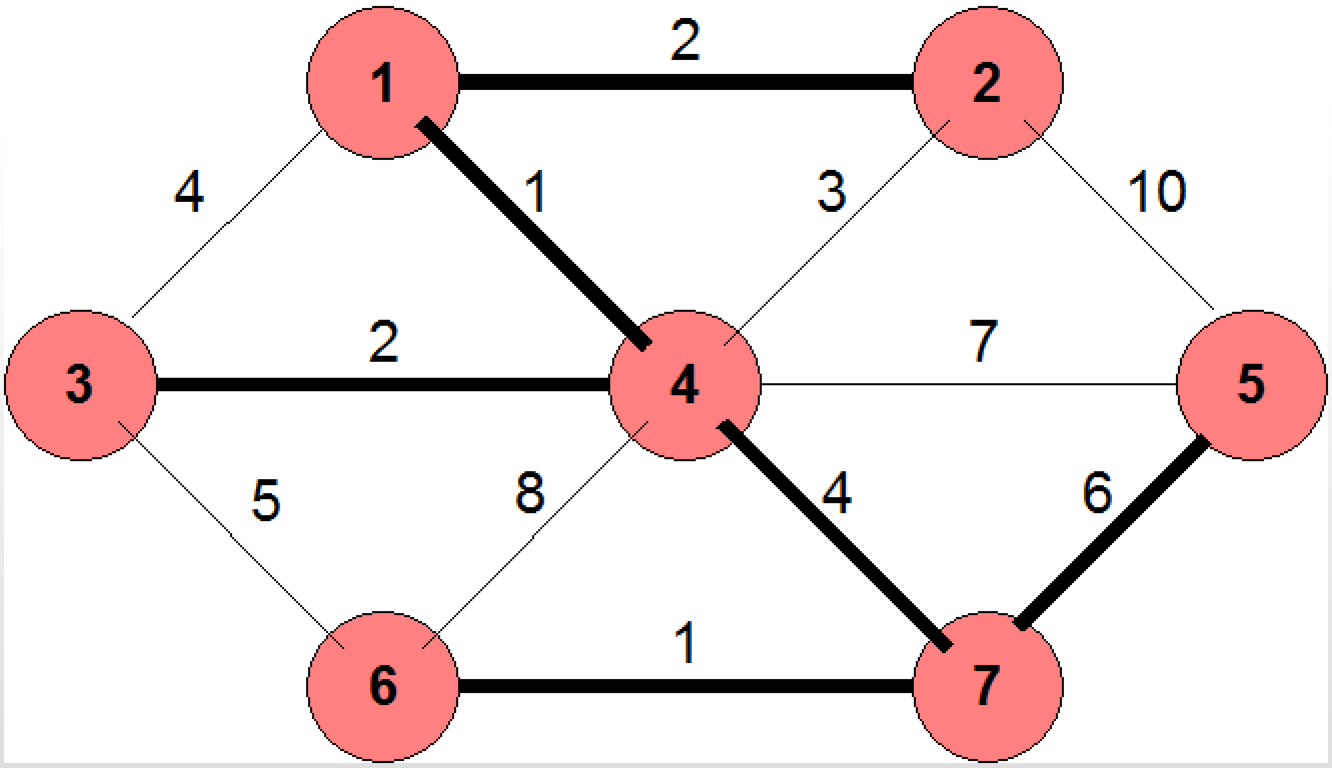
Prim's Algorithm



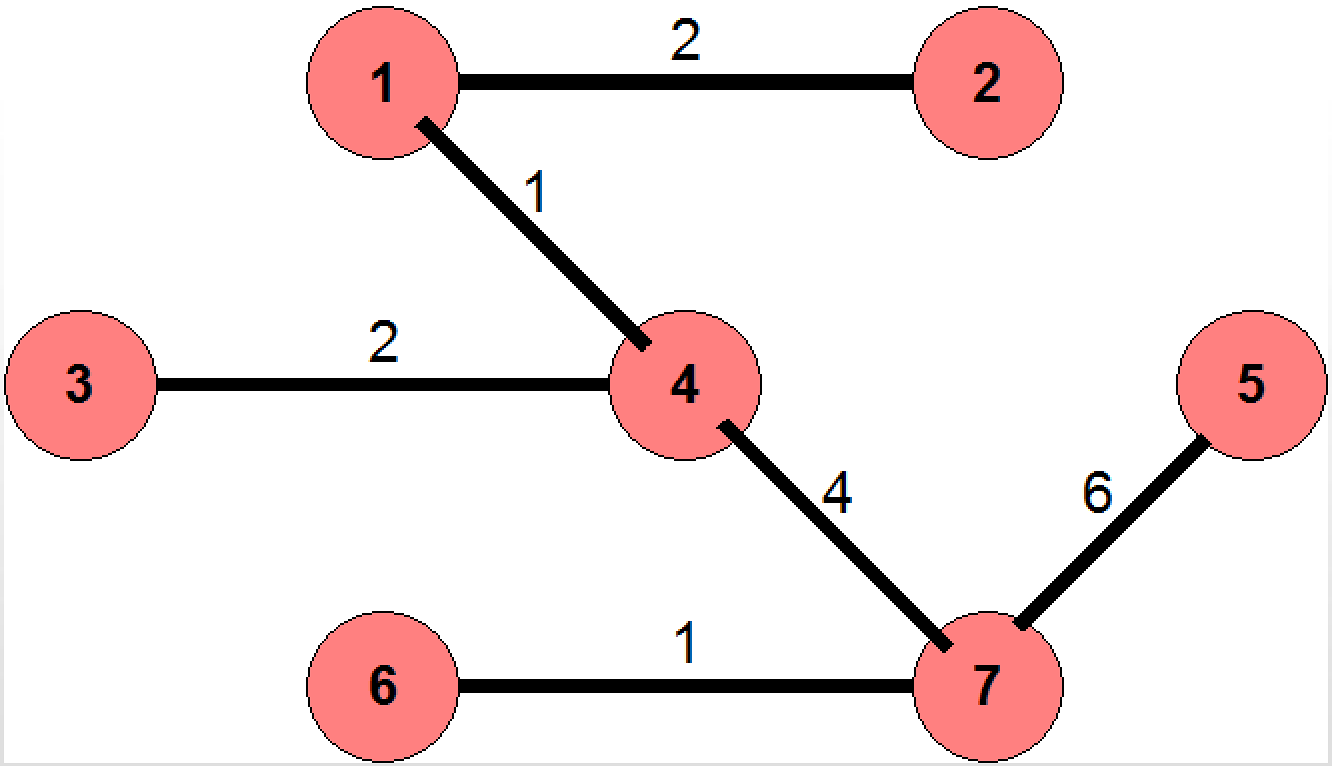
Prim's Algorithm



Prim's Algorithm



Prim's Algorithm



Prim's Algorithm

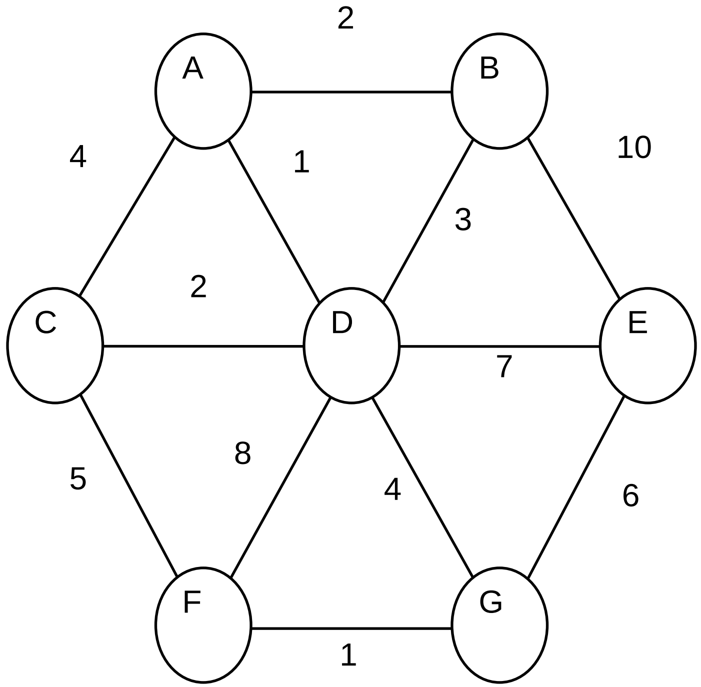
Outline

- *Step 1:* Initialize the set U to {1}.
- *Step 2:* Add to U the vertex $v \in (V-U)$ such that the edge (u,v) is the shortest or cheapest edge where $u \in U$.
- *Step 3:* Repeat step 2 until U=V.

Prim's Algorithm

```
Prims(G,c,d,p)
 begin
     for each vertex v ∈ V do
           d[v] = ∞
           p[v] = 0
           d[s] = 0
           S = ∅
           Q = V
     while Q ≠ ∅ do
           u = vertex x in Q such that d[x] is minimum
           S = S ∪ {u}
           Q = Q - {u}
        for each vertex v in Q and v adjacent to u do
            d[v] = min( d[v], c[u,v] )
           if c[u,v] is used
             then p[v] = u
end
```

Example



Matrix c[u,v] of edge weights

	A	B	C	D	E	F	G
A	0	2	4	1	∞	∞	∞
B	2	0	∞	3	10	∞	∞
C	4	∞	0	2	∞	5	∞
D	1	3	2	0	7	8	4
E	∞	10	∞	7	0	∞	6
F	∞	∞	5	8	∞	0	1
G	∞	∞	∞	4	6	1	0

Initial Configuration

for each vertex v $\in$ V do

$d[v] = \infty$

$p[v] = 0$

$d[s] = 0$

$S = \emptyset$

$Q = V$

$S = \emptyset$

$Q = \{A,B,C,D,E,F,G\}$

