CE 486
Urban Transportation Planning

Lec. 5
Shortest Path

Dr. Mahmoud Owais

Shortest Path Problems

Dijkstra's Algorithm

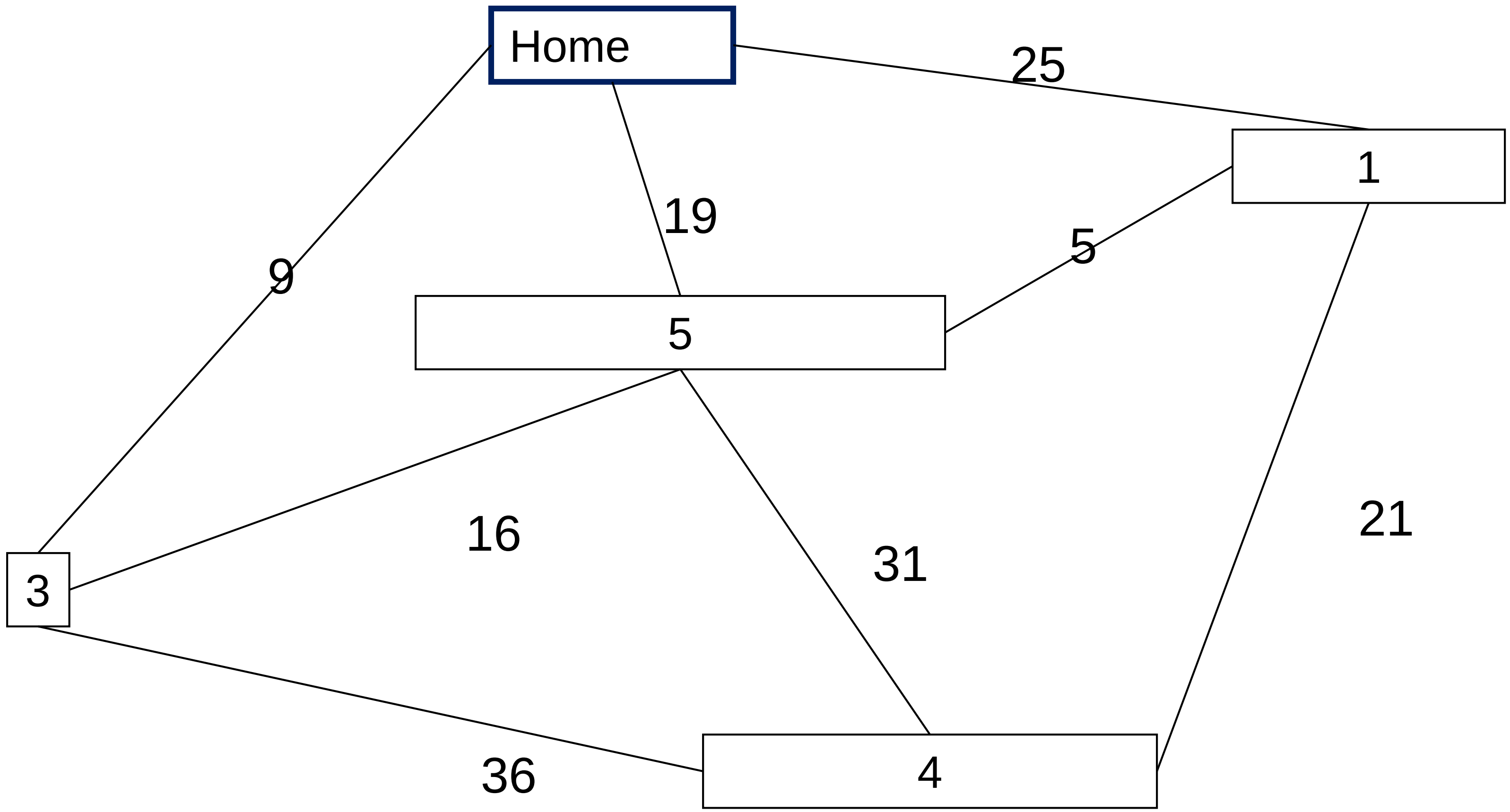
Introduction

- Many problems can be modeled using graphs with weights assigned to their edges:
  - Airline flight times
  - Telephone communication costs
  - Computer networks response times

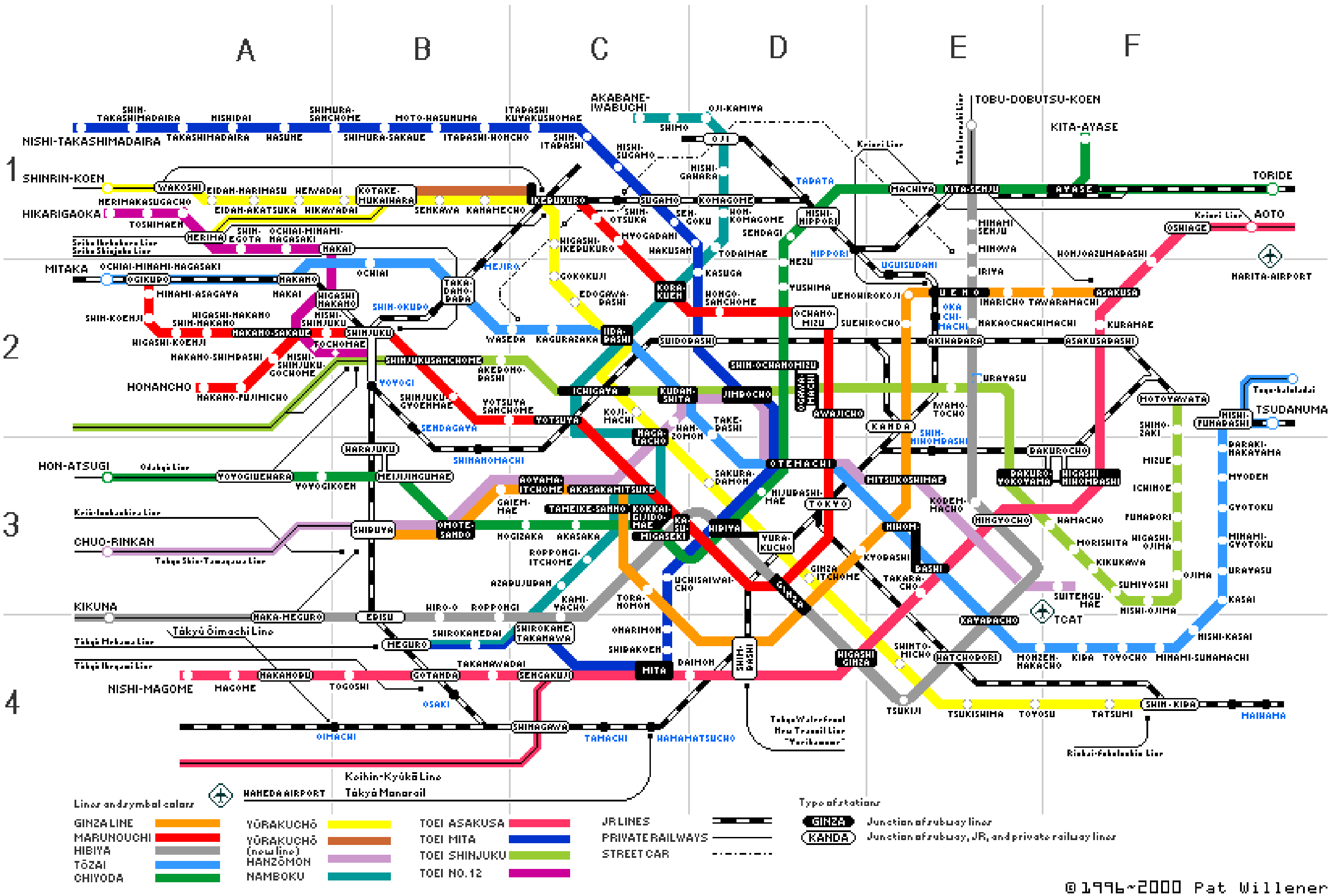
Where’s my motivation?

- Fastest way to get to school by car
- Finding the cheapest flight home

Optimal driving time



Tokyo Subway Map



Setup:

- G = weighted graph
- In our version, need POSITIVE weights.
- G is a simple connected graph.
  - A simple graph G = (V, E) consists of V, a nonempty set of vertices, and E, a set of unordered pairs of distinct elements of V called edges.
- A labeling procedure is carried out at each iteration
  - A vertex w is labeled with the length of the shortest path from a to w that contains only the vertices already in the distinguished set.

