ECE430.217 Data Structures

Dijkstra's Algorithms

Textbook: Weiss Chapter 9.3

Byoungyoung Lee

https://compsec.snu.ac.kr

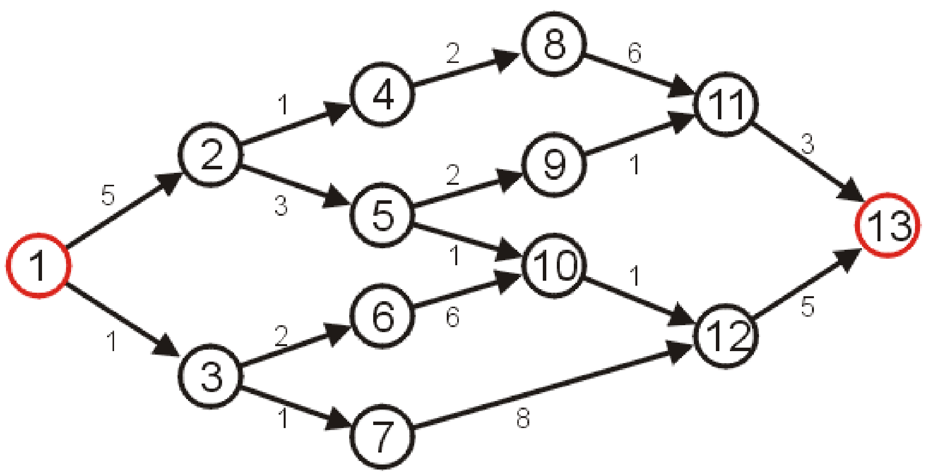
byoungyoung@snu.ac.kr

Lecture slides were prepared based on Douglas W. Harder's notes (**dwharder@alumni.uwaterloo.ca).**

Shortest Path

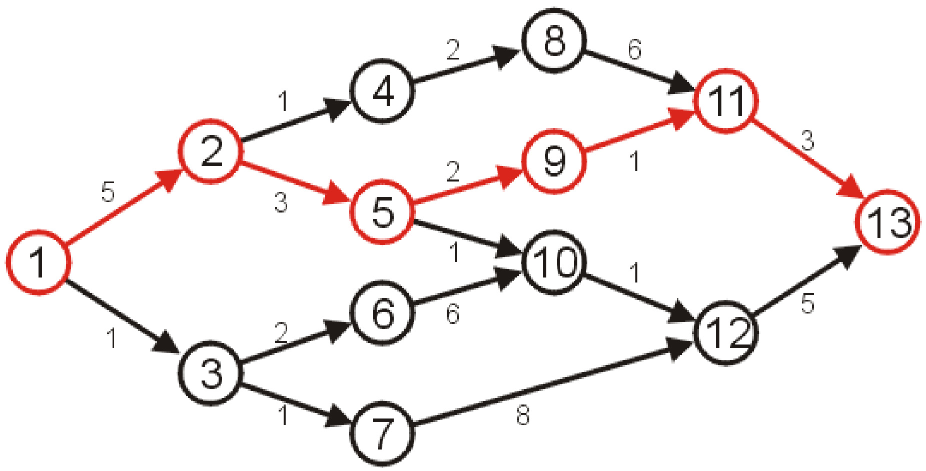
Given the graph, suppose you wish to **find the shortest path** from vertex 1 to vertex 13

- Length of the path is the sum of edge weights
- The graph is directed acyclic graph



Shortest Path

The shortest path has length 14



Other paths exists, but they are longer

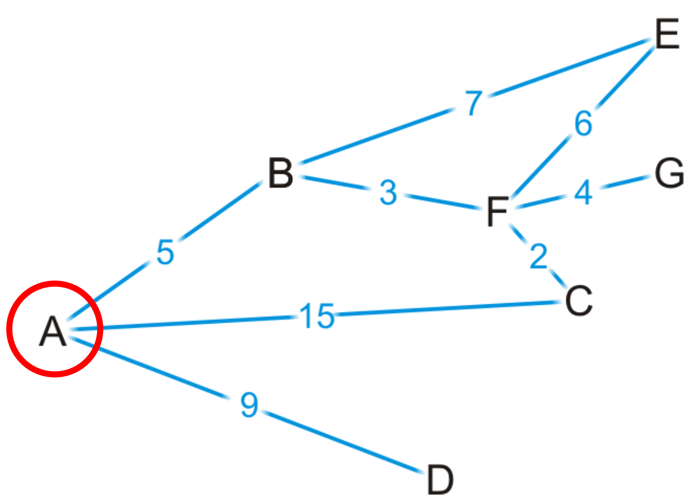
Dijkstra's algorithm

Dijkstra's algorithm solves the single-source shortest path problem

- It is very similar to Prim's algorithm
- Assumption: All the weights are positive

Suppose you are at vertex A

- You are aware of all adjacent vertices to A
- This information is either in an adjacency list or adjacency matrix



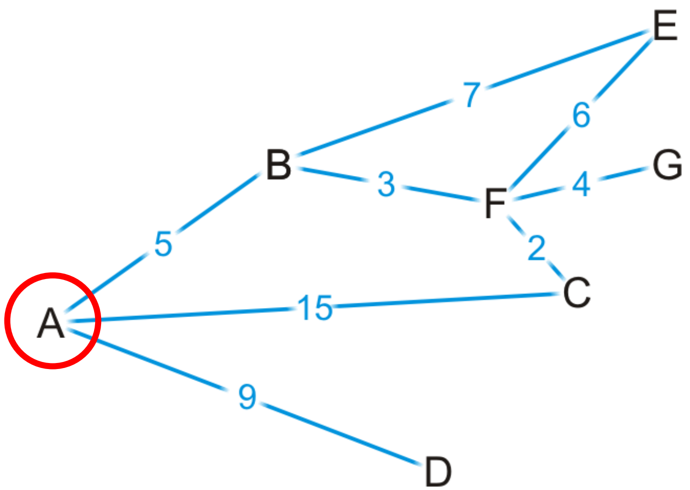
Strategy of Dijkstra's Algorithm

By looking at all the adjacent edges of A, what can you be sure?

Is 5 the shortest distance from A to B?

Is 15 the shorted distance from A to C?

Is 9 the shorted distance from A to D?



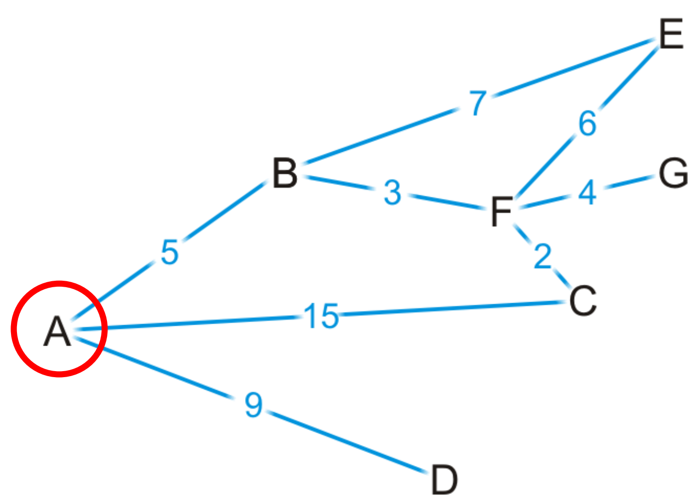
Strategy of Dijkstra's Algorithm

By looking at all the adjacent edges of A, what can you be sure?

Is 5 the shortest distance from A to B? (Yes)

Is 15 the shorted distance from A to C? (No)

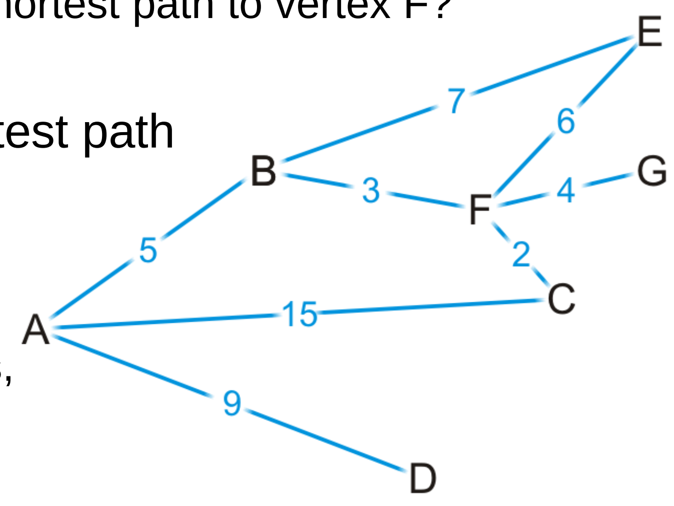
Is 9 the shorted distance from A to D? (No)



Dijkstra's algorithm

Like Prim's algorithm, we initially don't know the distance to any vertex except vertices adjacent to the initial vertex

- We require an array of distances, all initialized to infinity except for the source vertex, which is initialized to 0
- Each time we visit a vertex, we will examine all adjacent vertices
  - We need to track visited vertices—a Boolean table of size $|V|$
- Do we need to track the shortest path to each vertex?
  - That is, do I have to store (A, B, F) as the shortest path to vertex F?



- We really only have to record that the shortest path to vertex F came from vertex B
  - We would then determine that the shortest path to vertex B came from vertex A
  - Thus, we need an array of previous vertices, all initialized to null

Dijkstra's algorithm

Thus, we will iterate $|V|$ times:

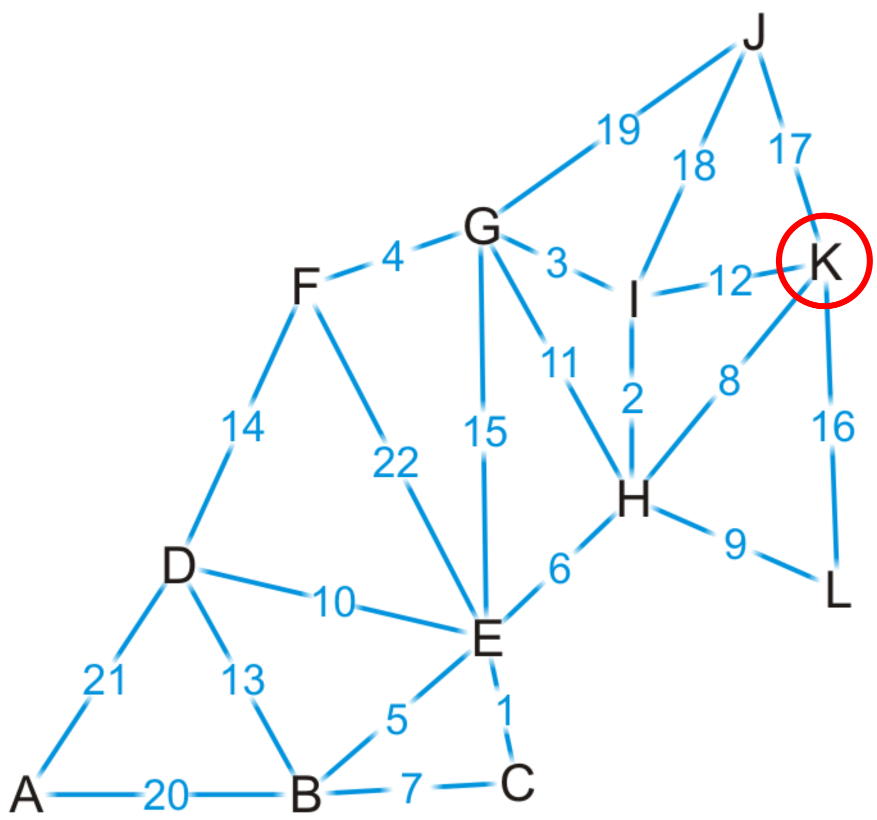
- Find that unvisited vertex $\boldsymbol{v}$ that has a minimum distance to it
- Mark it as having been visited
- Consider every adjacent vertex $\boldsymbol{w}$ that is unvisited:
  - Is the distance to $\boldsymbol{v}$ plus the weight of the edge ($\boldsymbol{v}$, $\boldsymbol{w}$) less than our currently known shortest distance to $\boldsymbol{w}$
  - If so, update the shortest distance to $\boldsymbol{w}$ and record $\boldsymbol{v}$ as the previous pointer

- Continue iterating until all vertices are visited or all remaining vertices have a distance to them of infinity

Example

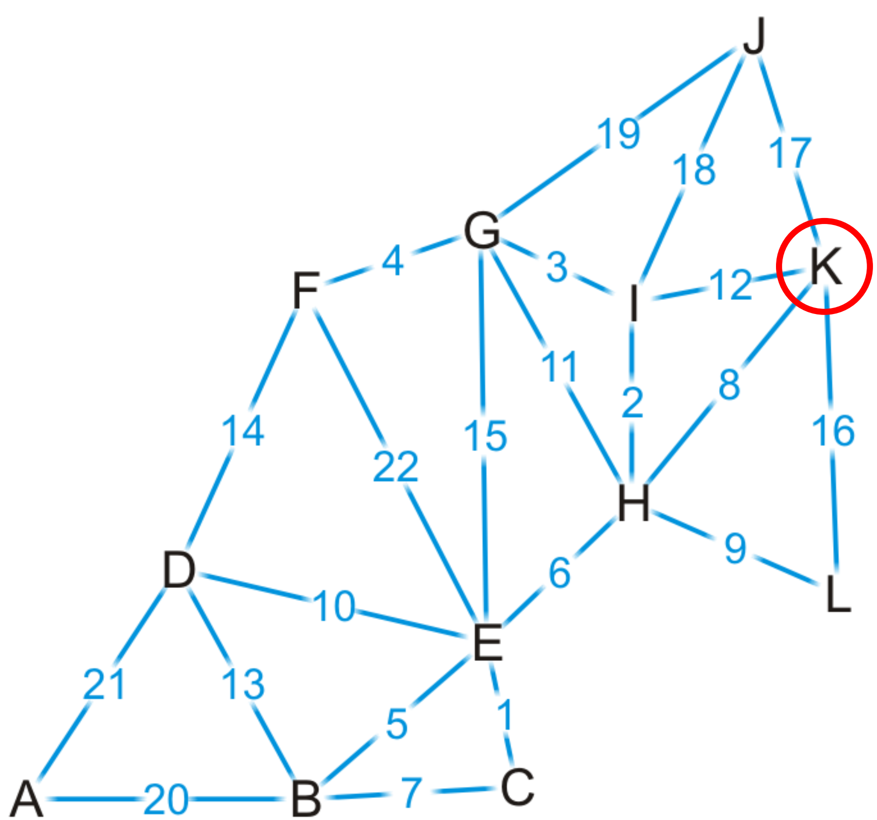
Let’s take an example with this graph

We will find the shortest distance from **the vertex K** to every other vertices



Example

We first set up the table.

