Lecture 21
WIDE AREA NETWORK

Routing & Congestion Control

Topics Covered

- Congestion Control
- Factors that Cause Congestion
- Congestion Control vs Flow Control
- Warning Bit
- Hop-by-Hop Choke Packets
- Load Shedding
- Random Early Discard (RED)
- Traffic Shaping
- What is Routing?
- Path Determination
- Shortest Path Problem
- Dijkstra’s Algorithm

Introduction

- Congestion control and routing are major issues to be handled in Wide Area Networks .
- Congestion is handled at transport layer and routing is handled at network layer.

Congestion Control

- When one part of the subnet (e.g. one or more routers in an area) becomes overloaded, congestion results.
- Because routers are receiving packets faster than they can forward them, one of two things must happen:
  - The subnet must prevent additional packets from entering the congested region until those already present can be processed.
  - The congested routers can discard queued packets to make room for those that are arriving.

Factors that Cause Congestion

- Packet arrival rate exceeds the outgoing link capacity.
- Insufficient memory to store arriving packets
- Bursty traffic
- Slow processor

Congestion Control vs Flow Control

- Congestion control is a global issue – involves every router and host within the subnet
- Flow control – scope is point-to-point; involves just sender and receiver.

Congestion Control (cont.)

- Congestion Control is concerned with efficiently using a network at high load.
- Several techniques can be employed. These include:
  - Warning bit
  - Choke packets
  - Load shedding
  - Random early discard
  - Traffic shaping
- The first 3 deal with congestion detection and recovery. The last 2 deal with congestion avoidance.

Warning Bit

- A special bit in the packet header is set by the router to warn the source when congestion is detected.
- The bit is copied and piggy-backed on the ACK and sent to the sender.
- The sender monitors the number of ACK packets it receives with the warning bit set and adjusts its transmission rate accordingly.

Choke Packets

- A more direct way of telling the source to slow down.
- A choke packet is a control packet generated at a congested node and transmitted to restrict traffic flow.
- The source, on receiving the choke packet must reduce its transmission rate by a certain percentage.
- An example of a choke packet is the ICMP Source Quench Packet

Hop-by-Hop Choke Packets

- Over long distances or at high speeds choke packets are not very effective.
- A more efficient method is to send to choke packets hop-by-hop.
- This requires each hop to reduce its transmission even before the choke packet arrive at the source.

Load Shedding

- When buffers become full, routers simply discard packets.
- Which packet is chosen to be the victim depends on the application and on the error strategy used in the data link layer.
- For a file transfer, for, e.g. cannot discard older packets since this will cause a gap in the received data.
- For real-time voice or video it is probably better to throw away old data and keep new packets.
- Get the application to mark packets with discard priority.

Random Early Discard (RED)

- This is a proactive approach in which the router discards one or more packets *before* the buffer becomes completely full.
- Each time a packet arrives, the RED algorithm computes the average queue length, ***avg***.
- If *avg* is lower than some lower threshold, congestion is assumed to be minimal or non-existent and the packet is queued.

RED, (Cont.)

- If *avg* is greater than some upper threshold, congestion is assumed to be serious and the packet is discarded.
- If *avg* is between the two thresholds, this might indicate the onset of congestion. The probability of congestion is then calculated.

Traffic Shaping

- Another method of congestion control is to "shape" the traffic before it enters the network.
- Traffic shaping controls the *rate* at which packets are sent (not just how many). Used in ATM and Integrated Services networks.
- At connection set-up time, the sender and carrier negotiate a traffic pattern (shape).

What is Routing?

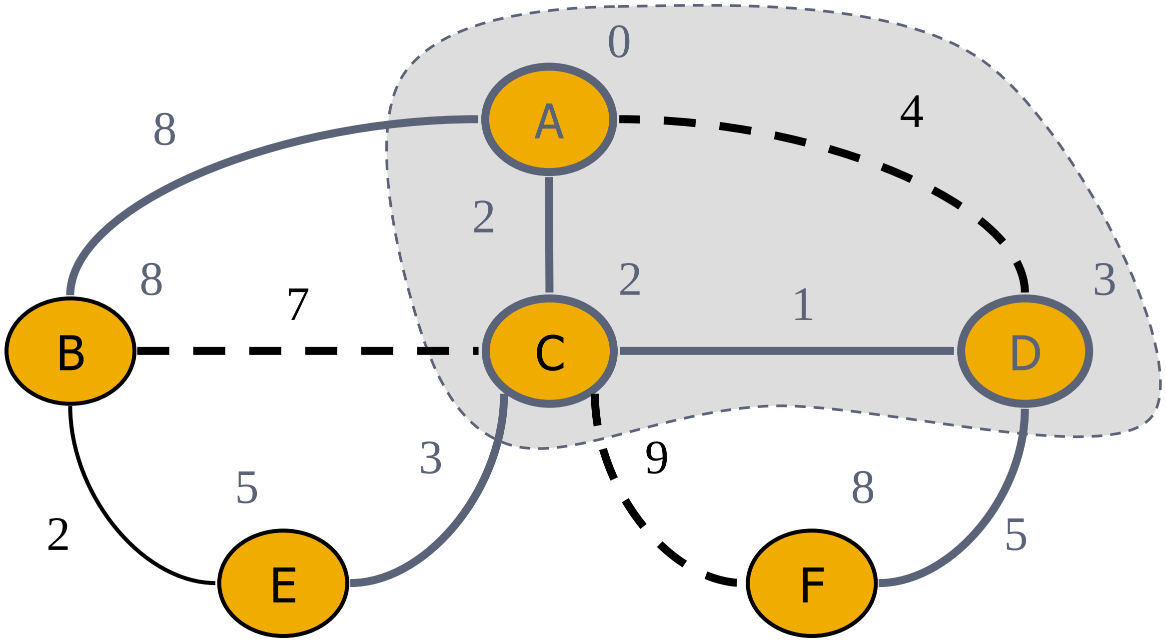
Moving information across the network from a source to a destination, typically through intermediate node(s). It consists of:

- Determining optimal routing paths
- Transporting information (e.g. grouped in packets, cells in packet switching)

Path Determination

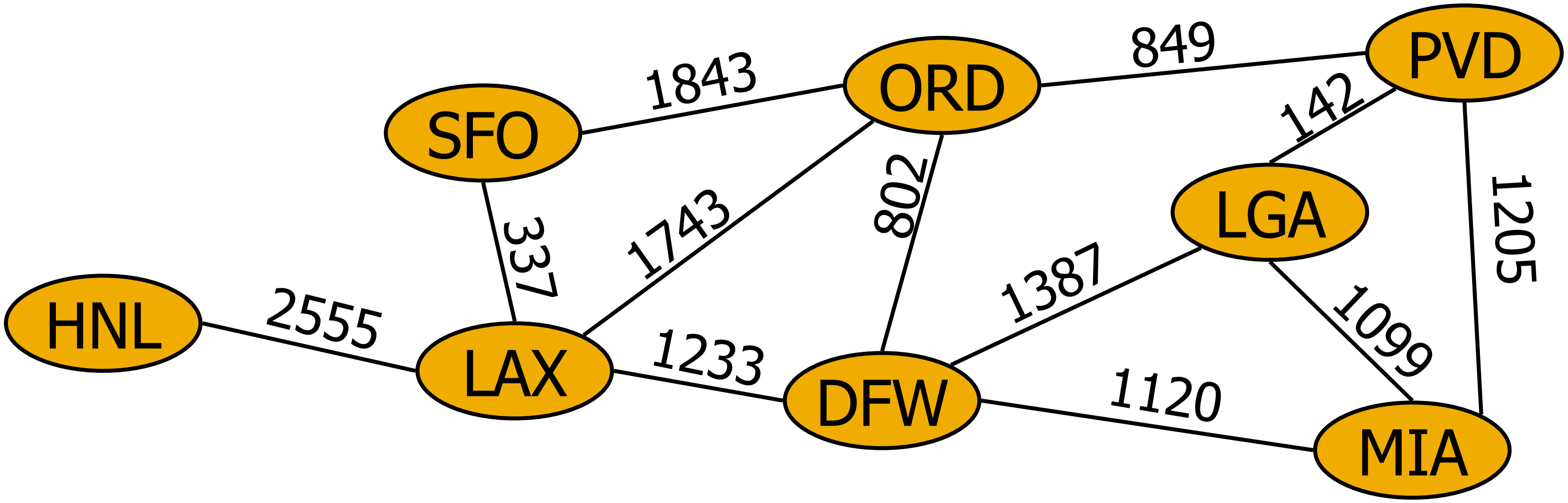
- Routing protocols use routing algorithms to populate routing tables, which contain the route information such as
  - destination/next hop association
  - desirability of a path, and other
- Routers build a picture of network topology based on routing information received from other routers

Shortest Path



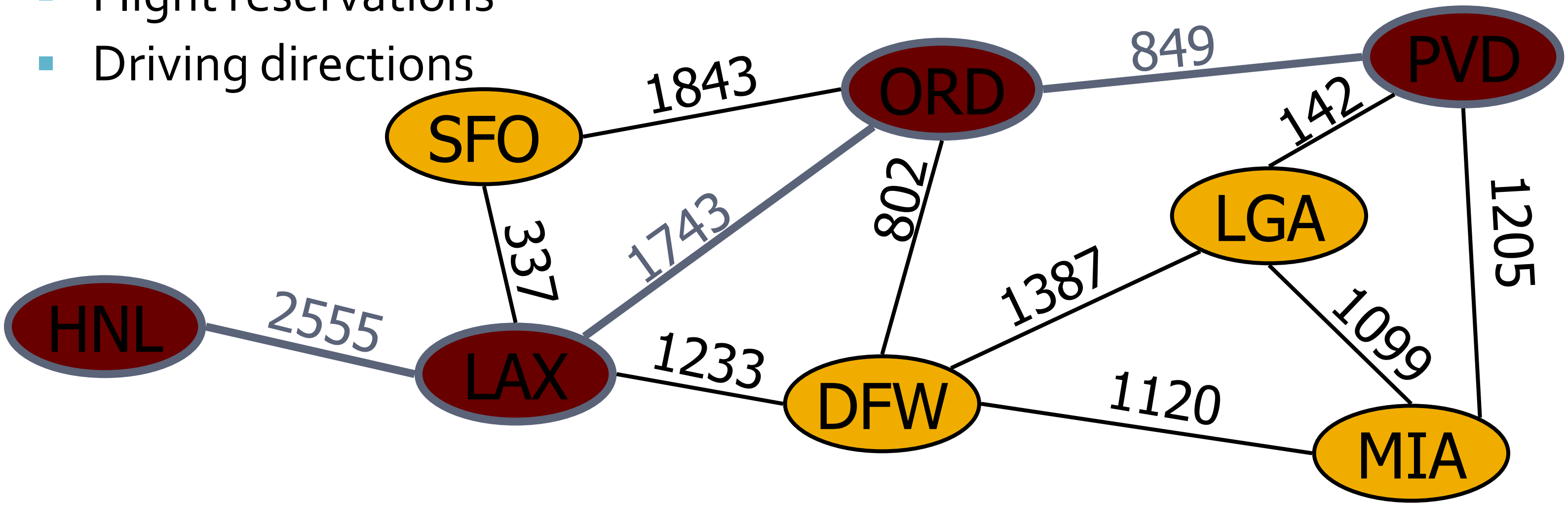
Weighted Graphs

- In a weighted graph, each edge has an associated numerical value, called the weight of the edge
- Edge weights may represent, distances, costs, etc.
- Example:
  - In a flight route graph, the weight of an edge represents the distance in miles between the endpoint airports