v	In S?	d[v]	p[v]
A		0	0
B		∞	0
C		∞	0
D		∞	0
E		∞	0
F		∞	0
G		∞	0

Start of 1st iteration

while Q $\neq \emptyset$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

if c[u,v] is used

then p[v] = u

S = $\emptyset$

Q = {A,B,C,D,E,F,G}

v	In S?	d[v]	p[v]
A		0	0
B		∞	0
C		∞	0
D		∞	0
E		∞	0
F		∞	0
G		∞	0

u=A

while $Q \neq \varnothing$ do

u = vertex x in Q such that d[x] is minimum

$S = S \cup \{u\}$

$Q = Q - \{u\}$

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

if c[u,v] is used

then p[v] = u

$S = \varnothing$

Q = {A,B,C,D,E,F,G}

v	In S?	d[v]	p[v]
A		0	0
B		∞	0
C		∞	0
D		∞	0
E		∞	0
F		∞	0
G		∞	0

u=A

while Q $\neq \emptyset$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

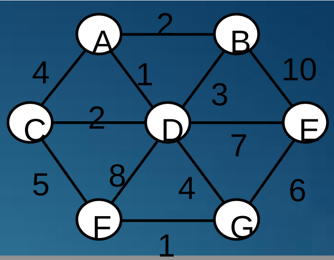
if c[u,v] is used

then p[v] = u

S = {A}
Q = {B,C,D,E,F,G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		∞	0
C		∞	0
D		∞	0
E		∞	0
F		∞	0
G		∞	0

u=A ; v={B,C,D}



while Q $\neq \emptyset$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

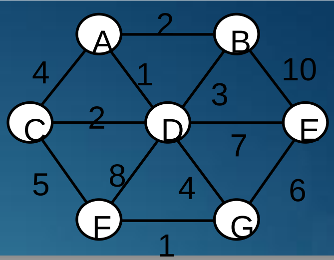
if c[u,v] is used

then p[v] = u

S = {A}
Q = {B,C,D,E,F,G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		∞	0
C		∞	0
D		∞	0
E		∞	0
F		∞	0
G		∞	0