Outline of Algorithm

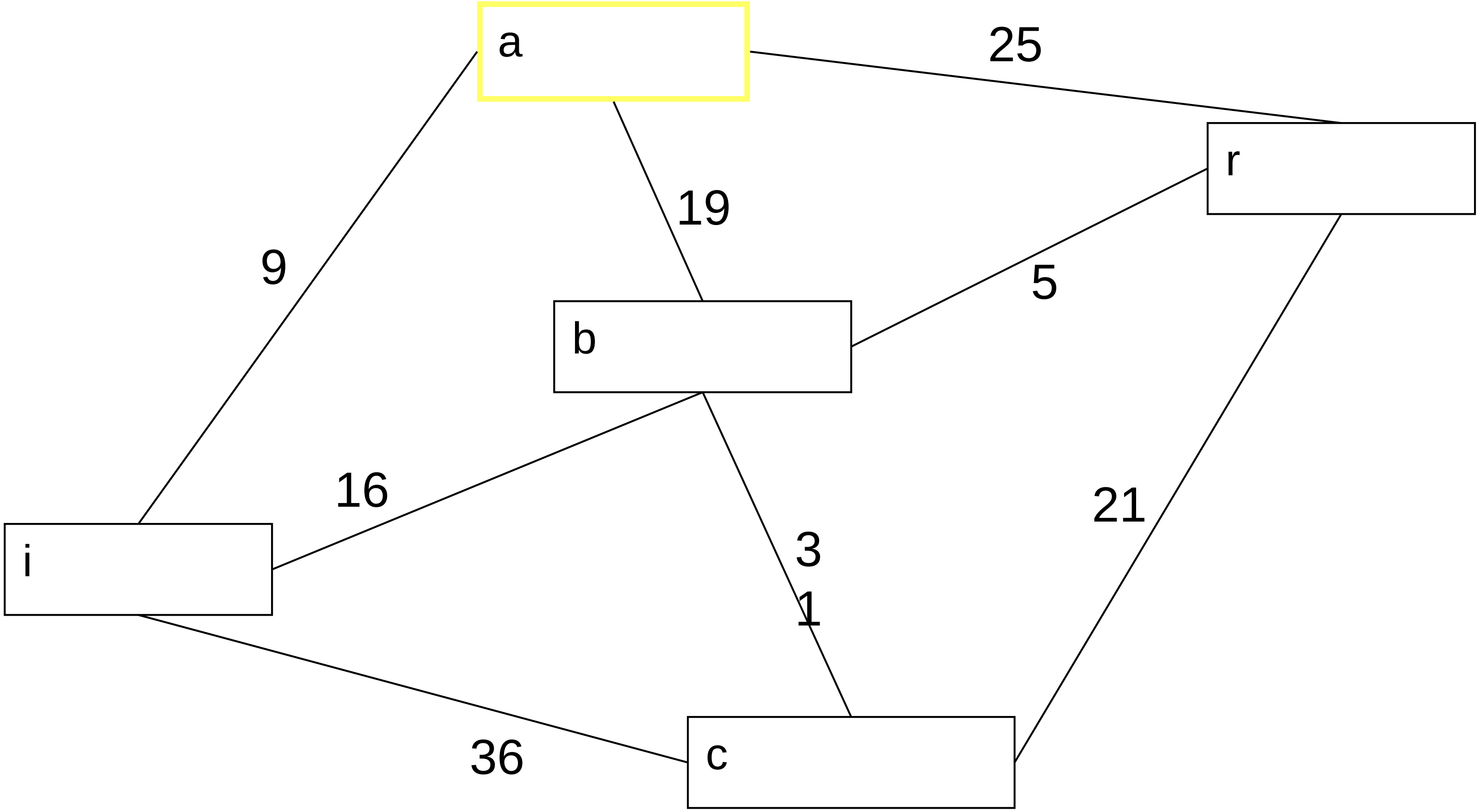
- Label a with 0 and all others with ∞. $L_0(a) = 0$ and $L_0(v) = \infty$
- Labels are shortest paths from a to vertices
- S_k = the distinguished set of vertices after k iterations. $S_0 = \emptyset$. The set S_k is formed by adding a vertex u NOT in S_{k-1} with the smallest label.
- Once u is added to S_k we update the labels of all the vertices not in S_k

To update labels:

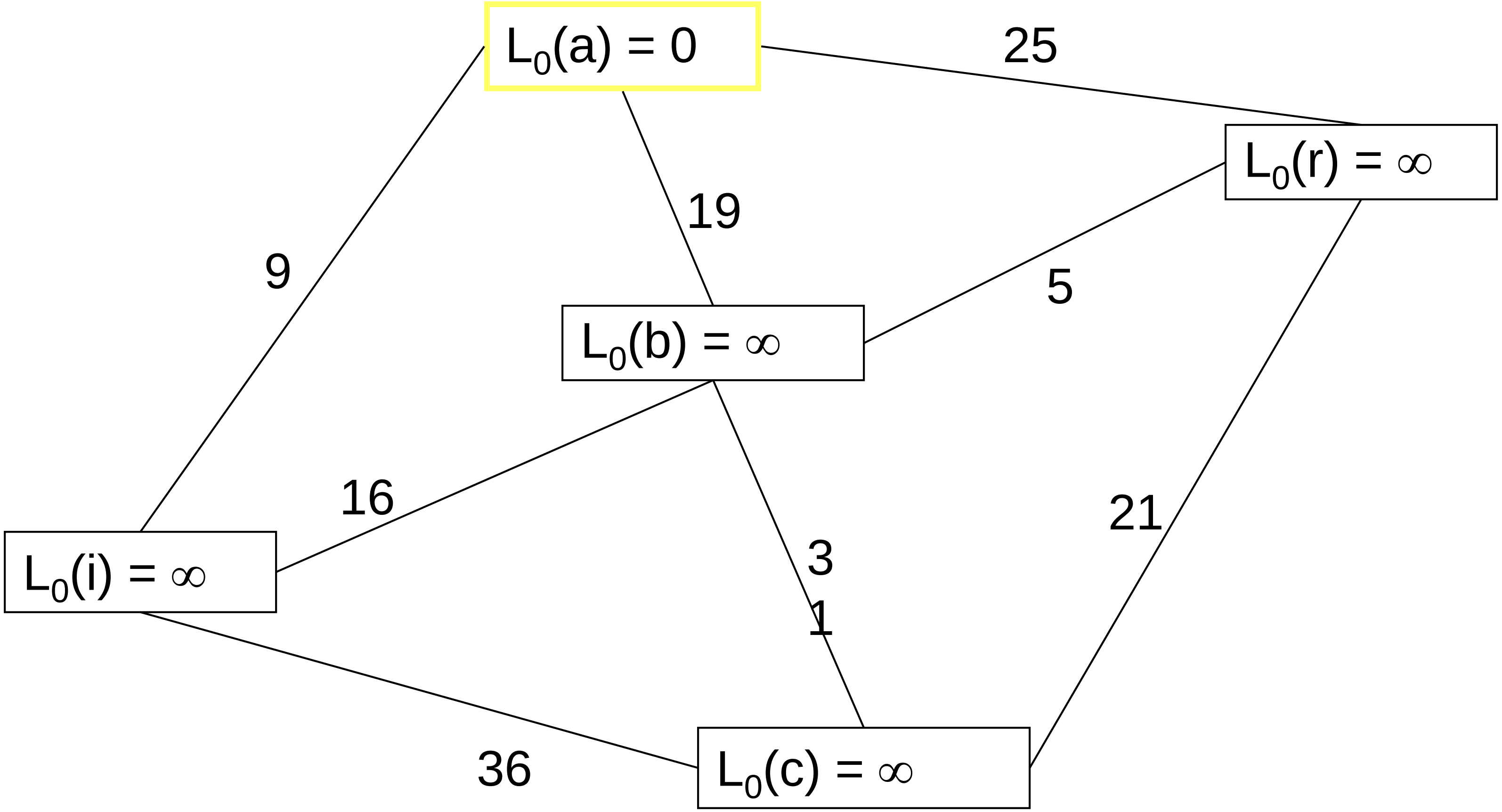
$L_k(a, v) = \min\{L_{k-1}(a, v),$

$L_{k-1}(a, u) + w(u, v)\}$

Using the previous example, we will find the shortest path from a to c.