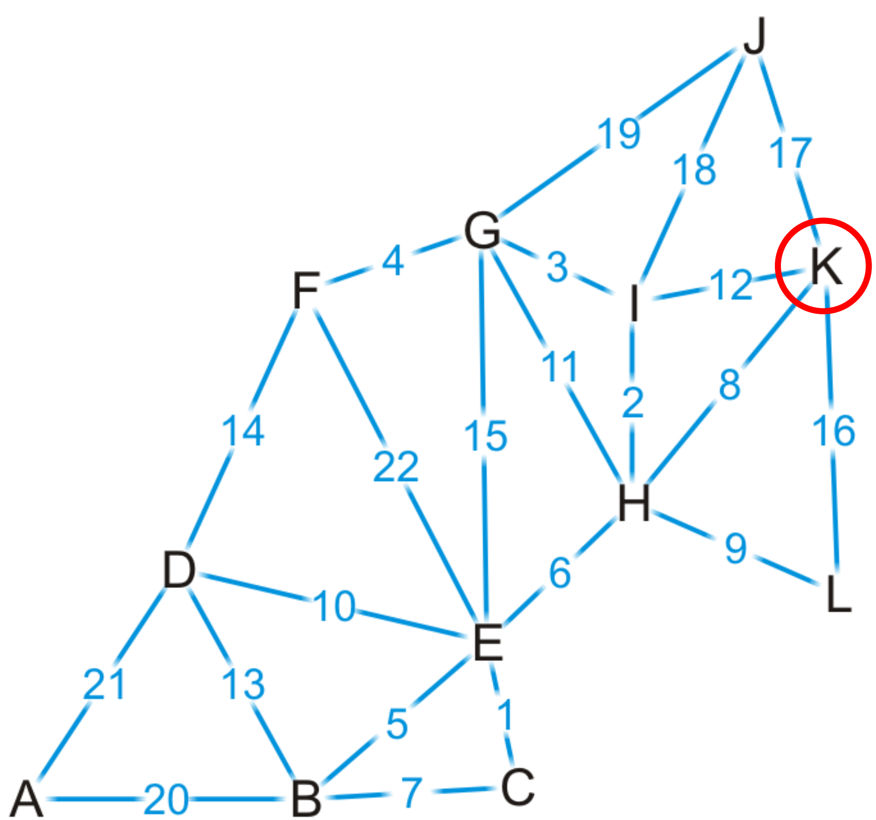


Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	∞	Ø
C	F	∞	Ø
D	F	∞	Ø
E	F	∞	Ø
F	F	∞	Ø
G	F	∞	Ø
H	F	∞	Ø
I	F	∞	Ø
J	F	∞	Ø
K	F	0	Ø
L	F	∞	Ø

Example

We visit vertex K.

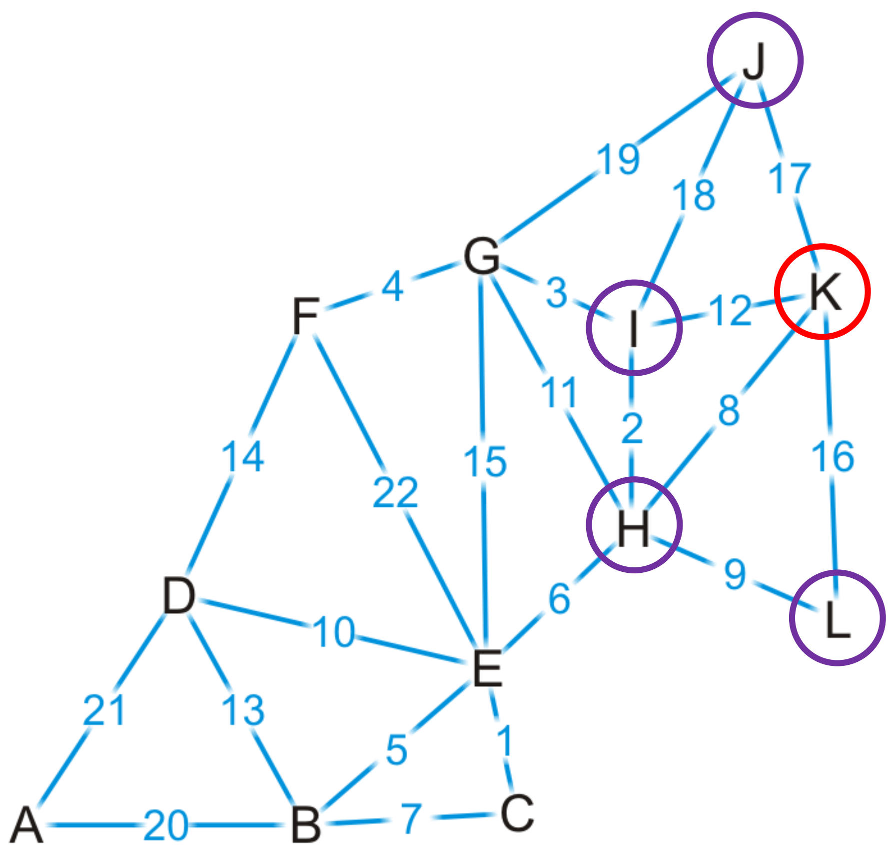
- Which unvisited vertex has the minimum distance to the vertex k?



Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	∞	Ø
C	F	∞	Ø
D	F	∞	Ø
E	F	∞	Ø
F	F	∞	Ø
G	F	∞	Ø
H	F	∞	Ø
I	F	∞	Ø
J	F	∞	Ø
K	**T**	**0**	**Ø**
L	F	∞	Ø

Example

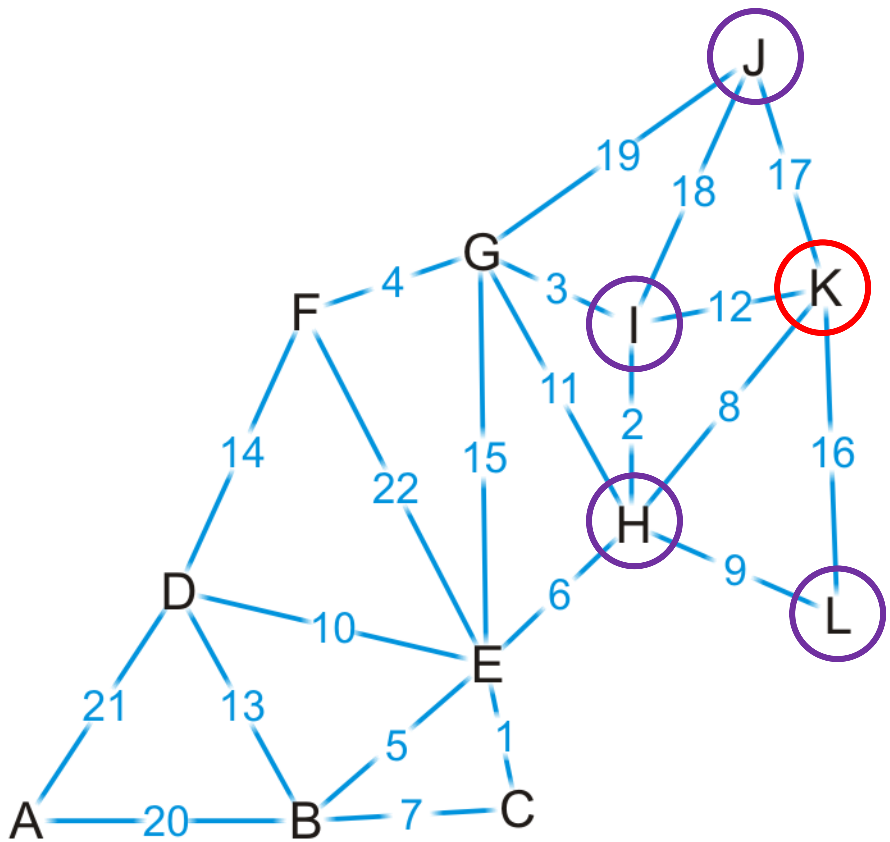
Vertex K has four neighbors: H, I, J and L



Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	∞	Ø
C	F	∞	Ø
D	F	∞	Ø
E	F	∞	Ø
F	F	∞	Ø
G	F	∞	Ø
H	F	∞	Ø
I	F	∞	Ø
J	F	∞	Ø
K	**T**	**0**	**Ø**
L	F	∞	Ø

Example

We have now found at least one path to each of these vertices

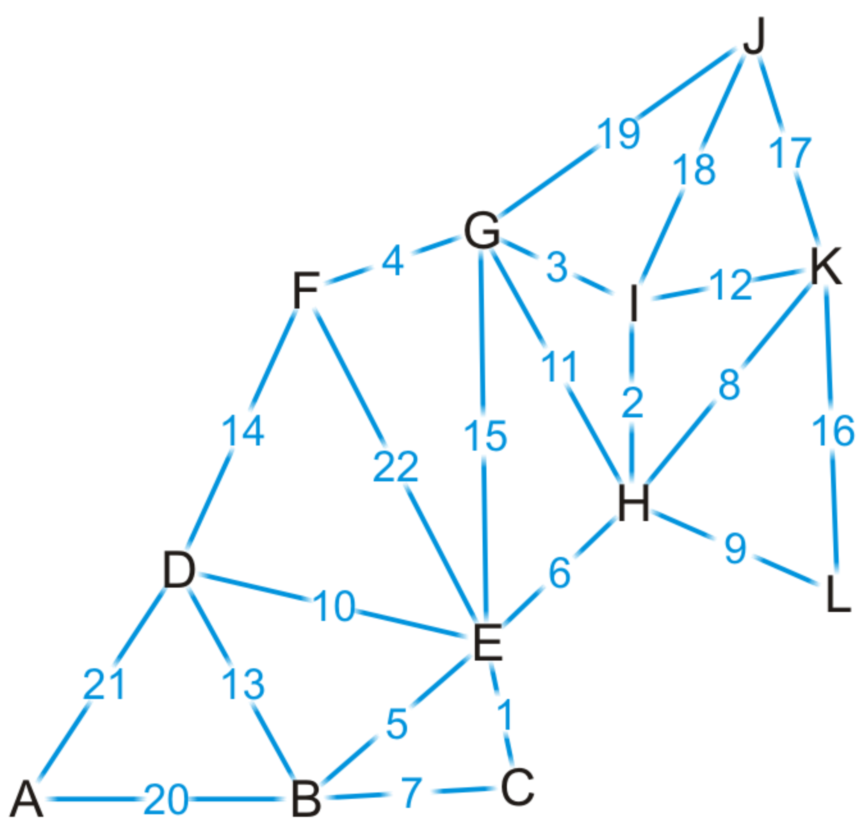


Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	∞	Ø
C	F	∞	Ø
D	F	∞	Ø
E	F	∞	Ø
F	F	∞	Ø
G	F	∞	Ø
H	F	**8**	**K**
I	F	**12**	**K**
J	F	**17**	**K**
K	**T**	**0**	**Ø**
L	F	**16**	**K**

Example

We're finished with vertex K

– To which vertex are we now guaranteed we have the shortest path?