u=A ; v={B,C,D}



while Q $\neq \emptyset$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

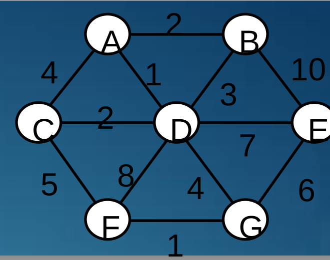
if c[u,v] is used

then p[v] = u

S = {A}
Q = {B,C,D,E,F,G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		∞	0
C		∞	0
D		∞	0
E		∞	0
F		∞	0
G		∞	0

u=A ; v={B,C,D}



while Q $\neq \varnothing$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

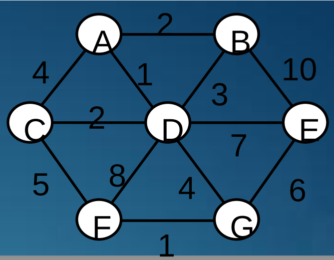
if c[u,v] is used

then p[v] = u

S = {A}
Q = {B,C,D,E,F,G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	0
C		∞	0
D		∞	0
E		∞	0
F		∞	0
G		∞	0

u=A ; v={B,C,D}



while Q $\neq \emptyset$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

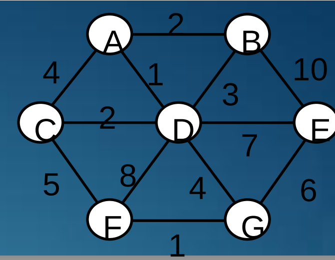
if c[u,v] is used

then p[v] = u

S = {A}
Q = {B,C,D,E,F,G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	0
C		∞	0
D		∞	0
E		∞	0
F		∞	0
G		∞	0

u=A ; v={B,C,D}



while Q $\neq \varnothing$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

if c[u,v] is used

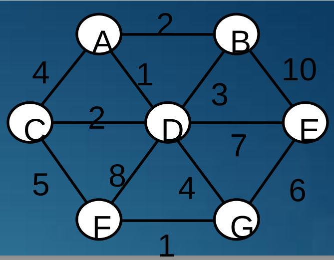
then p[v] = u

S = {A}

Q = {B,C,D,E,F,G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		∞	0
D		∞	0
E		∞	0
F		∞	0
G		∞	0

u=A ; v={B,C,D}



while Q $\neq \emptyset$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

if c[u,v] is used

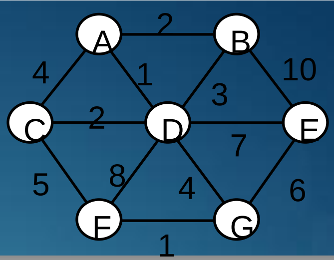
then p[v] = u

S = {A}

Q = {B,C,D,E,F,G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		∞	0
D		∞	0
E		∞	0
F		∞	0
G		∞	0

u=A ; v={B,C,D}



while Q $\neq \varnothing$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

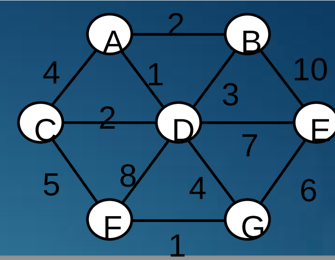
if c[u,v] is used

then p[v] = u

S = {A}
Q = {B,C,D,E,F,G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		∞	0
D		∞	0
E		∞	0
F		∞	0
G		∞	0

u=A ; v={B,C,D}



while Q $\neq \emptyset$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

if c[u,v] is used

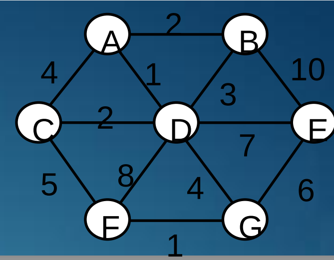
then p[v] = u

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		4	0
D		∞	0
E		∞	0
F		∞	0
G		∞	0

S = {A}

Q = {B,C,D,E,F,G}

u=A ; v={B,C,D}



while Q $\neq \emptyset$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

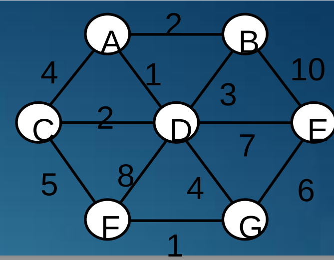
if c[u,v] is used

then p[v] = u

S = {A}
Q = {B,C,D,E,F,G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		4	0
D		∞	0
E		∞	0
F		∞	0
G		∞	0

u=A ; v={B,C,D}



while Q $\neq \emptyset$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

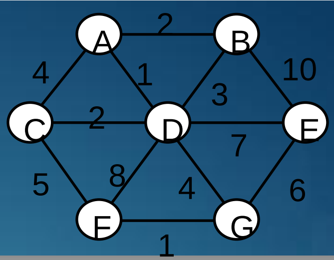
if c[u,v] is used

then p[v] = u

S = {A}
Q = {B,C,D,E,F,G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		4	A
D		∞	0
E		∞	0
F		∞	0
G		∞	0

u=A ; v={B,C,D}



while Q $\neq \emptyset$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

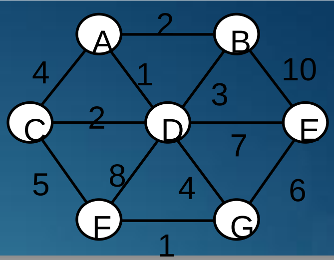
if c[u,v] is used

then p[v] = u

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		4	A
D		∞	0
E		∞	0
F		∞	0
G		∞	0