Shortest Path Problem

- Given a weighted graph and two vertices ***u*** and ***v***, we want to find a path of minimum total weight between ***u*** and ***v*.**
  - Length of a path is the sum of the weights of its edges.
- Example:
  - Shortest path between Providence and Honolulu
- Applications
  - Internet packet routing
  - Flight reservations
  - Driving directions



Shortest Path Properties

Property 1:

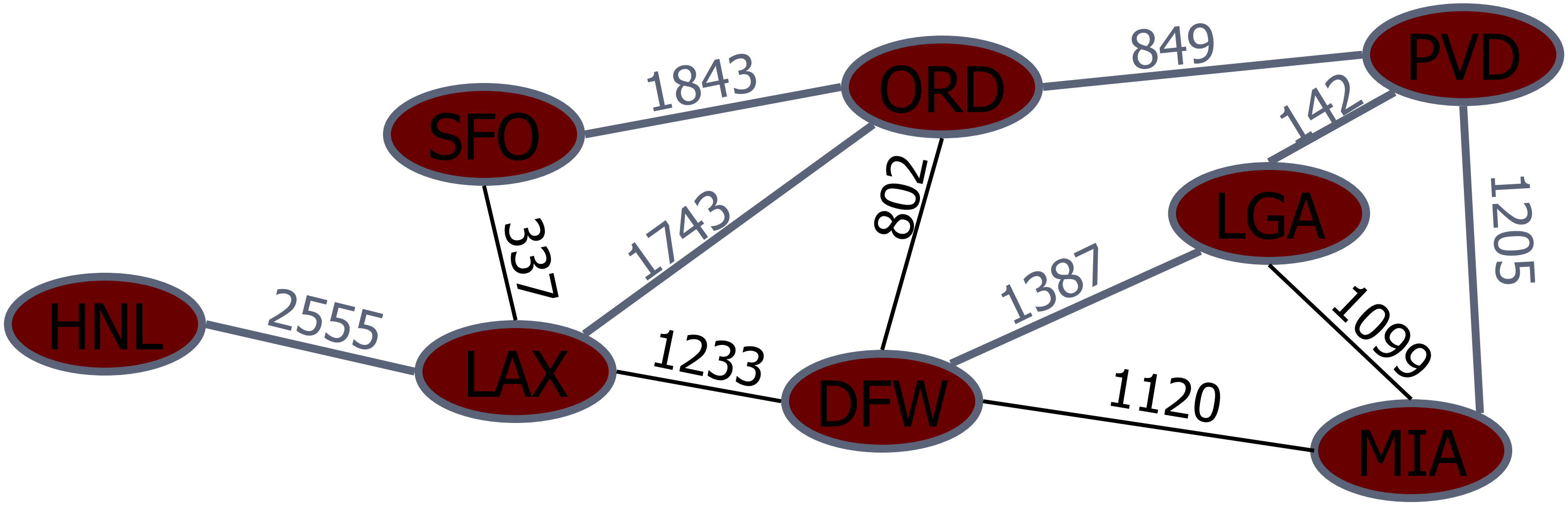
A subpath of a shortest path is itself a shortest path

Property 2:

There is a tree of shortest paths from a start vertex to all the other vertices

Example:

Tree of shortest paths from Providence

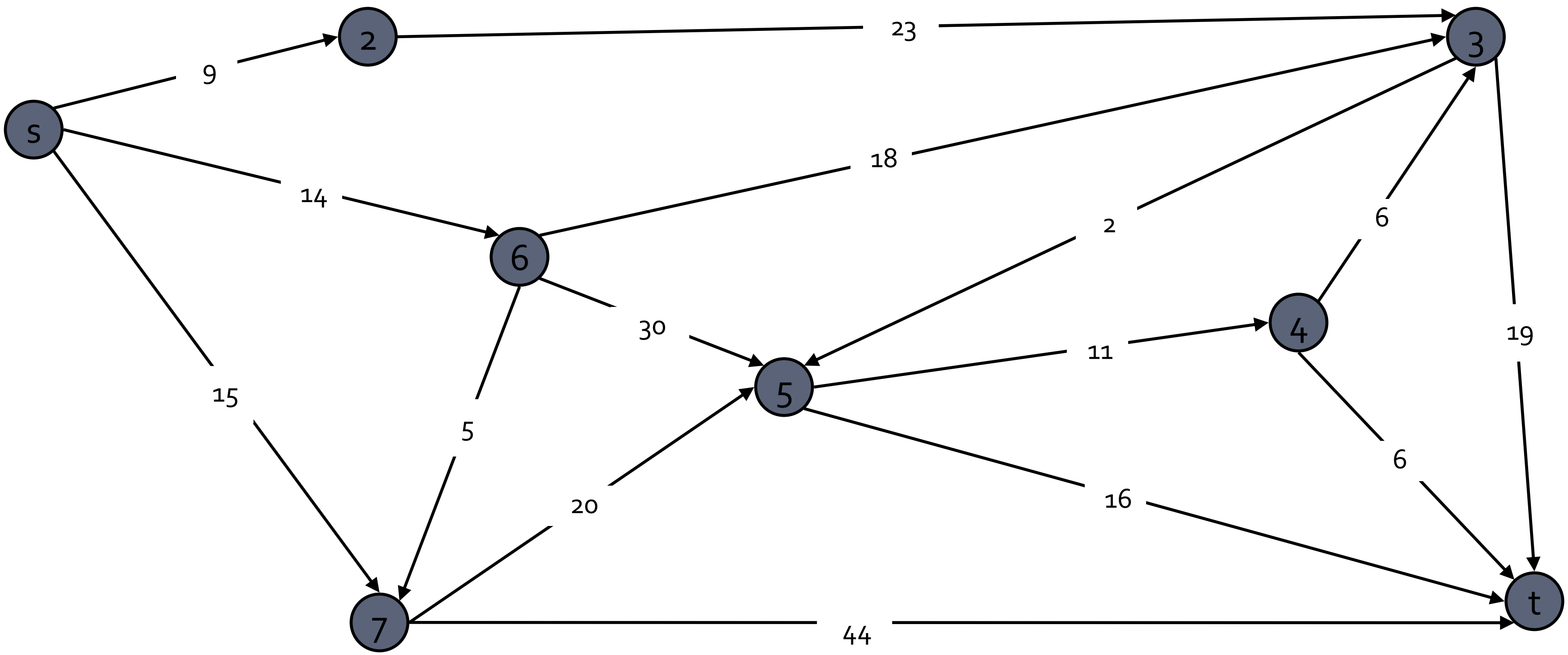


Dijkstra's Algorithm

- The distance of a vertex ***v*** from a vertex ***s*** is the length of a shortest path between ***s*** and ***v***
- Dijkstra's algorithm computes the distances of all the vertices from a given start vertex ***s***
- Assumptions:
  - the graph is connected
  - the edges are undirected
  - the edge weights are nonnegative
- We grow a "**cloud**" of vertices, beginning with ***s*** and eventually covering all the vertices
- We store with each vertex ***v*** a label ***d***(***v***) representing the distance of ***v*** from ***s*** in the subgraph consisting of the cloud and its adjacent vertices
- At each step
  - We add to the cloud the vertex ***u*** outside the cloud with the smallest distance label, ***d***(***u***)
  - We update the labels of the vertices adjacent to ***u***

Dijkstra's Shortest Path Algorithm

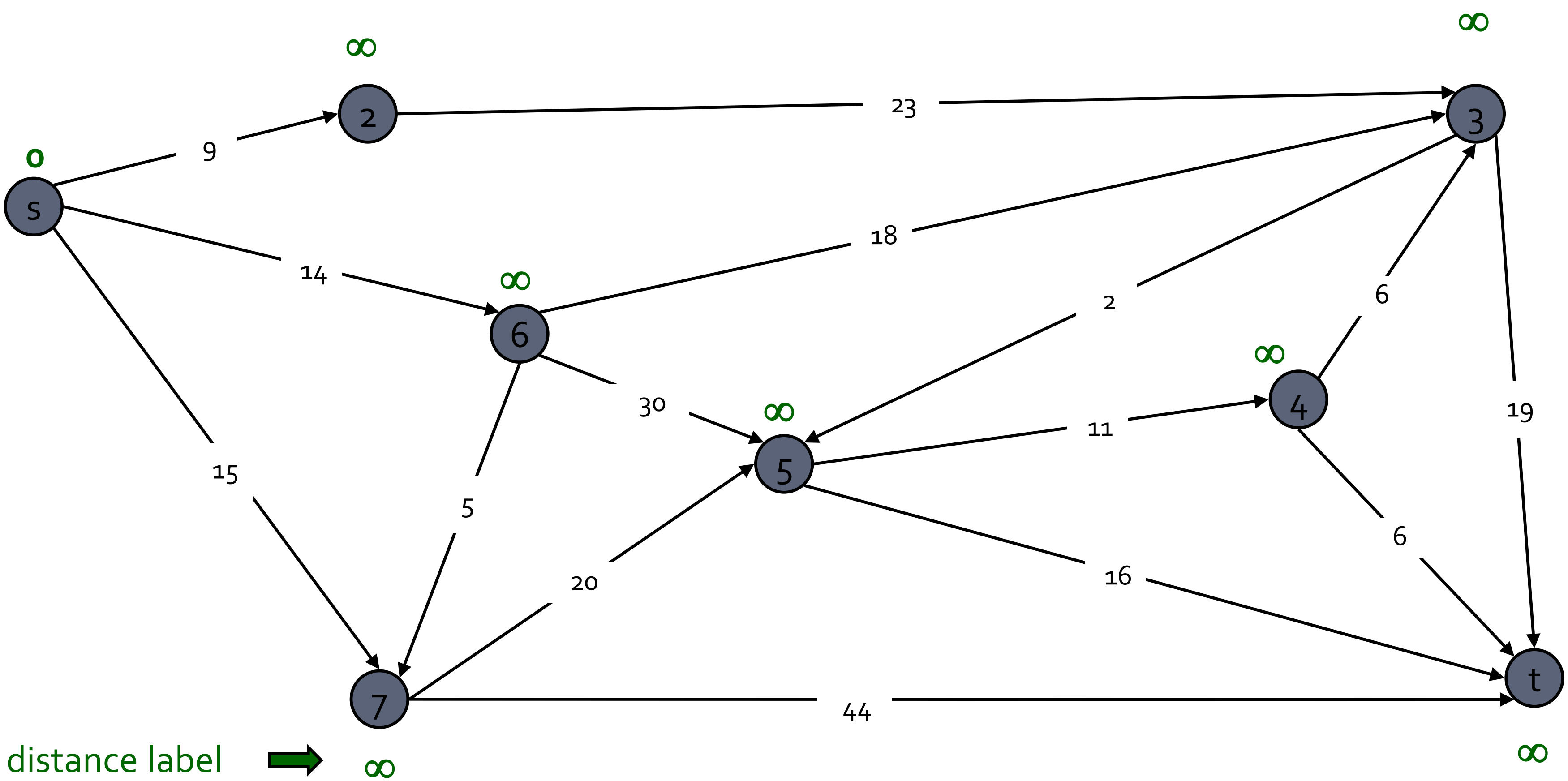
- Find shortest path from s to t.



