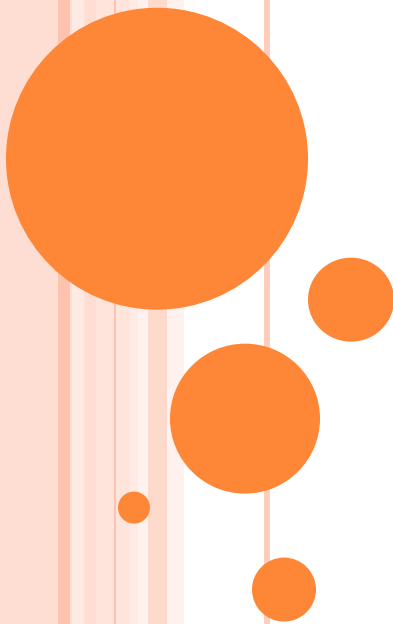


DIJKSTRA'S ALGORITHM

By Laksman Veeravagu and Luis Barrera



THE AUTHOR: EDSGER WYBE DIJKSTRA



"Computer Science is no more about computers than astronomy is about telescopes."

<http://www.cs.utexas.edu/~EWD/>



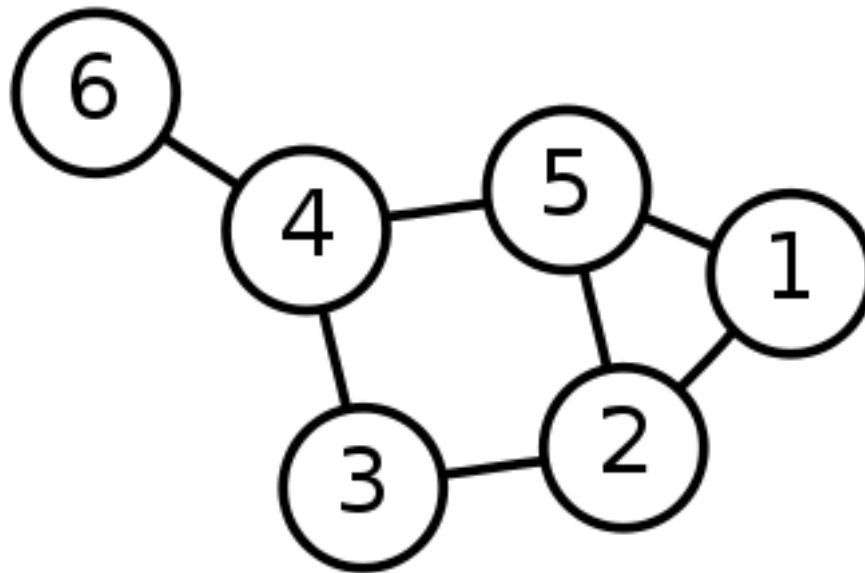
EDSGER WYBE DIJKSTRA

- May 11, 1930 – August 6, 2002
- Received the 1972 A. M. Turing Award, widely considered the most prestigious award in computer science.
- The Schlumberger Centennial Chair of Computer Sciences at The University of Texas at Austin from 1984 until 2000
- Made a strong case against use of the GOTO statement in programming languages and helped lead to its deprecation.
- Known for his many essays on programming.



SINGLE-SOURCE SHORTEST PATH PROBLEM

Single-Source Shortest Path Problem - The problem of finding shortest paths from a source vertex v to all other vertices in the graph.



DIJKSTRA'S ALGORITHM

Dijkstra's algorithm - is a solution to the single-source shortest path problem in graph theory.

Works on both directed and undirected graphs. However, all edges must have nonnegative weights.

Approach: Greedy

Input: Weighted graph $G=\{E,V\}$ and source vertex $v \in V$, such that all edge weights are nonnegative

Output: Lengths of shortest paths (or the shortest paths themselves) from a given source vertex $v \in V$ to all other vertices



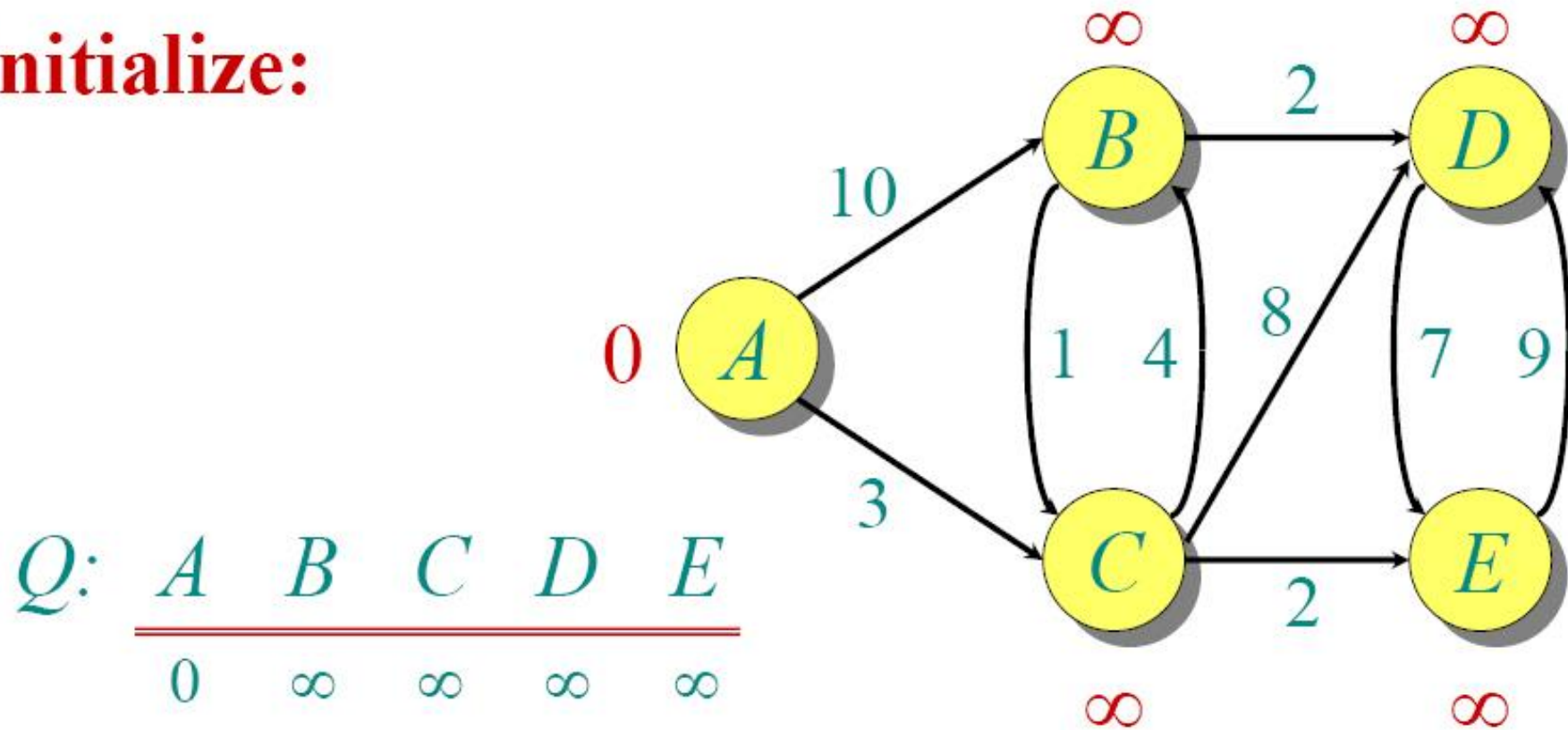
DIJKSTRA'S ALGORITHM - PSEUDOCODE

```
dist[s] ← 0                                (distance to source vertex is zero)
for all v ∈ V - {s}
    do dist[v] ← ∞                          (set all other distances to infinity)
S ← ∅                                        (S, the set of visited vertices is initially empty)
Q ← V                                       (Q, the queue initially contains all vertices)
while Q ≠ ∅                                (while the queue is not empty)
do u ← mindistance(Q, dist)                (select the element of Q with the min. distance)
    S ← S ∪ {u}                             (add u to list of visited vertices)
    for all v ∈ neighbors[u]
        do if dist[v] > dist[u] + w(u, v)    (if new shortest path found)
            then d[v] ← d[u] + w(u, v)      (set new value of shortest path)
            (if desired, add traceback code)
return dist
```