S = {A}
Q = {B,C,D,E,F,G}

u=A ; v={B,C,D}



while Q $\neq \varnothing$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

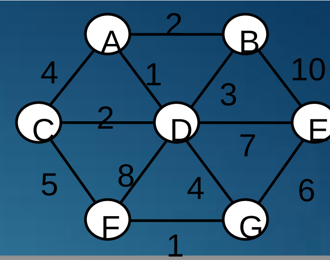
if c[u,v] is used

then p[v] = u

S = {A}
Q = {B,C,D,E,F,G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		4	A
D		∞	0
E		∞	0
F		∞	0
G		∞	0

u=A ; v={B,C,D}



while Q ≠ ∅ do

u = vertex x in Q such that d[x] is minimum

S = S ∪ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

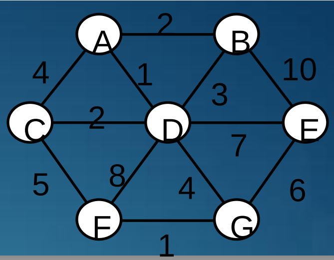
if c[u,v] is used

then p[v] = u

S = {A}
Q = {B,C,D,E,F,G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		4	A
D		1	0
E		∞	0
F		∞	0
G		∞	0

u=A ; v={B,C,D}



while Q ≠ ∅ do

u = vertex x in Q such that d[x] is minimum

S = S ∪ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

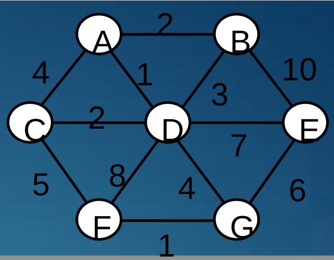
if c[u,v] is used

then p[v] = u

S = {A}
Q = {B,C,D,E,F,G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		4	A
D		1	0
E		∞	0
F		∞	0
G		∞	0

u=A ; v={B,C,D}



while Q $\neq \emptyset$ do
u = vertex x in Q such that d[x] is minimum
S = S $\cup$ {u}
Q = Q - {u}
for each vertex v in Q and v adjacent to u do
d[v] = min(d[v], c[u,v])
if c[u,v] is used
then p[v] = u

S = {A}
Q = {B,C,D,E,F,G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		4	A
D		1	A
E		∞	0
F		∞	0
G		∞	0

Table after 1st iteration

while Q $\neq \varnothing$ do
  u = vertex x in Q such that d[x] is minimum
    S = S $\cup$ {u}
    Q = Q - {u}
  for each vertex v in Q and v adjacent to u do
    d[v] = min(d[v], c[u,v])
    if c[u,v] is used
      then p[v] = u

S = {A}
Q = {B,C,D,E,F,G}
U = A

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		4	A
D		1	A
E		∞	0
F		∞	0
G		∞	0

Start of 2nd iteration

while Q $\neq \varnothing$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

if c[u,v] is used

then p[v] = u

S = {A}
Q = {B,C,D,E,F,G}
U = A

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		4	A
D		1	A
E		∞	0
F		∞	0
G		∞	0

u=D

while Q $\neq \emptyset$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

if c[u,v] is used

then p[v] = u

S = {A}
Q = {B,C,D,E,F,G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		4	A
D		1	A
E		∞	0
F		∞	0
G		∞	0

u=D

while Q $\neq \varnothing$ do

  u = vertex x in Q such that d[x]   is minimum

    S = S $\cup$ {u}

    Q = Q - {u}

  for each vertex v in Q and v adjacent to u do

    d[v] = min(d[v], c[u,v])

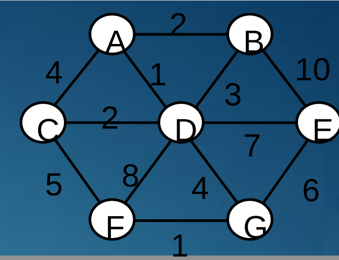
    if c[u,v] is used

     then p[v] = u

S = {A, D}
Q = {B, C, E, F, G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		4	A
D	✓	1	A
E		∞	0
F		∞	0
G		∞	0

U=D ; v={B,C,E,F,G}



while Q $\neq \emptyset$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

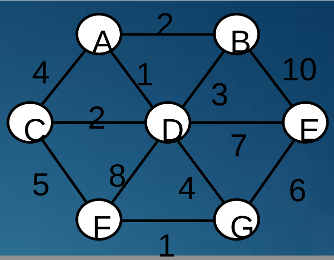
if c[u,v] is used

then p[v] = u

S = {A, D}
Q = {B, C, E, F, G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		4	A
D	✓	1	A
E		∞	0
F		∞	0
G		∞	0

U=D ; v={B,C,E,F,G}



while Q ≠ ∅ do

u = vertex x in Q such that d[x] is minimum

S = S ∪ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

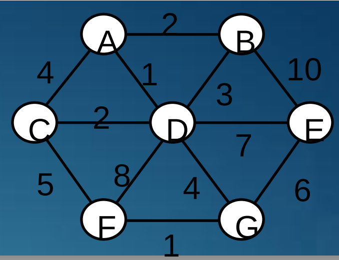
if c[u,v] is used

then p[v] = u

S = {A, D}
Q = {B, C, E, F, G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		4	A
D	✓	1	A
E		∞	0
F		∞	0
G		∞	0

U=D ; v={B,C,E,F,G}



while Q ≠ ∅ do

u = vertex x in Q such that d[x] is minimum

S = S ∪ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

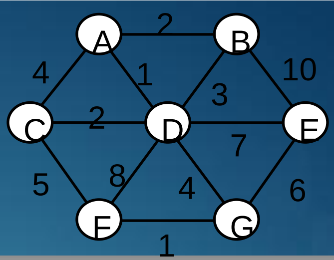
if c[u,v] is used

then p[v] = u

S = {A, D}
Q = {B, C, E, F, G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		4	A
D	✓	1	A
E		∞	0
F		∞	0
G		∞	0