Dijkstra's Shortest Path Algorithm

S = { }

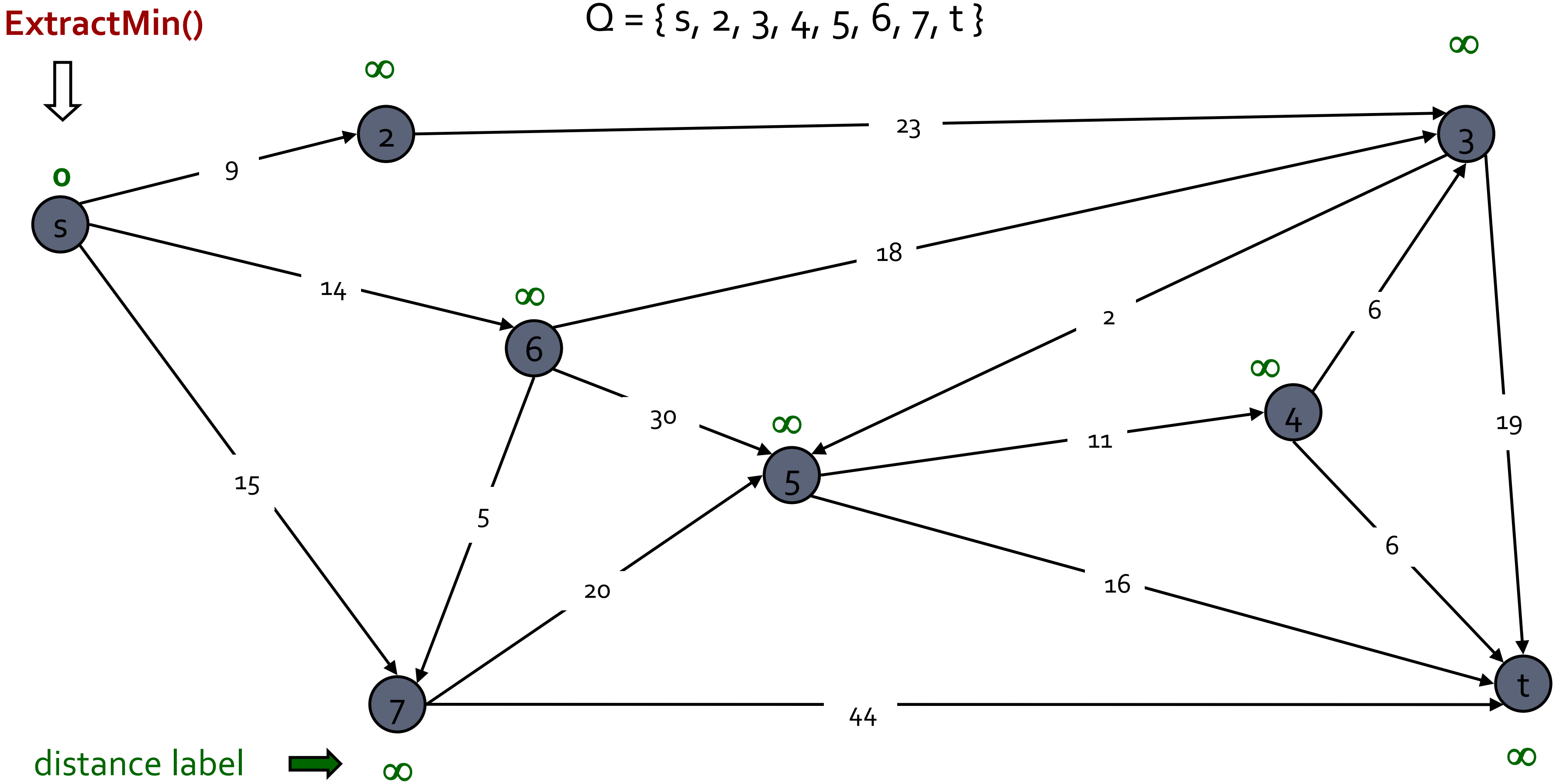
Q = { s, 2, 3, 4, 5, 6, 7, t }



Dijkstra's Shortest Path Algorithm

S = { }

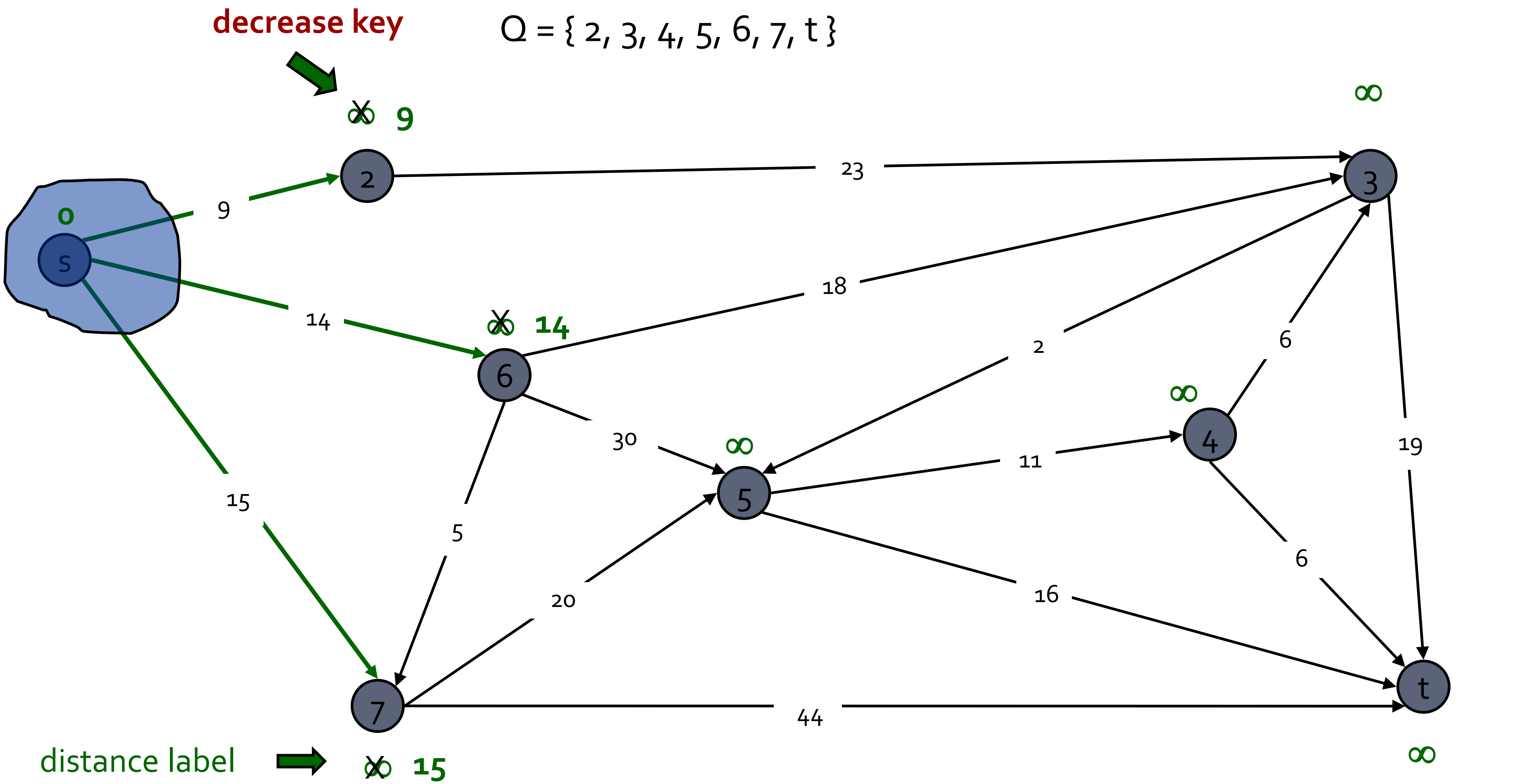
Q = { s, 2, 3, 4, 5, 6, 7, t }



Dijkstra's Shortest Path Algorithm

S = { s }

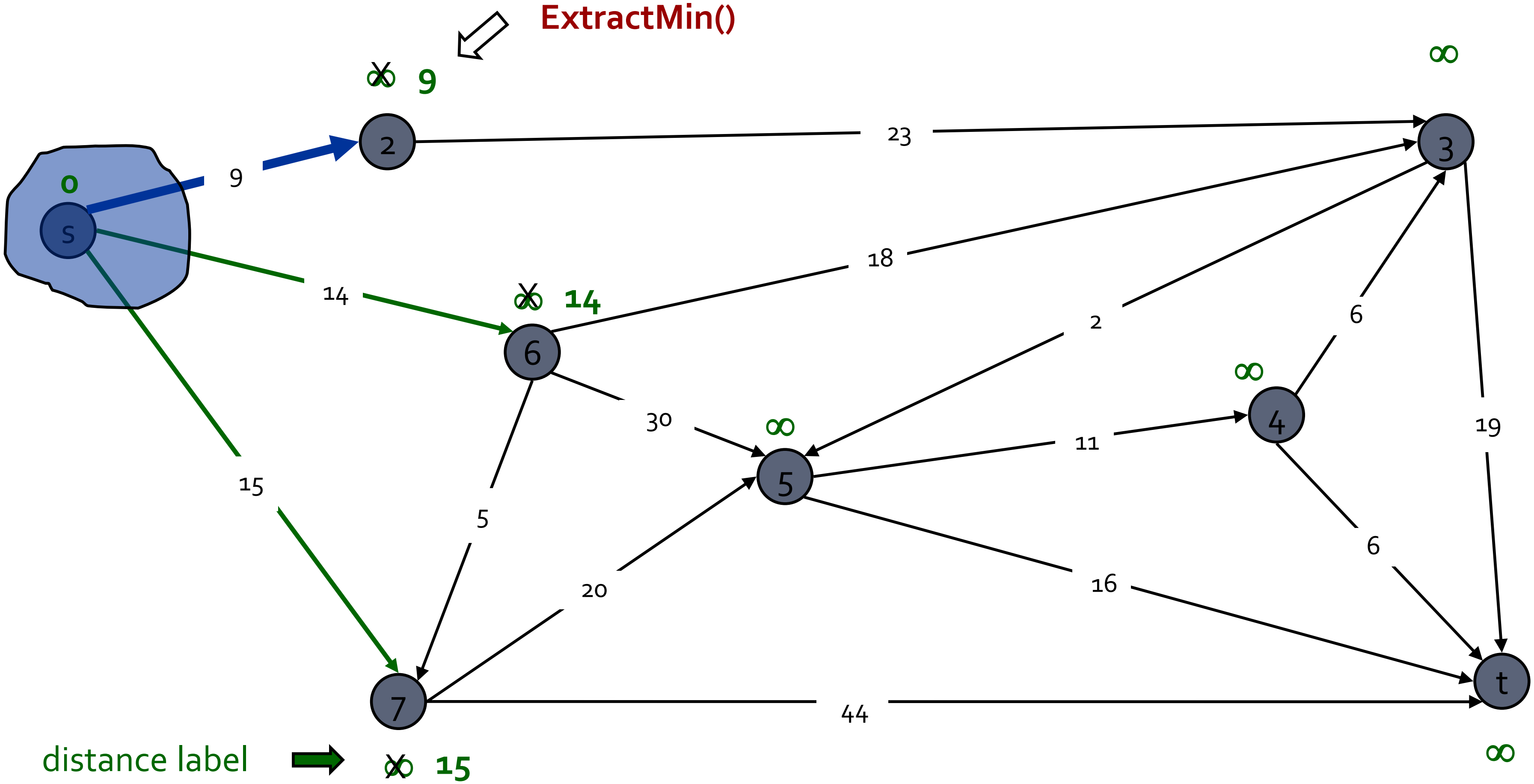
Q = { 2, 3, 4, 5, 6, 7, t }



Dijkstra's Shortest Path Algorithm

S = { s }

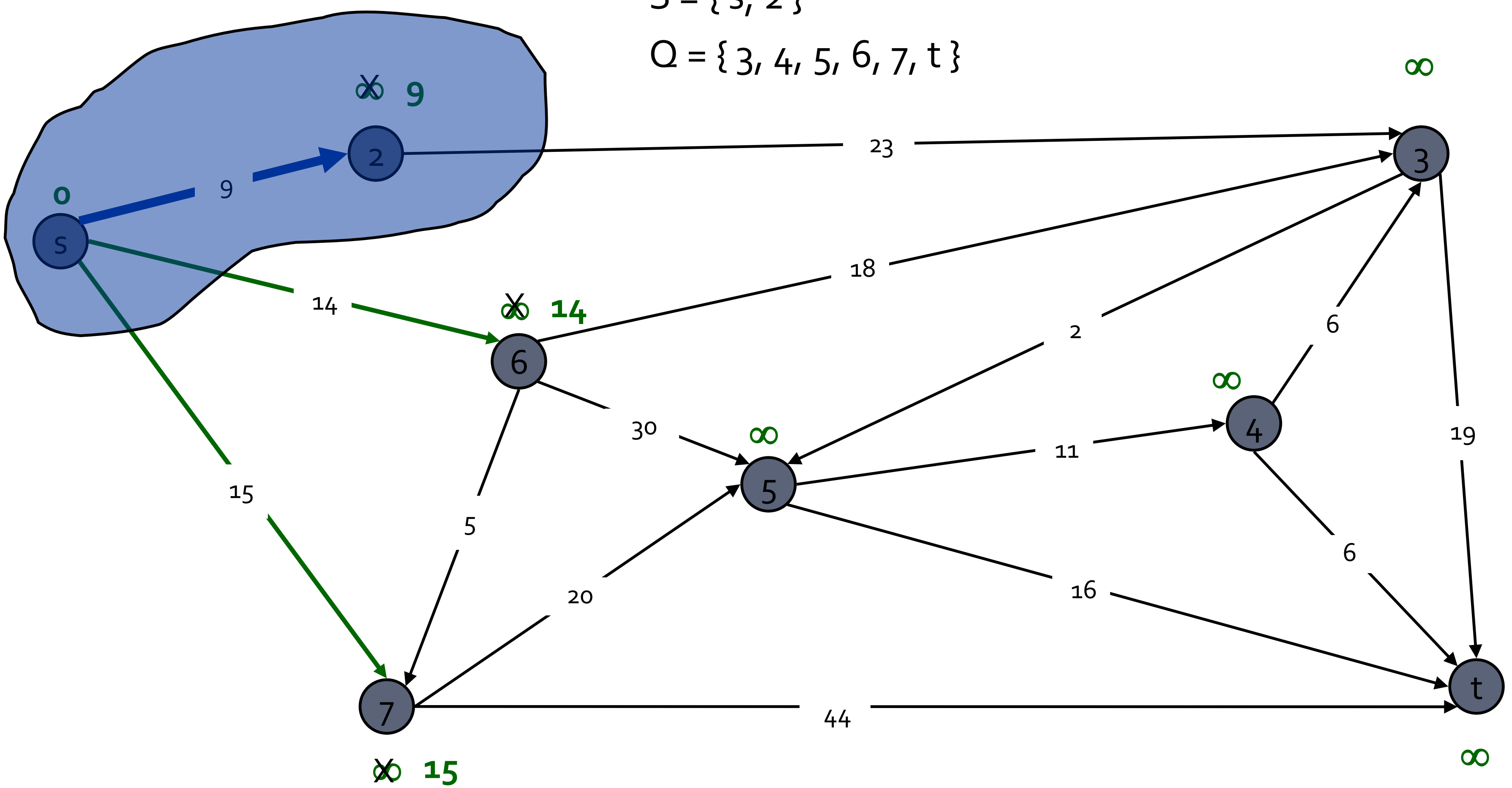
Q = { 2, 3, 4, 5, 6, 7, t }



Dijkstra's Shortest Path Algorithm

S = { s, 2 }

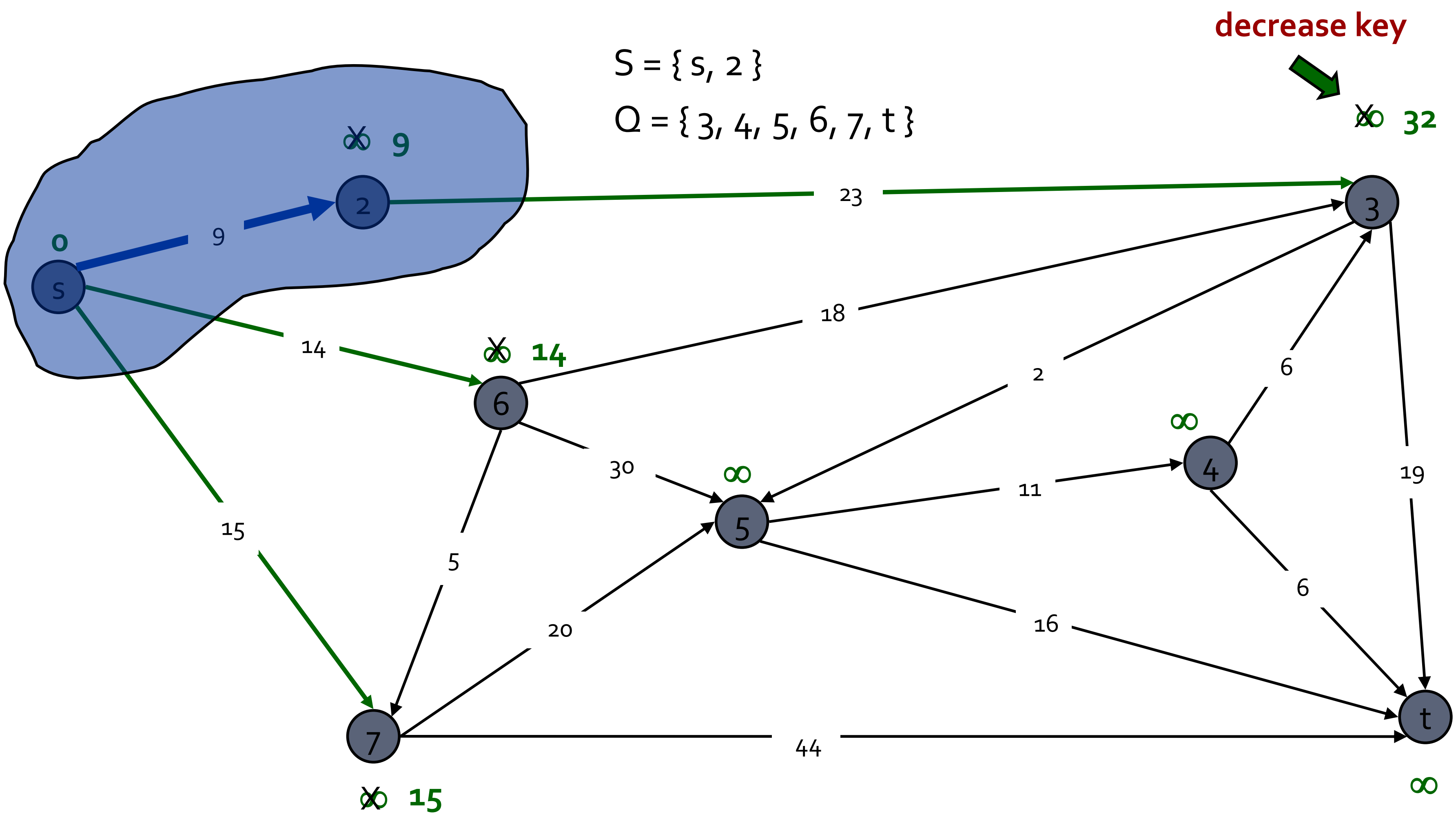
Q = { 3, 4, 5, 6, 7, t }



Dijkstra's Shortest Path Algorithm

S = { s, 2 }

