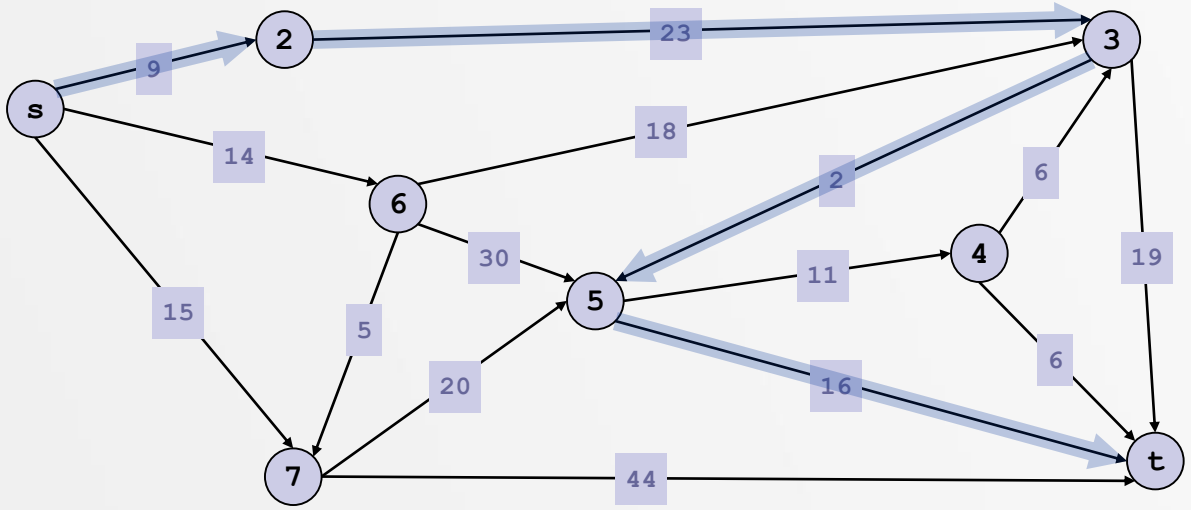
最短单元路径问题

最短单元路径问题

- 最短路径网络.
  - 有向图 G = (V, E).
  - 开始点 s, 目的地 t.
  - 长度 l_e = 边 e的长度.

- 最短路径问题：找到从 s 到 t的最短直接路径（通过路径上所有边权值之和的值的大小来评价路径长短）.



路径长度/代价
(s)-(2)-(3)-(5)-(t)
= 9 + 23 + 2 + 16
= 50.

如何进行贪心选择

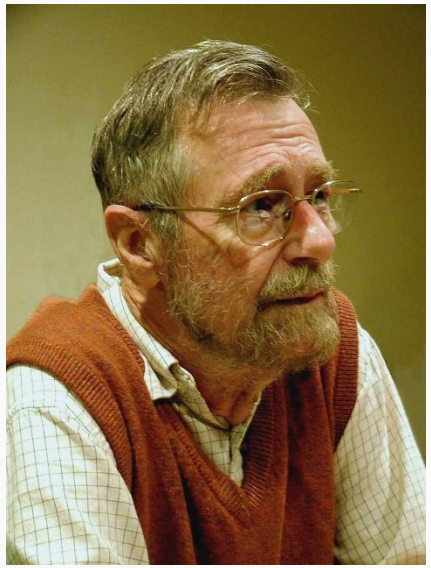
- 每步选择：选择一个节点，使得到达t更近
- 在每步中如何进行贪心选择?
- 目的：求s到所有节点的最短路径，最终求得s到t的最短路径
- 选择探索节点中距s节点最近的点及其到s的路径

Edsger W. Dijkstra

The question of whether computers can think is like the question of whether submarines can swim.

Do only what only you can do.

In their capacity as a tool, computers will be but a ripple on the surface of our culture. In their capacity as intellectual challenge, they are without precedent in the cultural history of mankind.



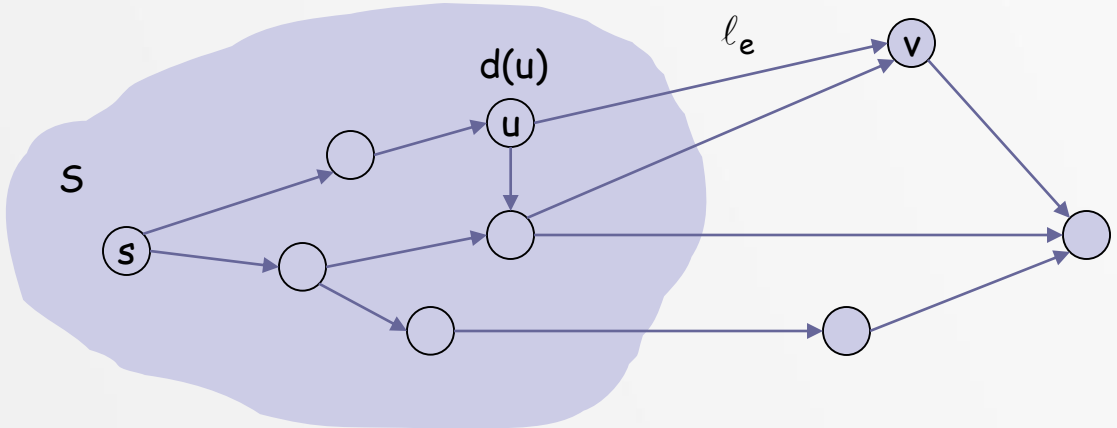
Dijkstra's 算法

- Dijkstra ‘s 算法.
  - 维护一个已被探索过的点集 S 。 S 中我们已经得到了从原点s 到点u的最短路径d(u).
  - 初始化时 S = { s }, d(s) = 0.
  - 重复选择未探索过的具有最小π(v)值的点 v

$$\pi(v) = \min_{e=(u,v):\, u \in S} d(u) + \ell_e,$$

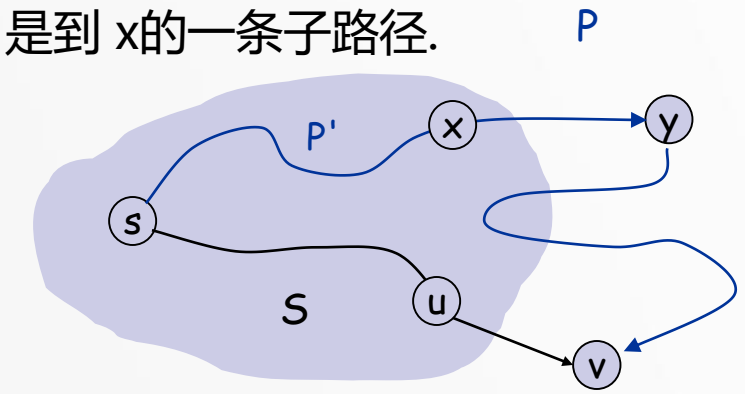
下一个加入探索节点集的点 = 具有最小 π(v)的点,假设为v′

  - 添加 v′ 到 S, 并设置d(v′) = π(v′).



Dijkstra's 算法的正确性

- 证明对于每个点 u ∈ S, d(u) 是 s到u 最短路径长度.
- 证明. (通过对 |S|进行归纳假设)
- |S| = 1 显然成立.
- 归纳假设: 假设 |S| = k ≥ 1时成立.
  - 令 v 为下一个加入到S中的点, 并令u-v 是被选择的边.
  - 最短 s-u 路径加上 (u, v) 是 s-v 长度为 π(v)的路径.
  - 记任意 s-v 的路径 P. 我们将发现没有比 π(v)更短的一条路径存在.
  - 令 x-y 为P的第一条离开S区域的边， 并令 P ‘ 是到 x的一条子路径.