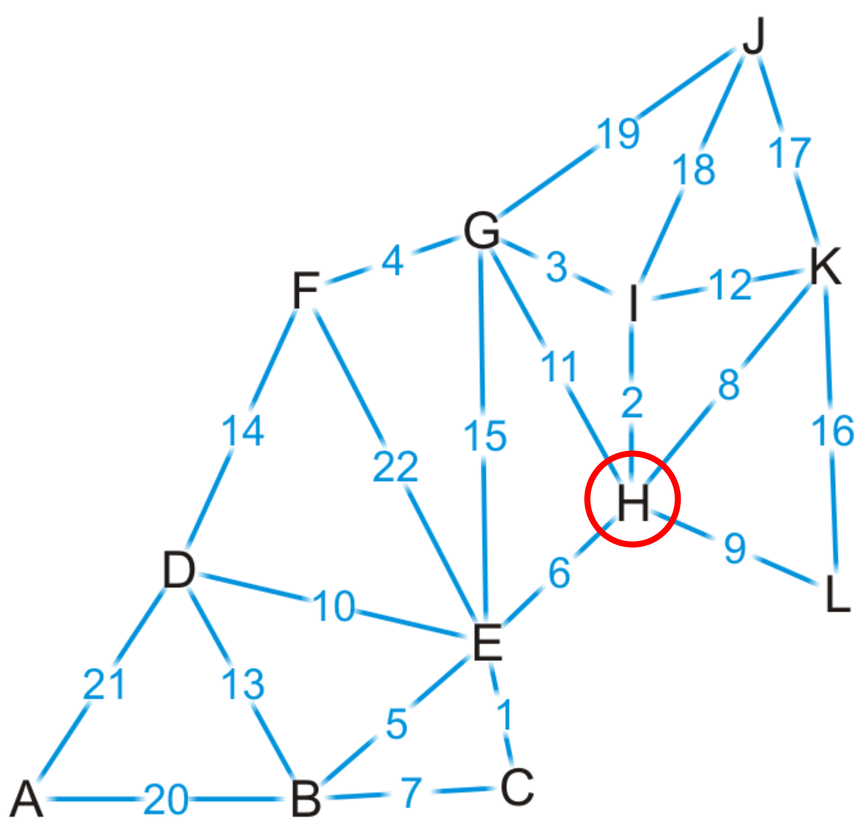


Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	∞	Ø
C	F	∞	Ø
D	F	∞	Ø
E	F	∞	Ø
F	F	∞	Ø
G	F	∞	Ø
H	F	8	K
I	F	12	K
J	F	17	K
K	T	0	Ø
L	F	16	K

Example

We visit vertex H: the shortest path is (K, H) of length 8

- Vertex H has four unvisited neighbors: E, G, I, L



Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	∞	Ø
C	F	∞	Ø
D	F	∞	Ø
E	F	∞	Ø
F	F	∞	Ø
G	F	∞	Ø
H	**T**	**8**	**K**
I	F	12	K
J	F	17	K
K	T	0	Ø
L	F	16	K

Example

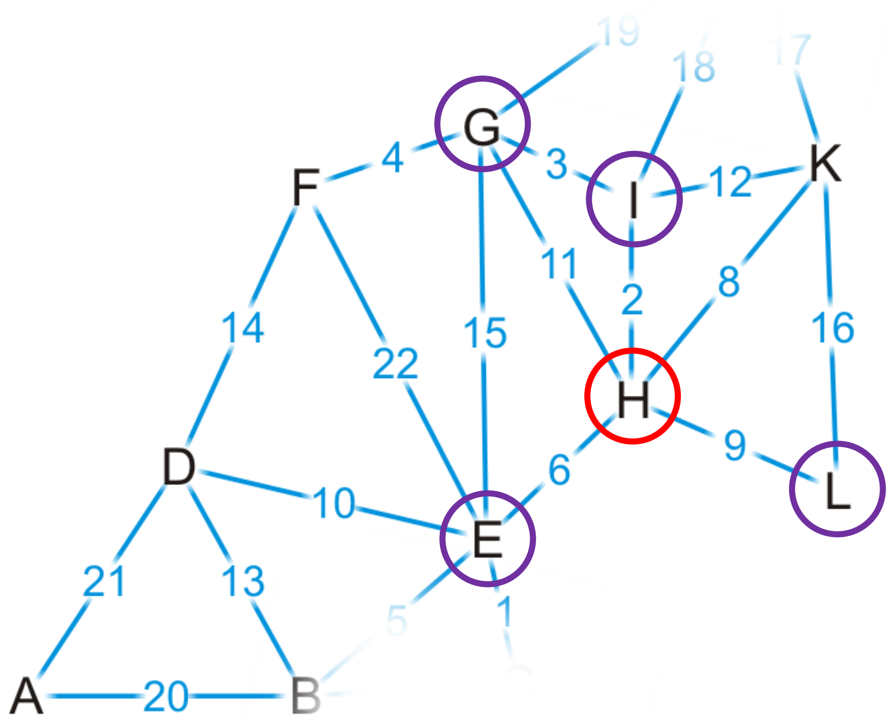
Consider these paths:

(K, H, E) of length 8 + 6 = 14

(K, H, G) of length 8 + 11 = 19

(K, H, I) of length 8 + 2 = 10

(K, H, L) of length 8 + 9 = 17

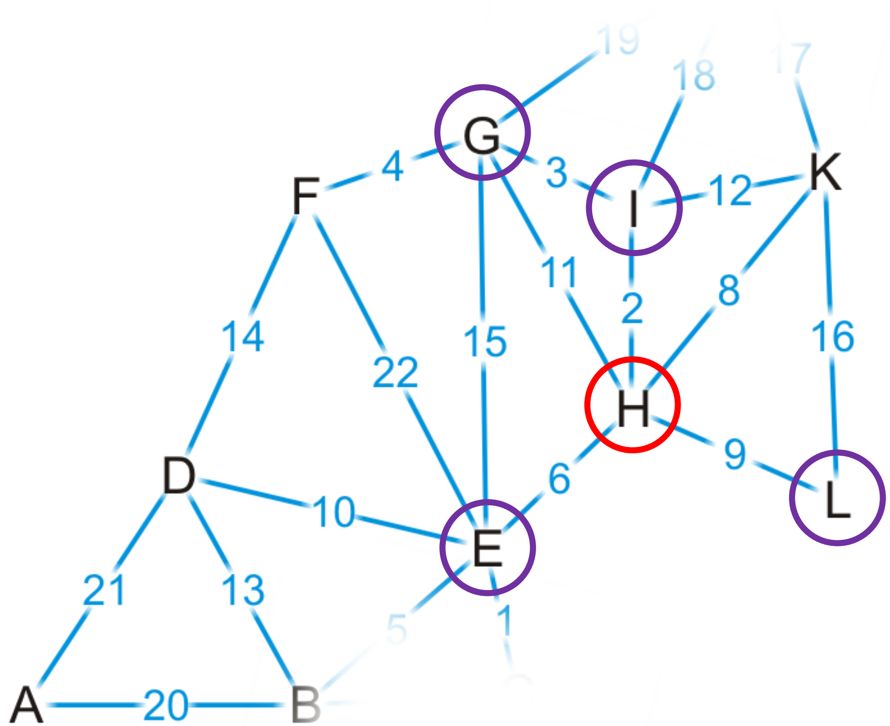


Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	∞	Ø
C	F	∞	Ø
D	F	∞	Ø
E	F	∞	Ø
F	F	∞	Ø
G	F	∞	Ø
H	**T**	**8**	**K**
I	F	12	K
J	F	17	K
K	T	0	Ø
L	F	16	K

Example

We already have a shorter path (K, L), so L is not updated.

But the other three (e.g., E, G, and I) are updated.

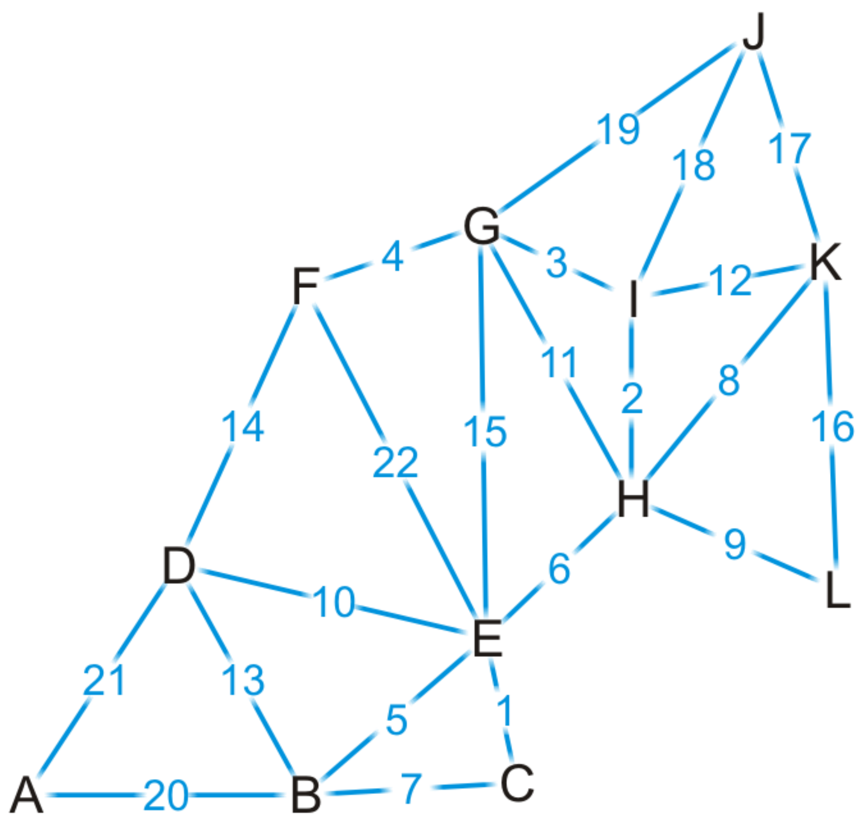


Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	∞	Ø
C	F	∞	Ø
D	F	∞	Ø
E	**F**	**14**	**H**
F	F	∞	Ø
G	**F**	**19**	**H**
H	**T**	**8**	**K**
I	**F**	**10**	**H**
J	F	17	K
K	T	0	Ø
L	F	16	K

Example

We are finished with vertex H

– Which vertex do we visit next?

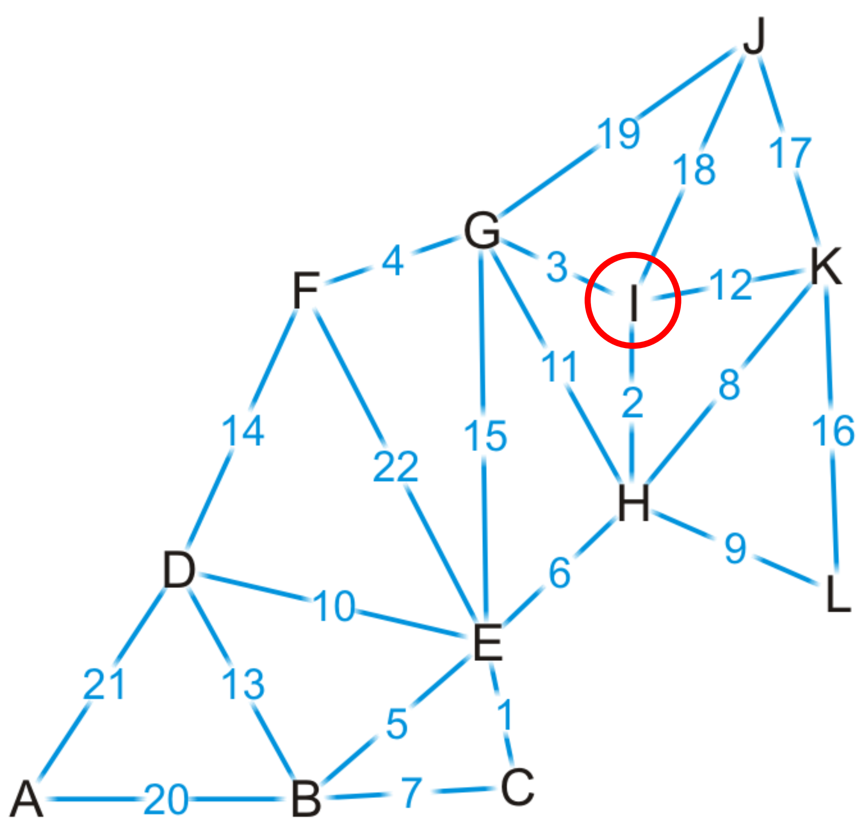


Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	∞	Ø
C	F	∞	Ø
D	F	∞	Ø
E	F	14	H
F	F	∞	Ø
G	F	19	H
H	T	8	K
I	F	10	H
J	F	17	K
K	T	0	Ø
L	F	16	K