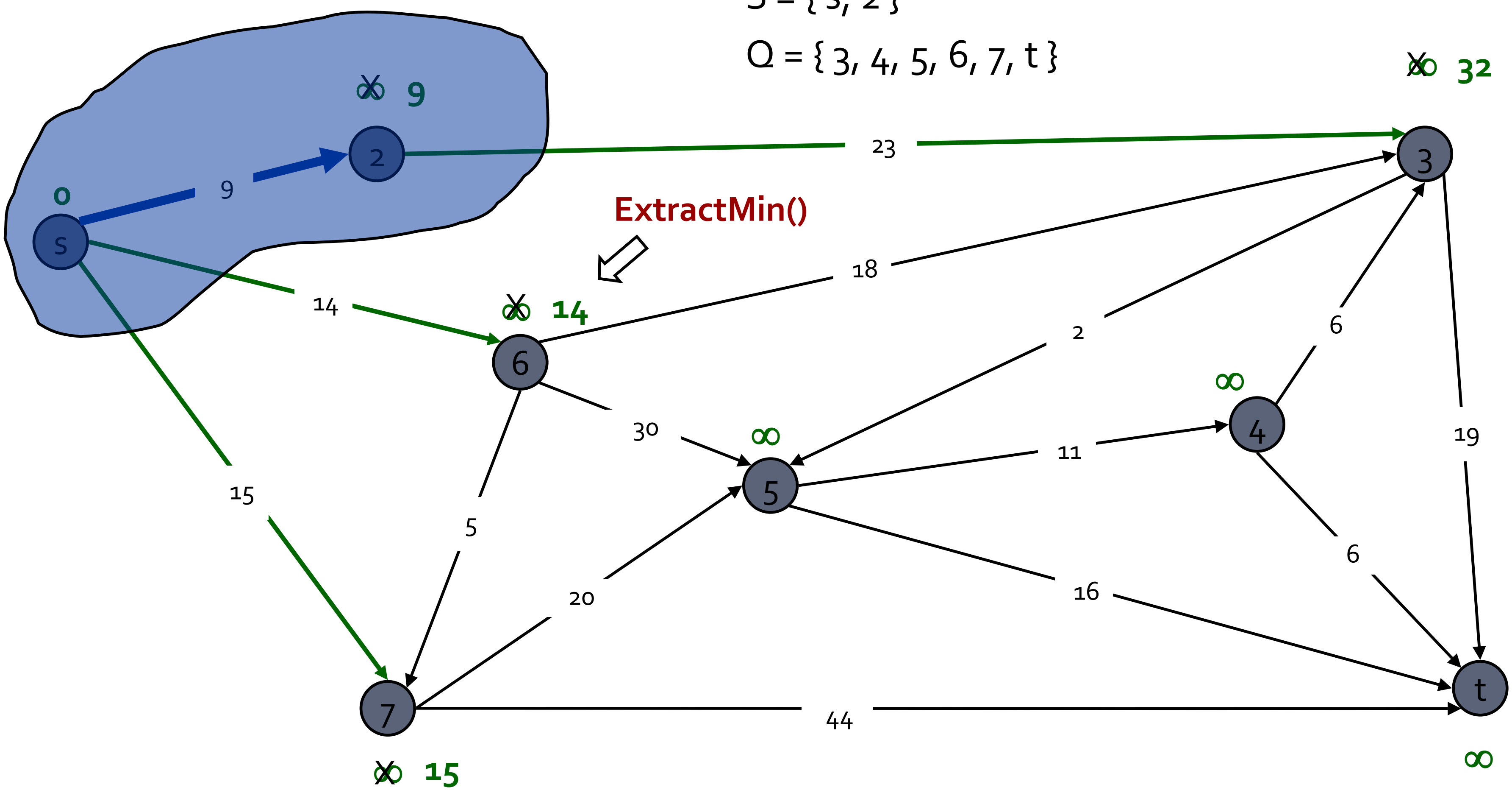
Q = { 3, 4, 5, 6, 7, t }



Dijkstra's Shortest Path Algorithm

S = { s, 2 }

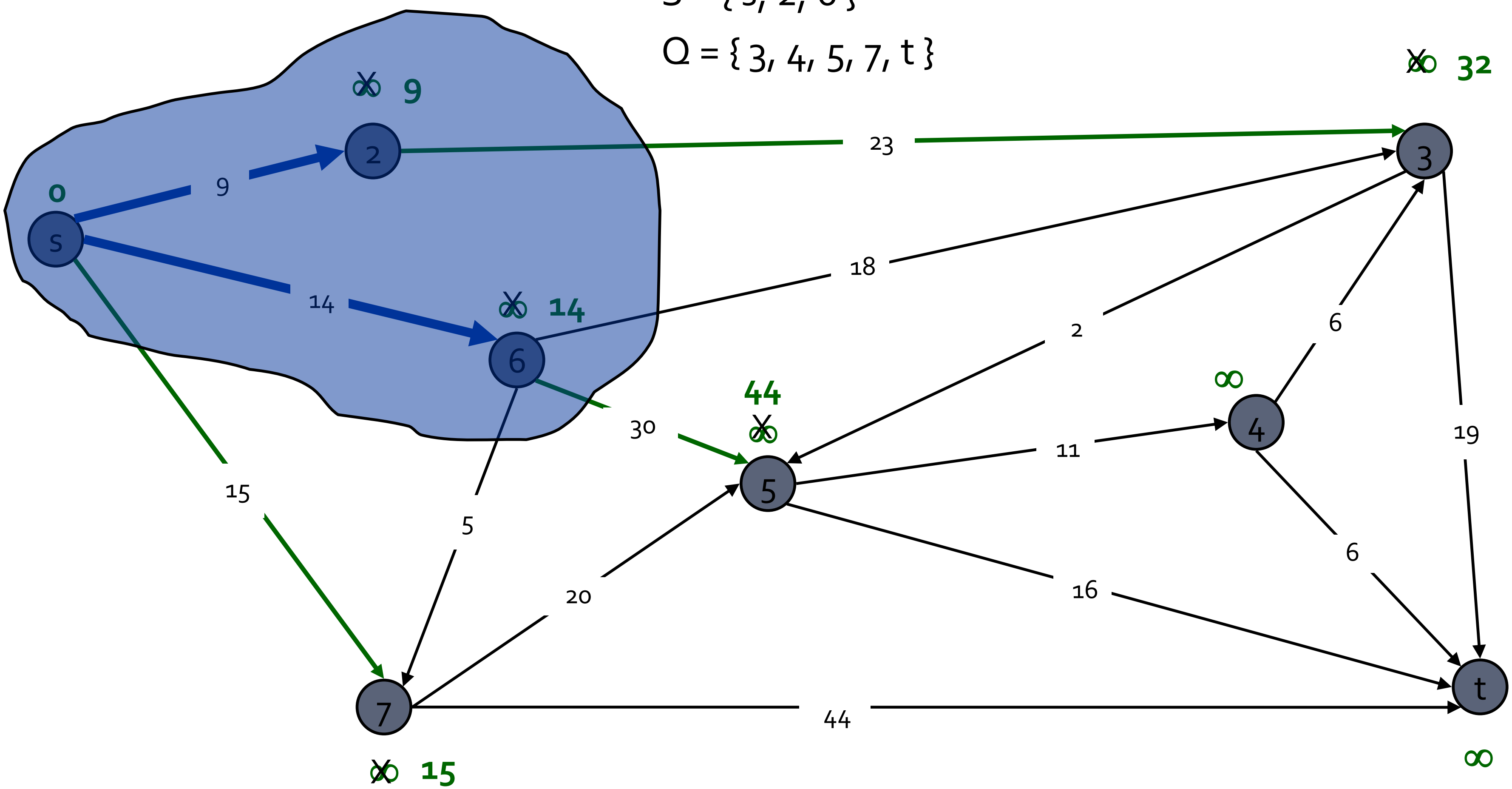
Q = { 3, 4, 5, 6, 7, t }



Dijkstra's Shortest Path Algorithm

S = { s, 2, 6 }

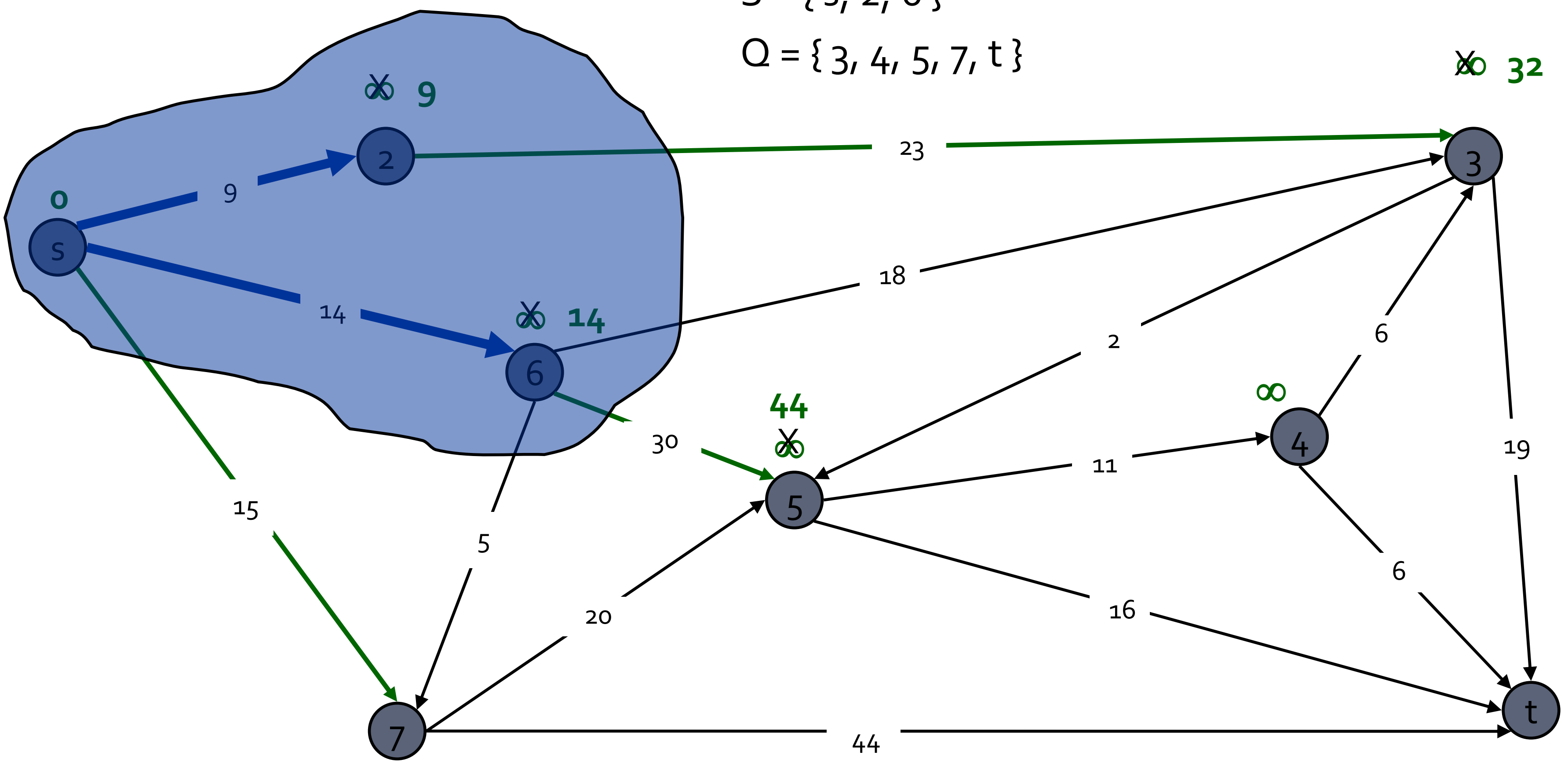
Q = { 3, 4, 5, 7, t }



Dijkstra's Shortest Path Algorithm

S = { s, 2, 6 }

Q = { 3, 4, 5, 7, t }

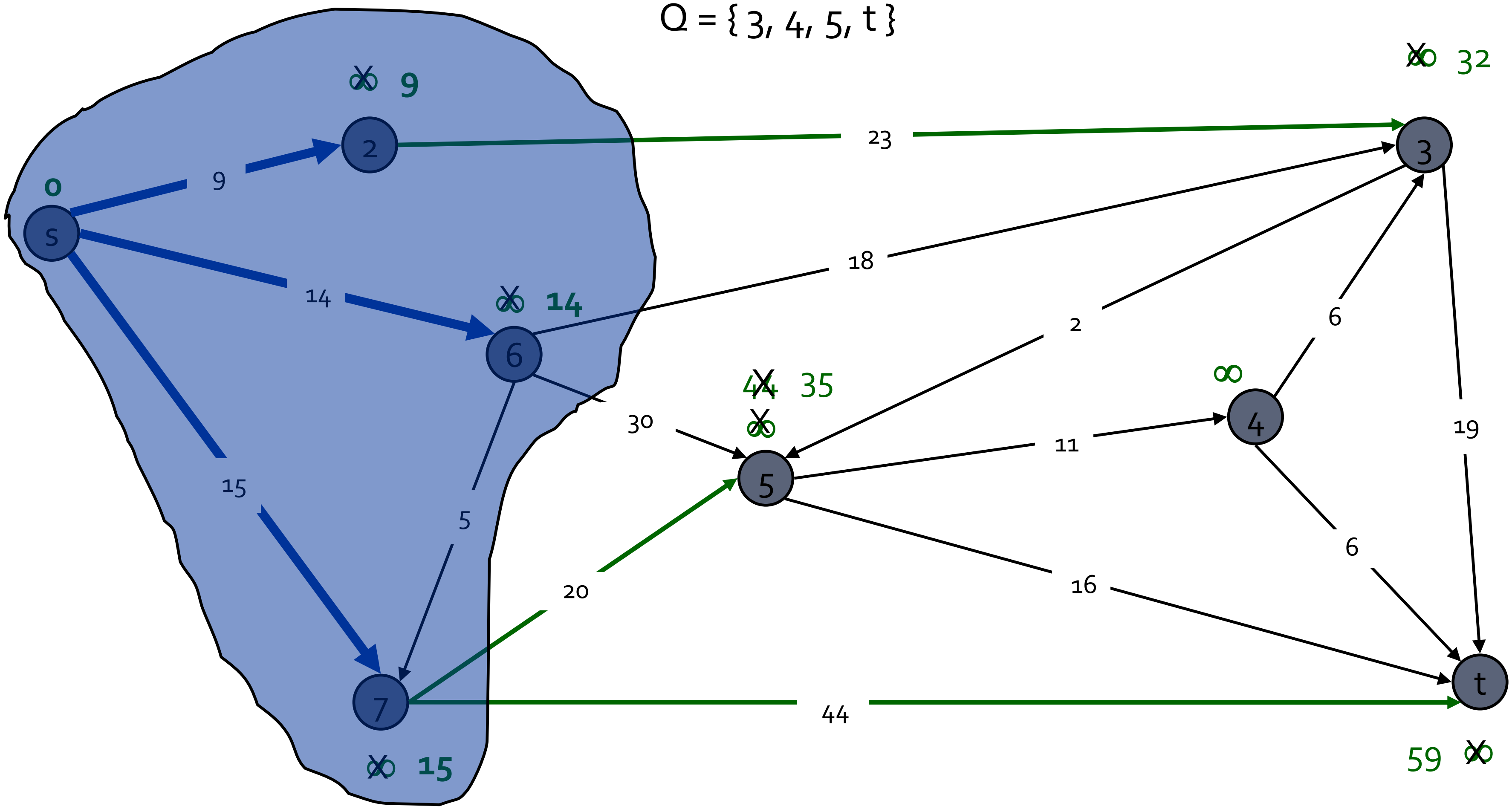


⇦ **ExtractMin()**

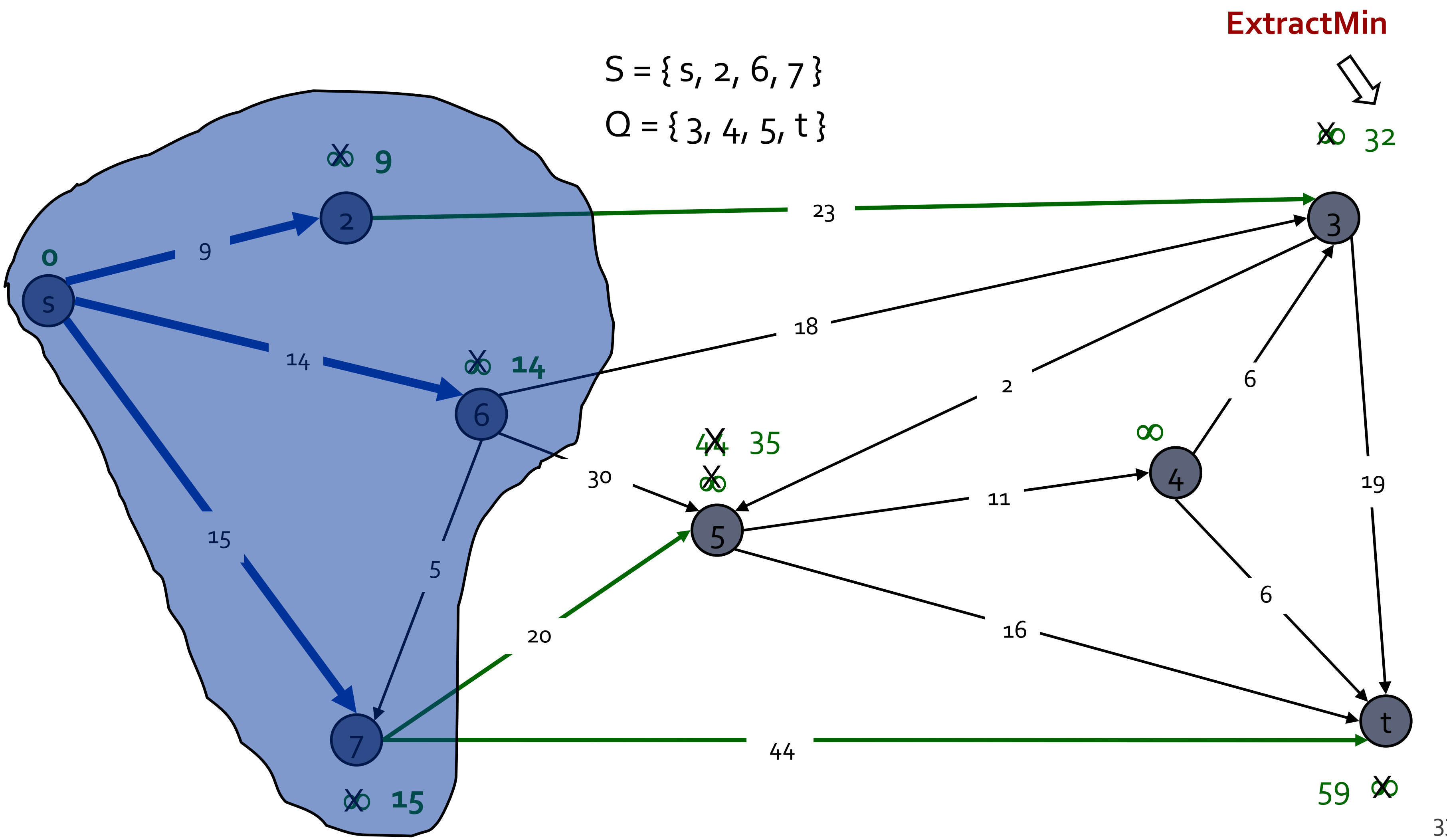
Dijkstra's Shortest Path Algorithm

S = { s, 2, 6, 7 }

Q = { 3, 4, 5, t }



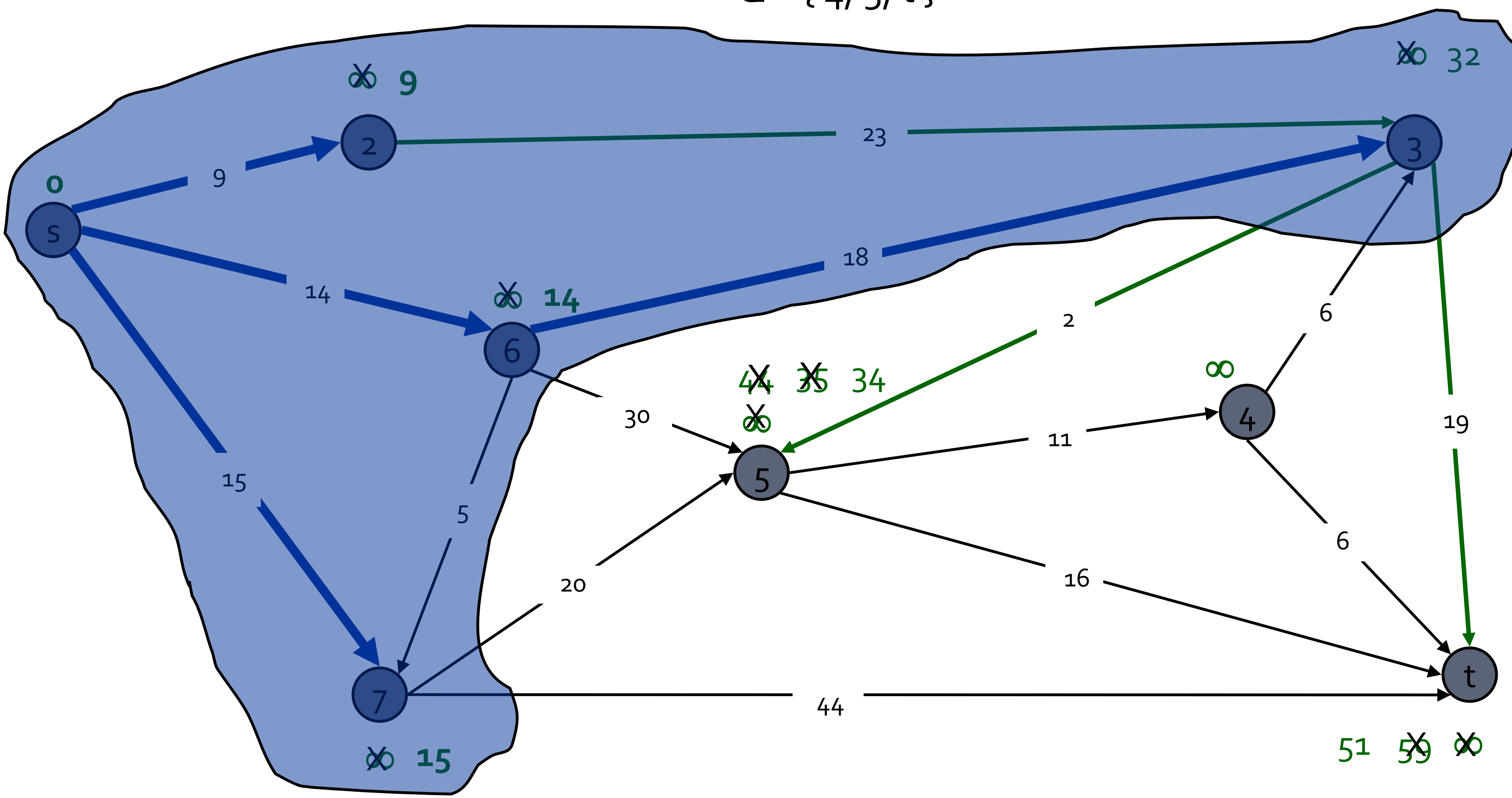
Dijkstra's Shortest Path Algorithm