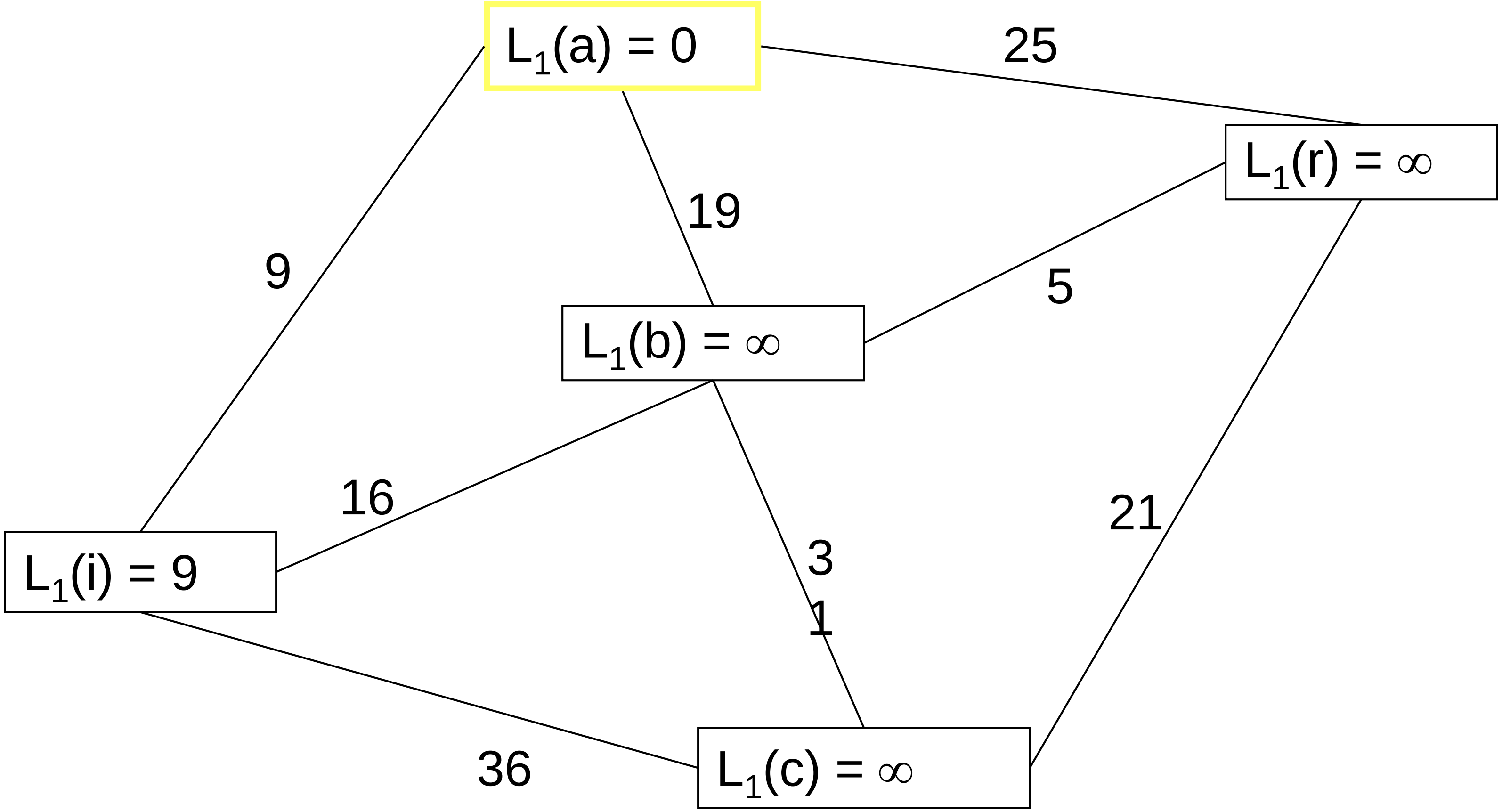


Label a with 0 and all others with ∞. $L_0(a) = 0$ and $L_0(v) = \infty$



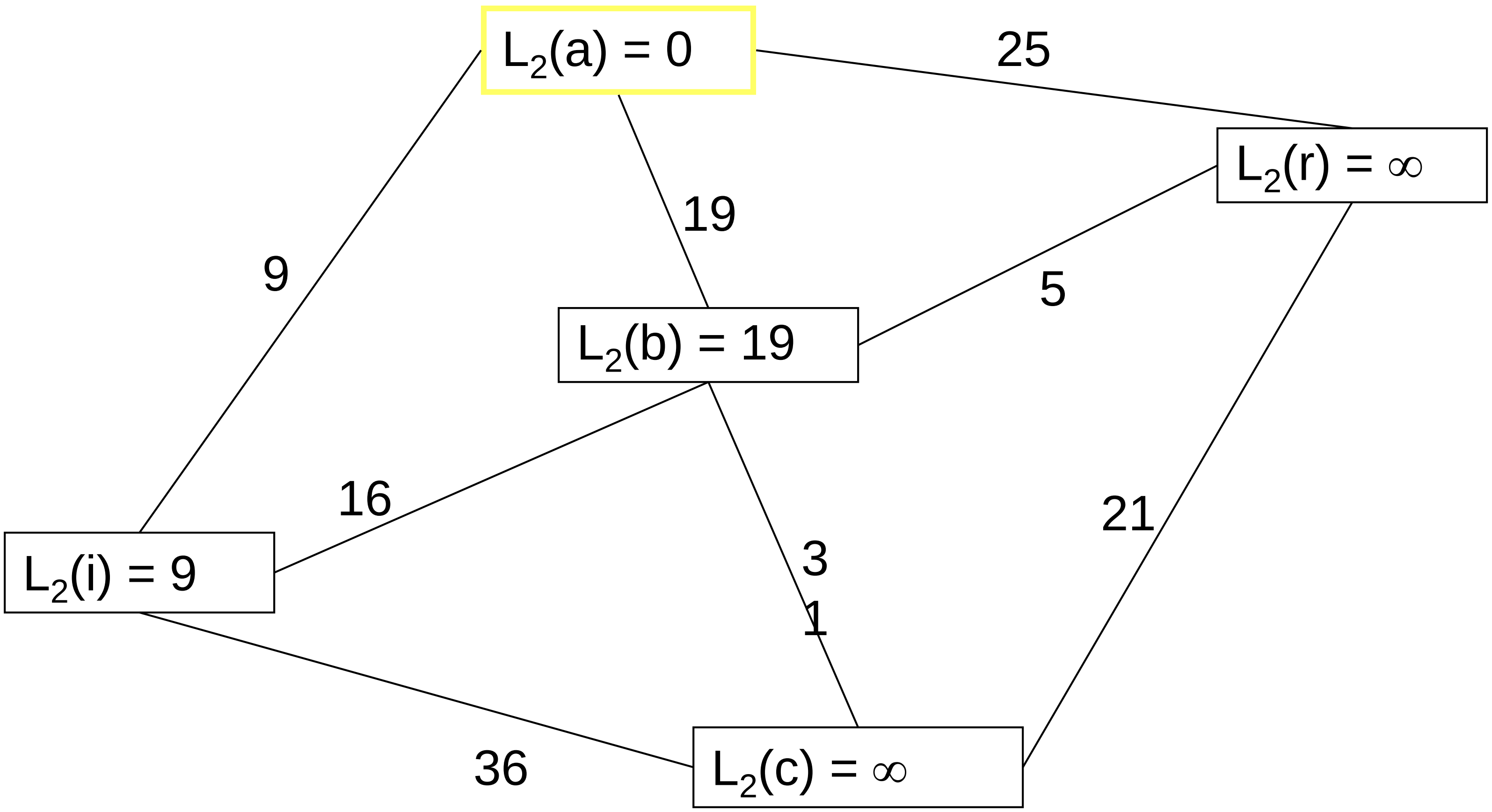
Labels are shortest paths from a to vertices.