DIJKSTRA ANIMATED EXAMPLE

Initialize:



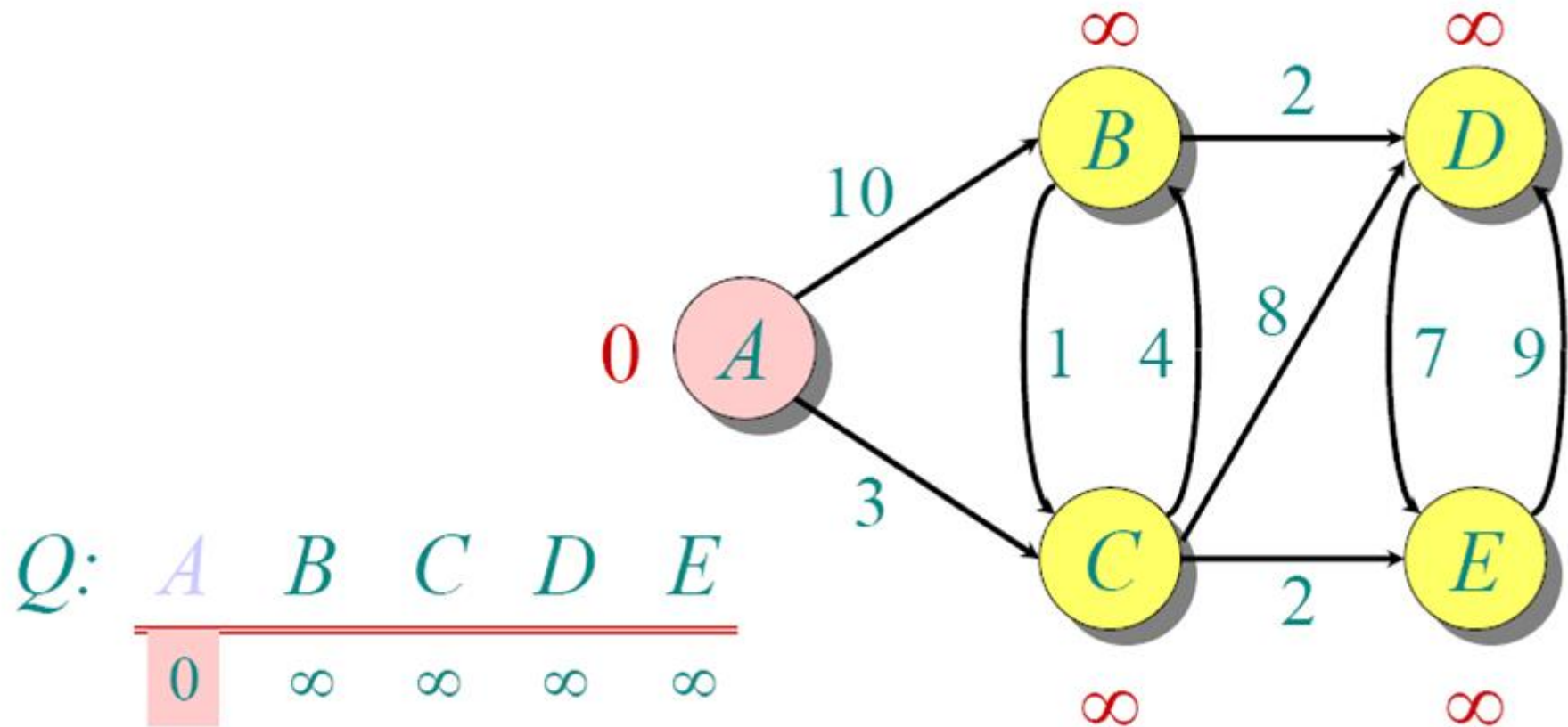
$Q:$

A	B	C	D	E
0	∞	∞	∞	∞

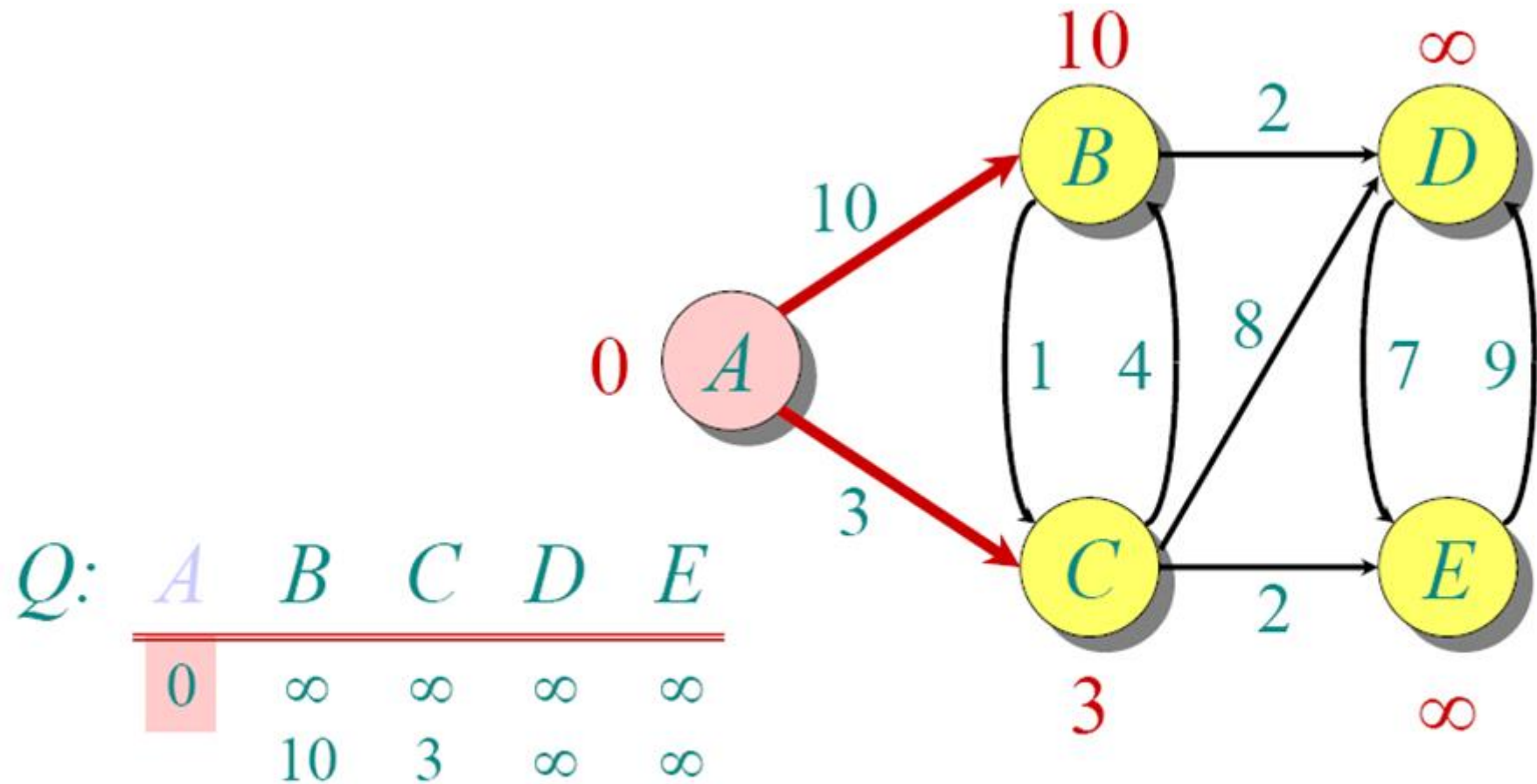
$S: \{\}$



DIJKSTRA ANIMATED EXAMPLE



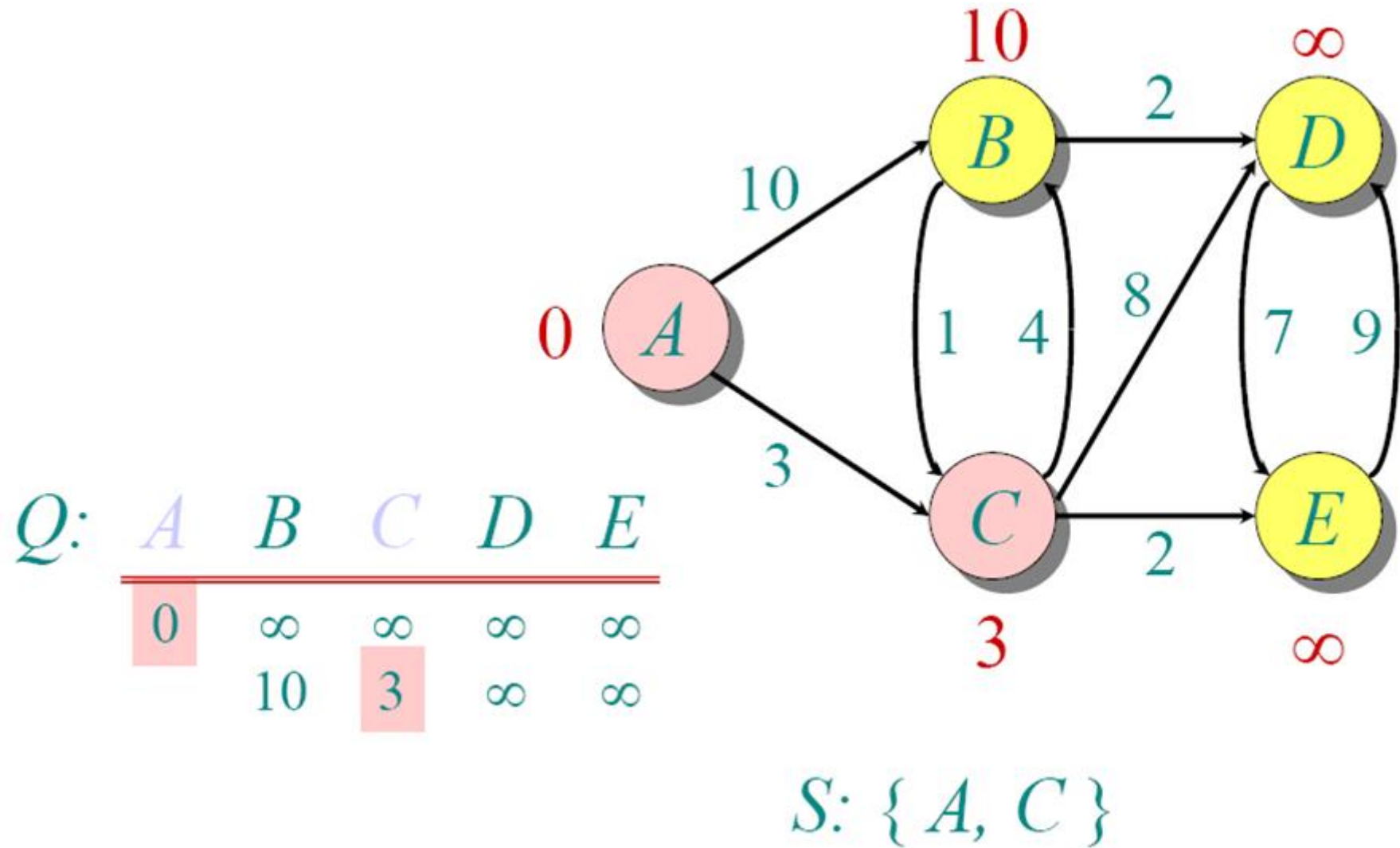
DIJKSTRA ANIMATED EXAMPLE



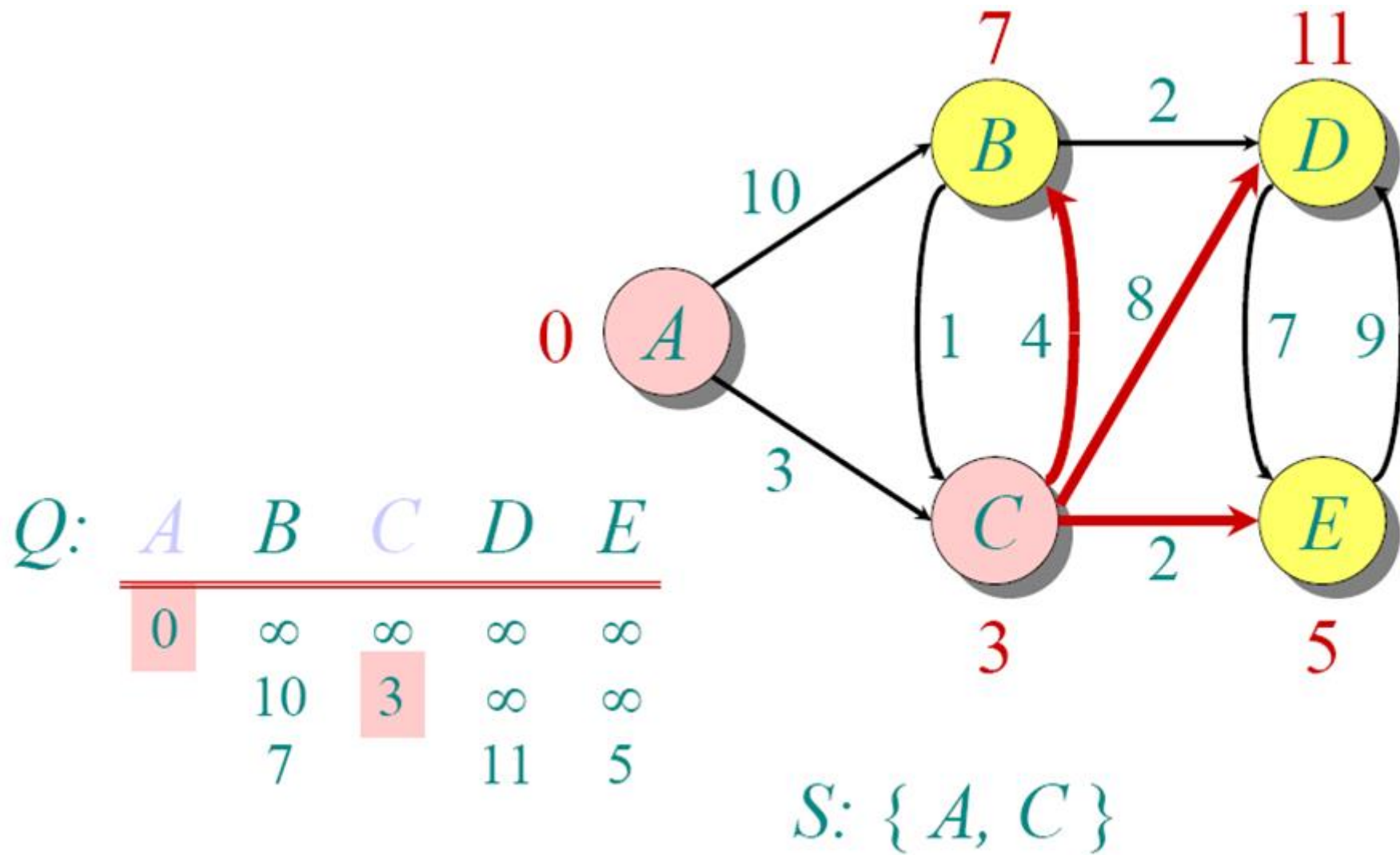
$S: \{ A \}$



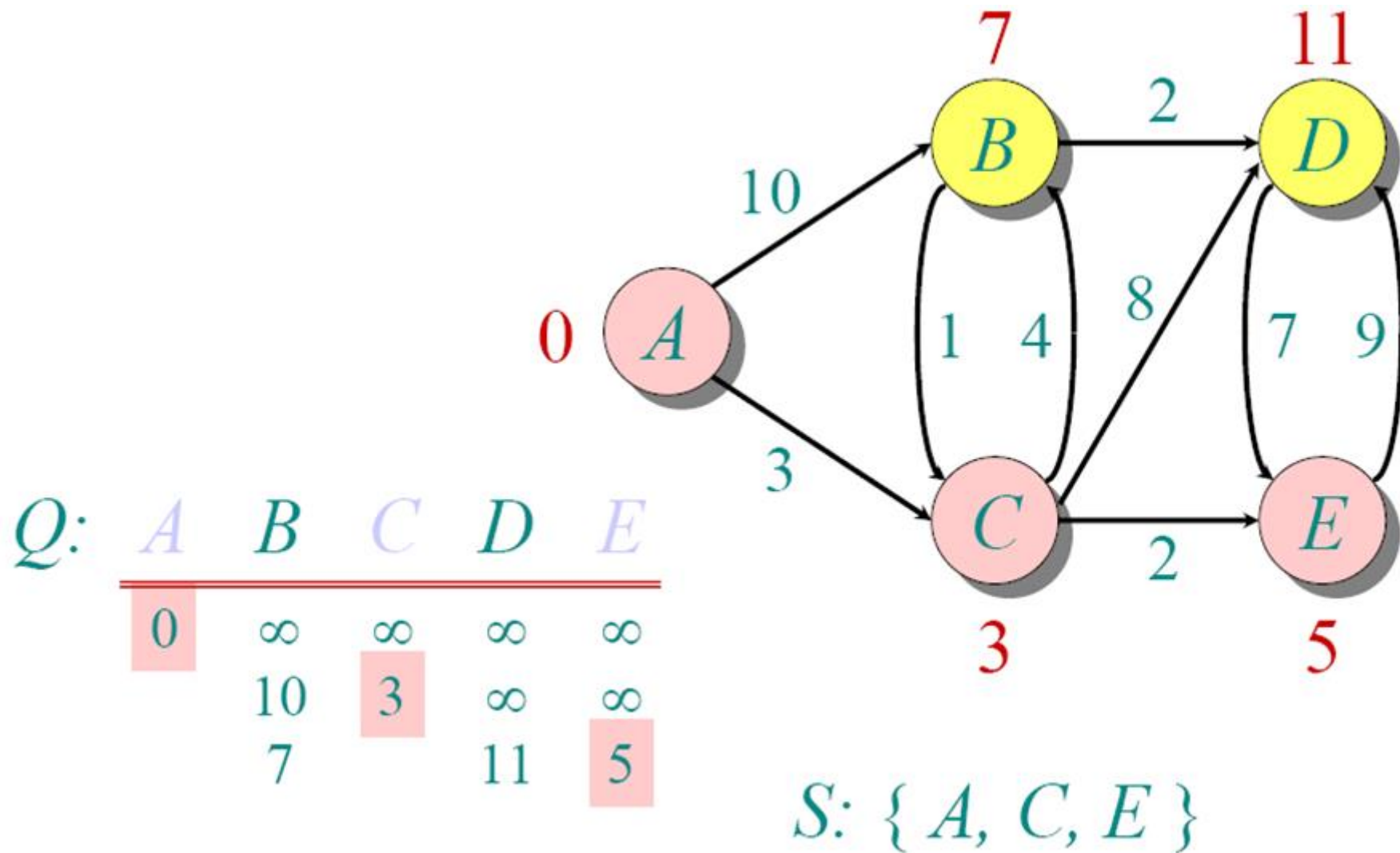
DIJKSTRA ANIMATED EXAMPLE



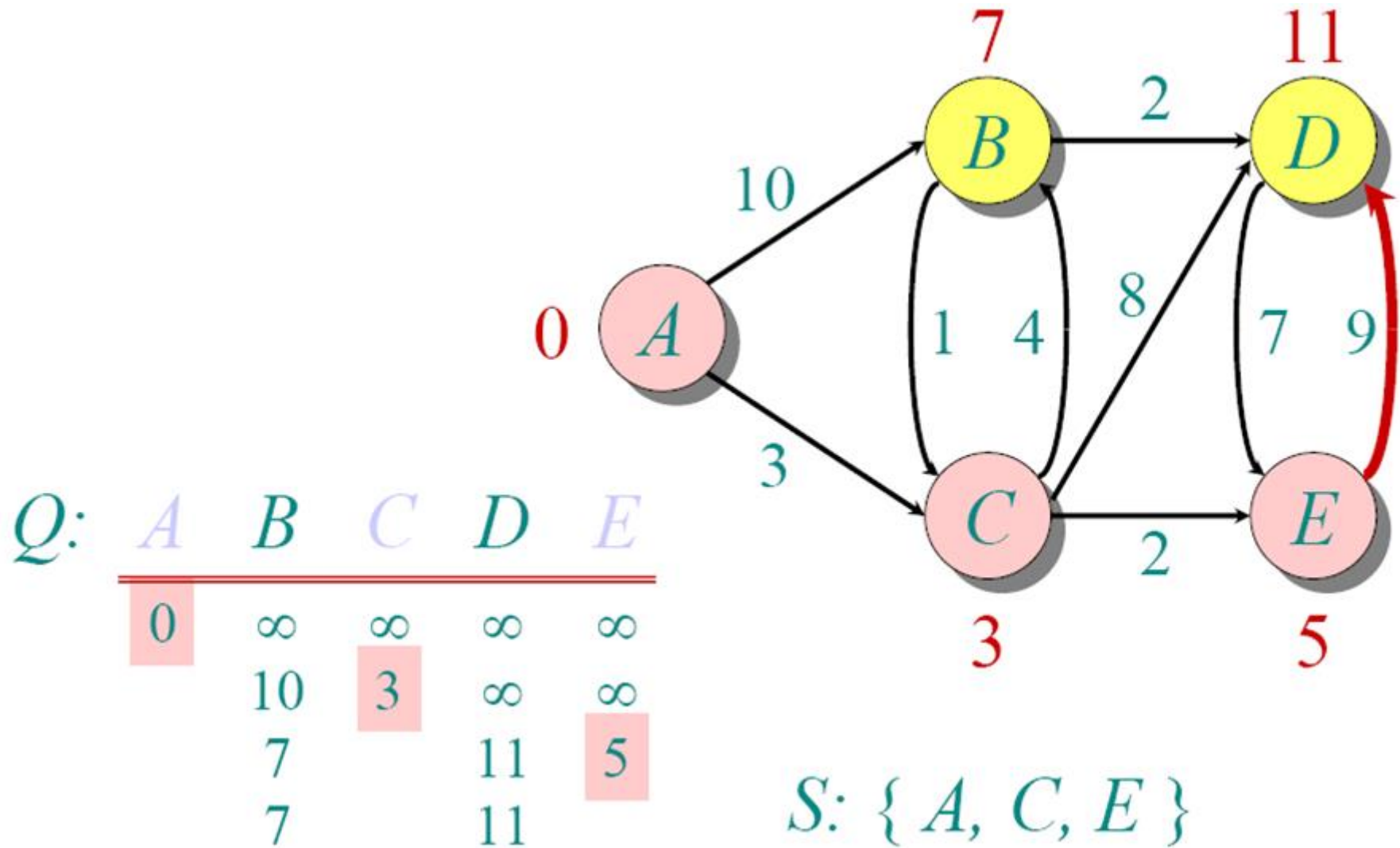