Dijkstra's Shortest Path Algorithm

S = { s, 2, 3, 6, 7 }

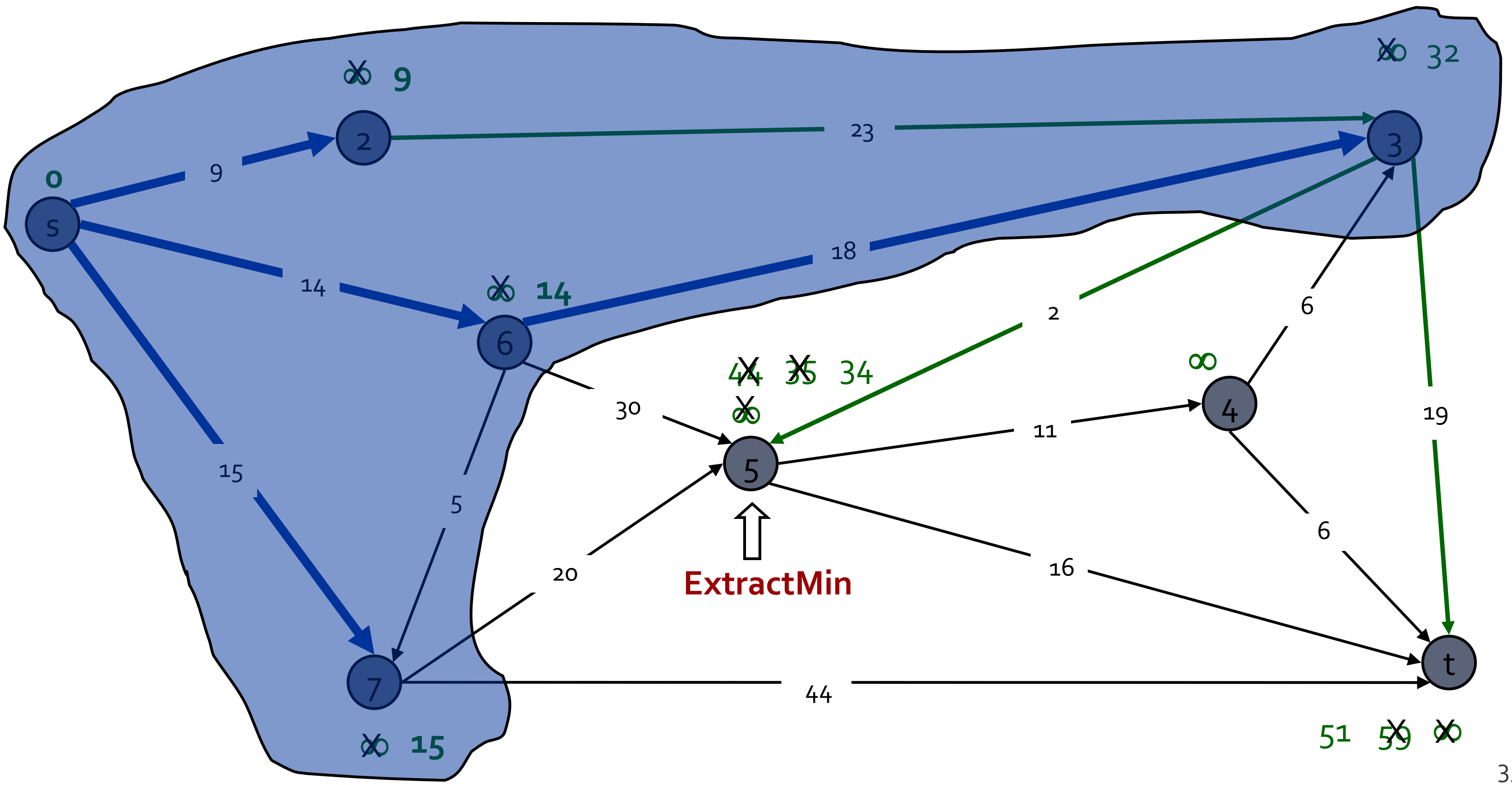
Q = { 4, 5, t }



Dijkstra's Shortest Path Algorithm

S = { s, 2, 3, 6, 7 }

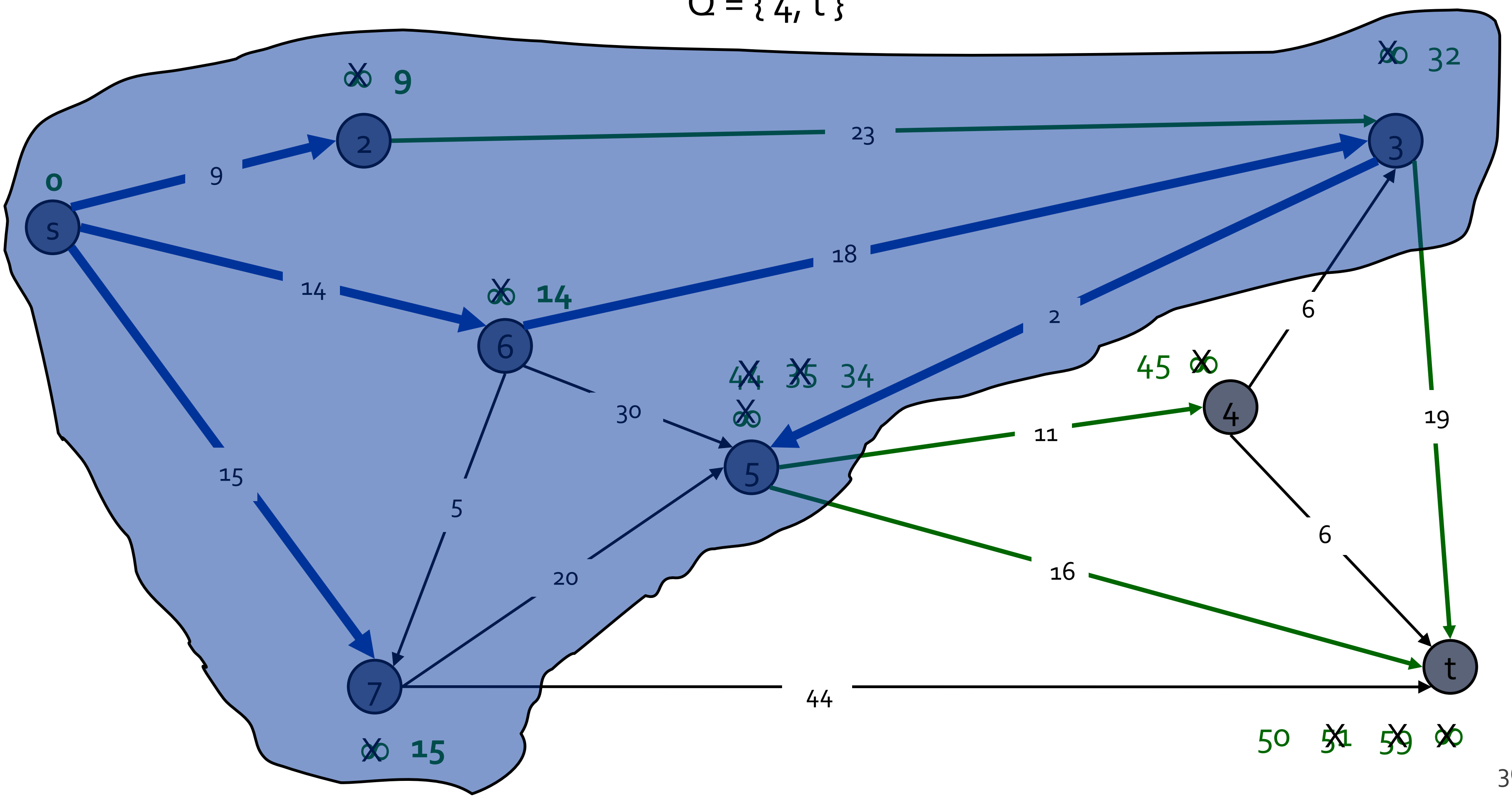
Q = { 4, 5, t }



Dijkstra's Shortest Path Algorithm

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

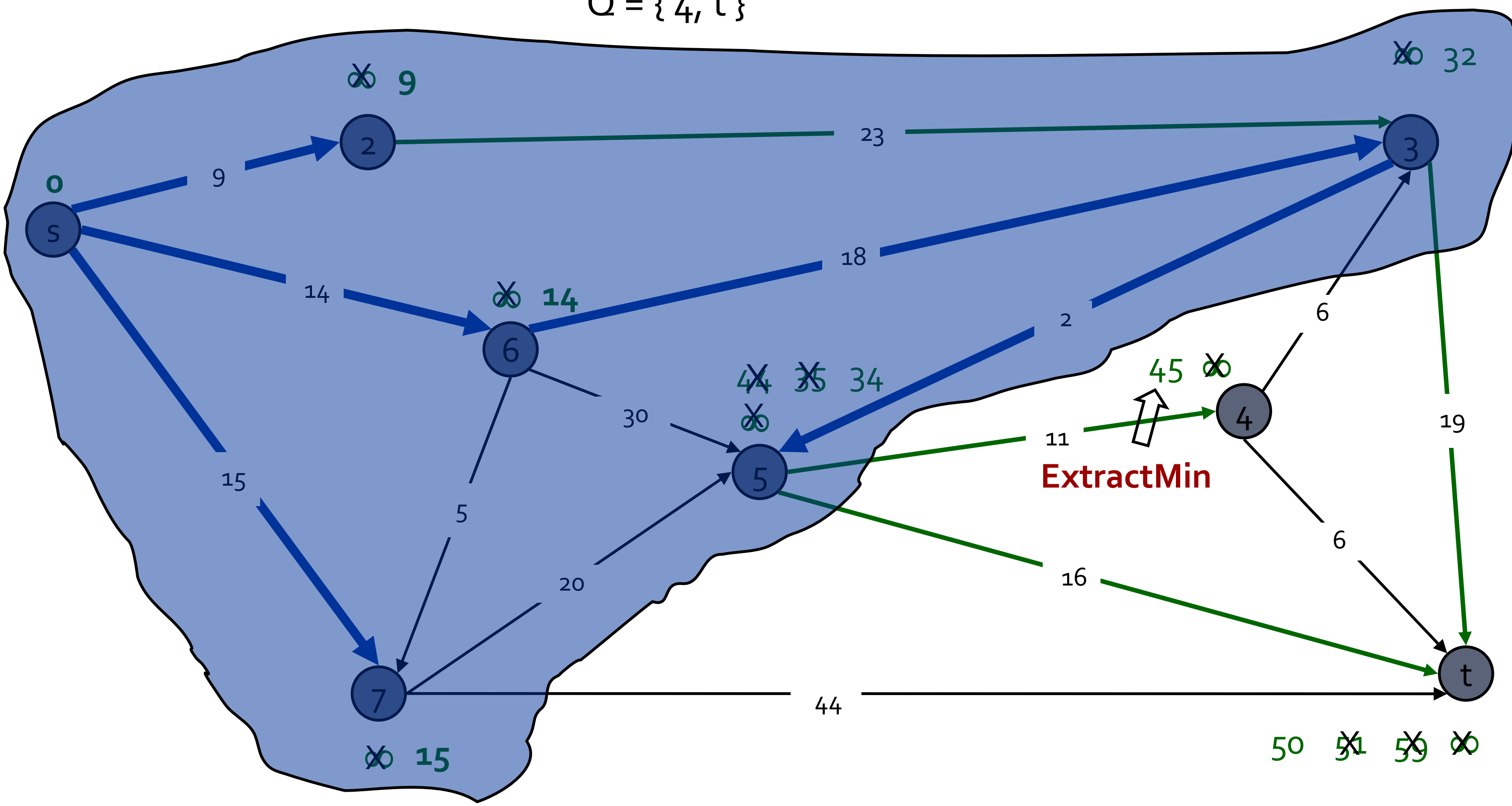
Q = { 4, t }



Dijkstra's Shortest Path Algorithm

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

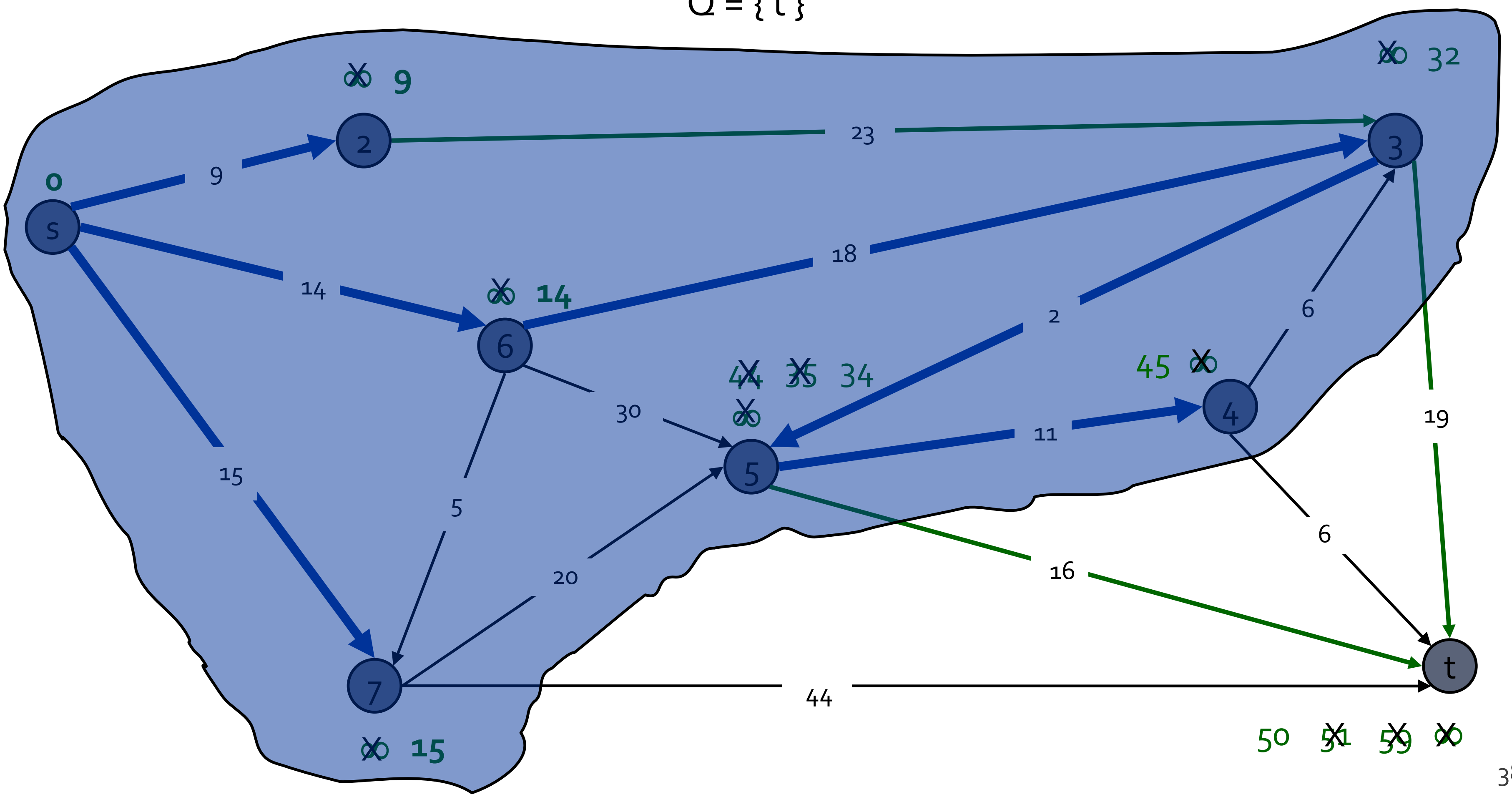
Q = { 4, t }



Dijkstra's Shortest Path Algorithm

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