U=D ; v={B,C,E,F,G}



while $Q \neq \varnothing$ do

u = vertex x in Q such that d[x] is minimum

$S = S \cup \{u\}$

$Q = Q - \{u\}$

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

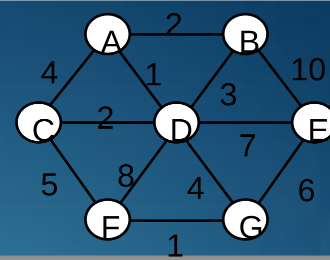
if c[u,v] is used

then p[v] = u

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		4	A
D	✓	1	A
E		∞	0
F		∞	0
G		∞	0

S = {A, D}
Q = {B, C, E, F, G}

U=D ; v={B,C,E,F,G}



while Q ≠ ∅ do

u = vertex x in Q such that d[x] is minimum

S = S ∪ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

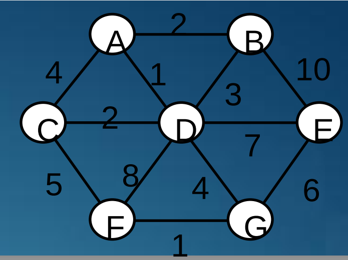
if c[u,v] is used

then p[v] = u

S = {A, D}
Q = {B, C, E, F, G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		4	A
D	✓	1	A
E		∞	0
F		∞	0
G		∞	0

U=D ; v={B,C,E,F,G}



while Q $\neq \varnothing$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

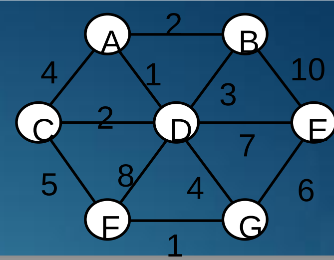
if c[u,v] is used

then p[v] = u

S = {A, D}
Q = {B, C, E, F, G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		2	A
D	✓	1	A
E		∞	0
F		∞	0
G		∞	0

U=D ; v={B,C,E,F,G}



while $Q \neq \emptyset$ do

u = vertex x in Q such that d[x] is minimum

$S = S \cup \{u\}$

$Q = Q - \{u\}$

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

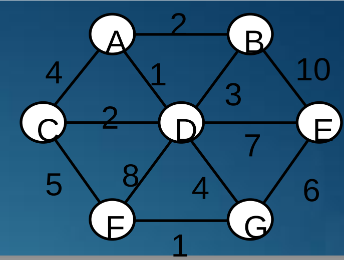
if c[u,v] is used

then p[v] = u

S = {A, D}
Q = {B, C, E, F, G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		2	A
D	✓	1	A
E		∞	0
F		∞	0
G		∞	0

U=D ; v={B,C,E,F,G}



while Q $\neq \varnothing$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

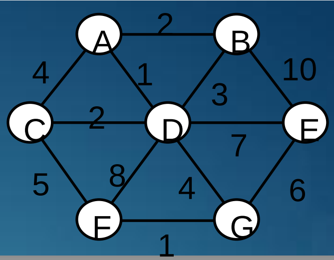
if c[u,v] is used

then p[v] = u

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		2	D
D	✓	1	A
E		∞	0
F		∞	0
G		∞	0

S = {A, D}
Q = {B, C, E, F, G}

U=D ; v={B,C,E,F,G}



while Q $\neq \emptyset$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

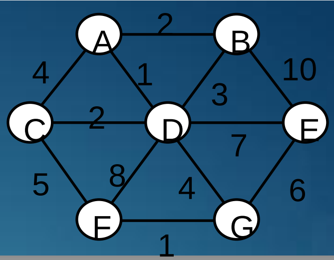
if c[u,v] is used

then p[v] = u

S = {A, D}
Q = {B, C, E, F, G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		2	D
D	✓	1	A
E		∞	0
F		∞	0
G		∞	0

U=D ; v={B,C,E,F,G}



while Q $\neq \varnothing$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

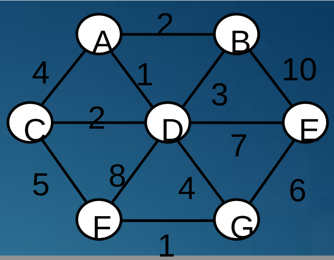
if c[u,v] is used

then p[v] = u

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		2	D
D	✓	1	A
E		∞	0
F		∞	0
G		∞	0

S = {A, D}
Q = {B, C, E, F, G}

U=D ; v={B,C,E,F,G}



while Q $\neq \varnothing$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

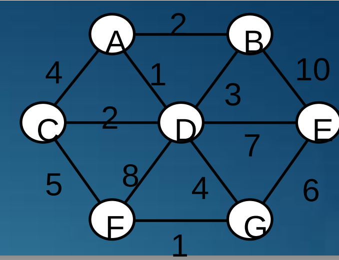
if c[u,v] is used

then p[v] = u

S = {A, D}
Q = {B, C, E, F, G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		2	D
D	✓	1	A
E		7	0
F		∞	0
G		∞	0

U=D ; v={B,C,E,F,G}



while Q $\neq \varnothing$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

if c[u,v] is used

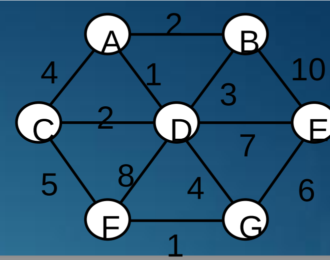
then p[v] = u

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		2	D
D	✓	1	A
E		7	0
F		∞	0
G		∞	0

S = {A, D}

Q = {B, C, E, F, G}

U=D ; v={B,C,E,F,G}

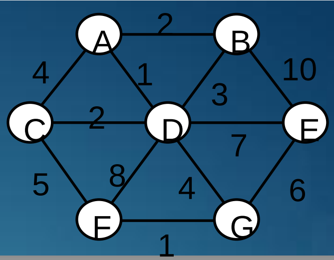


while Q $\neq \emptyset$ do
  u = vertex x in Q such that d[x] is minimum
    S = S $\cup$ {u}
    Q = Q - {u}
  for each vertex v in Q and v adjacent to u do
    d[v] = min(d[v], c[u,v])
    if c[u,v] is used
     then p[v] = u