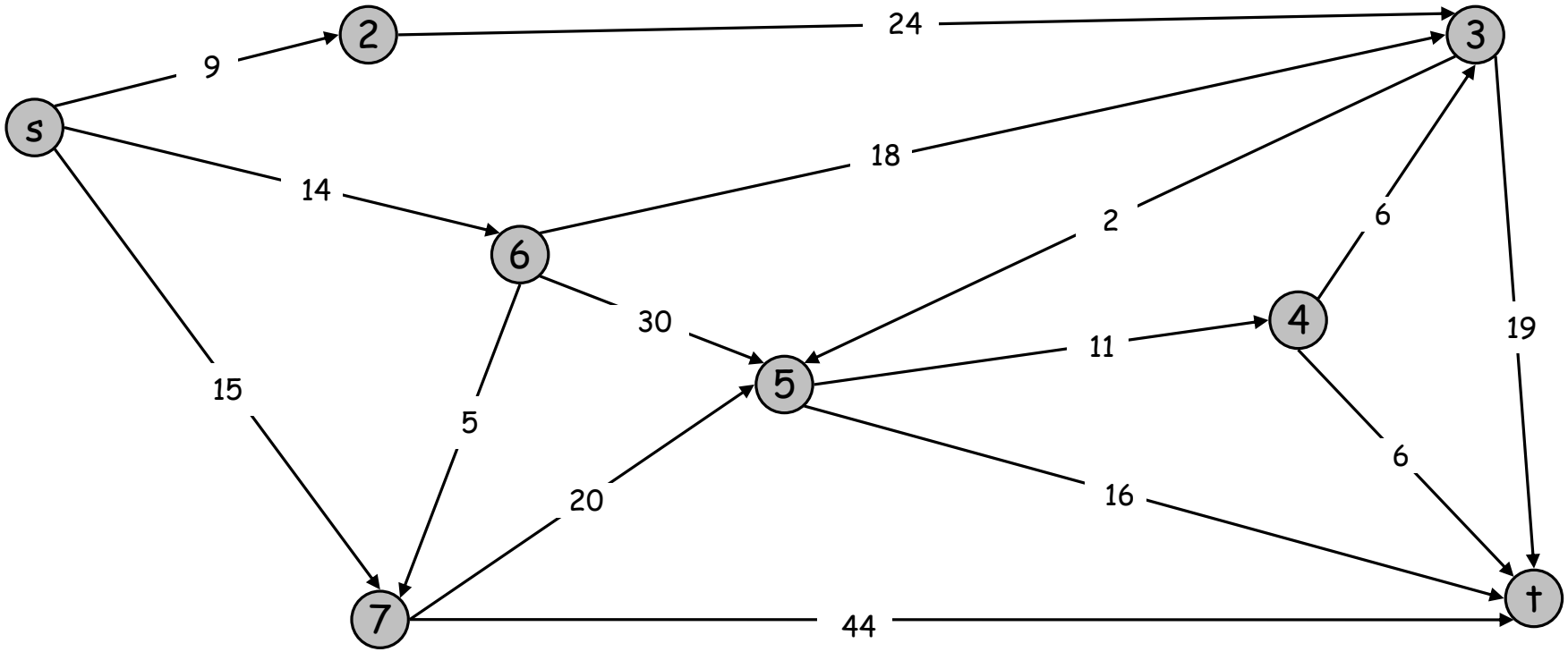
$$\ell(P) \geq \ell(P') + \ell(x,y) \geq d(x) + \ell(x,y) \geq \pi(y) \geq \pi(v)$$

- $\ell(P) \geq \ell(P') + \ell(x,y)$: 非负权重
- $\ell(P') + \ell(x,y) \geq d(x) + \ell(x,y)$: 归纳假设
- $d(x) + \ell(x,y) \geq \pi(y)$: 定义 π(y)
- $\pi(y) \geq \pi(v)$: Dijkstra 选择 v 代替 y

例子

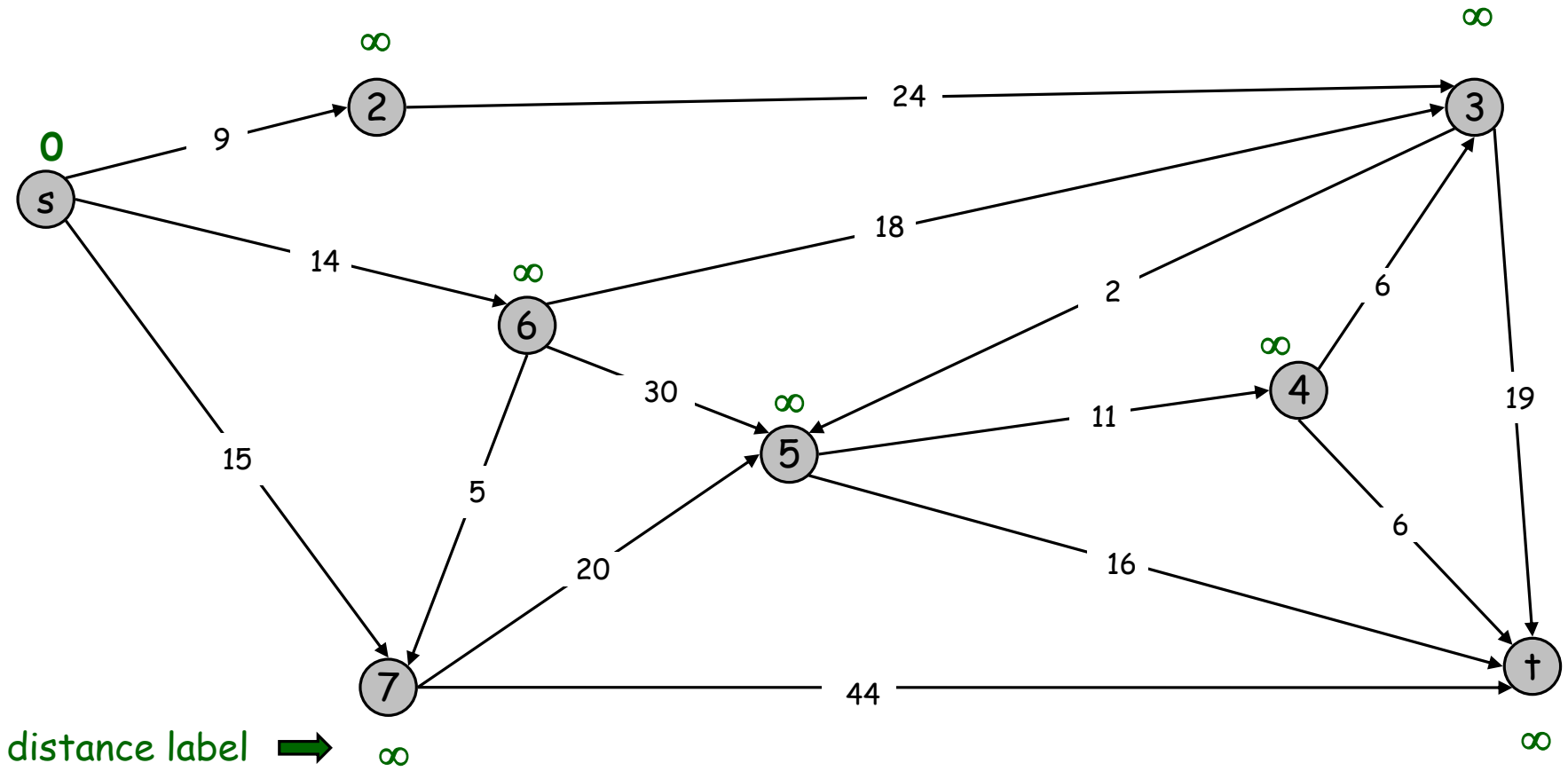
Dijkstra's

找到**s** 到 **t**的最短路径.