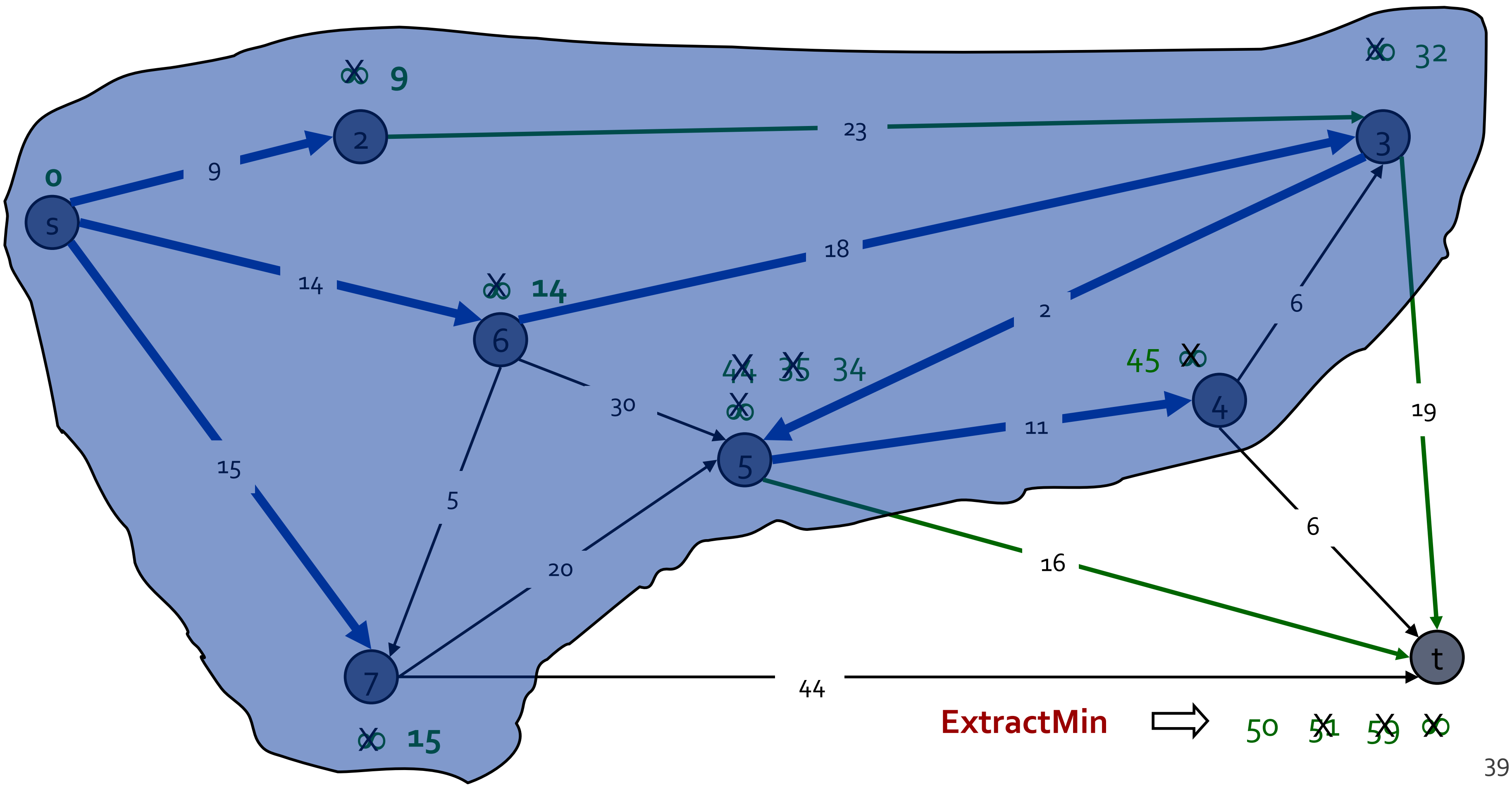
Q = { t }



Dijkstra's Shortest Path Algorithm

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

Q = { t }



Dijkstra's Algorithm

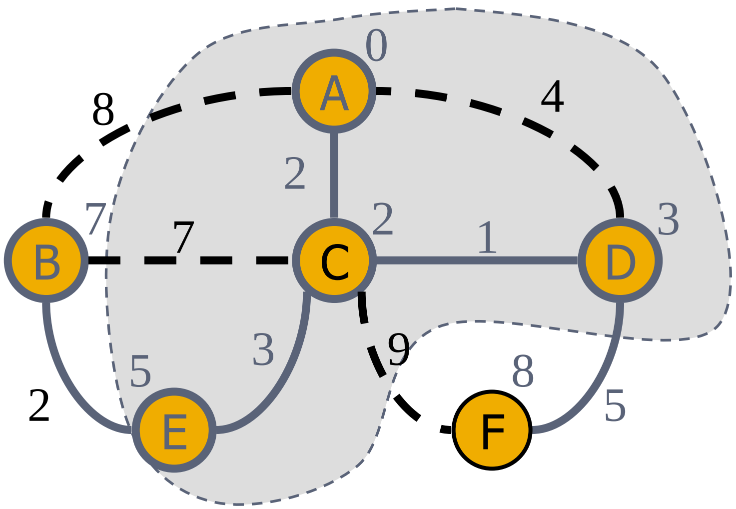
- A priority queue stores the vertices outside the cloud
  - Key: distance
  - Element: vertex
- Locator-based methods
  - *insert*(*k*,*e*) returns a locator
  - *replaceKey*(*l*,*k*) changes the key of an item
- We store two labels with each vertex:
  - Distance (d(v) label)
  - locator in priority queue

```
Algorithm DijkstraDistances(G, s)
  Q ← new heap-based priority queue
  for all v ∈ G.vertices()
    if v = s
      setDistance(v, 0)
    else
      setDistance(v, ∞)
    l ← Q.insert(getDistance(v), v)
    setLocator(v,l)
  while ¬Q.isEmpty()
    u ← Q.removeMin()
    for all e ∈ G.incidentEdges(u)
      { relax edge e }
      z ← G.opposite(u,e)
      r ← getDistance(u) + weight(e)
      if r < getDistance(z)
        setDistance(z,r)
        Q.replaceKey(getLocator(z),r)
```

Why Dijkstra's Algorithm Works

- Dijkstra's algorithm is based on the greedy method. It adds vertices by increasing distance.

  - Suppose it didn't find all shortest distances. Let F be the first wrong vertex the algorithm processed.
  - When the previous node, D, on the true shortest path was considered, its distance was correct.
  - But the edge (D,F) was **relaxed** at that time!
  - Thus, so long as d(F)≥d(D), F's distance cannot be wrong. That is, there is no wrong vertex.



Application

- Congestion and routing are two main areas of WAN which can help us to improve network performance.
- With congestion control, delay in packet delivery can be reduced to much extent.
- With optimal algorithms for routing, best possible routes can give much better network performance and faster delivery of packets.

Scope of Research

- Traffic management in wireless networks
- Route optimization in IPv6