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		2	D
D	✓	1	A
E		7	D
F		∞	0
G		∞	0

S = {A, D}
Q = {B, C, E, F, G}

U=D ; v={B,C,E,F,G}



while Q $\neq \varnothing$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

if c[u,v] is used

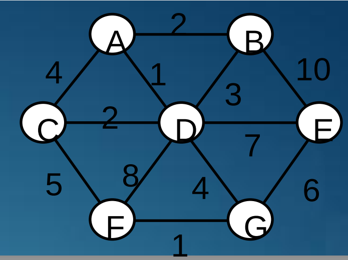
then p[v] = u

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		2	D
D	✓	1	A
E		7	D
F		∞	0
G		∞	0

S = {A, D}

Q = {B, C, E, F, G}

U=D ; v={B,C,E,F,G}



while Q $\neq \emptyset$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

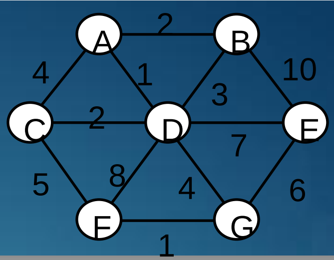
if c[u,v] is used

then p[v] = u

S = {A, D}
Q = {B, C, E, F, G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		2	D
D	✓	1	A
E		7	D
F		∞	0
G		∞	0

U=D ; v={B,C,E,F,G}

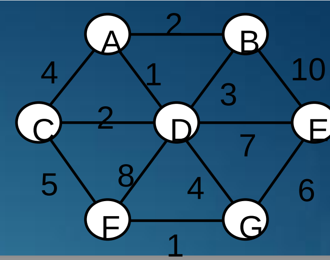


while $Q \neq \varnothing$ do
  u = vertex x in Q such that d[x] is minimum
    $S = S \cup \{u\}$
    $Q = Q - \{u\}$
  for each vertex v in Q and v adjacent to u do
    d[v] = min(d[v], c[u,v])
    if c[u,v] is used
      then p[v] = u

S = {A, D}
Q = {B, C, E, F, G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		2	D
D	✓	1	A
E		7	D
F		8	0
G		∞	0

U=D ; v={B,C,E,F,G}



while $Q \neq \varnothing$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

if c[u,v] is used

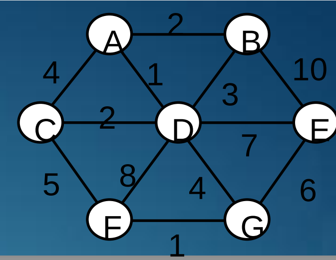
then p[v] = u

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		2	D
D	✓	1	A
E		7	D
F		8	0
G		∞	0

S = {A, D}

Q = {B, C, E, F, G}

U=D ; v={B,C,E,F,G}



while Q ≠ ∅ do

u = vertex x in Q such that d[x] is minimum

S = S ∪ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

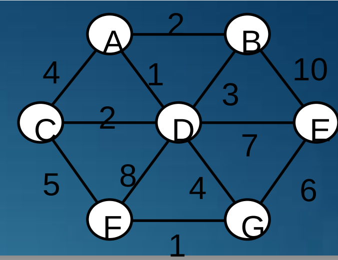
if c[u,v] is used

then p[v] = u

S = {A, D}
Q = {B, C, E, F, G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		2	D
D	✓	1	A
E		7	D
F		8	D
G		∞	0

U=D ; v={B,C,E,F,G}



while Q $\neq \varnothing$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

if c[u,v] is used

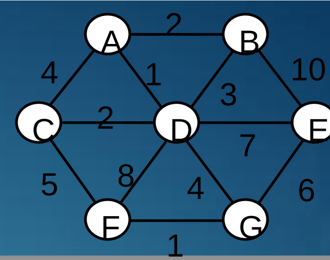
then p[v] = u

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		2	D
D	✓	1	A
E		7	D
F		8	D
G		∞	0

S = {A, D}

Q = {B, C, E, F, G}

U=D ; v={B,C,E,F,G}



while $Q \neq \varnothing$ do

u = vertex x in Q such that d[x] is minimum

$S = S \cup \{u\}$

$Q = Q - \{u\}$

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

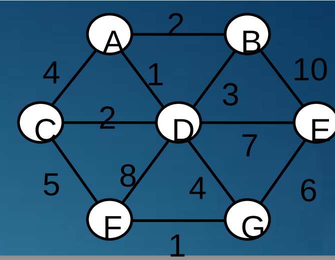
if c[u,v] is used

then p[v] = u

S = {A, D}
Q = {B, C, E, F, G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		2	D
D	✓	1	A
E		7	D
F		8	D
G		∞	0

U=D ; v={B,C,E,F,G}



while Q $\neq \varnothing$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

if c[u,v] is used

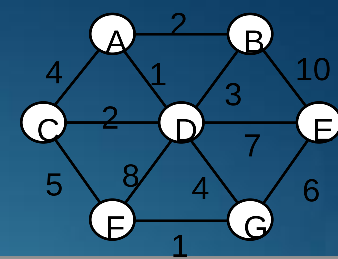
then p[v] = u

S = {A, D}

Q = {B, C, E, F, G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		2	D
D	✓	1	A
E		7	D
F		8	D
G		4	0

U=D ; v={B,C,E,F,G}



while Q ≠ ∅ do

u = vertex x in Q such that d[x] is minimum

S = S ∪ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

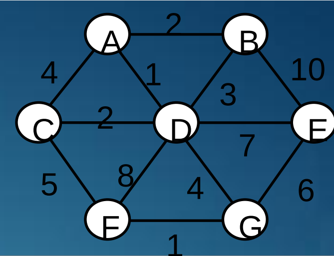
if c[u,v] is used

then p[v] = u

S = {A, D}
Q = {B, C, E, F, G}

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		2	D
D	✓	1	A
E		7	D
F		8	D
G		4	0