$S_1 = \{a, i\}$

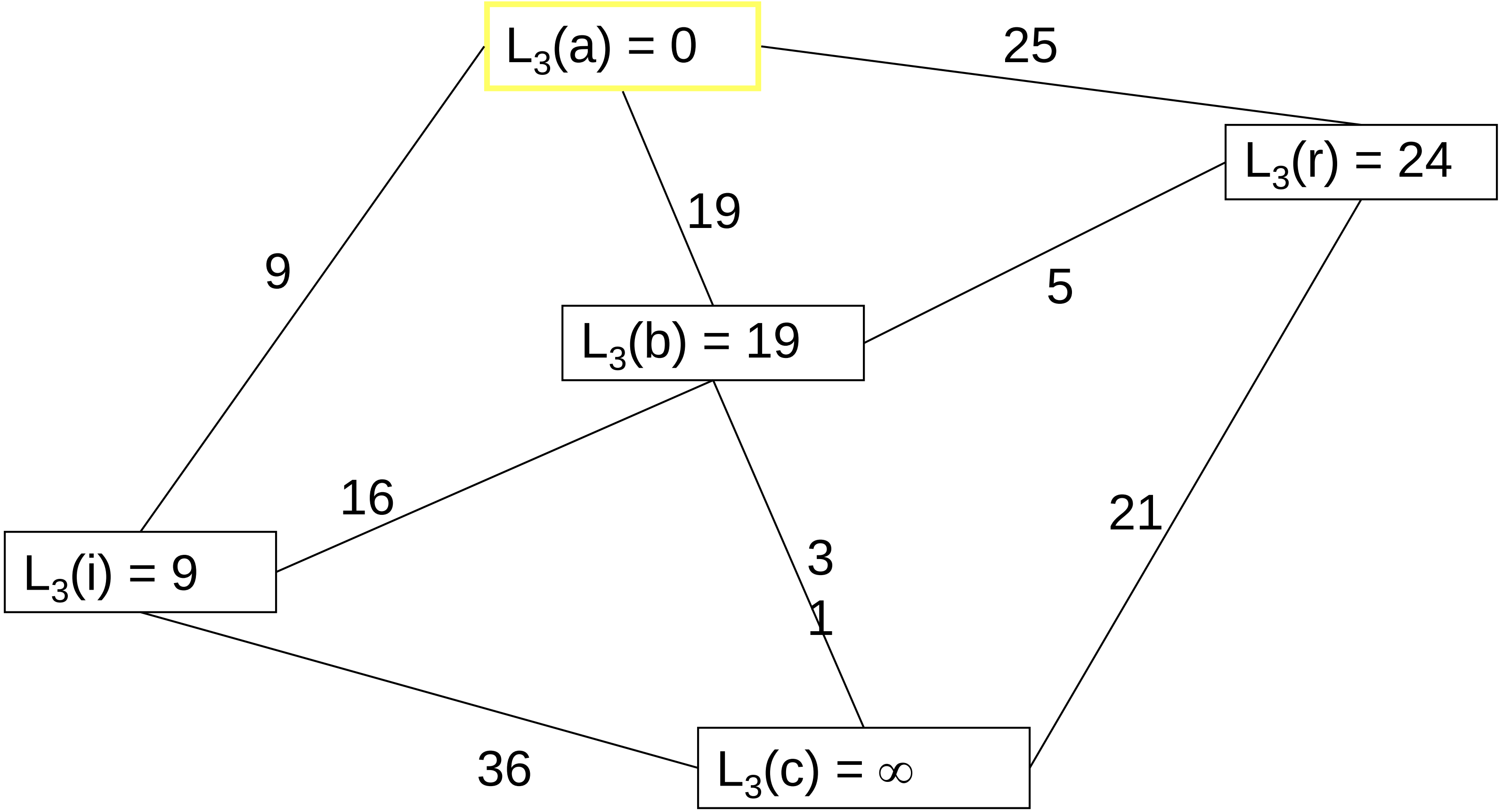


$L_k(a, v) = \min\{L_{k-1}(a, v), L_{k-1}(a, u) + w(u, v)\}$

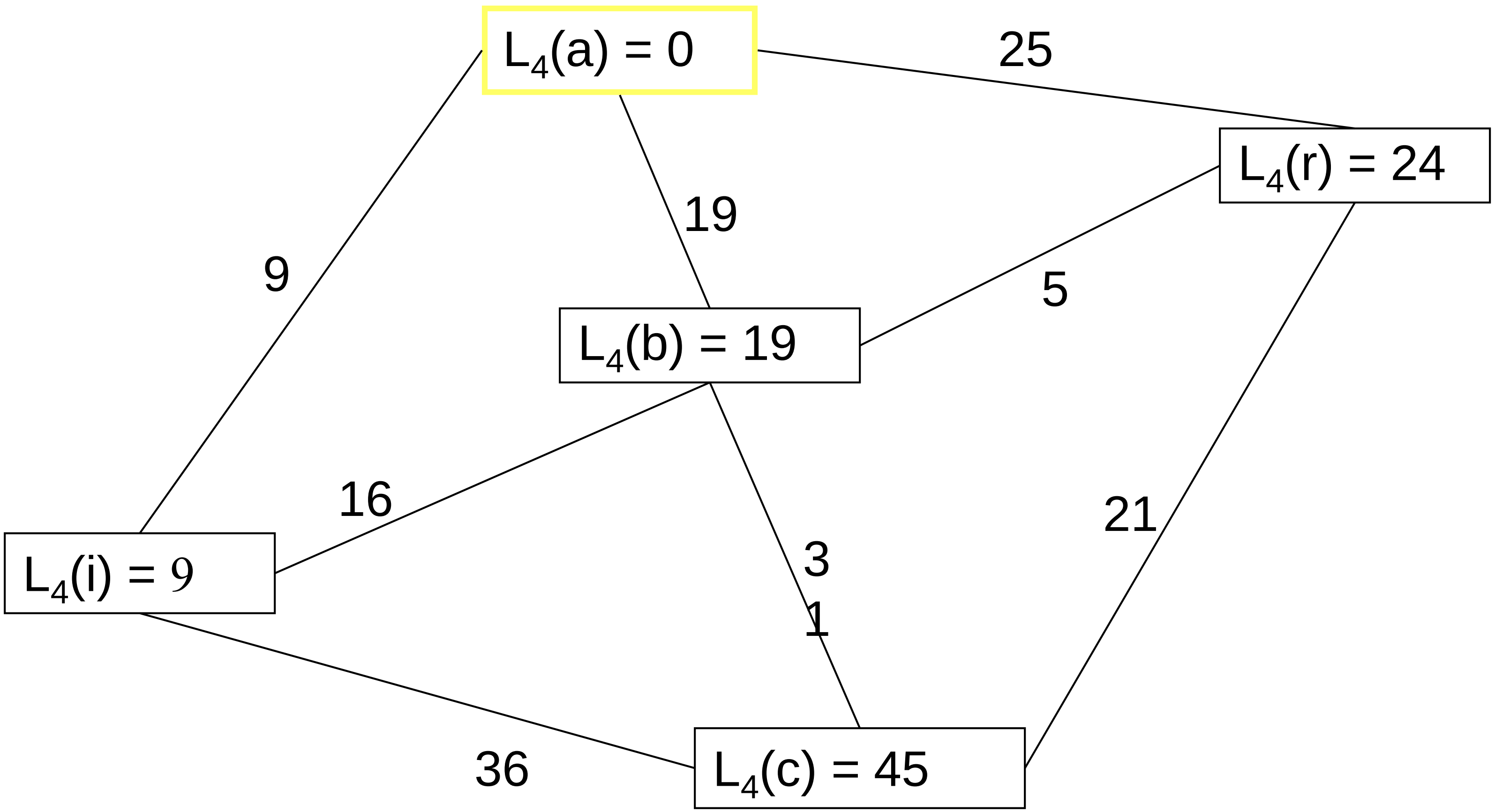
$S_2 = \{a, i, b\}$



S_3 = {a, i, b, r}



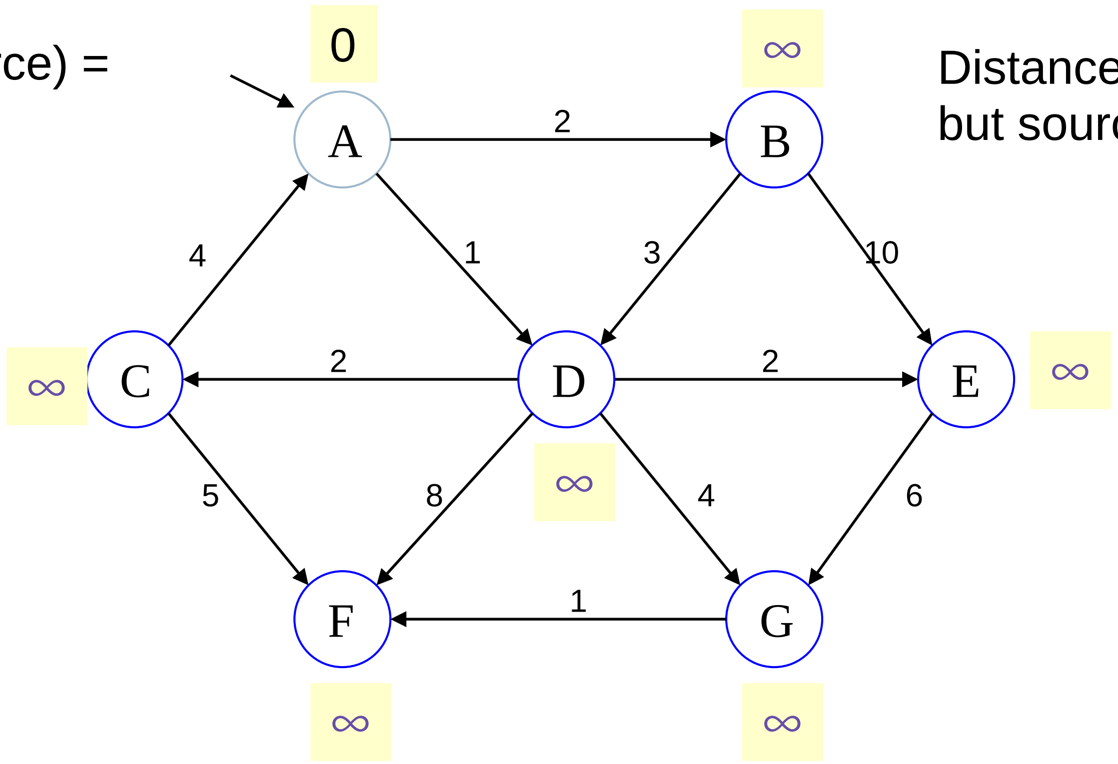
$S_4 = \{a, i, b, r, c\}$



Example2

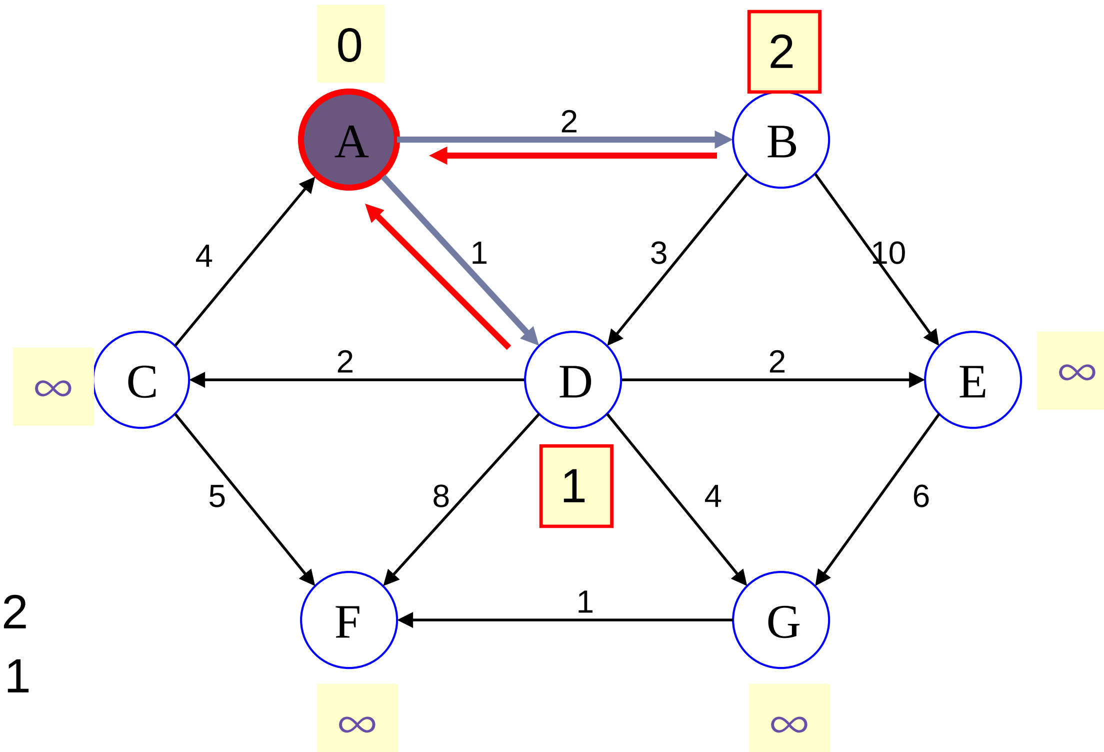
Distance(source) = 0

Distance (all vertices but source) = ∞



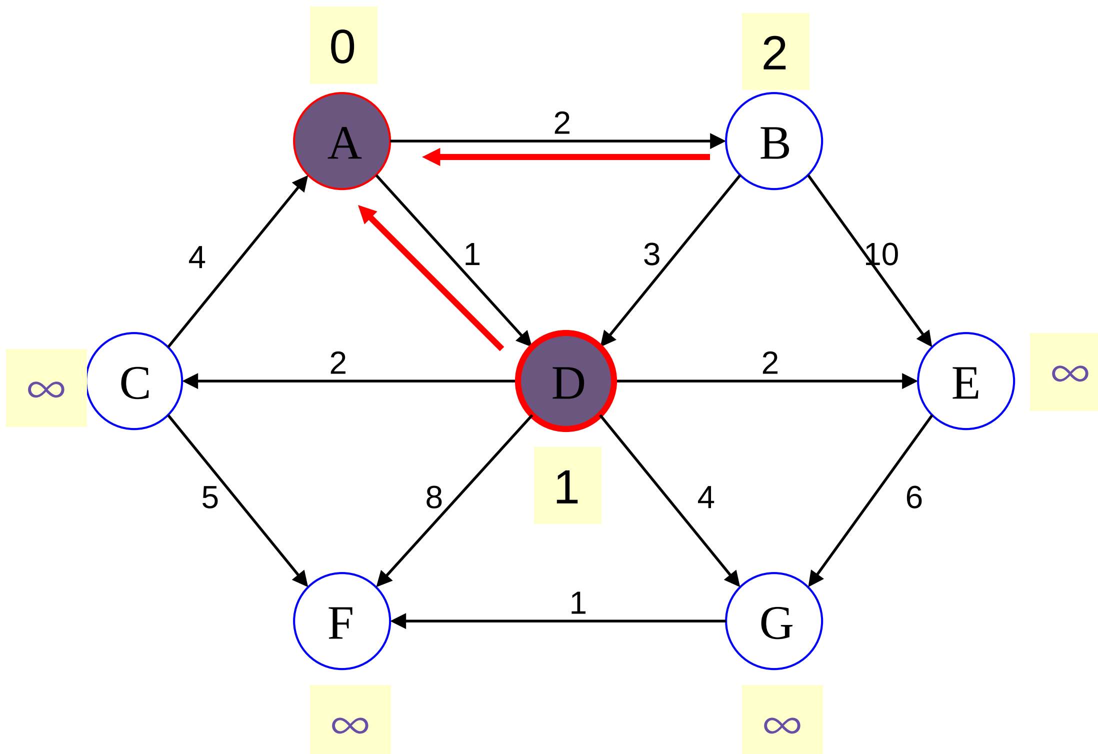
Pick vertex in List with minimum distance.