DIJKSTRA ANIMATED EXAMPLE



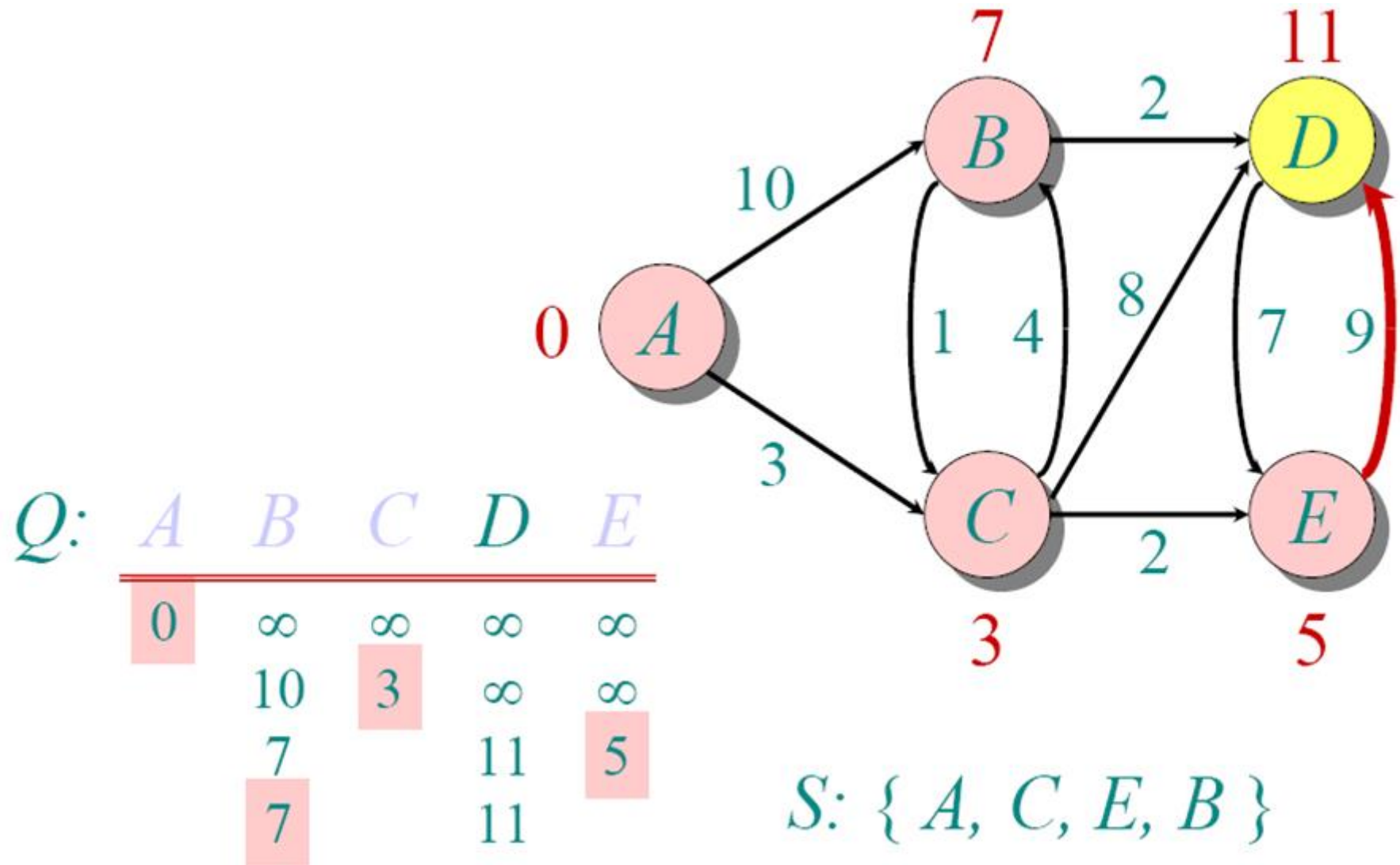
DIJKSTRA ANIMATED EXAMPLE



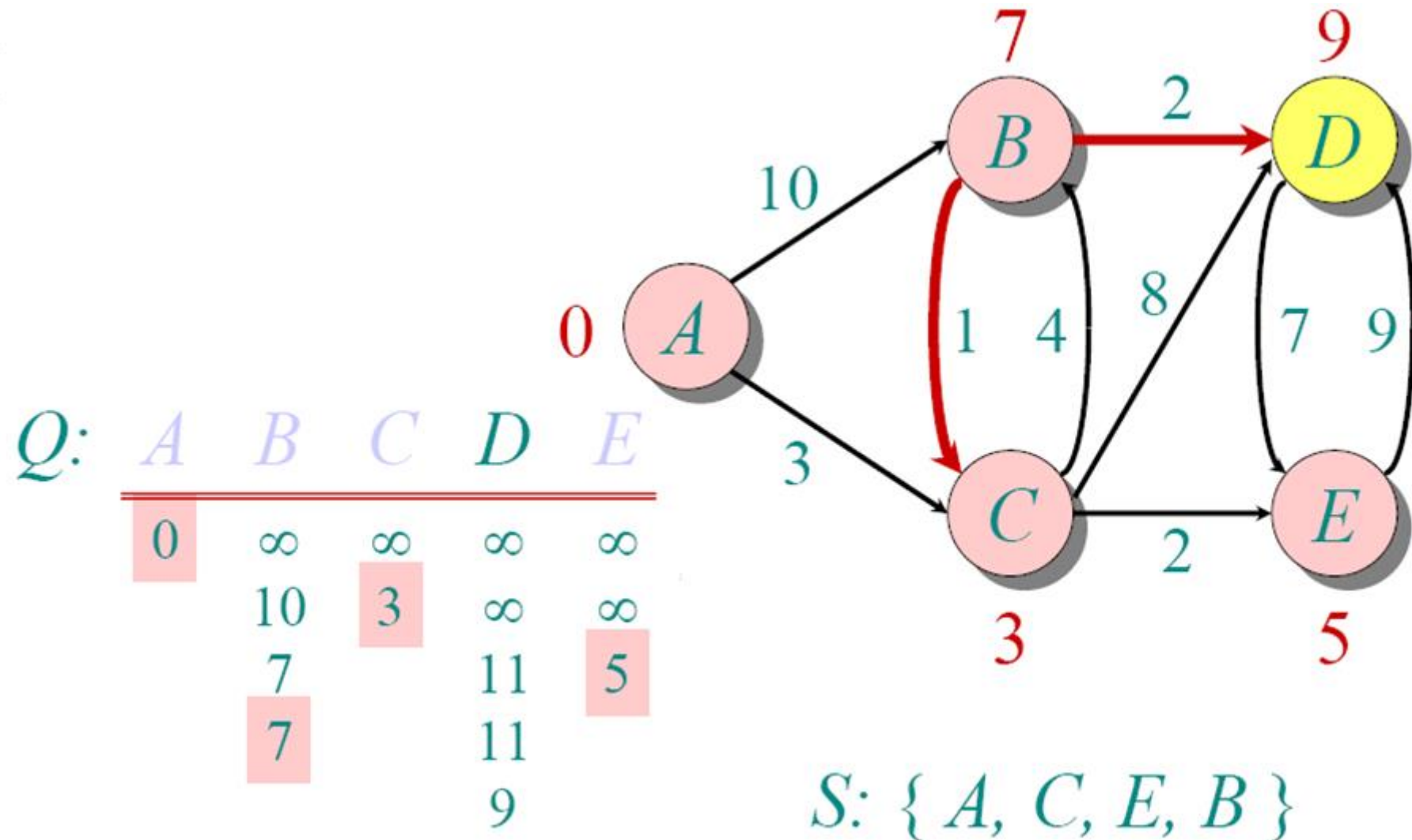
DIJKSTRA ANIMATED EXAMPLE



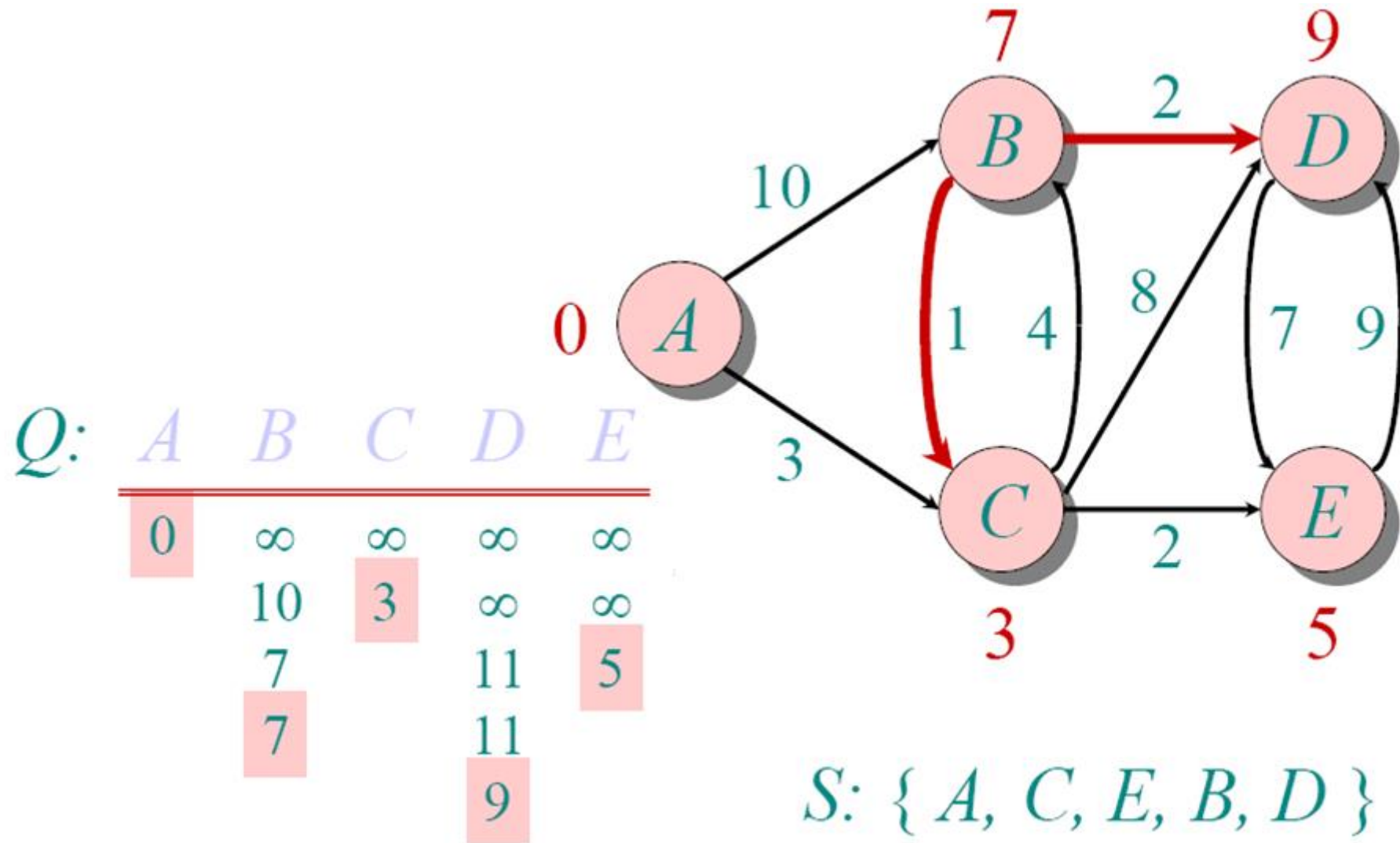
DIJKSTRA ANIMATED EXAMPLE



DIJKSTRA ANIMATED EXAMPLE



DIJKSTRA ANIMATED EXAMPLE



IMPLEMENTATIONS AND RUNNING TIMES

The simplest implementation is to store vertices in an array or linked list. This will produce a running time of