Example

The path (K, H, I) is the shortest path from K to I of length 10

– Vertex I has two unvisited neighbors: G and J



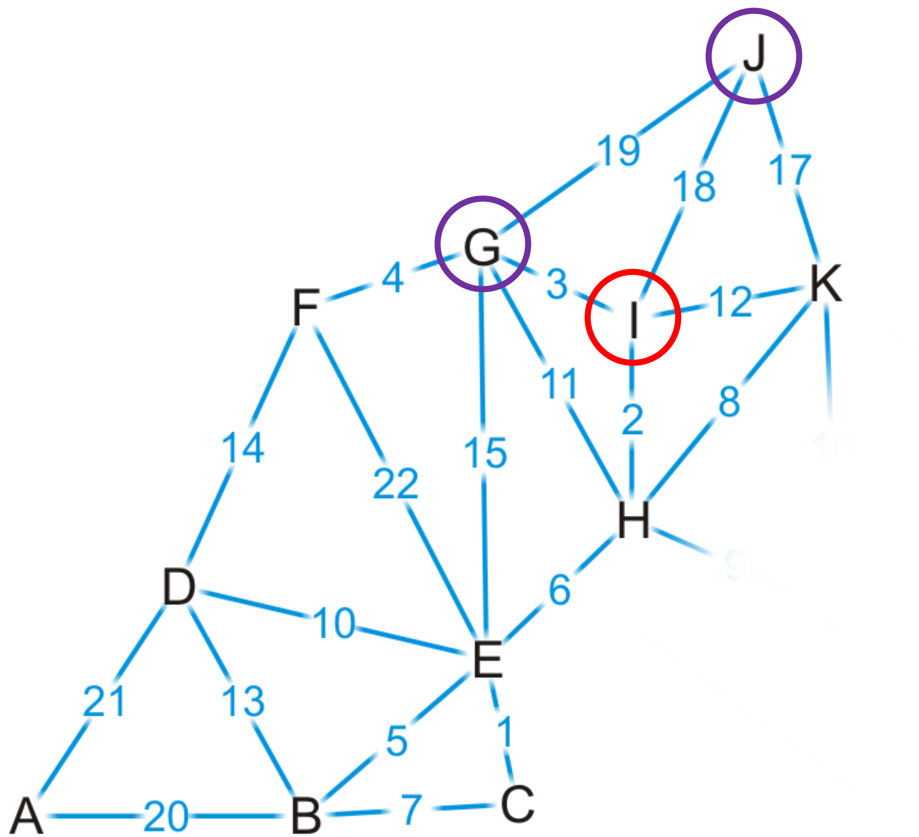
Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	∞	Ø
C	F	∞	Ø
D	F	∞	Ø
E	F	14	H
F	F	∞	Ø
G	F	19	H
H	T	8	K
I	**T**	**10**	**H**
J	F	17	K
K	T	0	Ø
L	F	16	K

Example

Consider these paths:

(K, H, I, G) of length 10 + 3 = 13

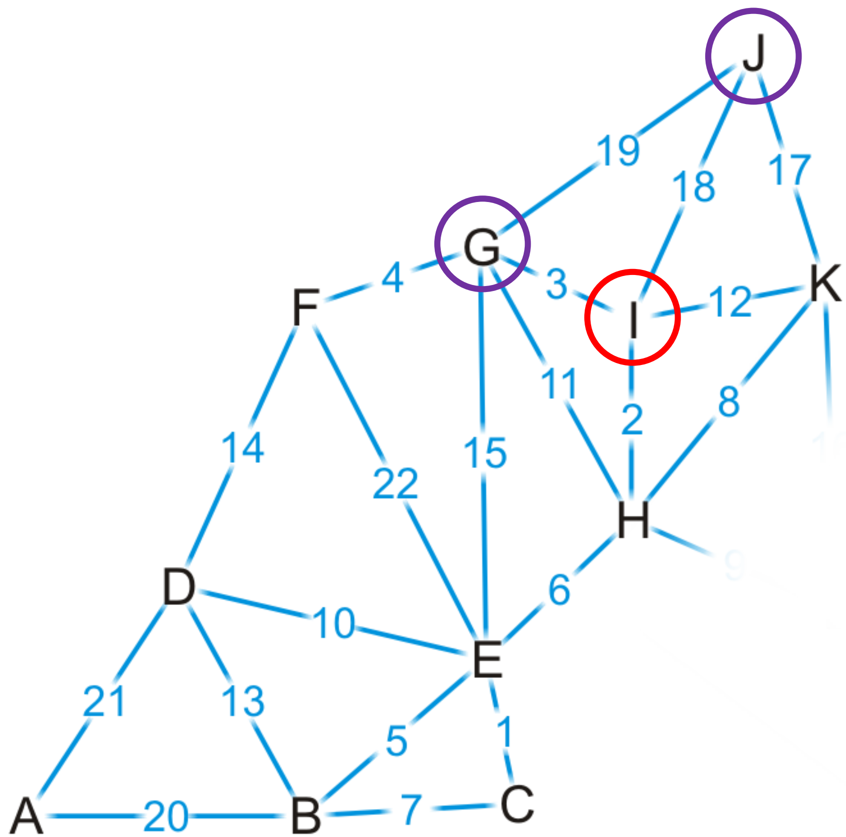
(K, H, I, J) of length 10 + 18 = 28



Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	∞	Ø
C	F	∞	Ø
D	F	∞	Ø
E	F	14	H
F	F	∞	Ø
G	F	19	H
H	T	8	K
I	**T**	**10**	**H**
J	F	17	K
K	T	0	Ø
L	F	16	K

Example

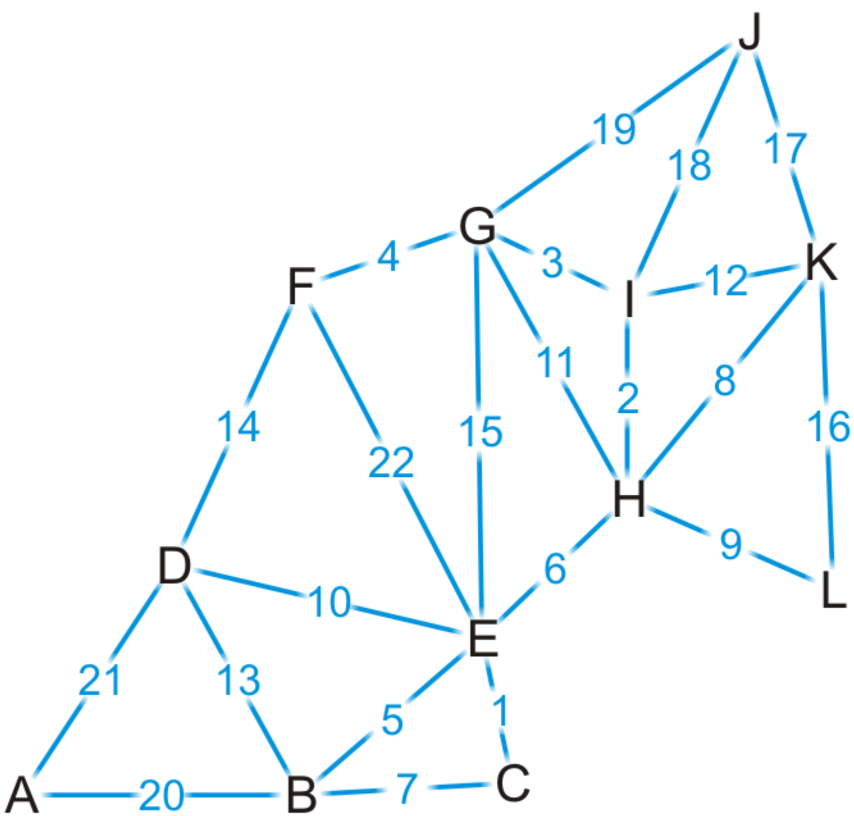
We have discovered a shorter path to vertex G, but (K, J) is still the shortest known path to vertex J



Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	∞	Ø
C	F	∞	Ø
D	F	∞	Ø
E	F	14	H
F	F	∞	Ø
G	F	**13**	**I**
H	T	8	K
I	**T**	**10**	**H**
J	F	17	K
K	T	0	Ø
L	F	16	K

Example

Which vertex can we visit next?

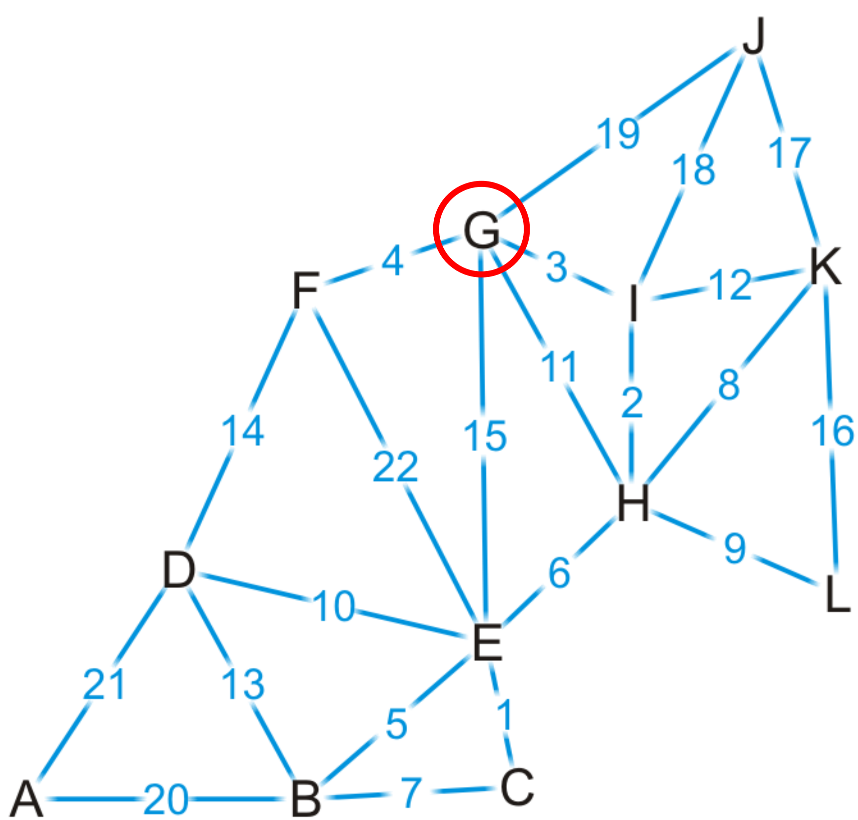


Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	∞	Ø
C	F	∞	Ø
D	F	∞	Ø
E	F	14	H
F	F	∞	Ø
G	F	13	I
H	T	8	K
I	T	10	H
J	F	17	K
K	T	0	Ø
L	F	16	K

Example

The path (K, H, I, G) is the shortest path from K to G of length 13

– Vertex G has three unvisited neighbors: E, F and J



Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	∞	Ø
C	F	∞	Ø
D	F	∞	Ø
E	F	14	H
F	F	∞	Ø
G	**T**	**13**	**I**
H	T	8	K
I	T	10	H
J	F	17	K
K	T	0	Ø
L	F	16	K

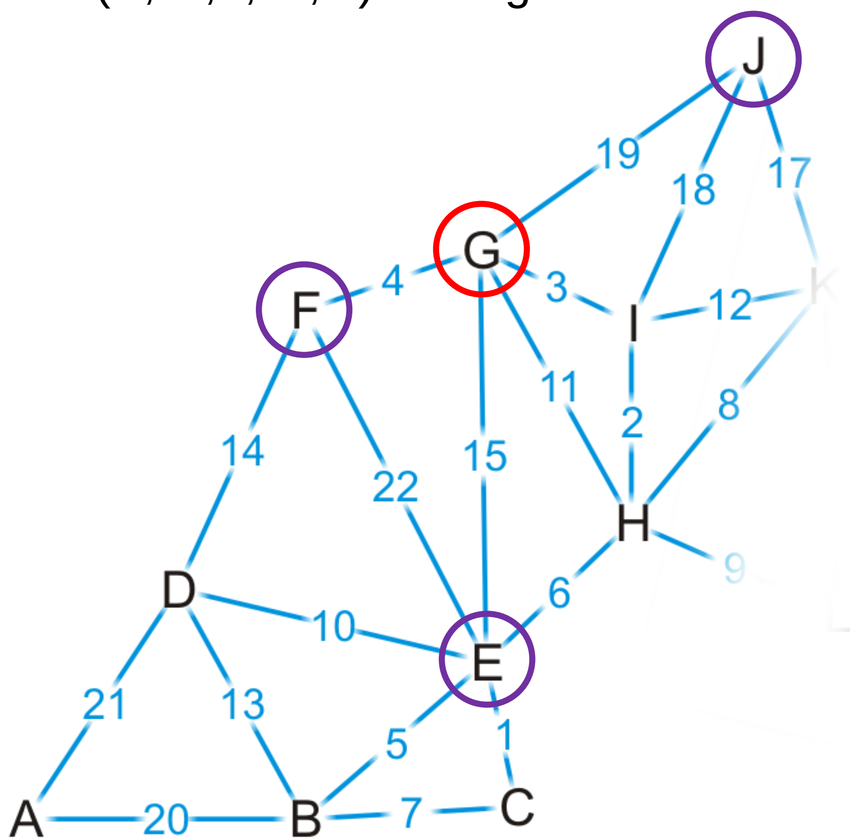
Example

Consider these paths:

(K, H, I, G, E) of length 13 + 15 = 28

(K, H, I, G, F) of length 13 + 4 = 17

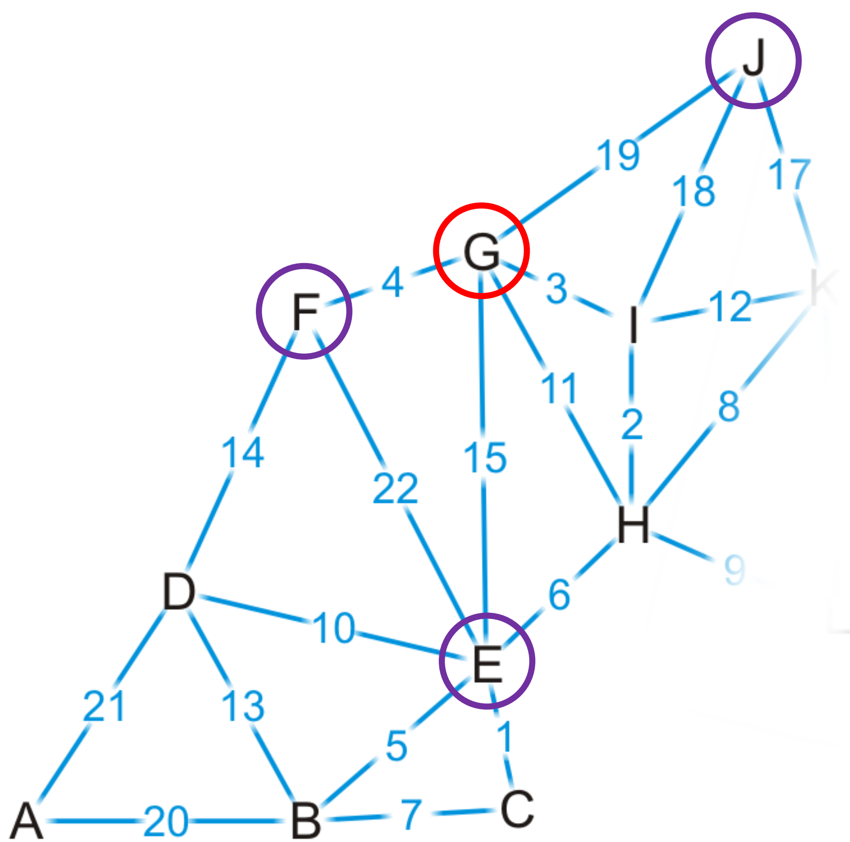
(K, H, I, G, J) of length 13 + 19 = 32



Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	∞	Ø
C	F	∞	Ø
D	F	∞	Ø
E	F	14	H
F	F	∞	Ø
G	**T**	**13**	**I**
H	T	8	K
I	T	10	H
J	F	17	K
K	T	0	Ø
L	F	16	K

Example

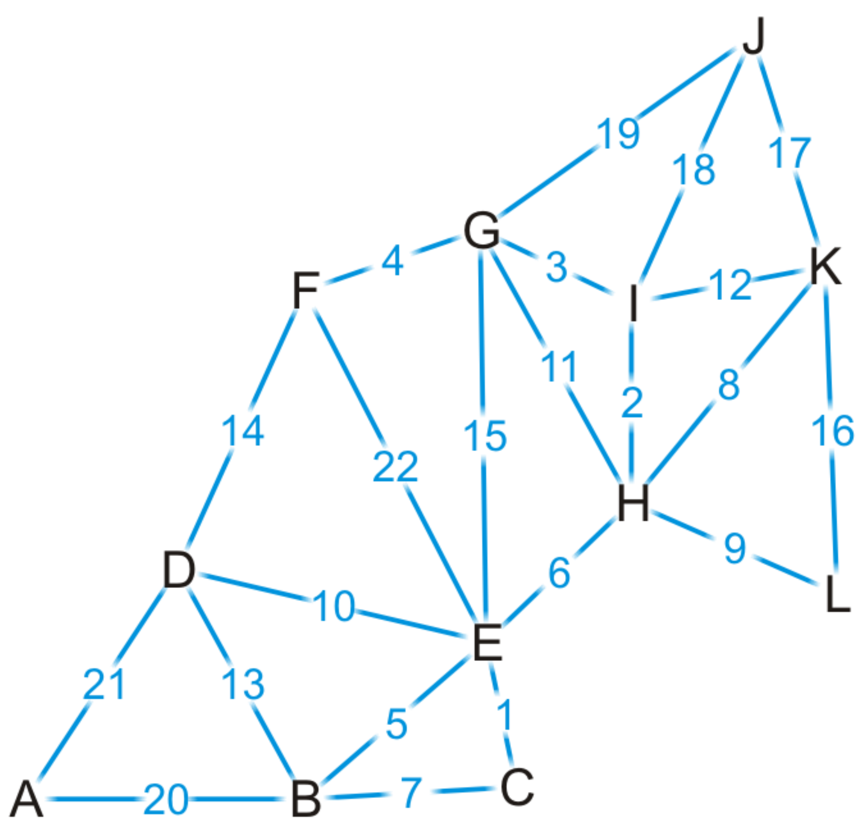
We have now found a path to vertex F



Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	∞	Ø
C	F	∞	Ø
D	F	∞	Ø
E	F	14	H
F	F	**17**	**G**
G	**T**	**13**	**I**
H	T	8	K
I	T	10	H
J	F	17	K
K	T	0	Ø
L	F	16	K

Example

Where do we visit next?

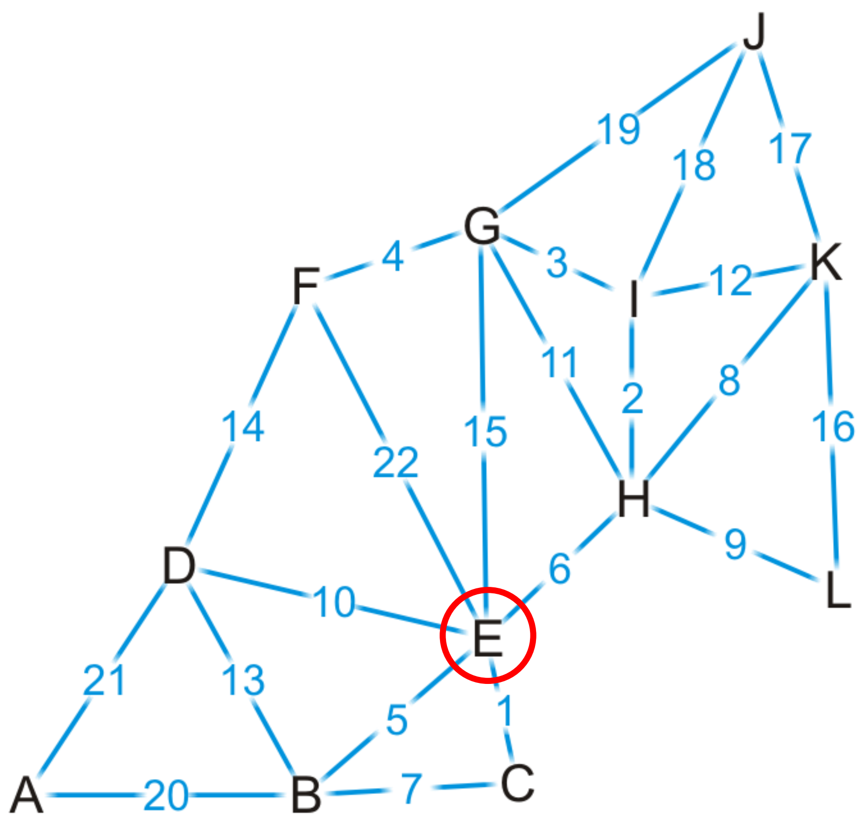


Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	∞	Ø
C	F	∞	Ø
D	F	∞	Ø
E	F	14	H
F	F	17	G
G	T	13	I
H	T	8	K
I	T	10	H
J	F	17	K
K	T	0	Ø
L	F	16	K

Example

The path (K, H, E) is the shortest path from K to E of length 14

– Vertex G has four unvisited neighbors: B, C, D and F

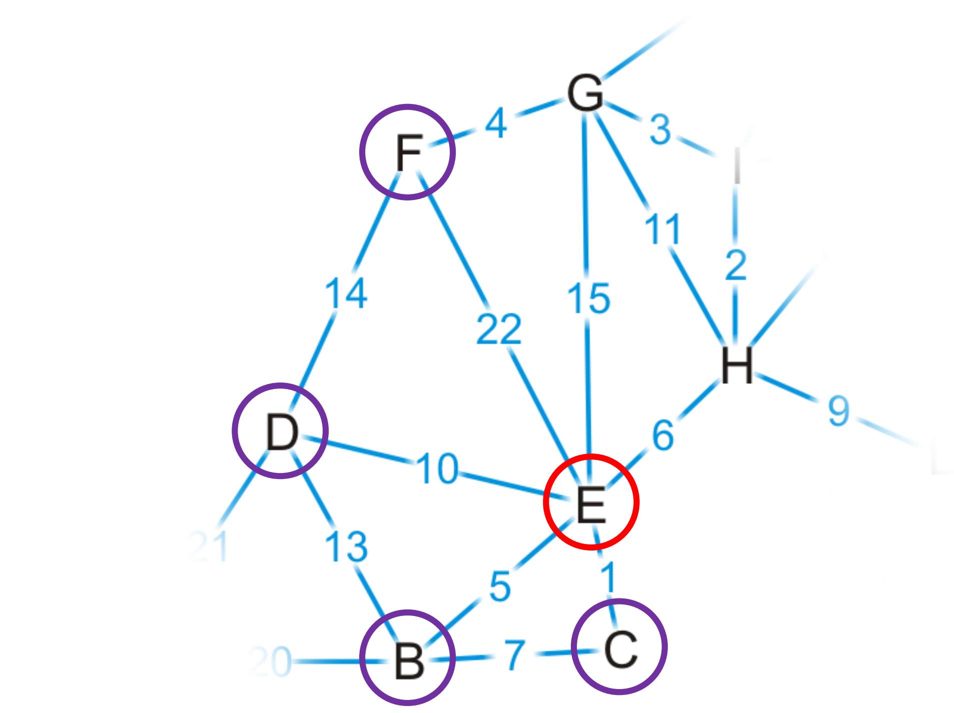


Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	∞	Ø
C	F	∞	Ø
D	F	∞	Ø
E	**T**	**14**	**H**
F	F	17	G
G	T	13	I
H	T	8	K
I	T	10	H
J	F	17	K
K	T	0	Ø
L	F	16	K

Example

The path (K, H, E) is the shortest path from K to E of length 14

– Vertex G has four unvisited neighbors: B, C, D and F



Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	∞	Ø
C	F	∞	Ø
D	F	∞	Ø
E	**T**	**14**	**H**
F	F	17	G
G	T	13	I
H	T	8	K
I	T	10	H
J	F	17	K
K	T	0	Ø
L	F	16	K

Example

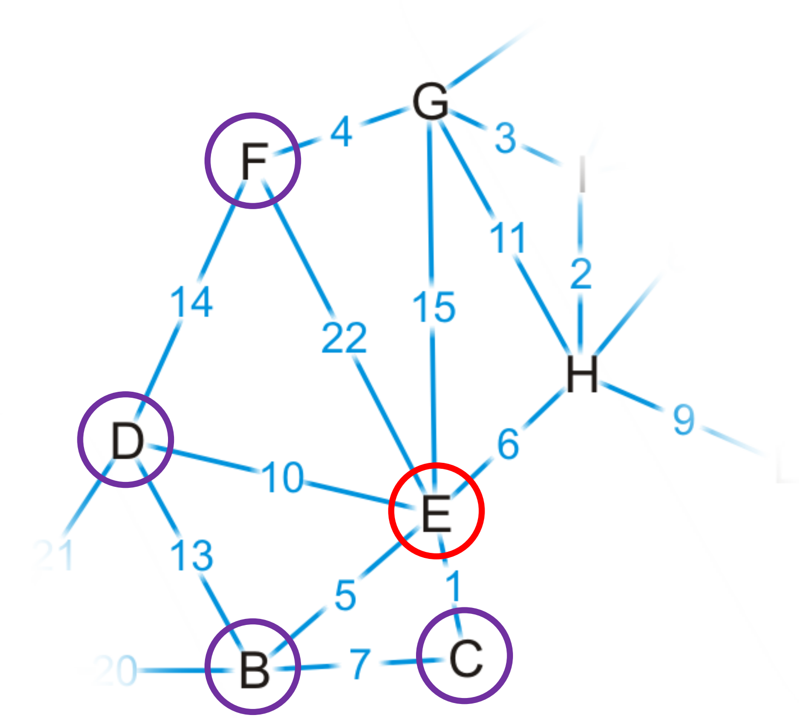
Consider these paths:

(K, H, E, B) of length 14 + 5 = 19

(K, H, E, C) of length 14 + 1 = 15

(K, H, E, D) of length 14 + 10 = 24

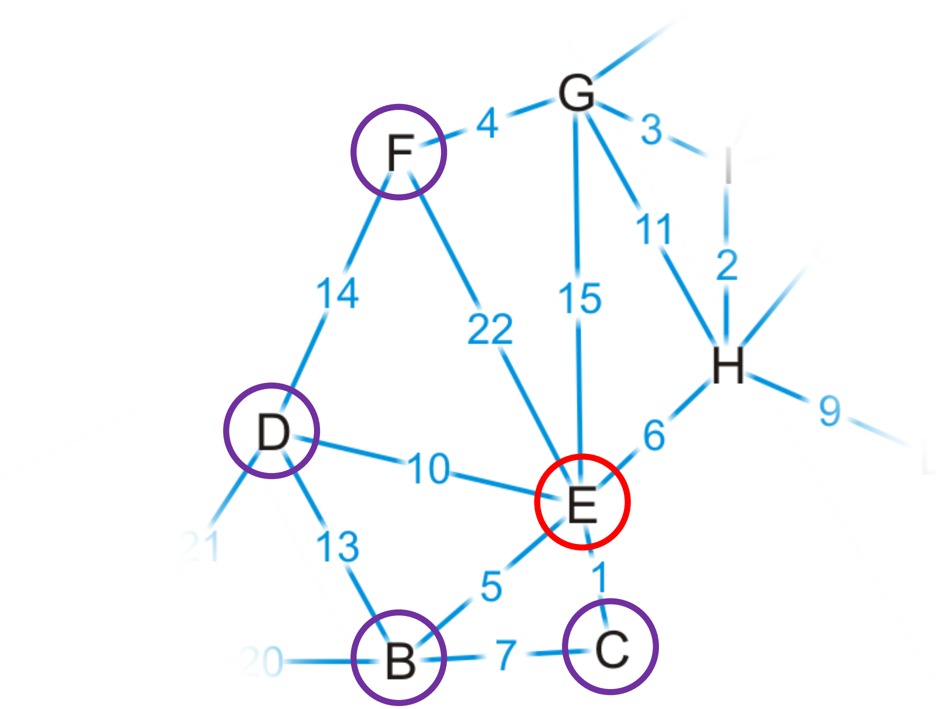
(K, H, E, F) of length 14 + 22 = 36



Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	∞	Ø
C	F	∞	Ø
D	F	∞	Ø
E	**T**	**14**	**H**
F	F	17	G
G	T	13	I
H	T	8	K
I	T	10	H
J	F	17	K
K	T	0	Ø
L	F	16	K