Example: Update neighbors' distance



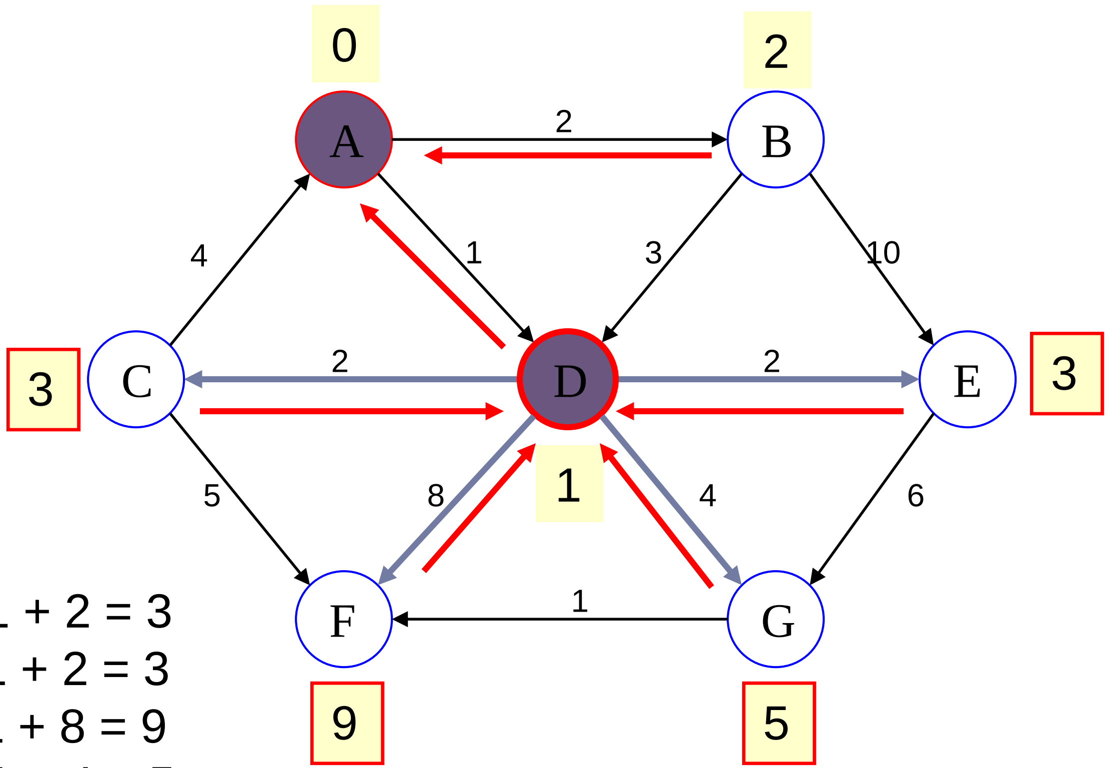
Distance(B) = 2
Distance(D) = 1

Example: Remove vertex with minimum distance



Pick vertex in List with minimum distance, i.e., D

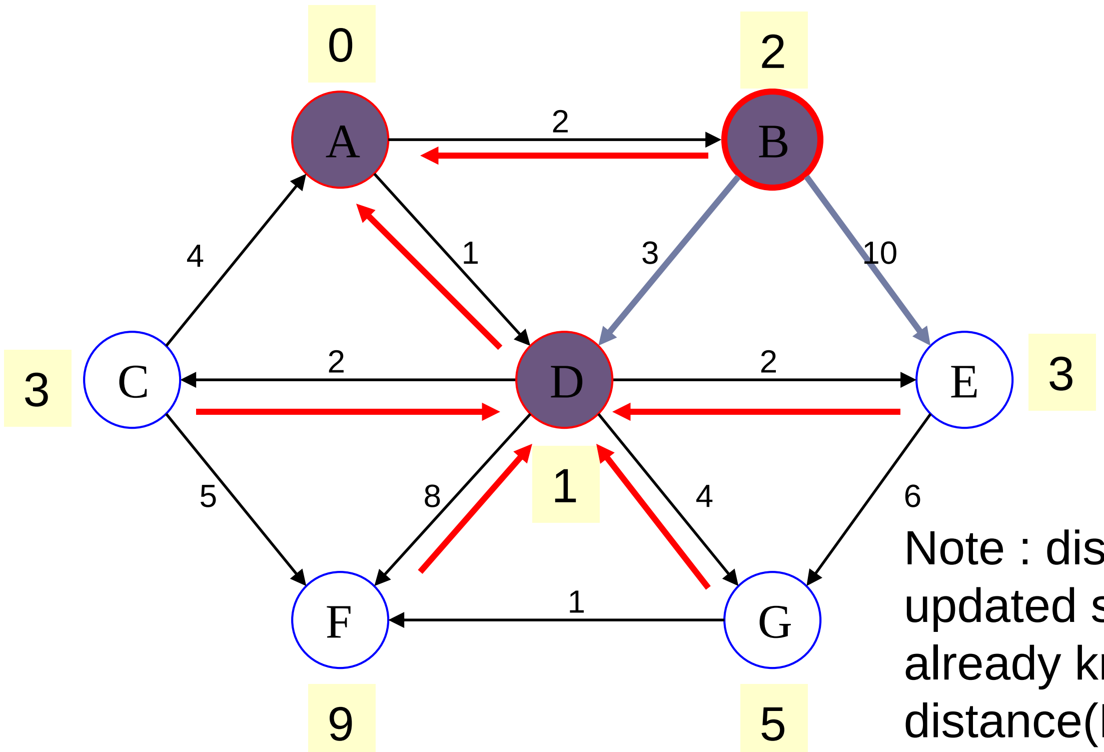
Example: Update neighbors



Distance(C) = 1 + 2 = 3
Distance(E) = 1 + 2 = 3
Distance(F) = 1 + 8 = 9
Distance(G) = 1 + 4 = 5

Example: Continued...

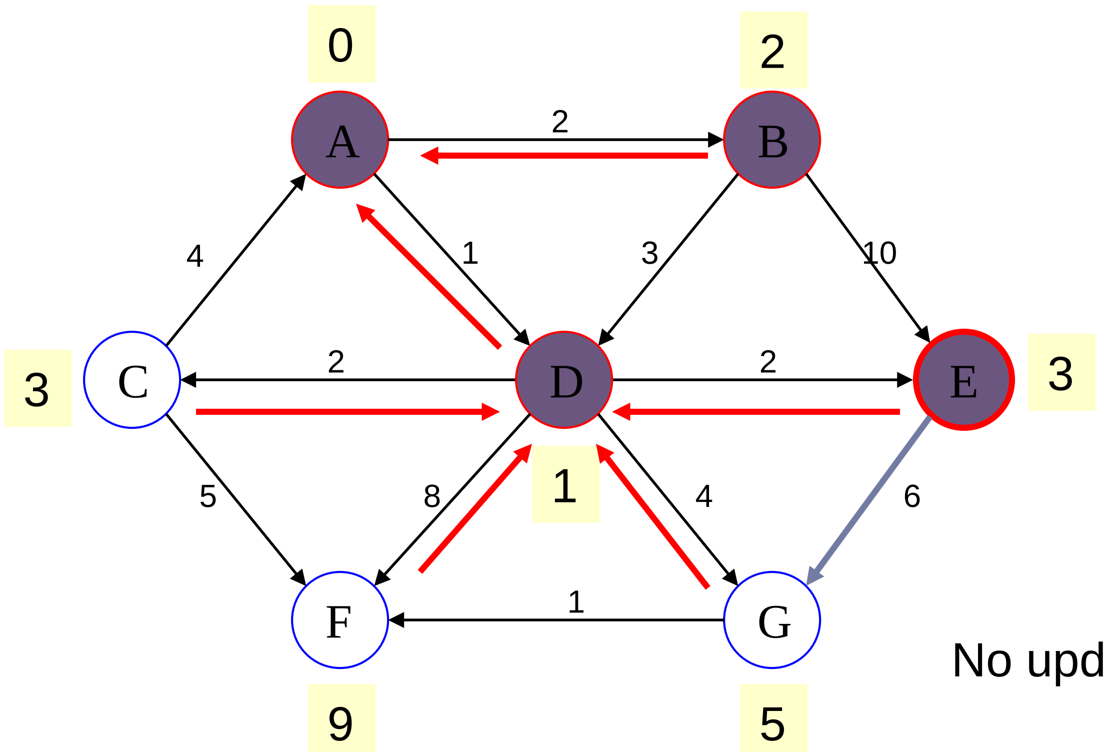
Pick vertex in List with minimum distance (B) and update neighbors



Note : distance(D) not updated since D is already known and distance(E) not updated since it is larger than previously computed

Example: Continued...

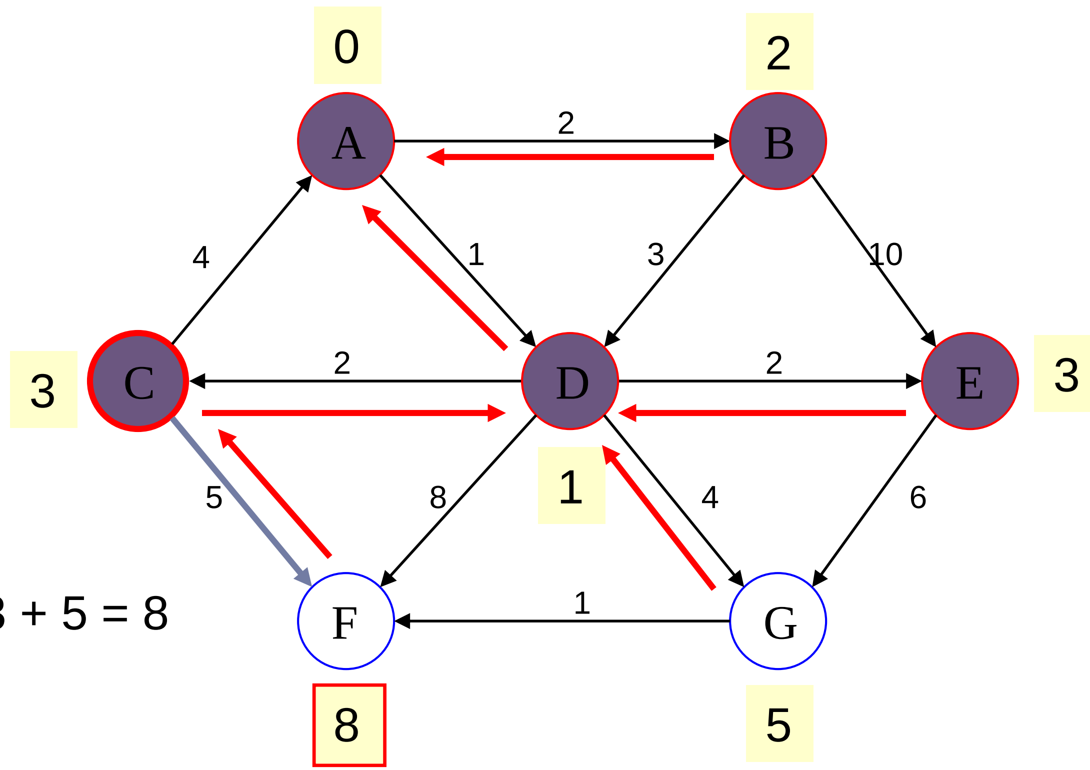
Pick vertex List with minimum distance (E) and update neighbors



No updating

Example: Continued...