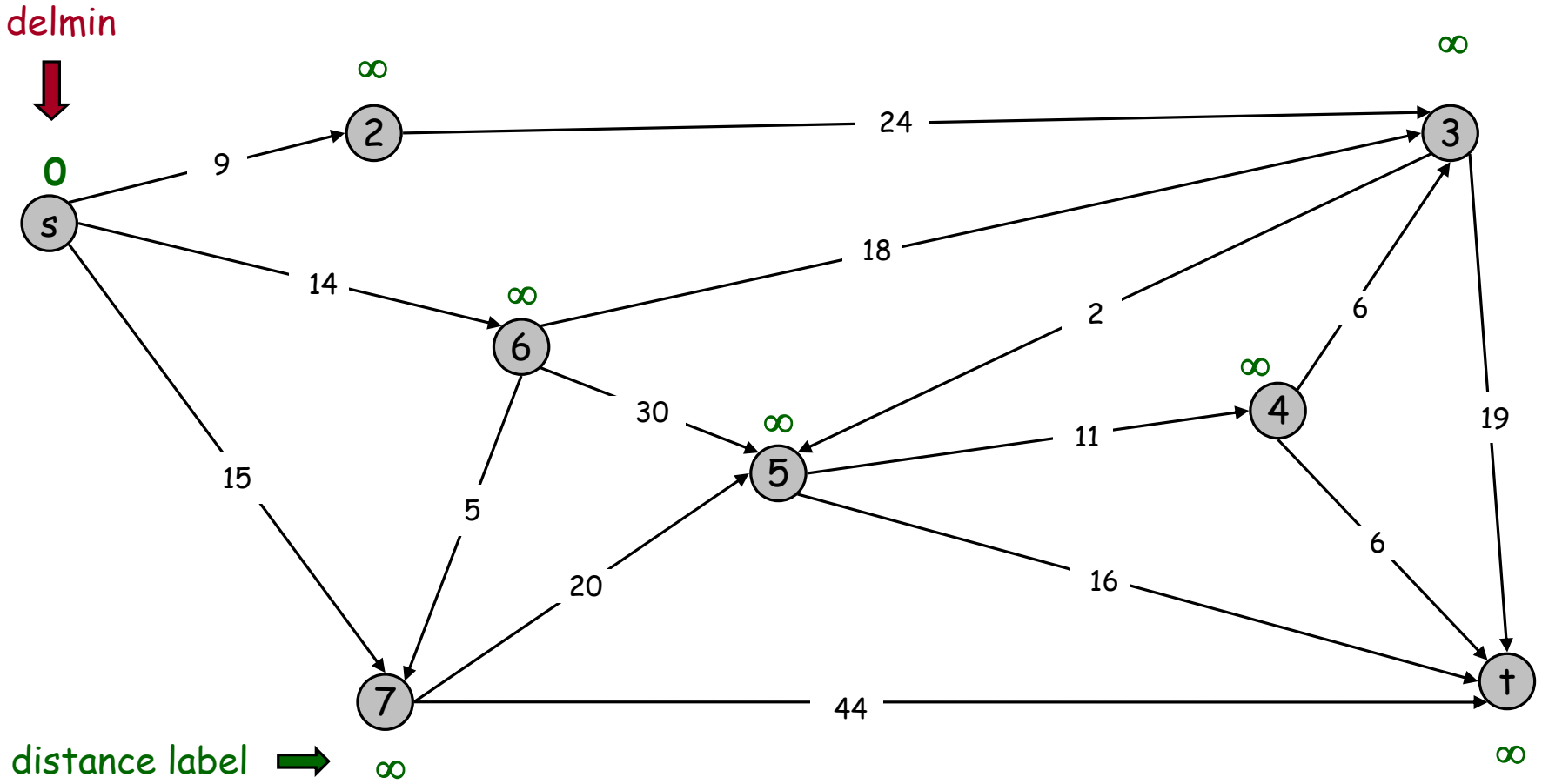
Dijkstra's Shortest Path Algorithm

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



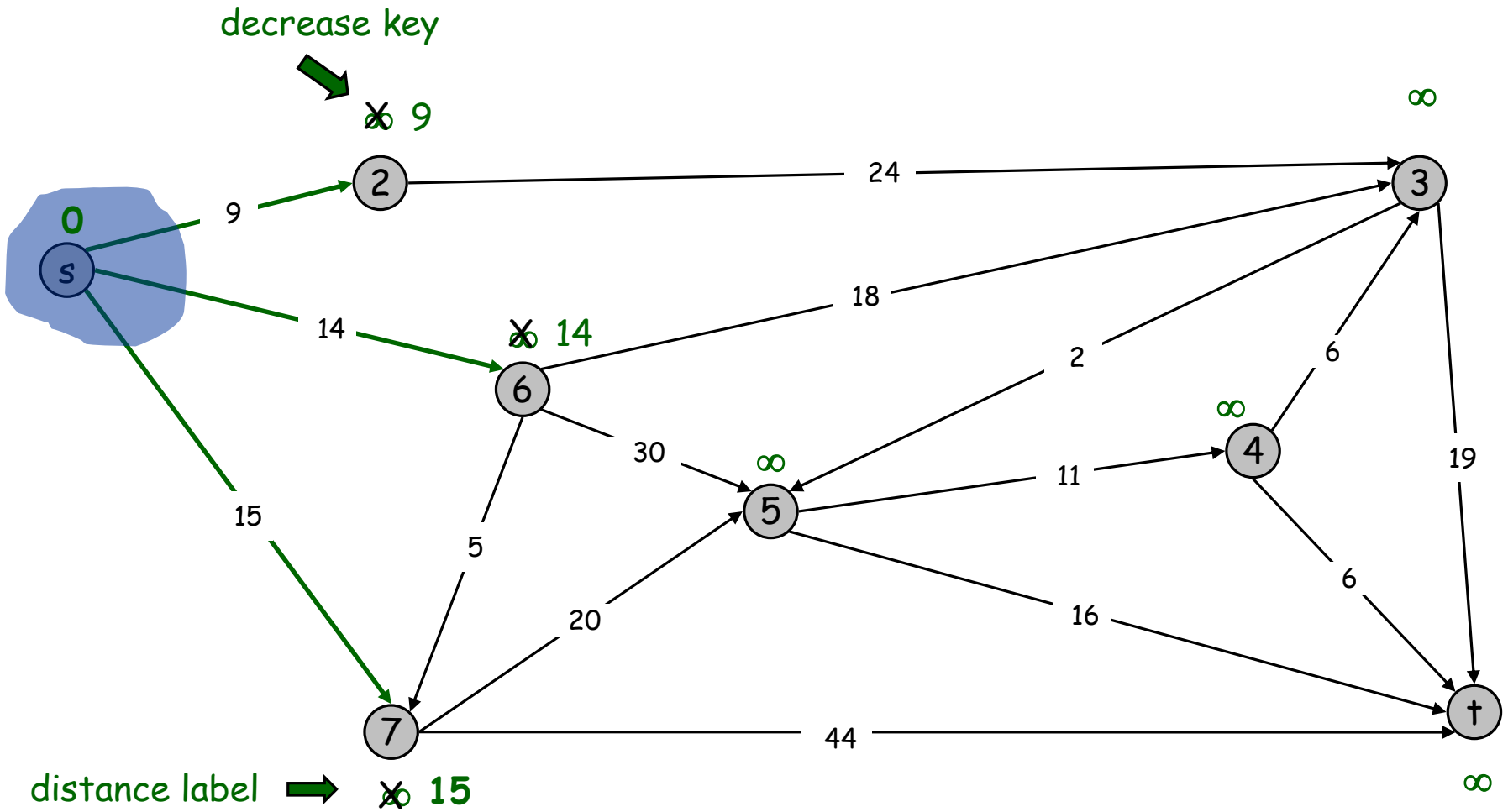
Dijkstra's Shortest Path Algorithm

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



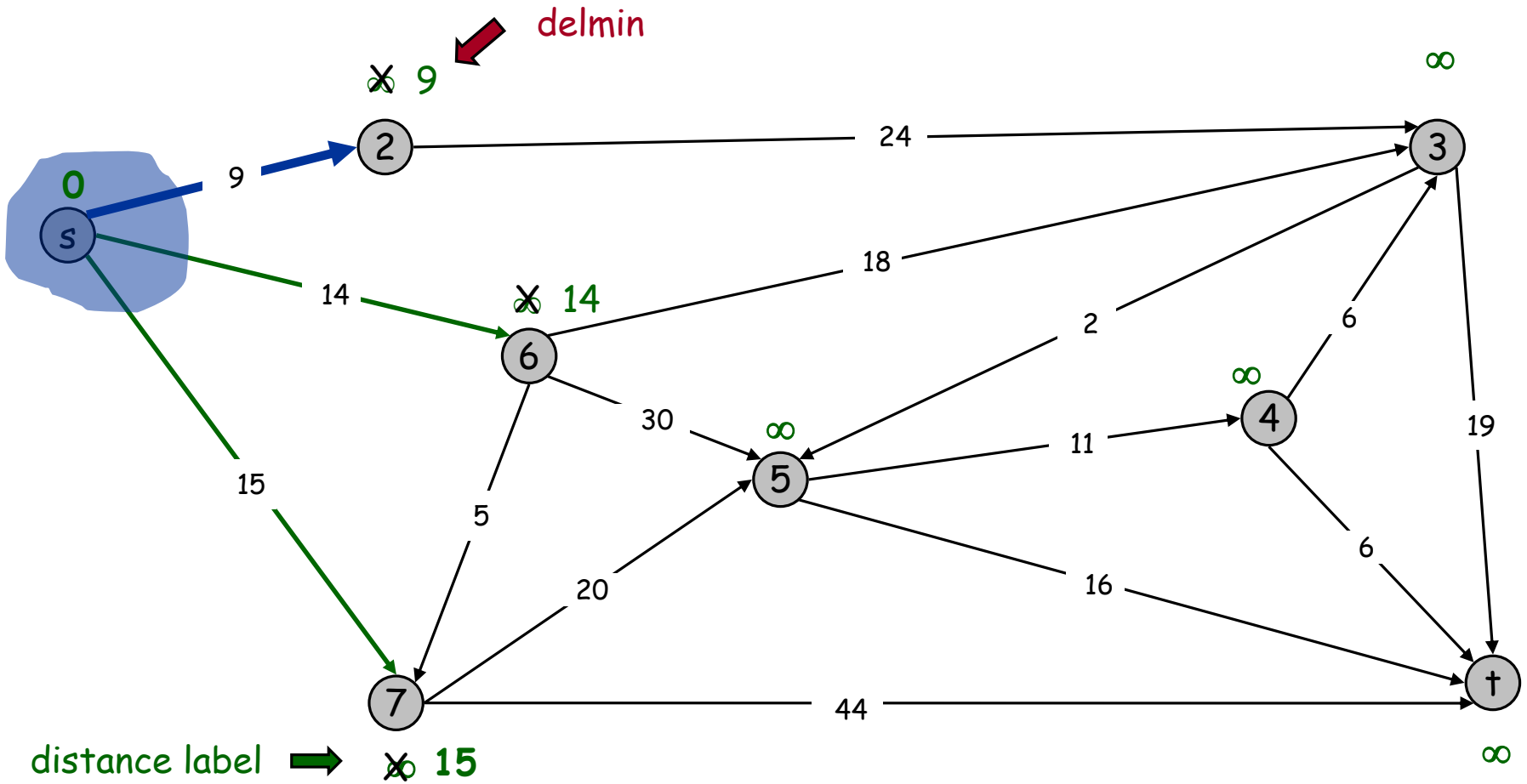