U=D ; v={B,C,E,F,G}



while Q ≠ ∅ do

u = vertex x in Q such that d[x] is minimum

S = S ∪ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

if c[u,v] is used

then p[v] = u

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		2	D
D	✓	1	A
E		7	D
F		8	D
G		4	D

S = {A, D}

Q = {B, C, E, F, G}

Table after 2nd iteration

while $Q \neq \varnothing$ do

u = vertex x in Q such that d[x] is minimum

$S = S \cup \{u\}$

$Q = Q - \{u\}$

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

if c[u,v] is used

then p[v] = u

S = {A, D}
Q = {B, C, E, F, G}
U = D

v	In S?	d[v]	p[v]
A	✓	0	0
B		2	A
C		2	D
D	✓	1	A
E		7	D
F		8	D
G		4	D

Table after 3rd iteration

while $Q \neq \emptyset$ do

u = vertex x in Q such that d[x] is minimum

$S = S \cup \{u\}$

$Q = Q - \{u\}$

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

if c[u,v] is used

then p[v] = u

S = {A, D, B}
Q = {C, E, F, G}
U = B

v	In S?	d[v]	p[v]
A	✓	0	0
B	✓	2	A
C		2	D
D	✓	1	A
E		7	D
F		8	D
G		4	D

Table after 4^{th} iteration

while Q $\neq \emptyset$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

if c[u,v] is used

then p[v] = u

S = {A, B, D, C}
Q = {E, F, G}
U = C

v	In S?	d[v]	p[v]
A	✓	0	0
B	✓	2	A
C	✓	2	D
D	✓	1	A
E		7	D
F		5	C
G		4	D

Table after 5th iteration

while Q $\neq \varnothing$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

if c[u,v] is used

then p[v] = u

S = {A, B, D, C, G}
Q = { E, F}
U = G

v	In S?	d[v]	p[v]
A	✓	0	0
B	✓	2	A
C	✓	2	D
D	✓	1	A
E		6	G
F		1	G
G	✓	4	D

Table after 6th iteration

while $Q \neq \emptyset$ do

u = vertex x in Q such that d[x] is minimum

$S = S \cup \{u\}$

$Q = Q - \{u\}$

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

if c[u,v] is used

then p[v] = u

S = {A, B, D, C, G,F}
Q = {E}
U = F

v	In S?	d[v]	p[v]
A	✓	0	0
B	✓	2	A
C	✓	2	D
D	✓	1	A
E		6	G
F	✓	1	G
G	✓	4	D

Table after 7th iteration (FINAL tree)

while Q $\neq \emptyset$ do

u = vertex x in Q such that d[x] is minimum

S = S $\cup$ {u}

Q = Q - {u}

for each vertex v in Q and v adjacent to u do

d[v] = min(d[v], c[u,v])

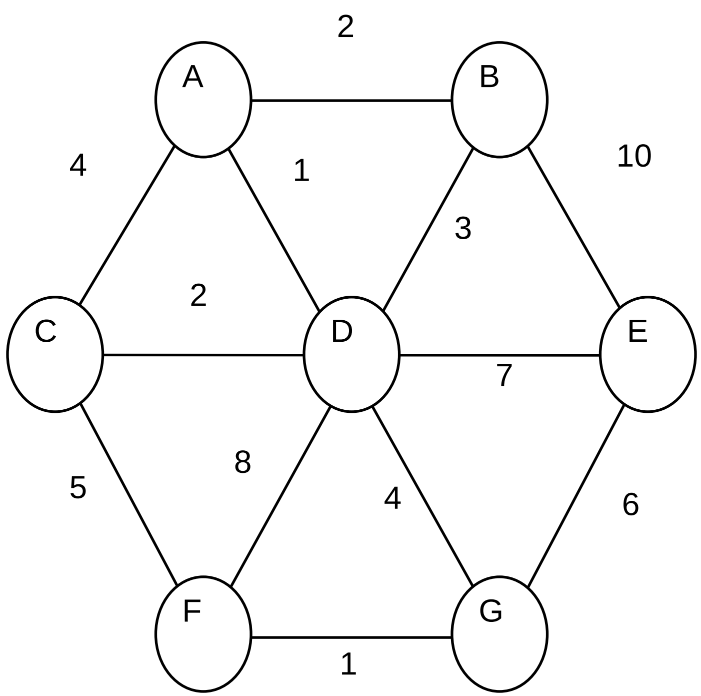
if c[u,v] is used

then p[v] = u

S = {A,B,D,C,G,F,E}
Q = {}
U = E

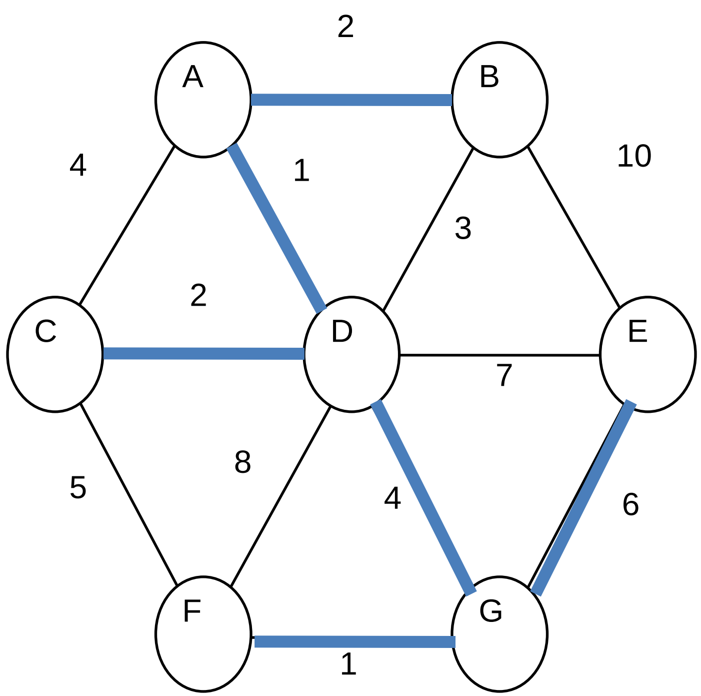
v	In S?	d[v]	p[v]
A	✓	0	0
B	✓	2	A
C	✓	2	D
D	✓	1	A
E	✓	6	G
F	✓	1	G
G	✓	4	D