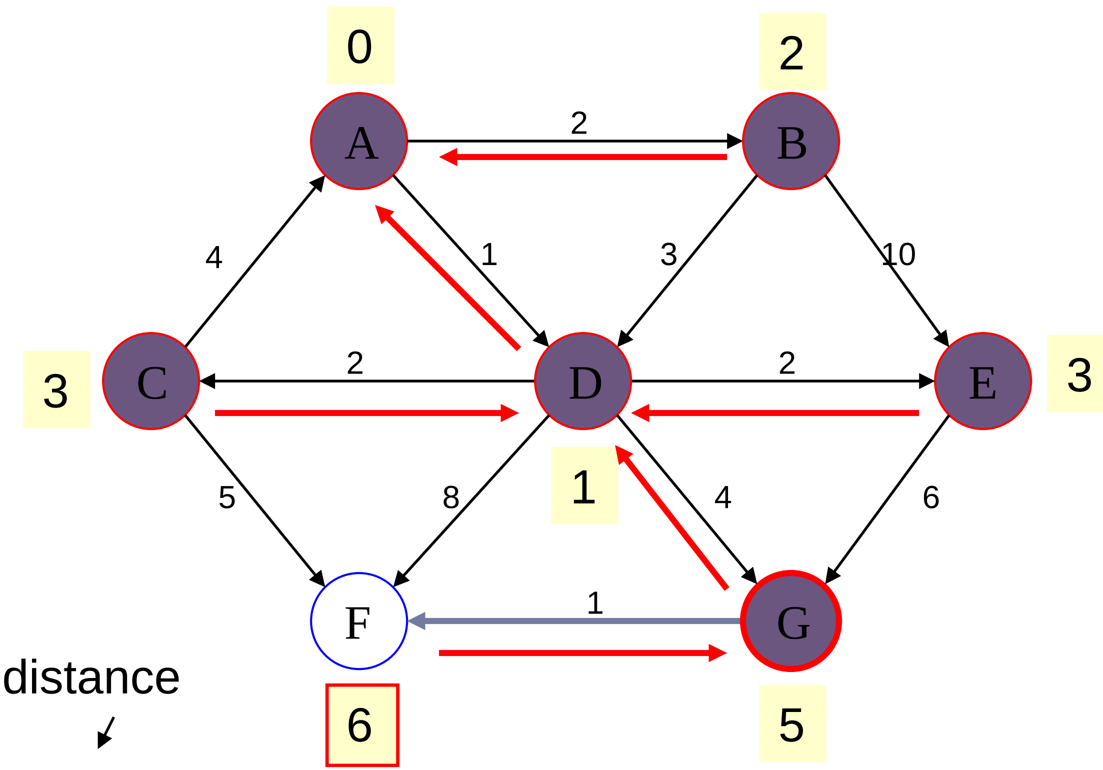
Pick vertex List with minimum distance (C) and update neighbors



Distance(F) = 3 + 5 = 8

Example: Continued...

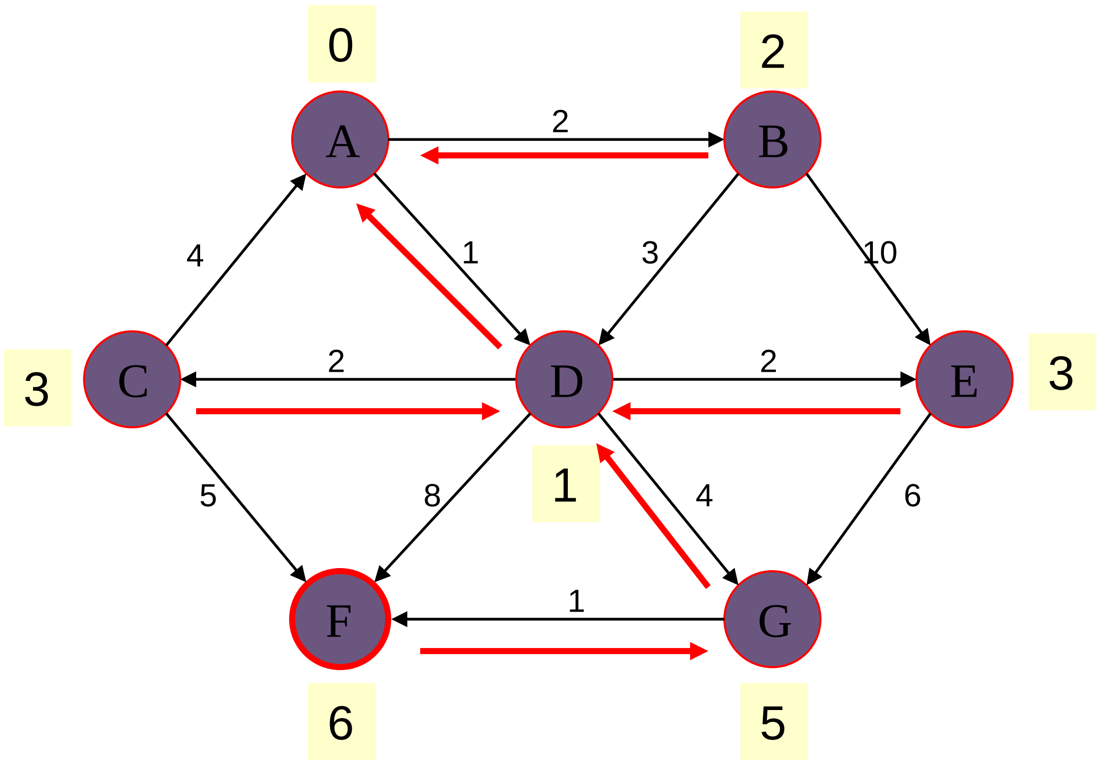
Pick vertex List with minimum distance (G) and update neighbors



Previous distance

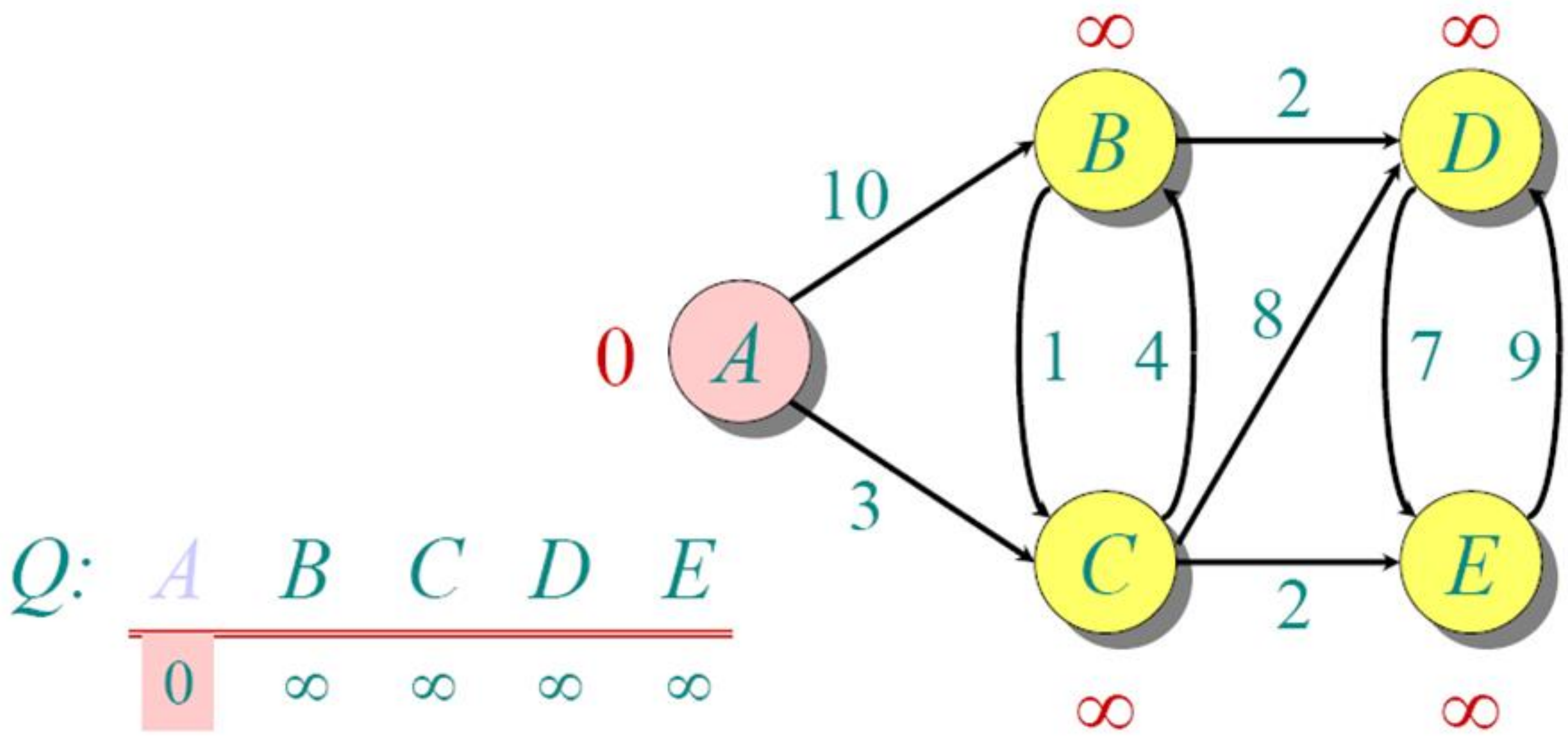
Distance(F) = min (8, 5+1) = 6

Example (end)