Dijkstra's Shortest Path Algorithm

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



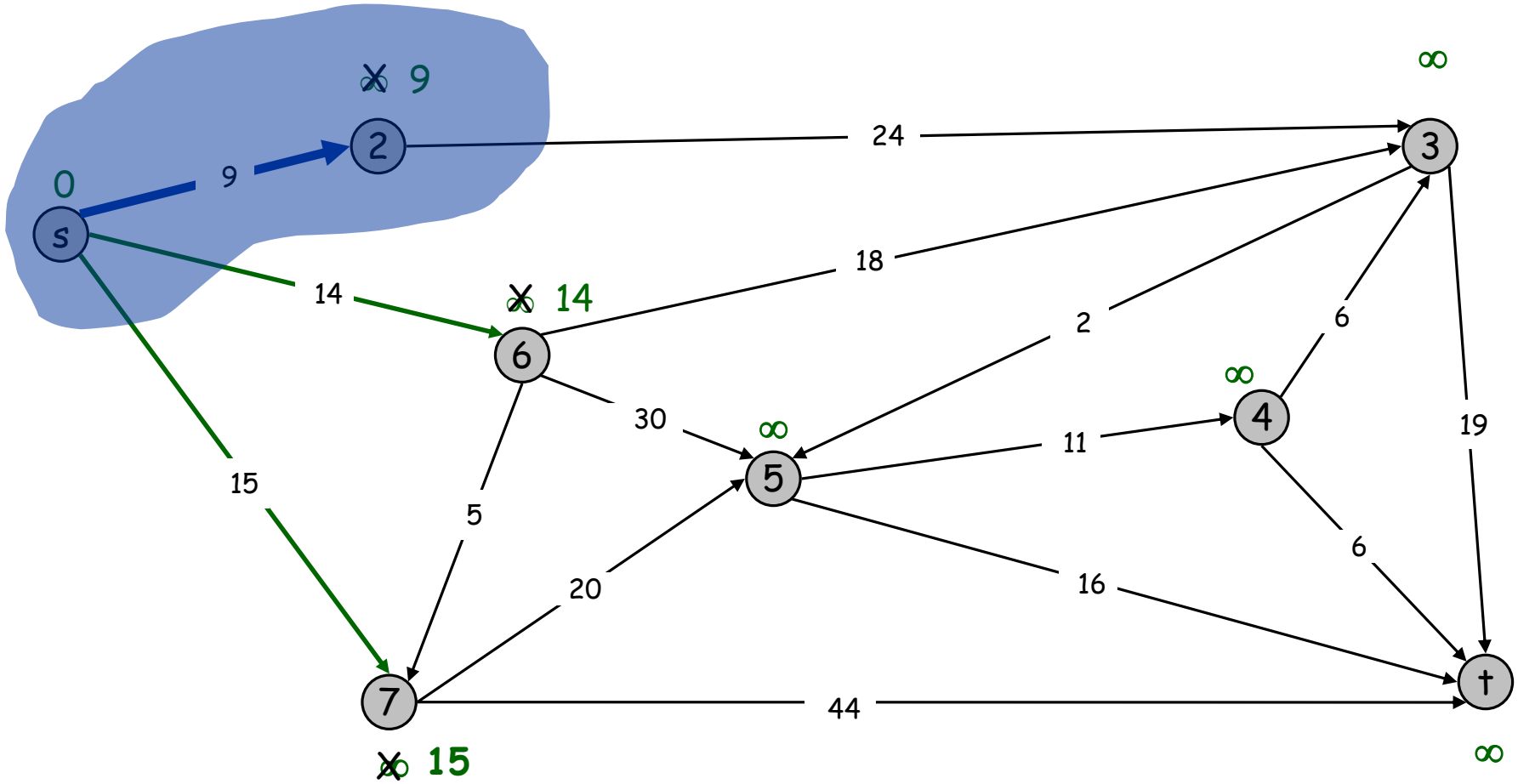
Dijkstra's Shortest Path Algorithm

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



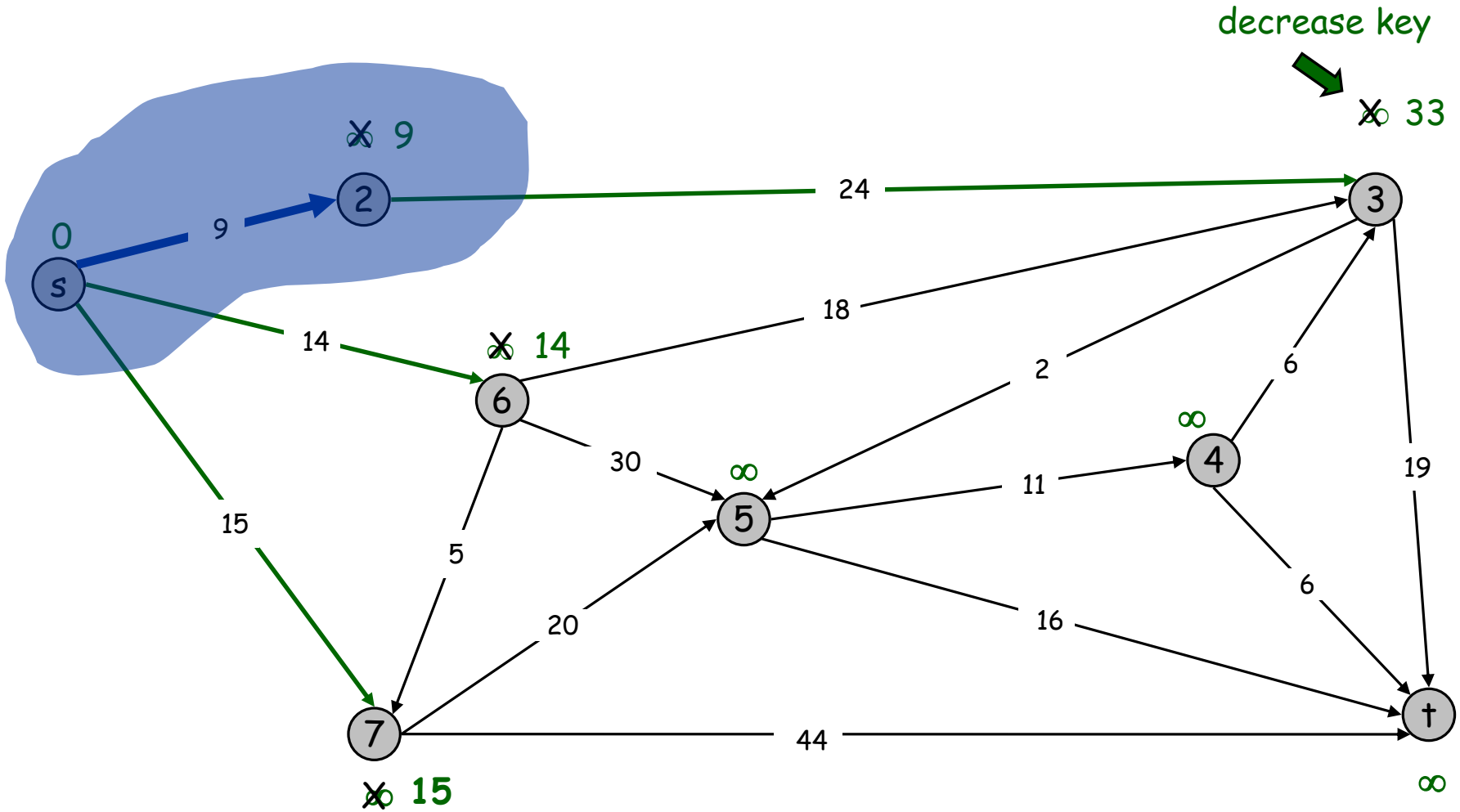