Example

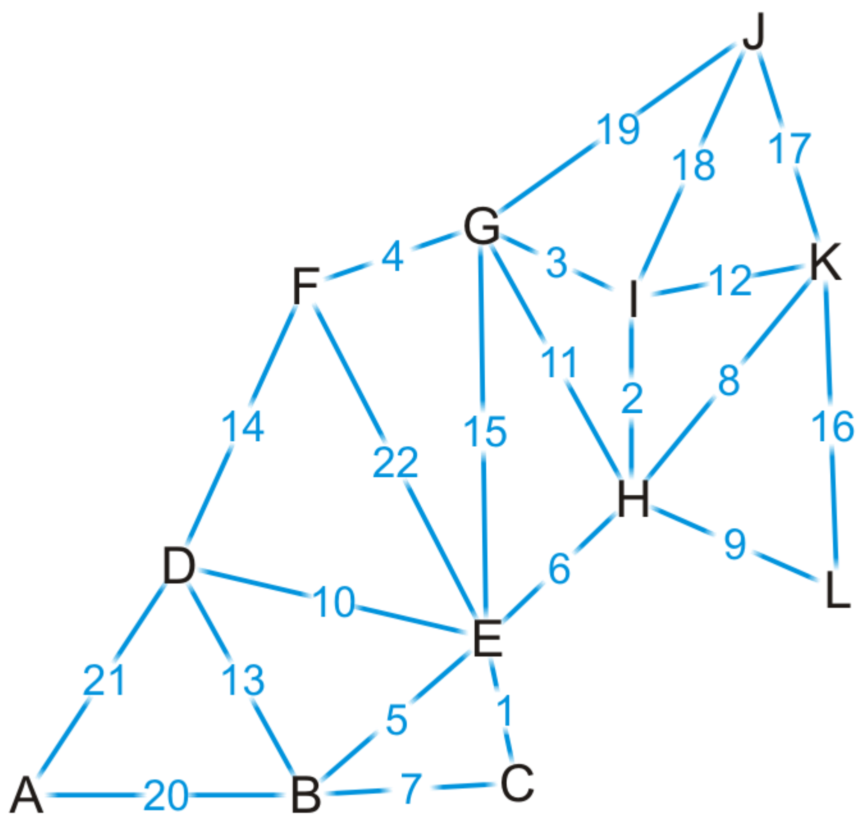
We've discovered paths to vertices B, C, D



Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	**19**	**E**
C	F	**15**	**E**
D	F	**24**	**E**
E	**T**	**14**	**H**
F	F	17	G
G	T	13	I
H	T	8	K
I	T	10	H
J	F	17	K
K	T	0	Ø
L	F	16	K

Example

Which vertex is next?

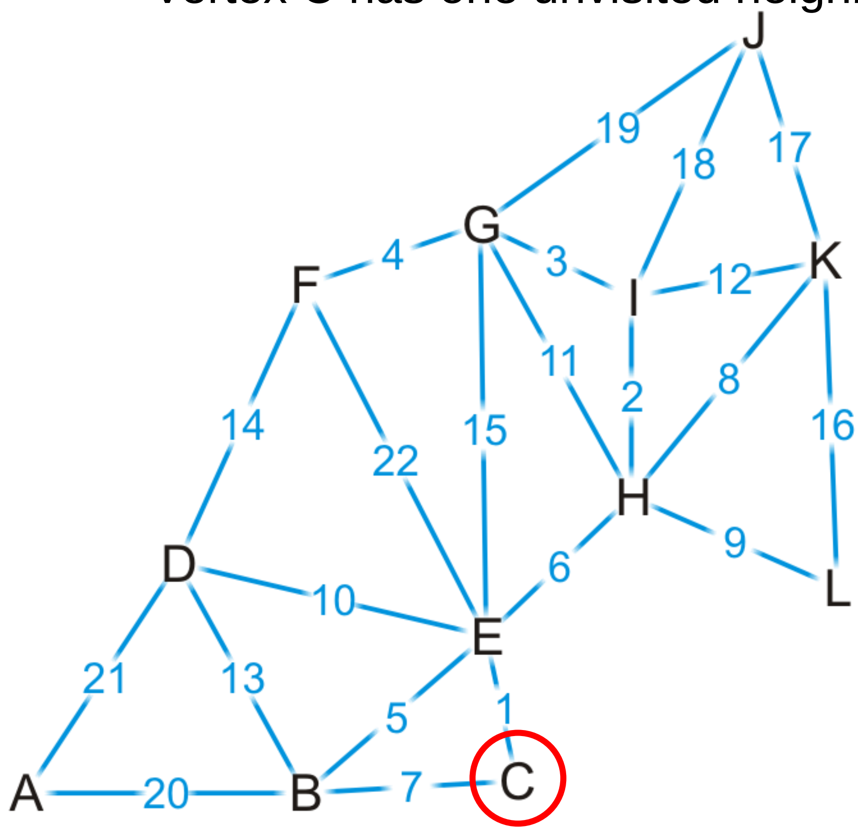


Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	19	E
C	F	15	E
D	F	24	E
E	T	14	H
F	F	17	G
G	T	13	I
H	T	8	K
I	T	10	H
J	F	17	K
K	T	0	Ø
L	F	16	K

Example

We've found that the path (K, H, E, C) of length 15 is the shortest path from K to C

– Vertex C has one unvisited neighbor, B

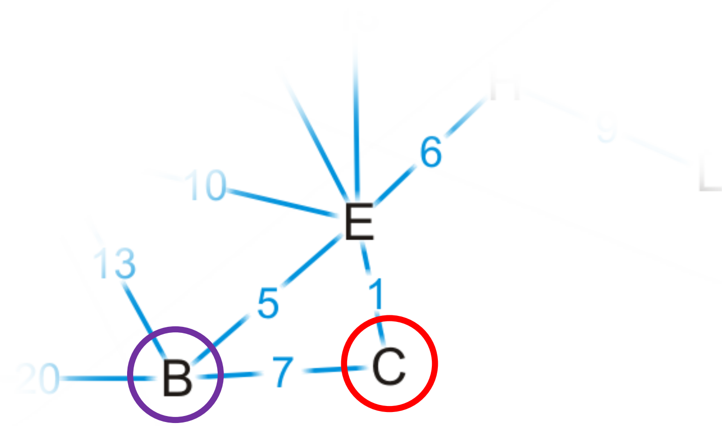


Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	19	E
C	**T**	**15**	**E**
D	F	24	E
E	T	14	H
F	F	17	G
G	T	13	I
H	T	8	K
I	T	10	H
J	F	17	K
K	T	0	Ø
L	F	16	K

Example

The path (K, H, E, C, B) is of length 15 + 7 = 22

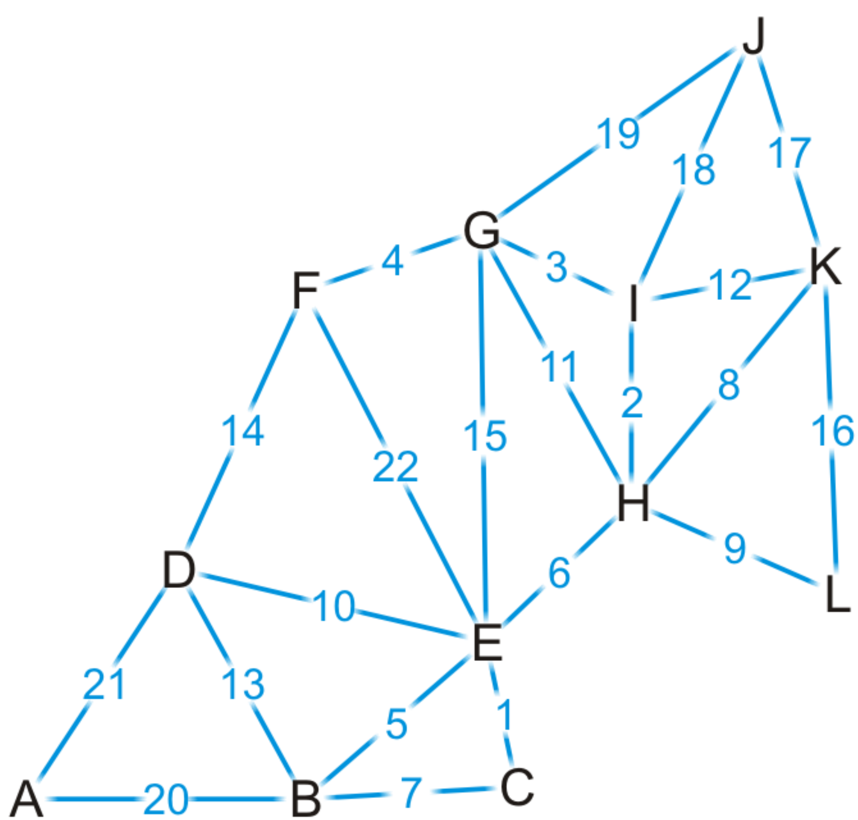
– We have already discovered a shorter path through vertex E



Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	19	E
C	**T**	**15**	**E**
D	F	24	E
E	T	14	H
F	F	17	G
G	T	13	I
H	T	8	K
I	T	10	H
J	F	17	K
K	T	0	Ø
L	F	16	K

Example

Where to next?

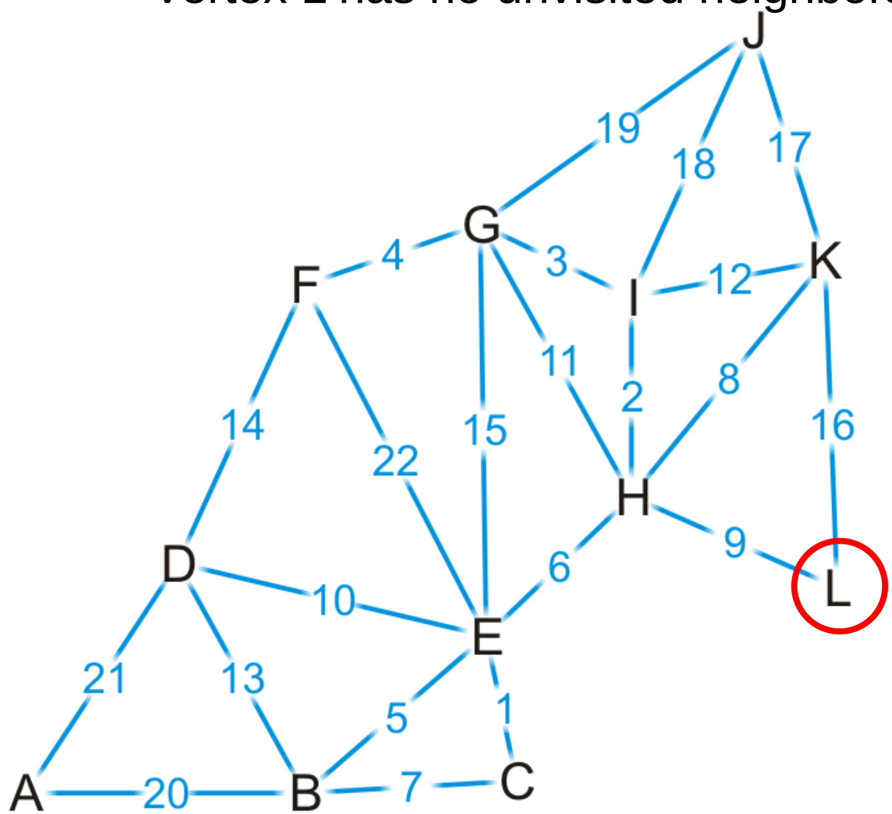


Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	19	E
C	T	15	E
D	F	24	E
E	T	14	H
F	F	17	G
G	T	13	I
H	T	8	K
I	T	10	H
J	F	17	K
K	T	0	Ø
L	F	16	K