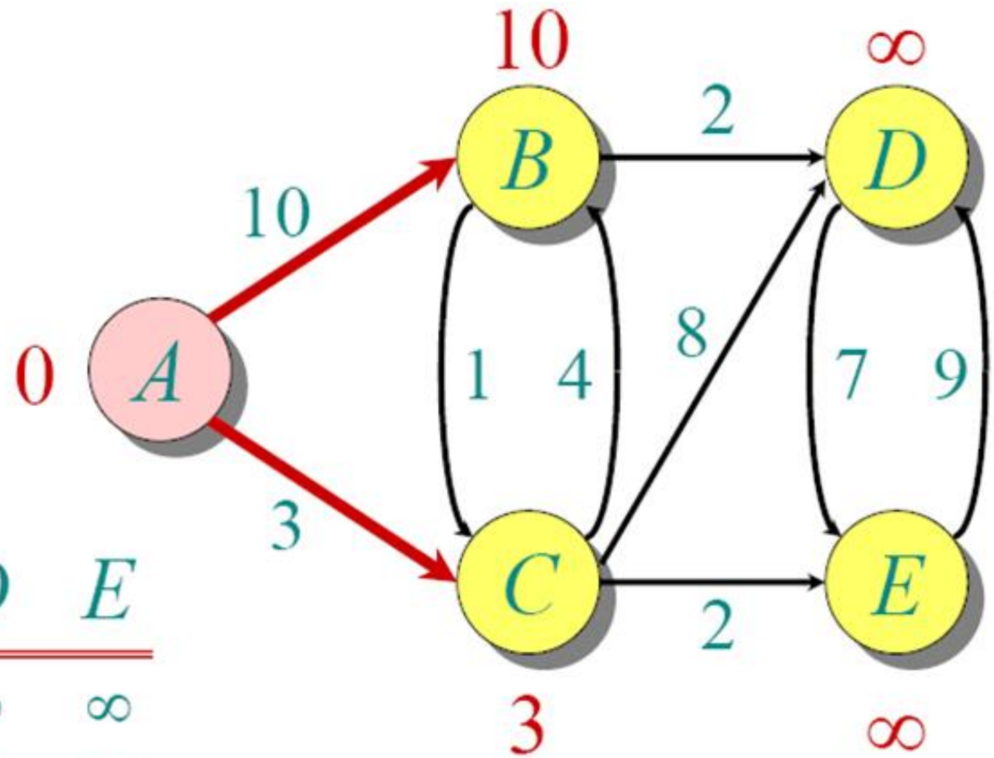


Pick vertex not in S with lowest cost (F) and update neighbors

Another Example



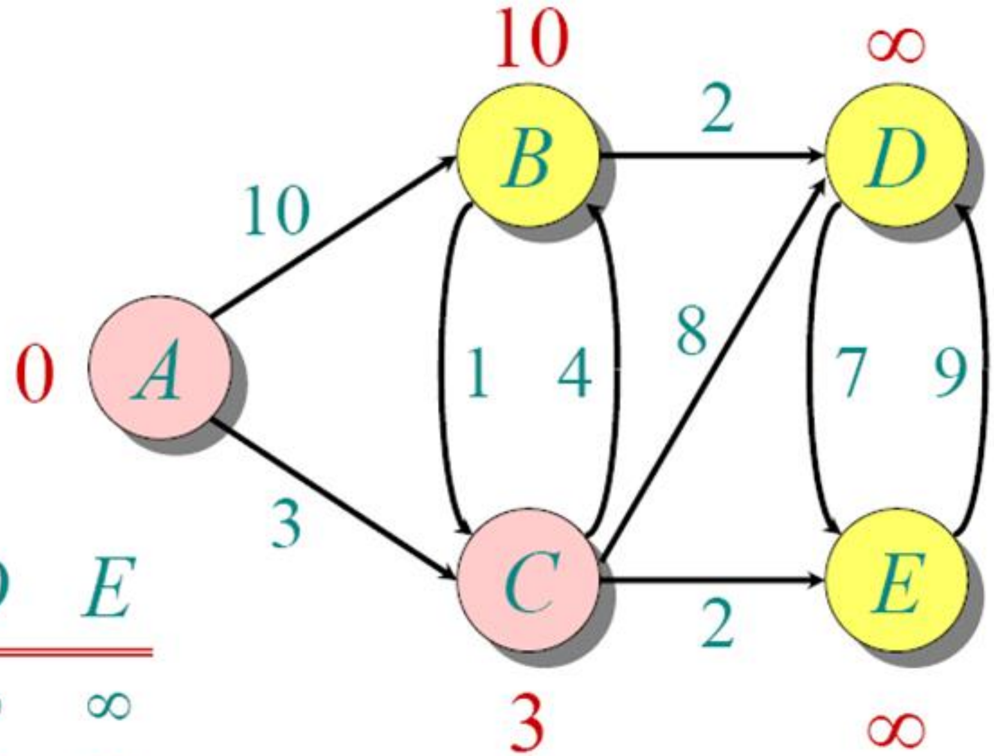
Another Example



Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞

S: { A }

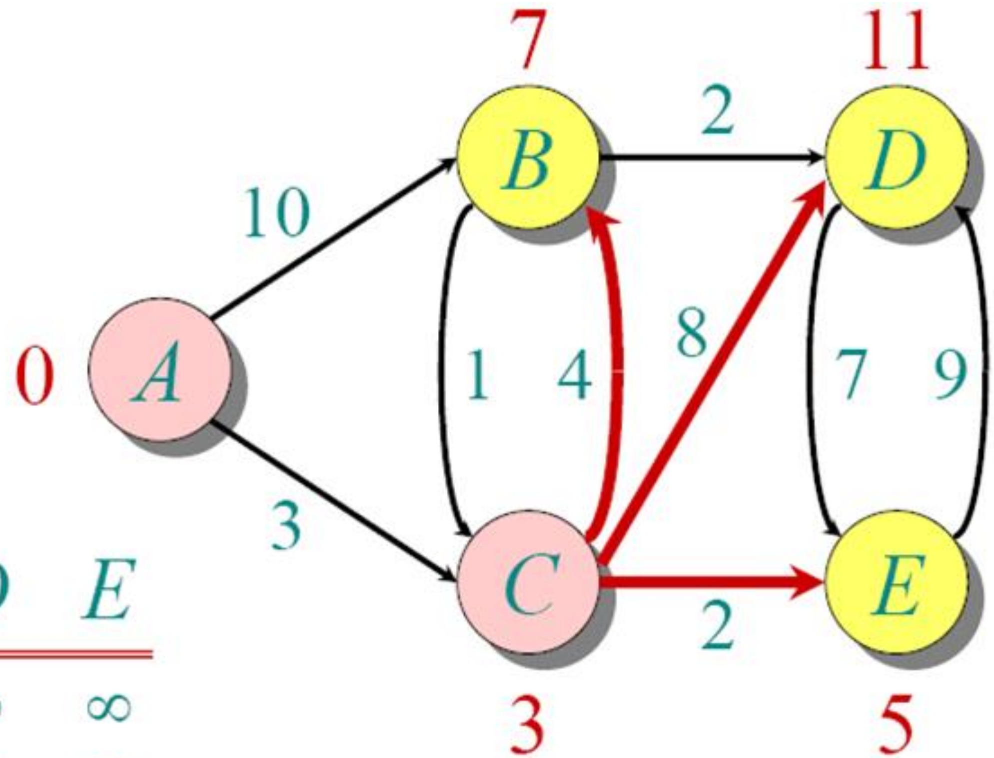
Another Example



Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞

S: { *A*, *C* }

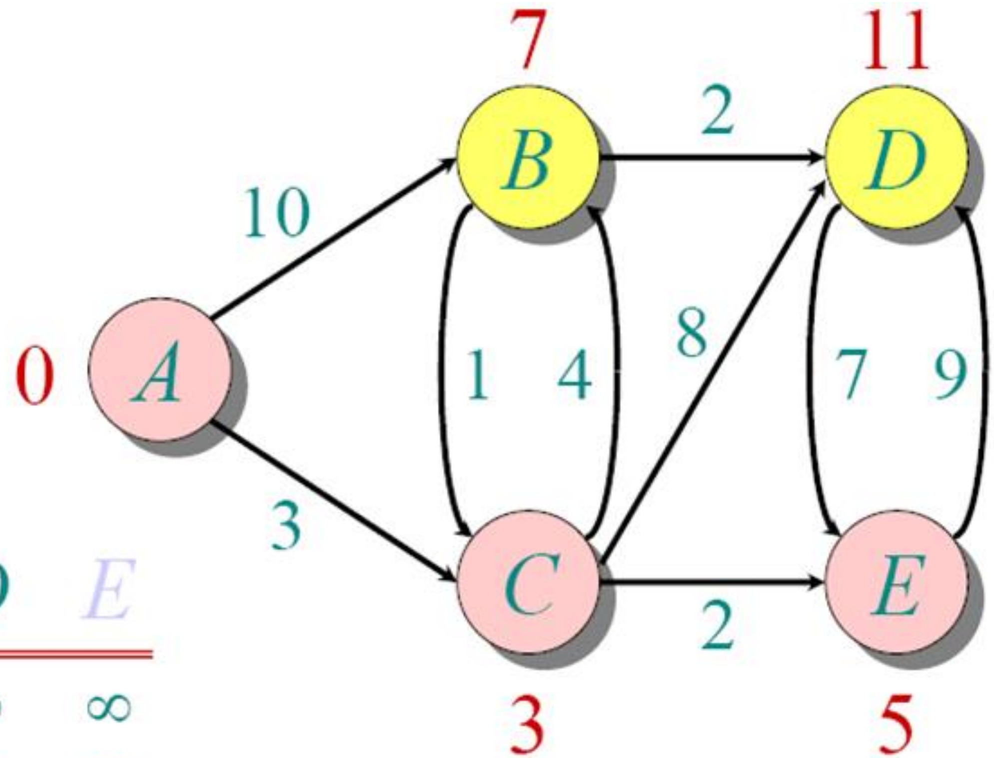
Another Example



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

$S: \{ A, C \}$

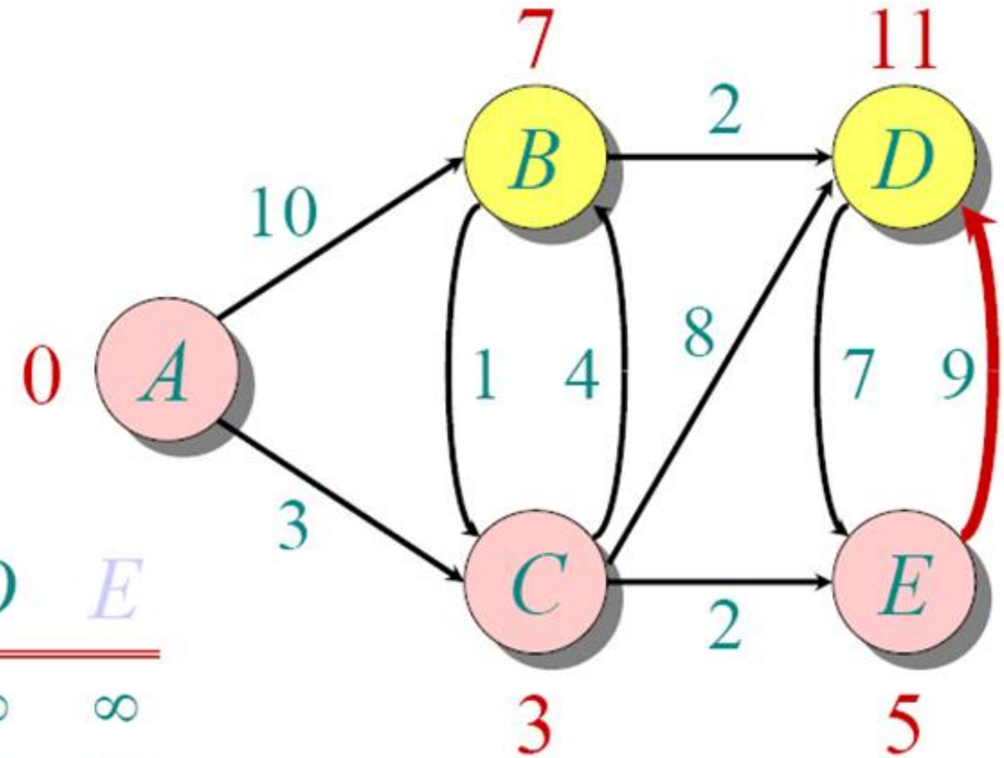
Another Example