Dijkstra's Shortest Path Algorithm

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



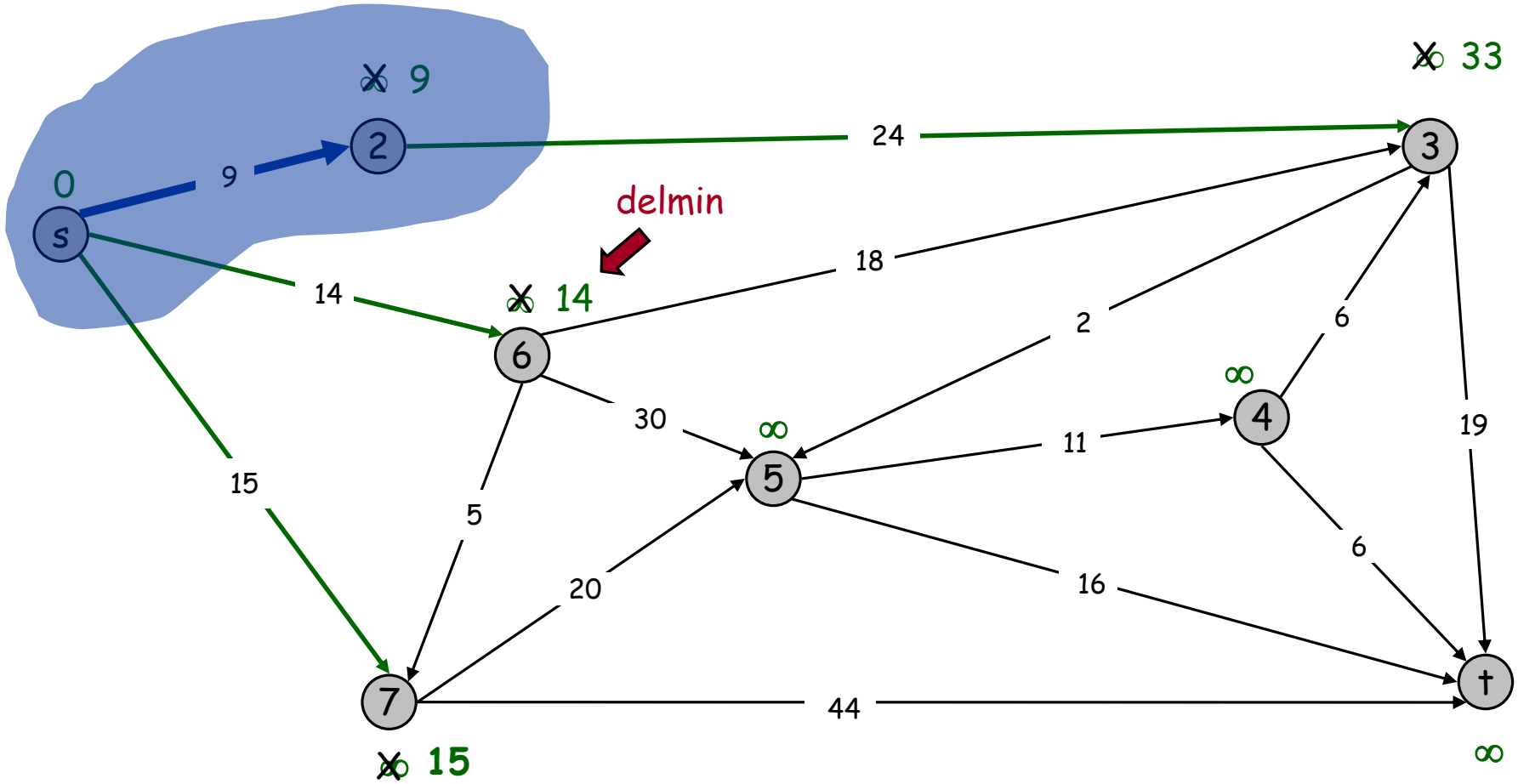
Dijkstra's Shortest Path Algorithm

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



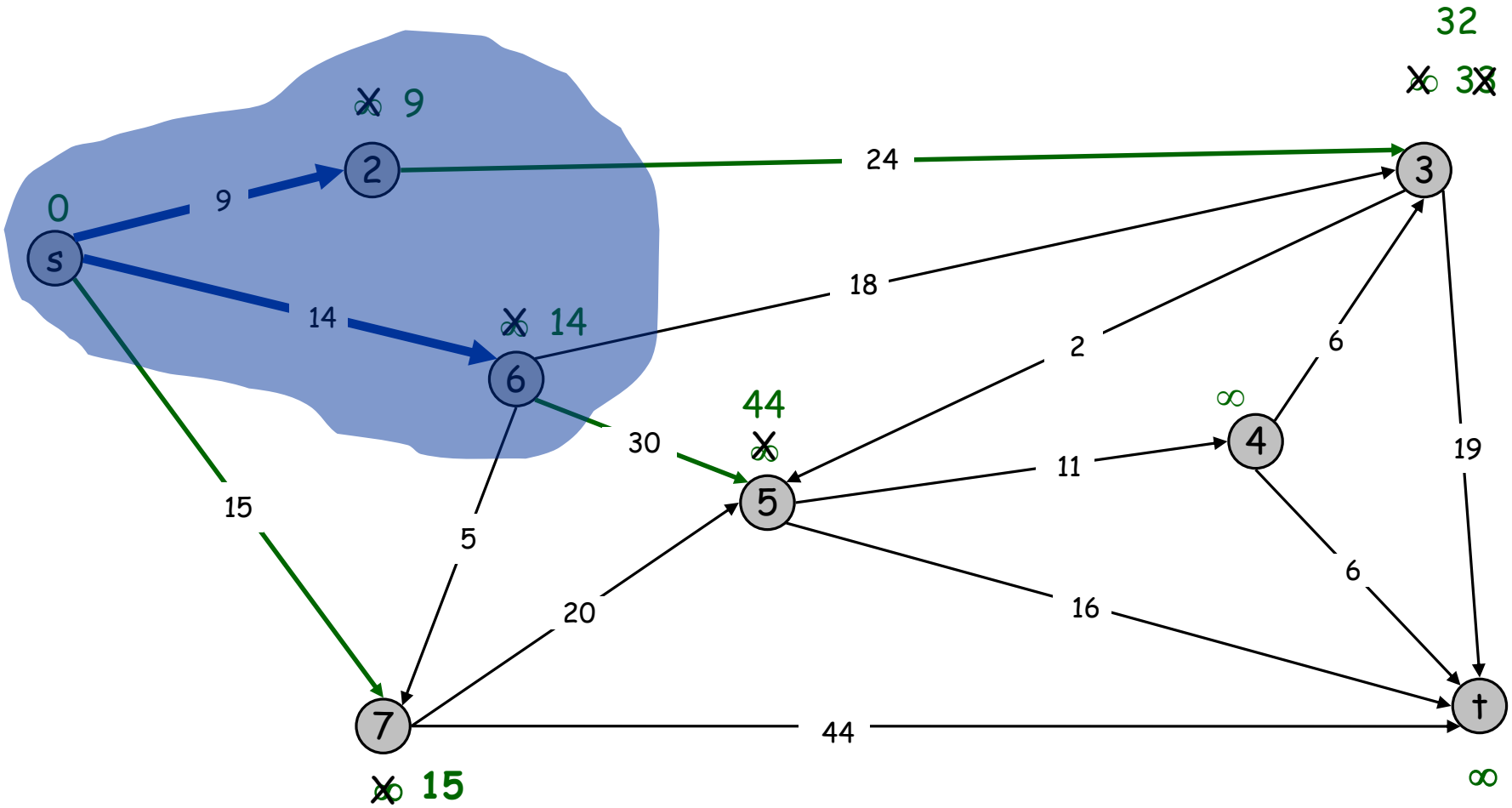
Dijkstra's Shortest Path Algorithm