Example

We now know that (K, L) is the shortest path between these two points

- Vertex L has no unvisited neighbors

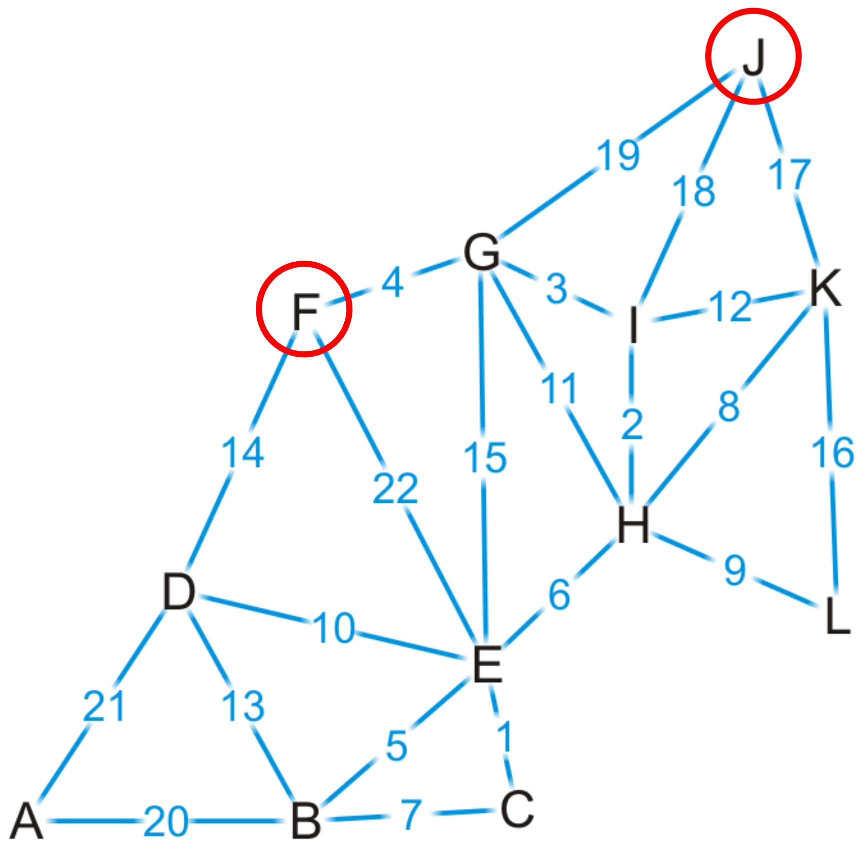


Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	19	E
C	T	15	E
D	F	24	E
E	T	14	H
F	F	17	G
G	T	13	I
H	T	8	K
I	T	10	H
J	F	17	K
K	T	0	Ø
L	**T**	**16**	**K**

Example

Where to next?

– Does it matter if we visit vertex F first or vertex J first?

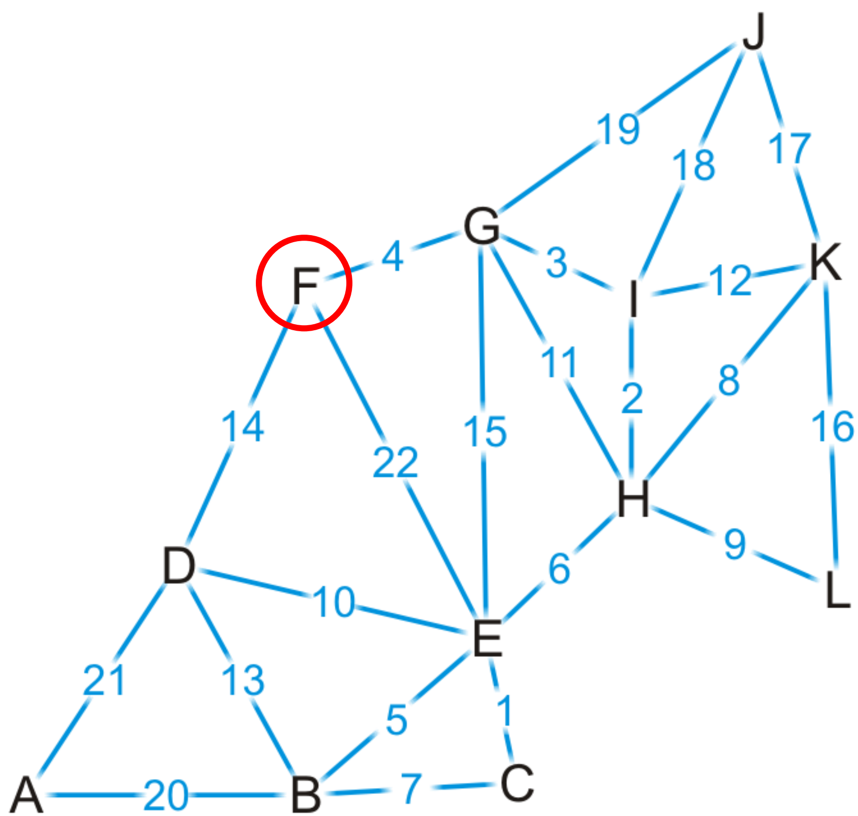


Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	19	E
C	T	15	E
D	F	24	E
E	T	14	H
F	F	17	G
G	T	13	I
H	T	8	K
I	T	10	H
J	F	17	K
K	T	0	Ø
L	T	16	K

Example

Let's visit vertex F first

– It has one unvisited neighbor, vertex D

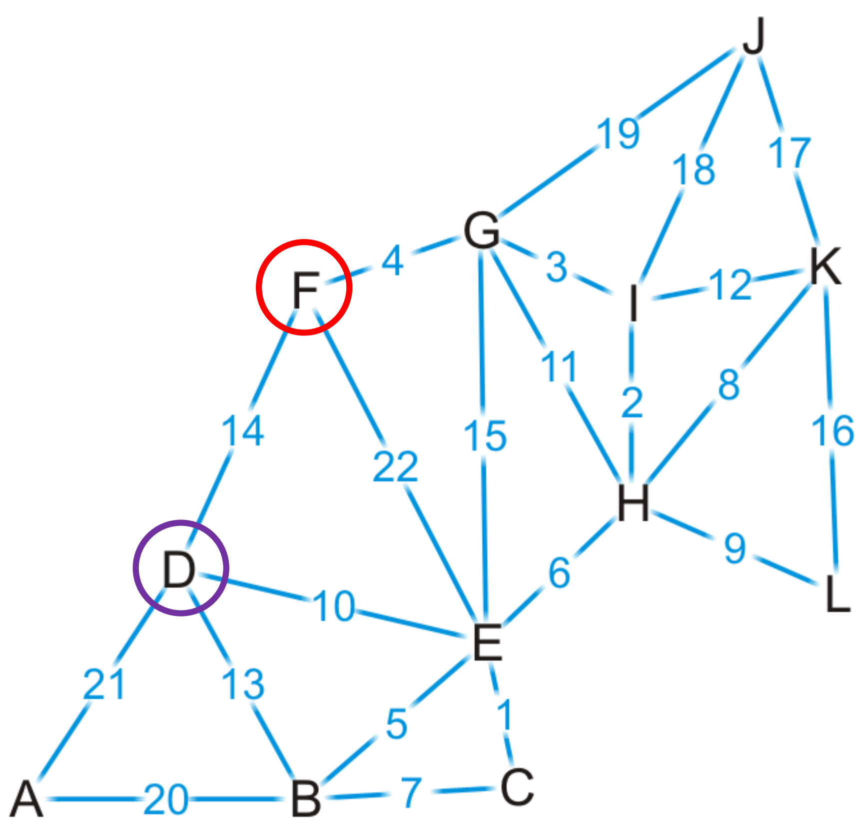


Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	19	E
C	T	15	E
D	F	24	E
E	T	14	H
F	**T**	**17**	**G**
G	T	13	I
H	T	8	K
I	T	10	H
J	F	17	K
K	T	0	Ø
L	T	16	K

Example

The path (K, H, I, G, F, D) is of length 17 + 14 = 31

– This is longer than the path we've already discovered

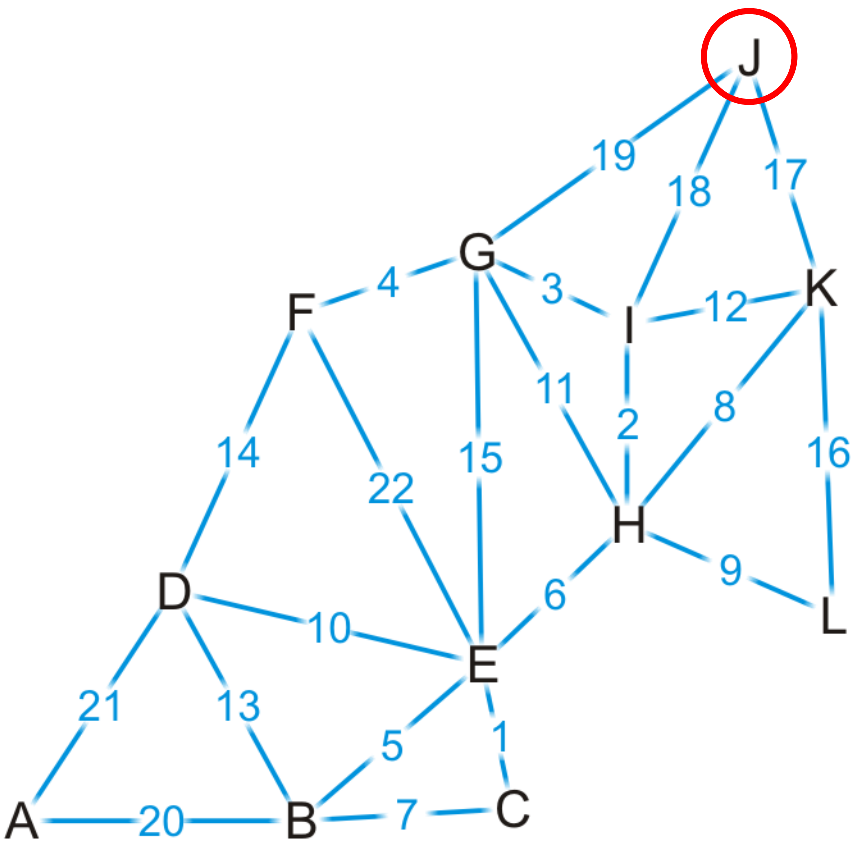


Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	19	E
C	T	15	E
D	F	24	E
E	T	14	H
F	**T**	**17**	**G**
G	T	13	I
H	T	8	K
I	T	10	H
J	F	17	K
K	T	0	Ø
L	T	16	K

Example

Now we visit vertex J

– It has no unvisited neighbors



Vertex	Visited	Distance	Previous
A	F	∞	Ø
B	F	19	E
C	T	15	E
D	F	24	E
E	T	14	H
F	T	17	G
G	T	13	I
H	T	8	K
I	T	10	H
J	**T**	**17**	**K**
K	T	0	Ø
L	T	16	K

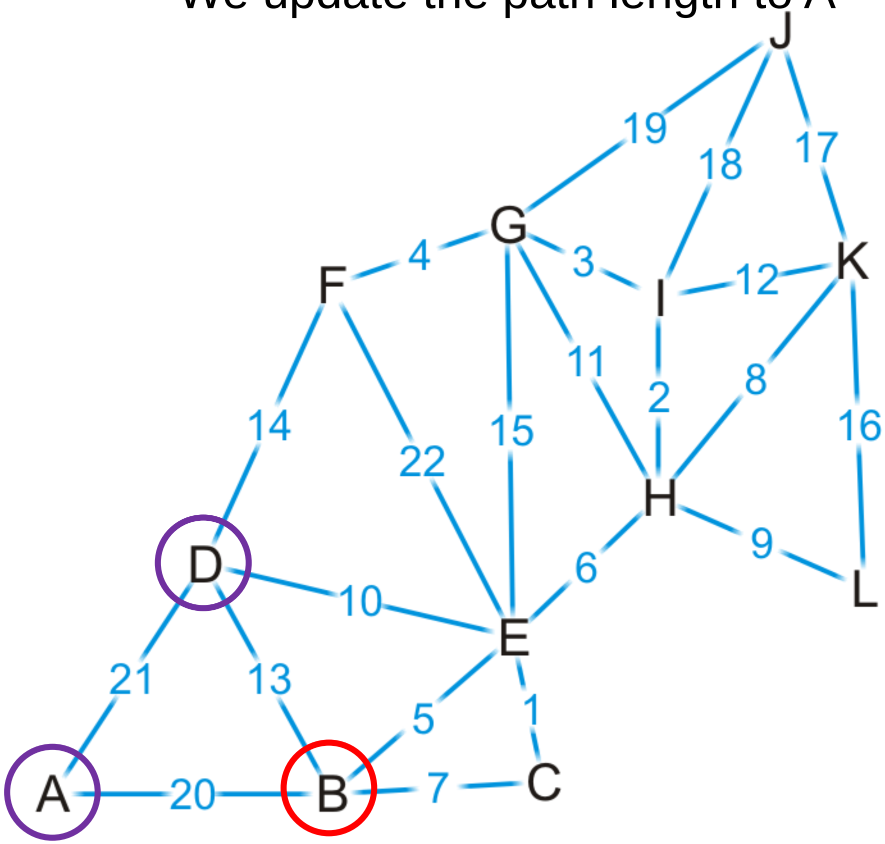
Example

Next we visit vertex B, which has two unvisited neighbors:

(K, H, E, B, A) of length 19 + 20 = 39

(K, H, E, B, D) of length 19 + 13 = 32

– We update the path length to A

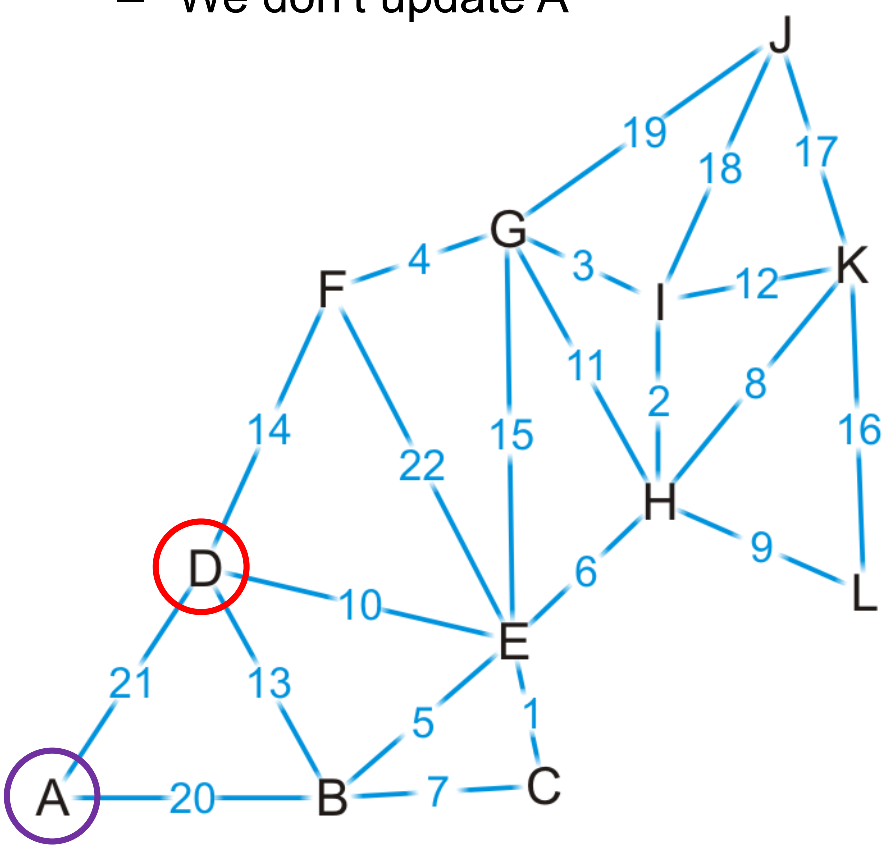


Vertex	Visited	Distance	Previous
A	F	**39**	**B**
B	**T**	**19**	**E**
C	T	15	E
D	F	24	E
E	T	14	H
F	T	17	G
G	T	13	I
H	T	8	K
I	T	10	H
J	T	17	K
K	T	0	Ø
L	T	16	K

Example

Next we visit vertex D

- The path (K, H, E, D, A) is of length 24 + 21 = 45
- We don't update A

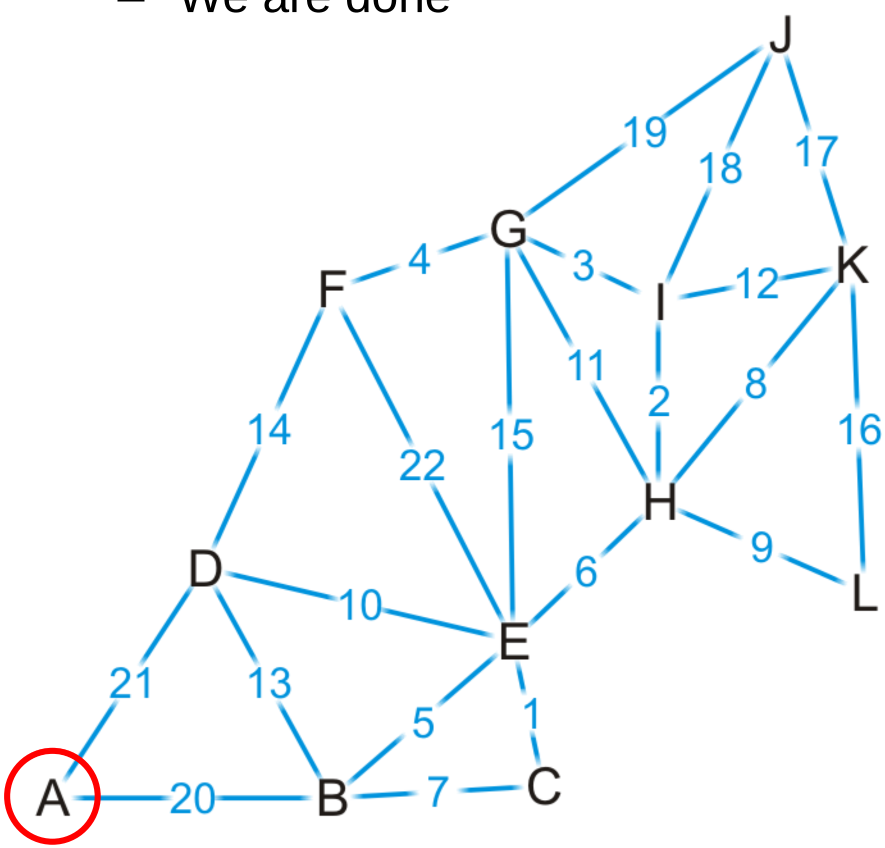


Vertex	Visited	Distance	Previous
A	F	39	B
B	T	19	E
C	T	15	E
D	**T**	**24**	**E**
E	T	14	H
F	T	17	G
G	T	13	I
H	T	8	K
I	T	10	H
J	T	17	K
K	T	0	Ø
L	T	16	K

Example

Finally, we visit vertex A

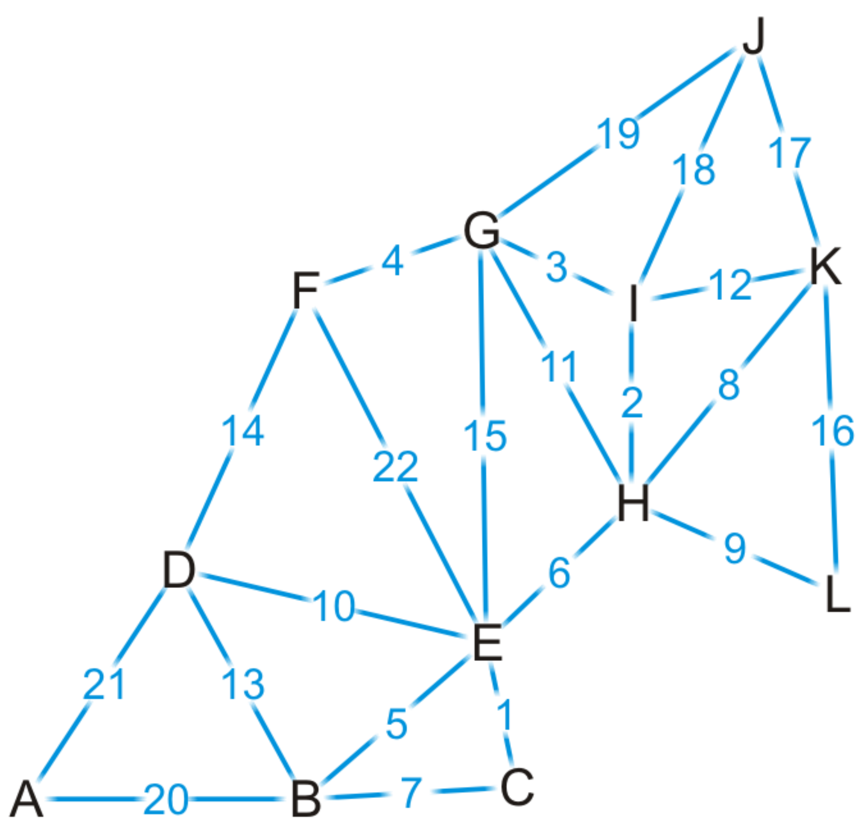
- It has no unvisited neighbors and there are no unvisited vertices left
- We are done



Vertex	Visited	Distance	Previous
A	**T**	**39**	**B**
B	T	19	E
C	T	15	E
D	T	24	E
E	T	14	H
F	T	17	G
G	T	13	I
H	T	8	K
I	T	10	H
J	T	17	K
K	T	0	Ø
L	T	16	K

Example

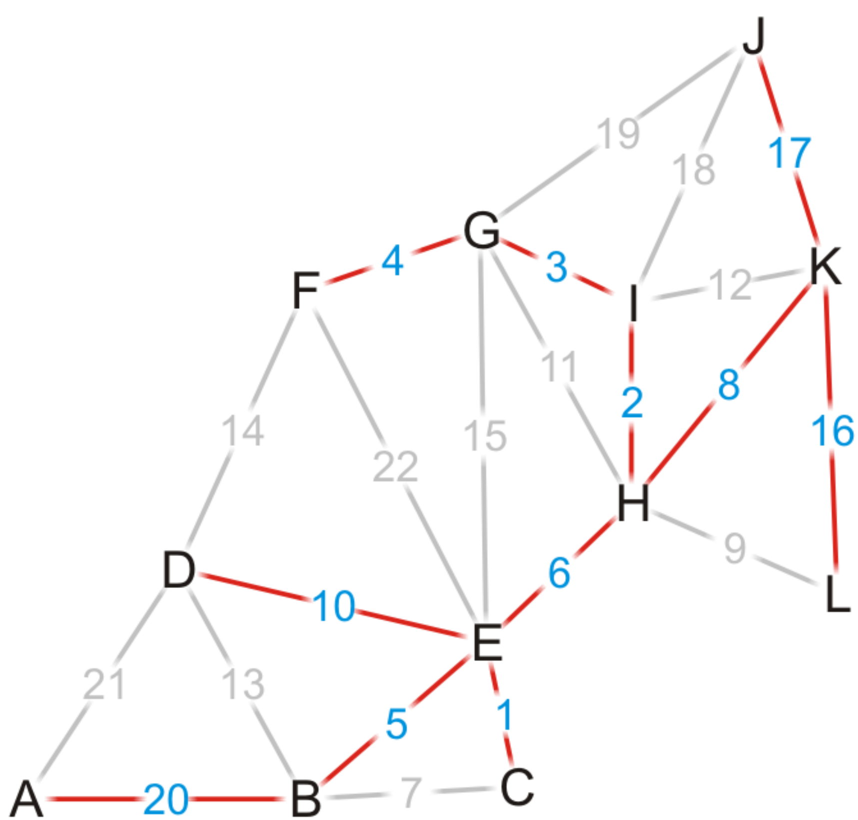
Thus, we have found the shortest path from vertex K to each of the other vertices



Vertex	Visited	Distance	Previous
A	T	39	B
B	T	19	E
C	T	15	E
D	T	24	E
E	T	14	H
F	T	17	G
G	T	13	I
H	T	8	K
I	T	10	H
J	T	17	K
K	T	0	Ø
L	T	16	K

Example

Using the *previous* pointers, we can reconstruct the paths



Vertex	Visited	Distance	Previous
A	T	39	B
B	T	19	E
C	T	15	E
D	T	24	E
E	T	14	H
F	T	17	G
G	T	13	I
H	T	8	K
I	T	10	H
J	T	17	K
K	T	0	Ø
L	T	16	K

Comments on Dijkstra's algorithm

Questions:

- Q1. What if at some point, all unvisited vertices have a distance ∞?
- Q2. What if we just want to find the shortest path from v_j to v_k?
- Q3. Does the algorithm change if we have a directed graph?

Runtime Analysis

The runtime of Dijkstra's algorithm is the same as Prim's algorithm

- With an adjacency matrix, the run time is $\Theta(|V|(|V| + |V|)) = \Theta(|V|^2)$
- With an adjacency list, the run time is $\Theta(|V|^2 + |E|) = \Theta(|V|^2)$ as $|E| = \mathrm{O}(|V|^2)$

```
for _ in range(|V|-1): // until visiting all vertices
    // visit the closest vertex
    v = table.find_min_dist_vertex()
    table.mark_visit(v)

    // update the shortest path if needed
    for j in graph.get_adj_vertices(v):
        if (table.get_dist(v) + graph.get_dist(v, j) < table.get_dist(j)):
            new_dist = table.get_dist(v) + graph.get_dist(v, j)
            table.set_dist(j, new_dist)
```

Again, using the binary heap may show the better runtime

– $\mathrm{O}(|V| \ln(|V|) + |E| \ln(|V|)) = \mathrm{O}(|E| \ln(|V|))$

Summary

We have seen an algorithm for finding single-source shortest paths
- Start with the initial vertex
- Continue finding the next vertex that is closest

Dijkstra's algorithm always finds the next closest vertex
- It solves the problem in $O(|E| \ln(|V|))$ time

References

Cormen, Leiserson, Rivest and Stein,
Introduction to Algorithms, The MIT Press, 2001, §25.2, pp.629-35.

Mark A. Weiss,
Data Structures and Algorithm Analysis in C++, 3rd Ed., Addison Wesley, 2006, Ch.?, p.?.

Joh Kleinberg and Eva Tardos,
Algorithm Design, Pearson, 2006.

Elliot B. Koffman and Paul A.T. Wolfgang,
Objects, Abstractions, Data Structures and Design using C++, Wiley, 2006.