$$O(|V|^2 + |E|)$$

For sparse graphs, or graphs with very few edges and many nodes, it can be implemented more efficiently storing the graph in an adjacency list using a binary heap or priority queue. This will produce a running time of

$$O((|E| + |V|) \log |V|)$$



DIJKSTRA'S ALGORITHM - WHY IT WORKS

- As with all greedy algorithms, we need to make sure that it is a correct algorithm (e.g., it *always* returns the right solution if it is given correct input).
- A formal proof would take longer than this presentation, but we can understand how the argument works intuitively.
- If you can't sleep unless you see a proof, see the second reference or ask us where you can find it.



DIJKSTRA'S ALGORITHM - WHY IT WORKS

- To understand how it works, we'll go over the previous example again. However, we need two mathematical results first:
- **Lemma 1:** Triangle inequality
If $\delta(u,v)$ is the shortest path length between u and v ,
$$\delta(u,v) \leq \delta(u,x) + \delta(x,v)$$
- **Lemma 2:**
The subpath of any shortest path is itself a shortest path.
- The key is to understand why we can claim that anytime we put a new vertex in S , we can say that we already know the shortest path to it.
- Now, back to the example...



DIJKSTRA'S ALGORITHM - WHY USE IT?

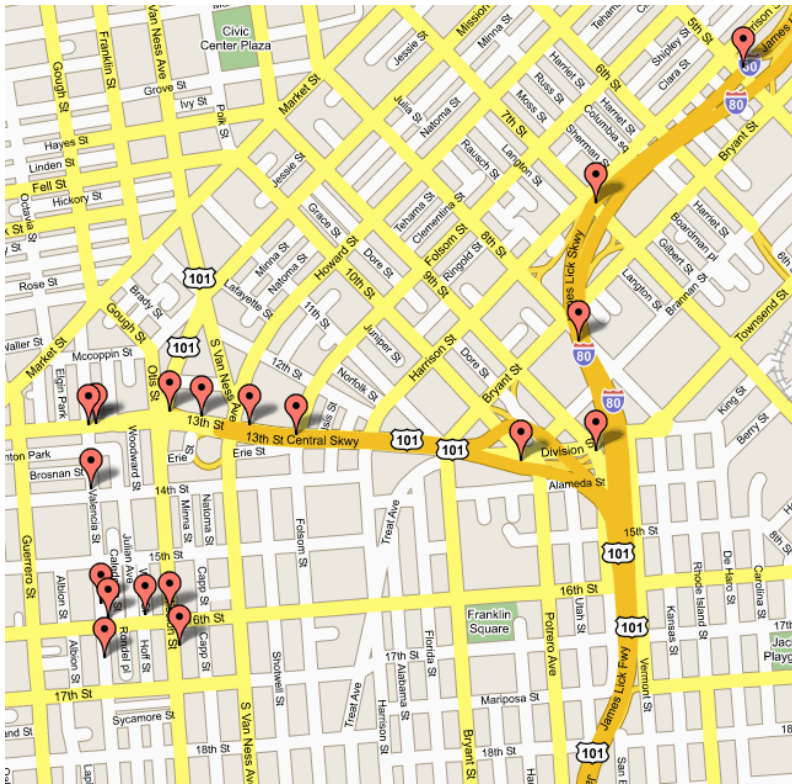
- As mentioned, Dijkstra's algorithm calculates the shortest path to every vertex.
- However, it is about as computationally expensive to calculate the shortest path from vertex u to every vertex using Dijkstra's as it is to calculate the shortest path to some particular vertex v .
- Therefore, anytime we want to know the optimal path to some other vertex from a determined origin, we can use Dijkstra's algorithm.