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



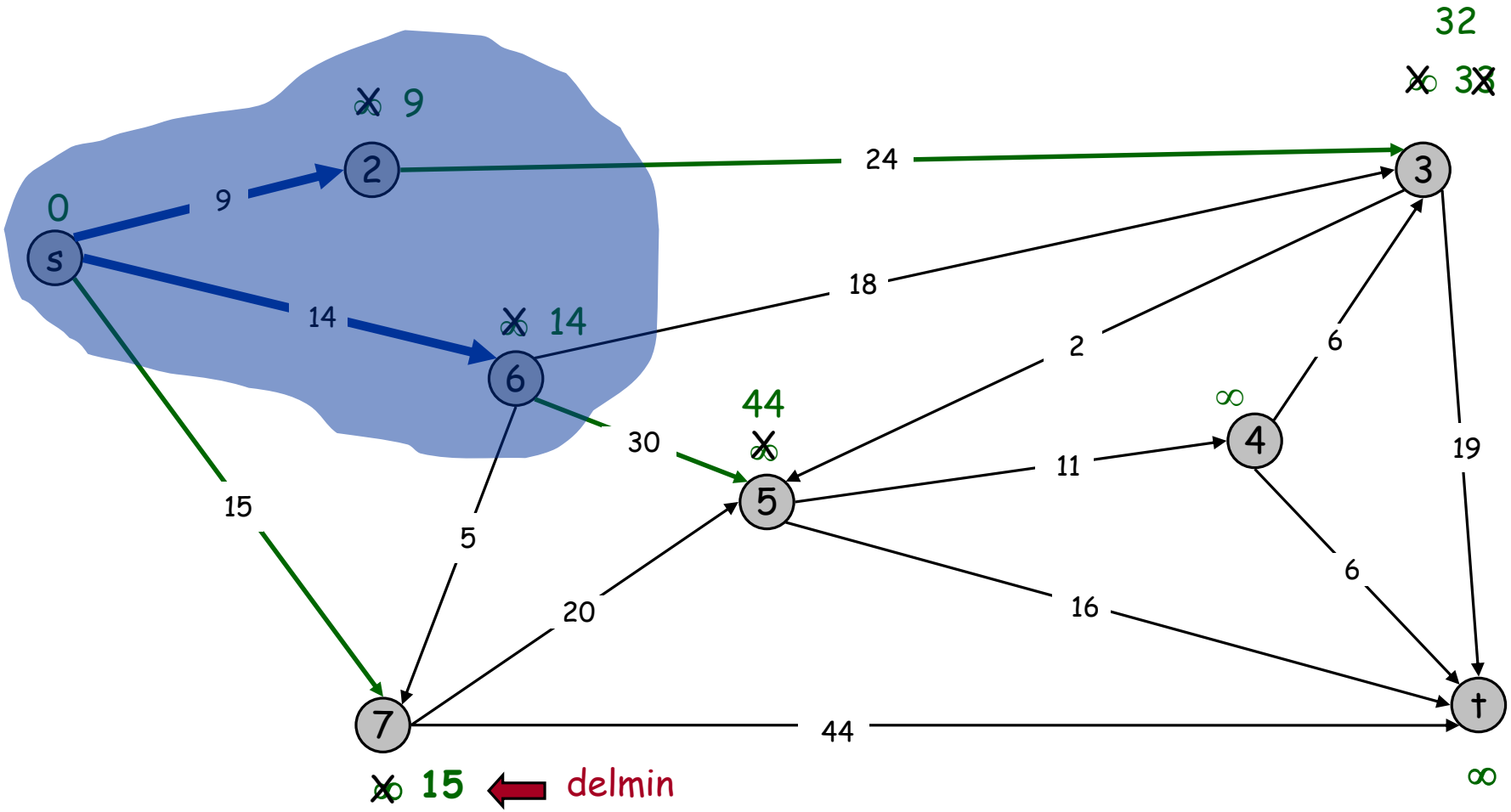
Dijkstra's Shortest Path Algorithm

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



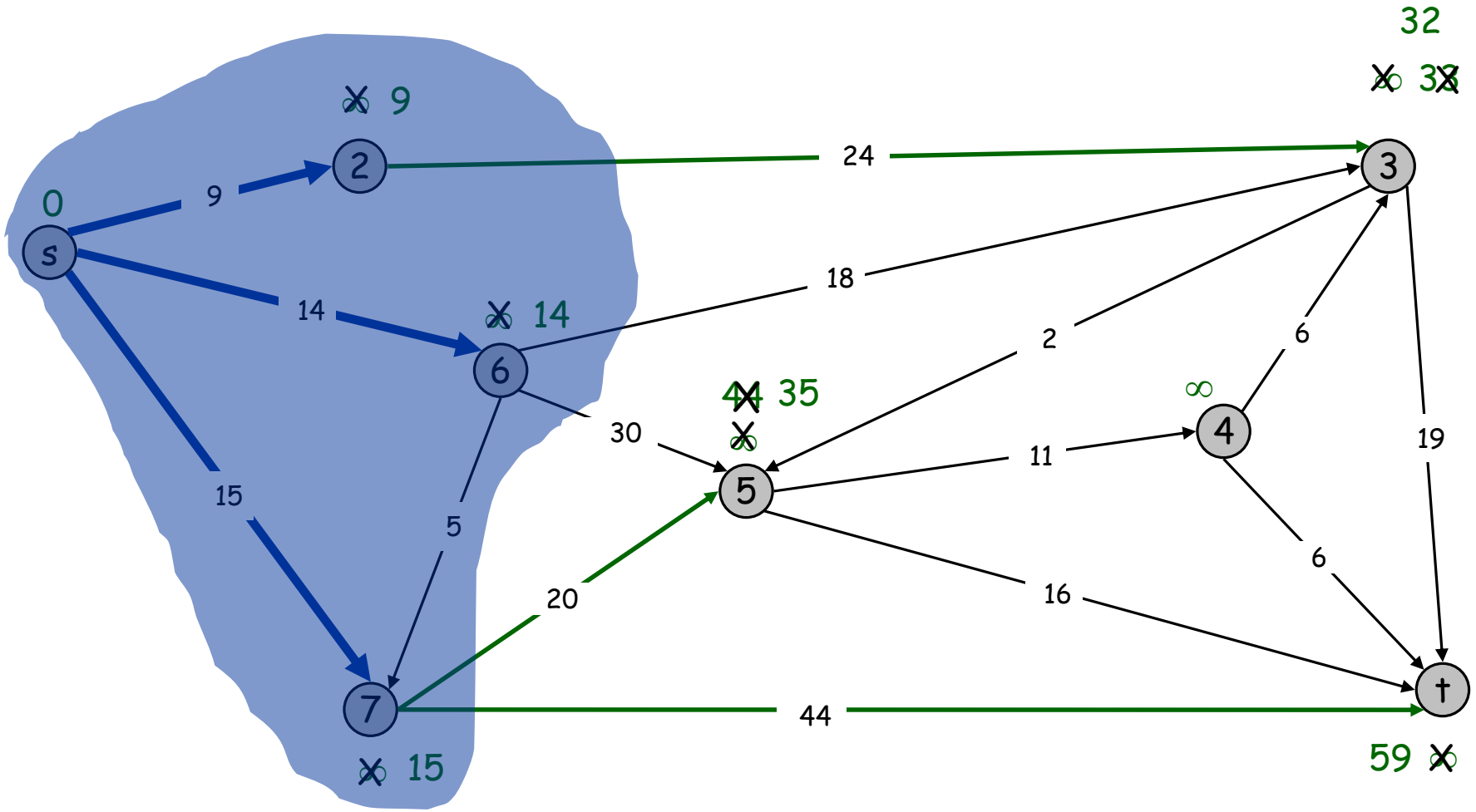
Dijkstra's Shortest Path Algorithm