Example



v	In S?	d[v]	p[v]
A	✓	0	0
B	✓	2	A
C	✓	2	D
D	✓	1	A
E	✓	6	G
F	✓	1	G
G	✓	4	D

Example



v	In S?	d[v]	p[v]
A	✓	0	0
B	✓	2	A
C	✓	2	D
D	✓	1	A
E	✓	6	G
F	✓	1	G
G	✓	4	D

Travelling Salesman Problem - TSP

Hamiltonian Cycle – a simple cycle that includes all the vertices of the graph.

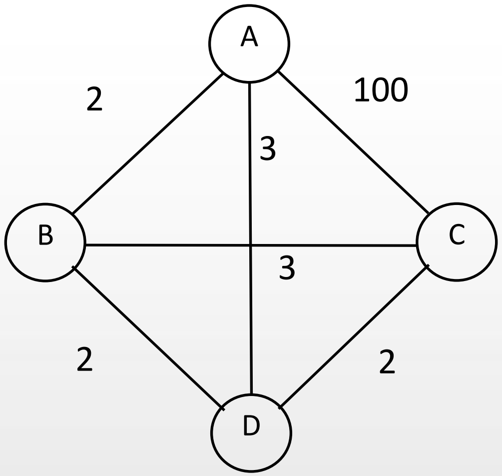
TSP – Given n cities, find the shortest roundrip route connecting them all with no city visited twice

– Find a hamiltonian cycle whose sum of edge-weights is the minimum.

Algorithm : Modified Kruskal's/Prim's Algorithm

- No new edge should form a cycle except for the last edge.
- No new edge should cause a vertex to have a degree more than two.

Traveling Salesman Problem - TSP



Greedy Algorithms will not find the optimal TSP tour in the sample graph on the left.

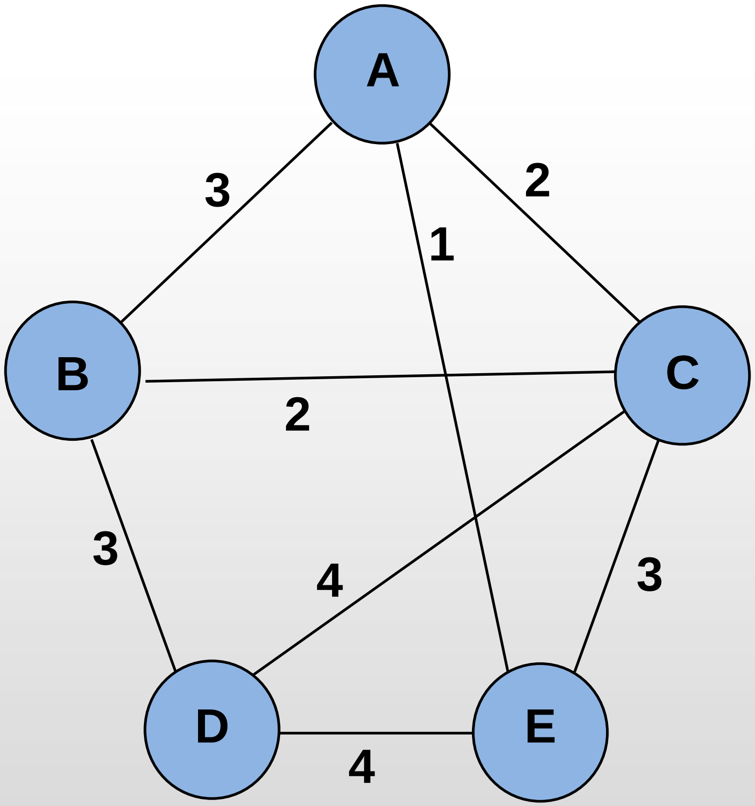
Shortest edge and nearest neighbor algorithms would choose ABDC then would be forced to select CA.

They won’t be able to find minimal TSP tours such as ADCBA or ABCDA.

The next step would be to generate all possible paths, to generate hamiltionian cycles and find a cycle with least sum of edge weights.

This leads us to the concept of backtracking…

Quiz - MST



Single Source Shortest Path

```
dijkstra_sssp(int source, int graphsize)  {
    int j;
    S={source};
    for (j=1;j<graphsize;j++) d[j]=cost[0][j];
    for (j=1;j<graphsize-1;j++) {                       /*O(n)*/
        choose a vertex w in V-S  such that
                d[w] is a minimum;                      /*O(n)*/
        insert(w,S);
        for each vertex v in V-S
                d[v]=min(d[v], d[w] + cost[w][v]);      /*O(n)*/
    }
}
```

/* $T(n) = O(n) * (O(n) + O(n)) = O(n^2)$ */

All Pairs Shortest Path

- ```
  dijkstra_apsp(int graphsize)      /* T(n) = O(n) * O(n^2) = O(n^3) */
        for(i=0; i<graphsize; i++)
              dijkstra_sssp(i , graphsize);
  ```

- ```
  floyd_apsp(int graphsize)         /*T(n) = O(n^2) + O(n^3) = O(n^3) */
  {
        int i,j,k;
        for(i=0;i<graphsize;i++)                                 /* O(n^2) */
          for(j=0;j<graphsize;j++) a[i][j]=cost[i][j];

        for(k=0;k<graphsize;k++)                                 /* O(n^3) */
          for(i=0;i<graphsize;i++)
            for(j=0;j<graphsize;j++)
              a[i][j]=min(a[i][j],a[i][k]+a[k][j]);
  }
  ```