APPLICATIONS OF DIJKSTRA'S ALGORITHM

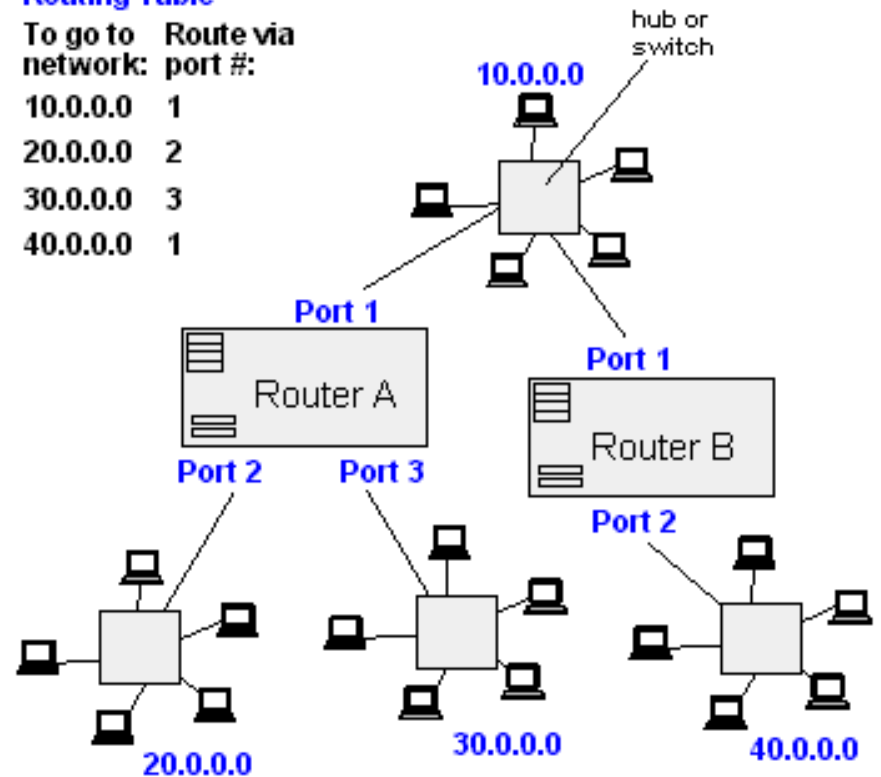
- Traffic Information Systems are most prominent use
- Mapping (Map Quest, Google Maps)
- Routing Systems

From Computer Desktop Encyclopedia
© 1998 The Computer Language Co. Inc.



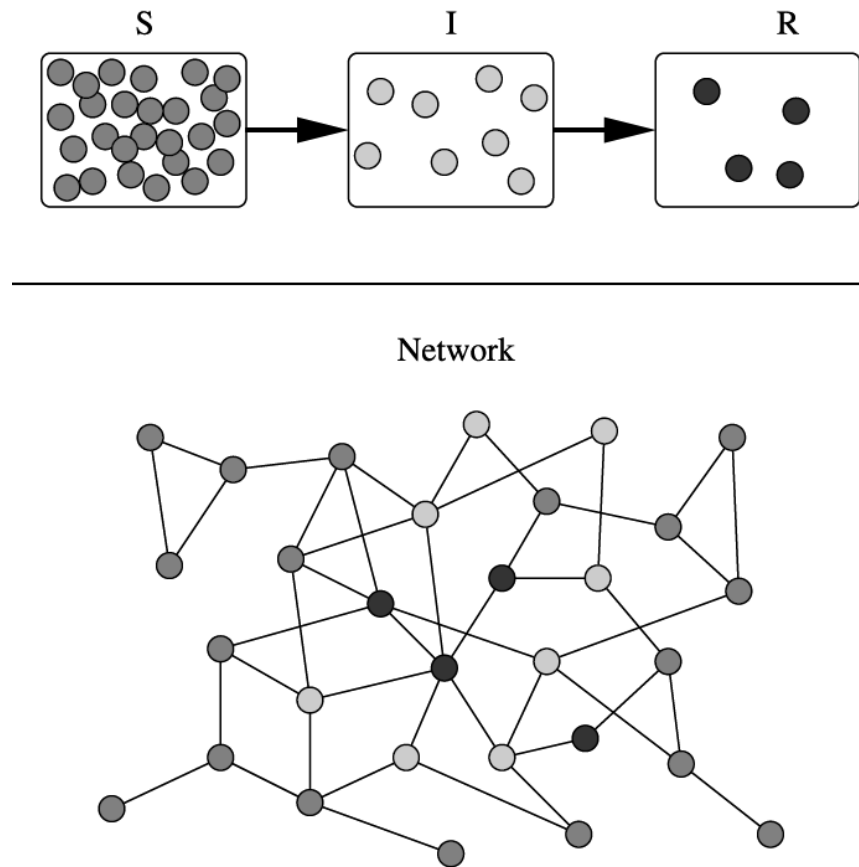
Router A Routing Table

To go to network:	Route via port #:
10.0.0.0	1
20.0.0.0	2
30.0.0.0	3
40.0.0.0	1



APPLICATIONS OF DIJKSTRA'S ALGORITHM

- One particularly relevant this week: epidemiology
- Prof. Lauren Meyers (Biology Dept.) uses networks to model the spread of infectious diseases and design prevention and response strategies.
- Vertices represent individuals, and edges their possible contacts. It is useful to calculate how a particular individual is connected to others.
- Knowing the shortest path lengths to other individuals can be a relevant indicator of the potential of a particular individual to infect others.



REFERENCES

- Dijkstra's original paper:
E. W. Dijkstra. (1959) *A Note on Two Problems in Connection with Graphs*. Numerische Mathematik, 1. 269-271.
- MIT OpenCourseware, 6.046J Introduction to Algorithms.
< <http://ocw.mit.edu/OcwWeb/Electrical-Engineering-and-Computer-Science/6-046JFall-2005/CourseHome/> > Accessed 4/25/09
- Meyers, L.A. (2007) Contact network epidemiology: Bond percolation applied to infectious disease prediction and control. *Bulletin of the American Mathematical Society* 44: 63-86.
- Department of Mathematics, University of Melbourne. *Dijkstra's Algorithm*.
<<http://www.ms.unimelb.edu.au/~moshe/620-261/dijkstra/dijkstra.html>> Accessed 4/25/09