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



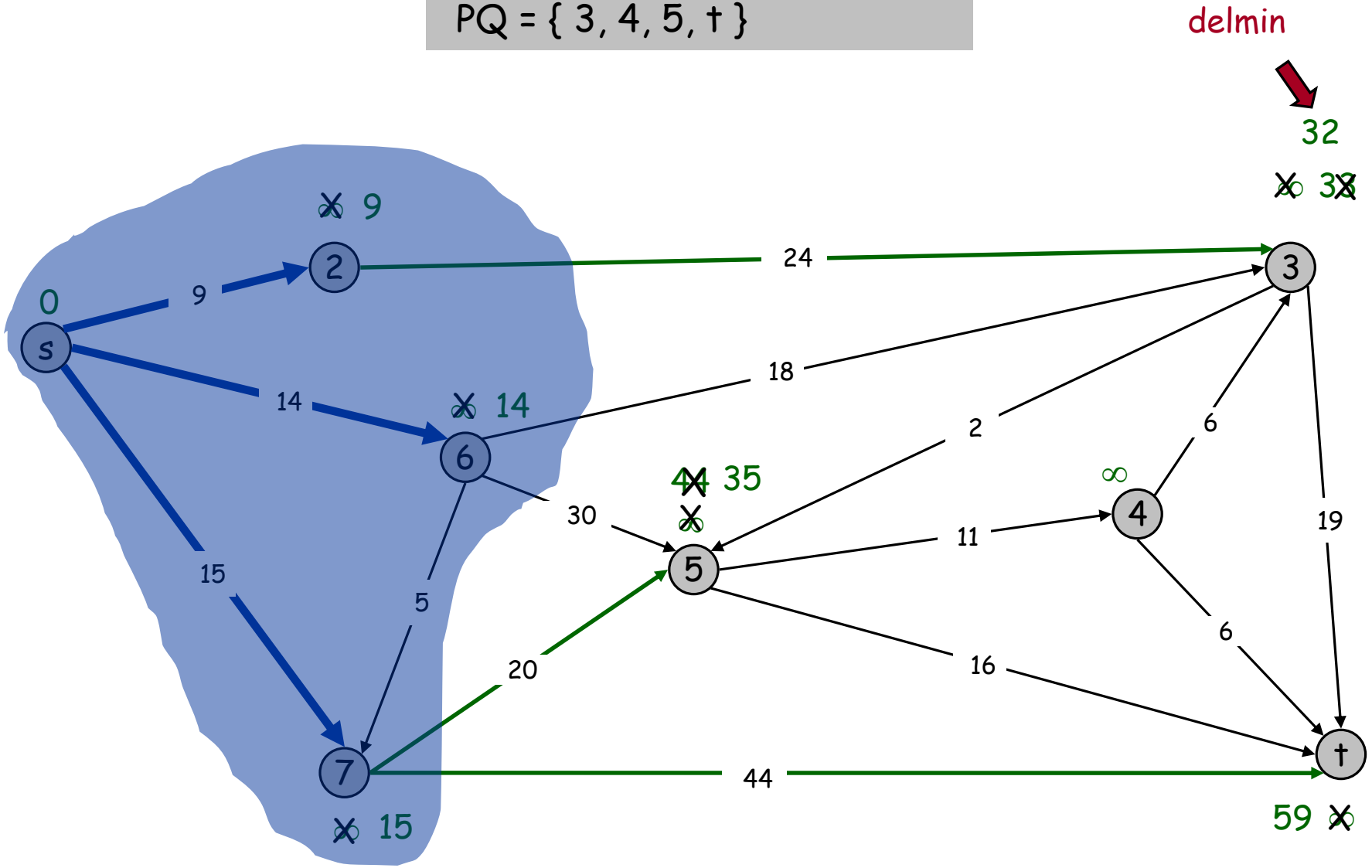
Dijkstra's Shortest Path Algorithm

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



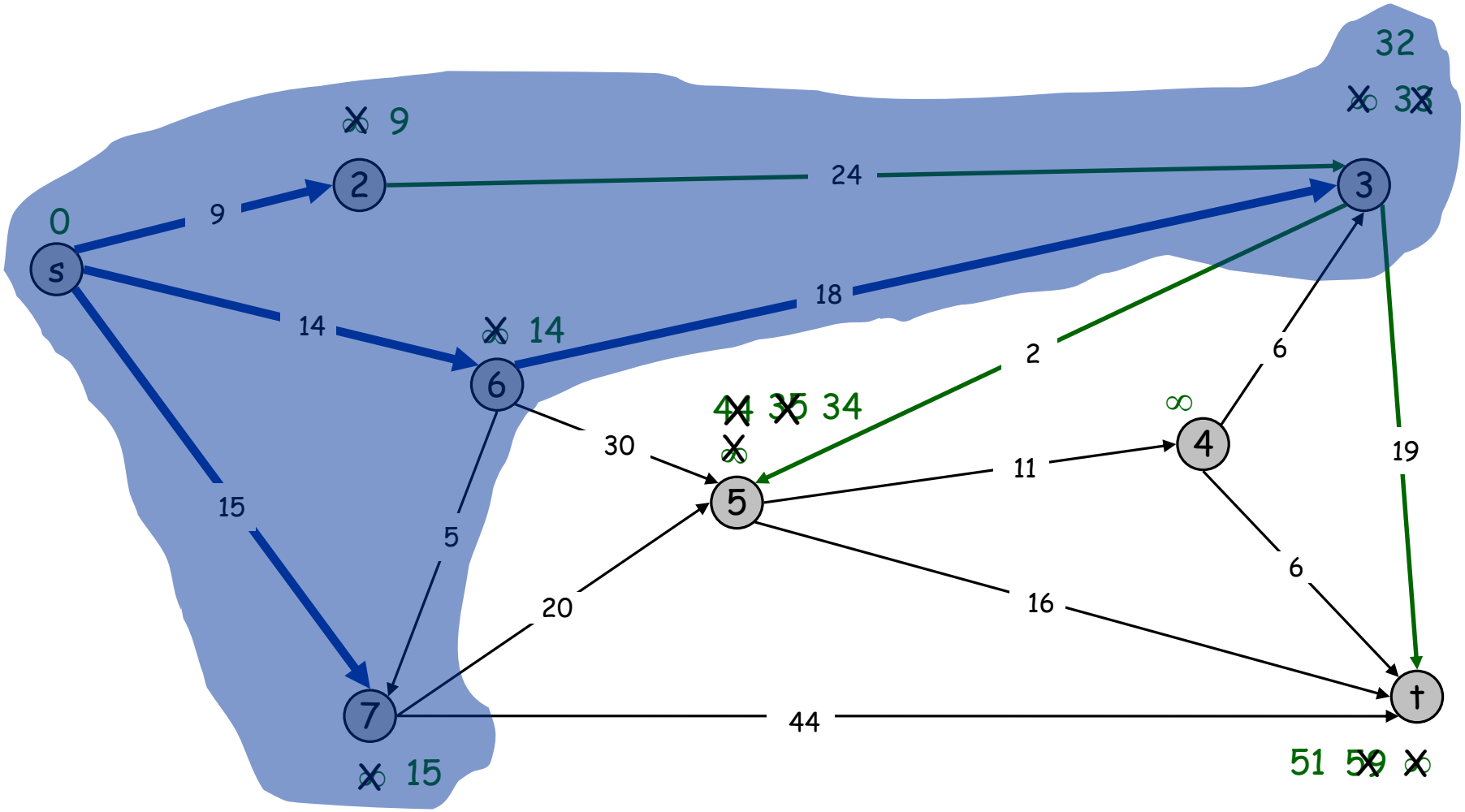
Dijkstra's Shortest Path Algorithm

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