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

S: { A, C, E }

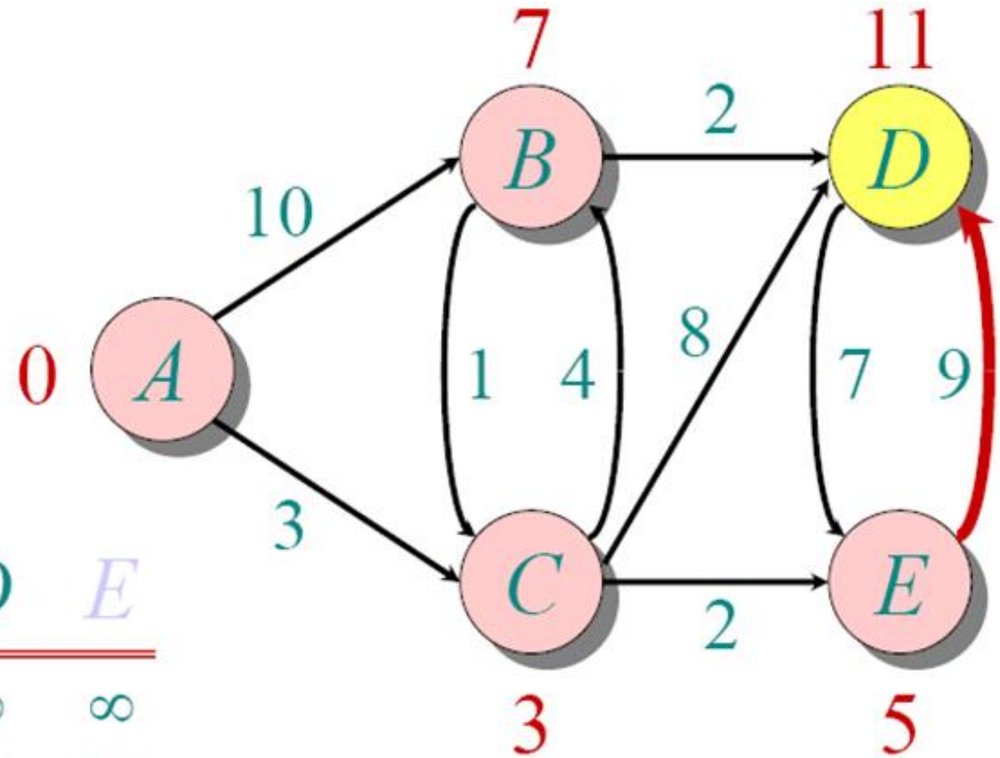
Another Example



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

S: { A, C, E }

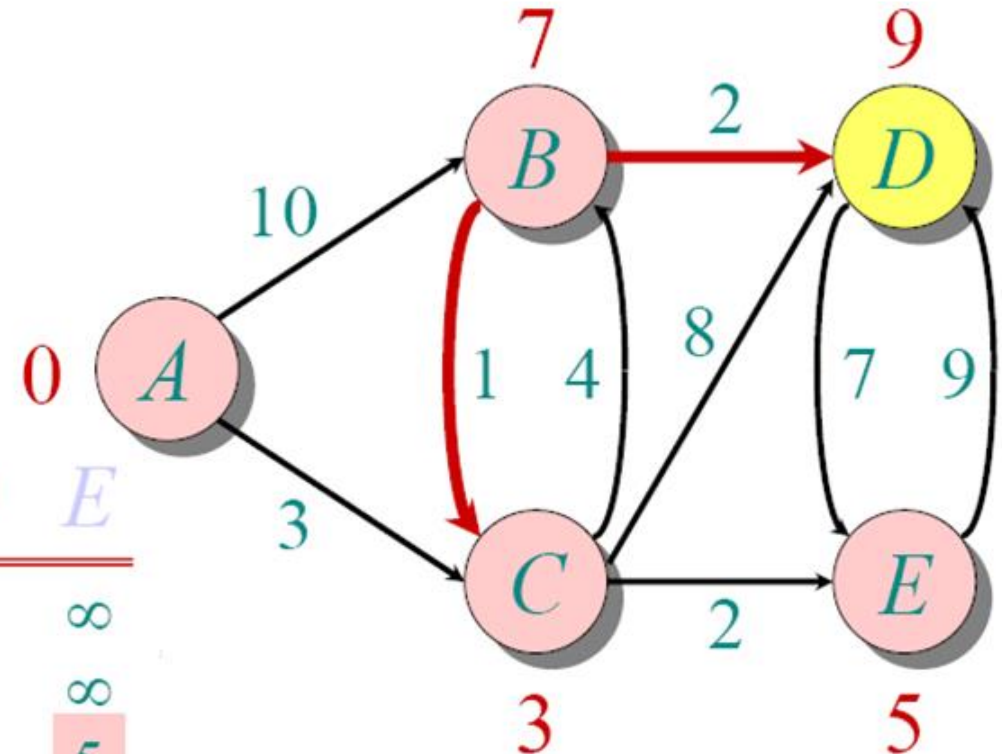
Another Example



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

S: { A, C, E, B }

Another Example

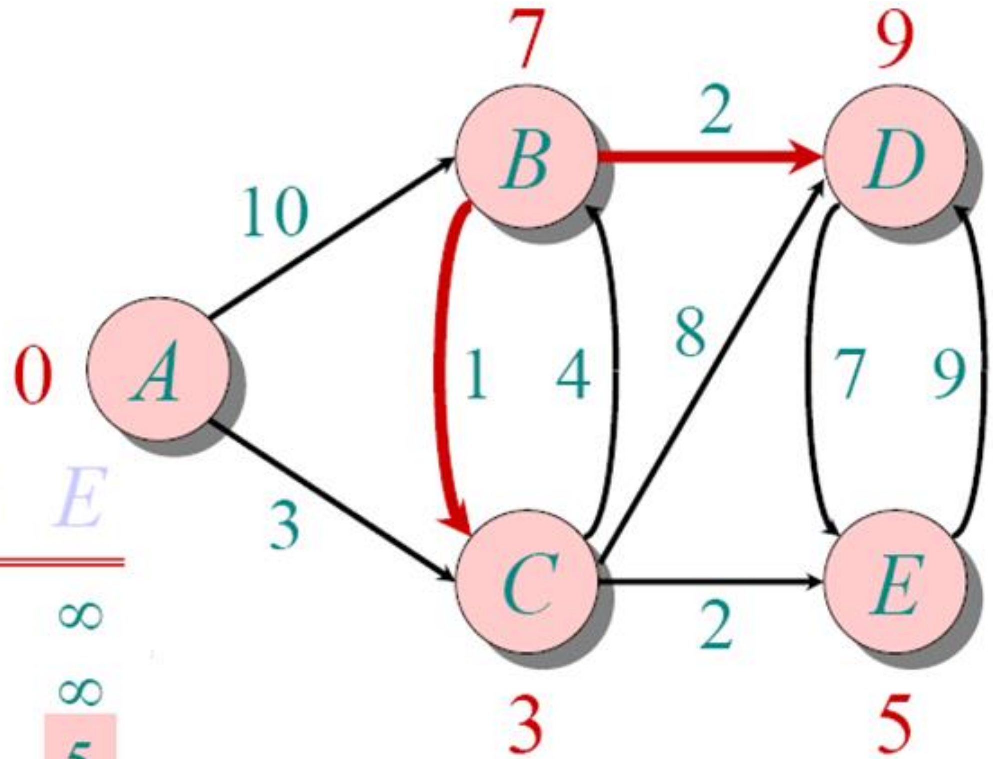


Q:

A	B	C	D	E
0	∞	∞	∞	∞
	10	3	∞	∞
	7		11	5
	7		11	
			9	

S: { A, C, E, B }

Another Example



Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	
				9	

S: { A, C, E, B, D }