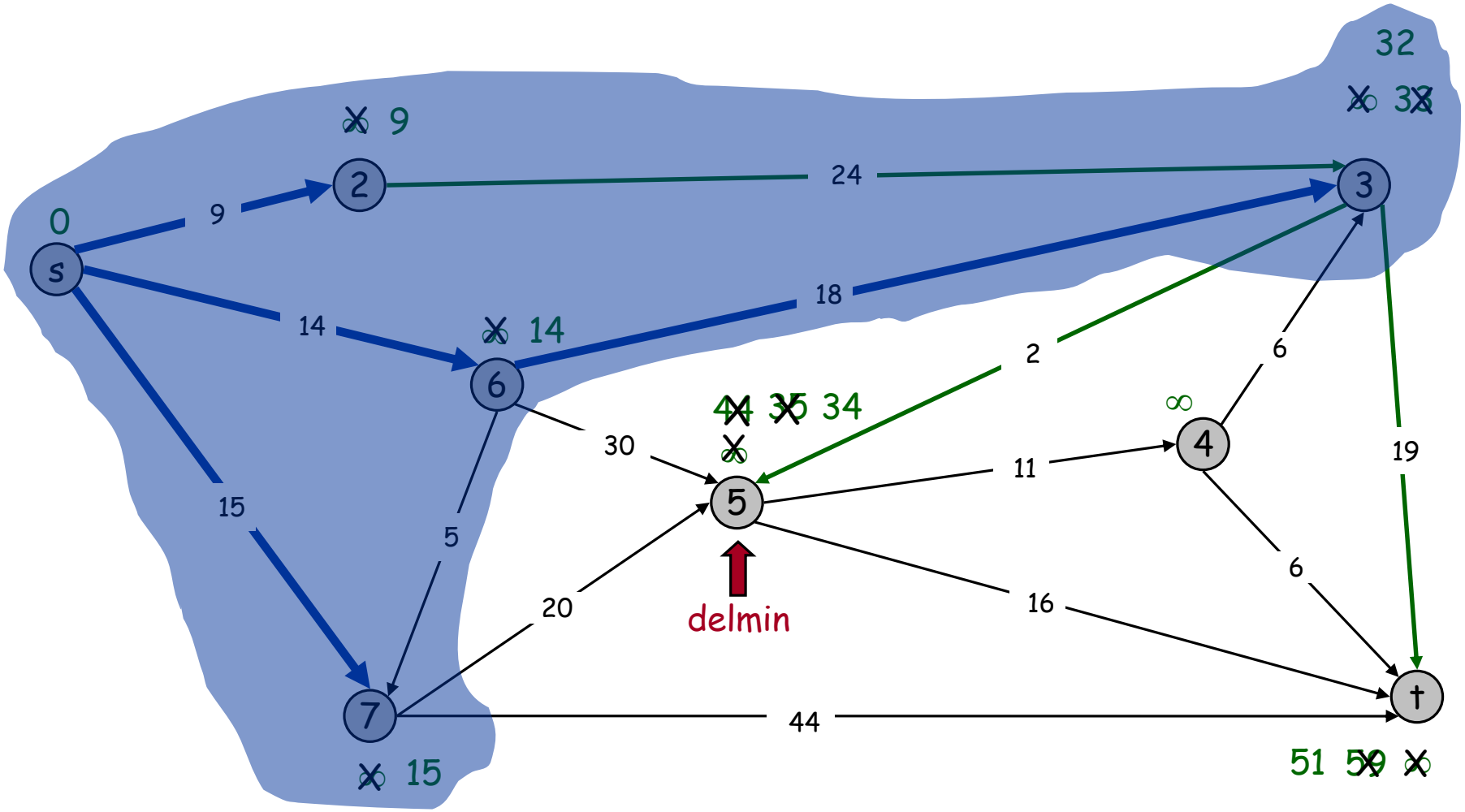
Dijkstra's Shortest Path Algorithm

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



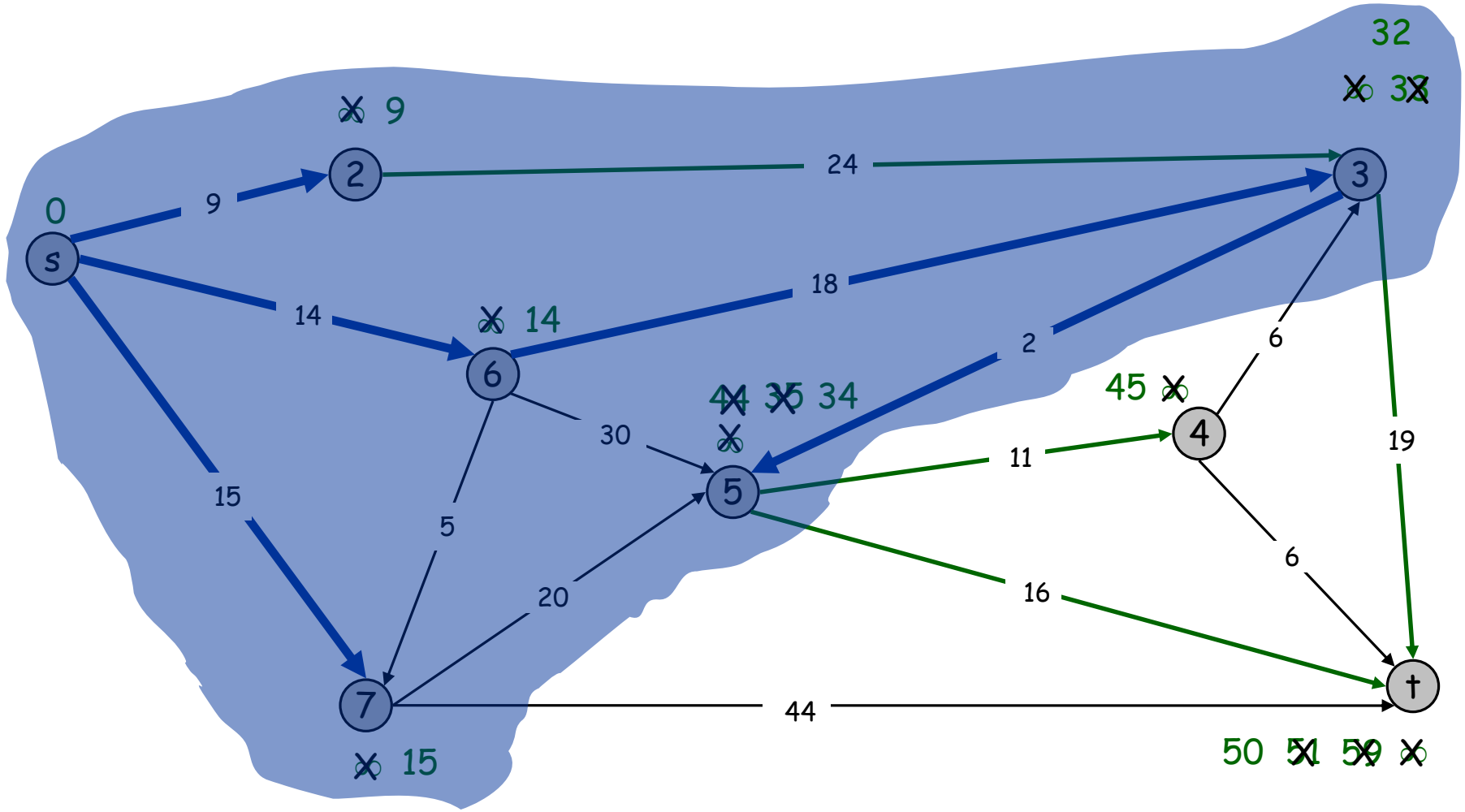
Dijkstra's Shortest Path Algorithm

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



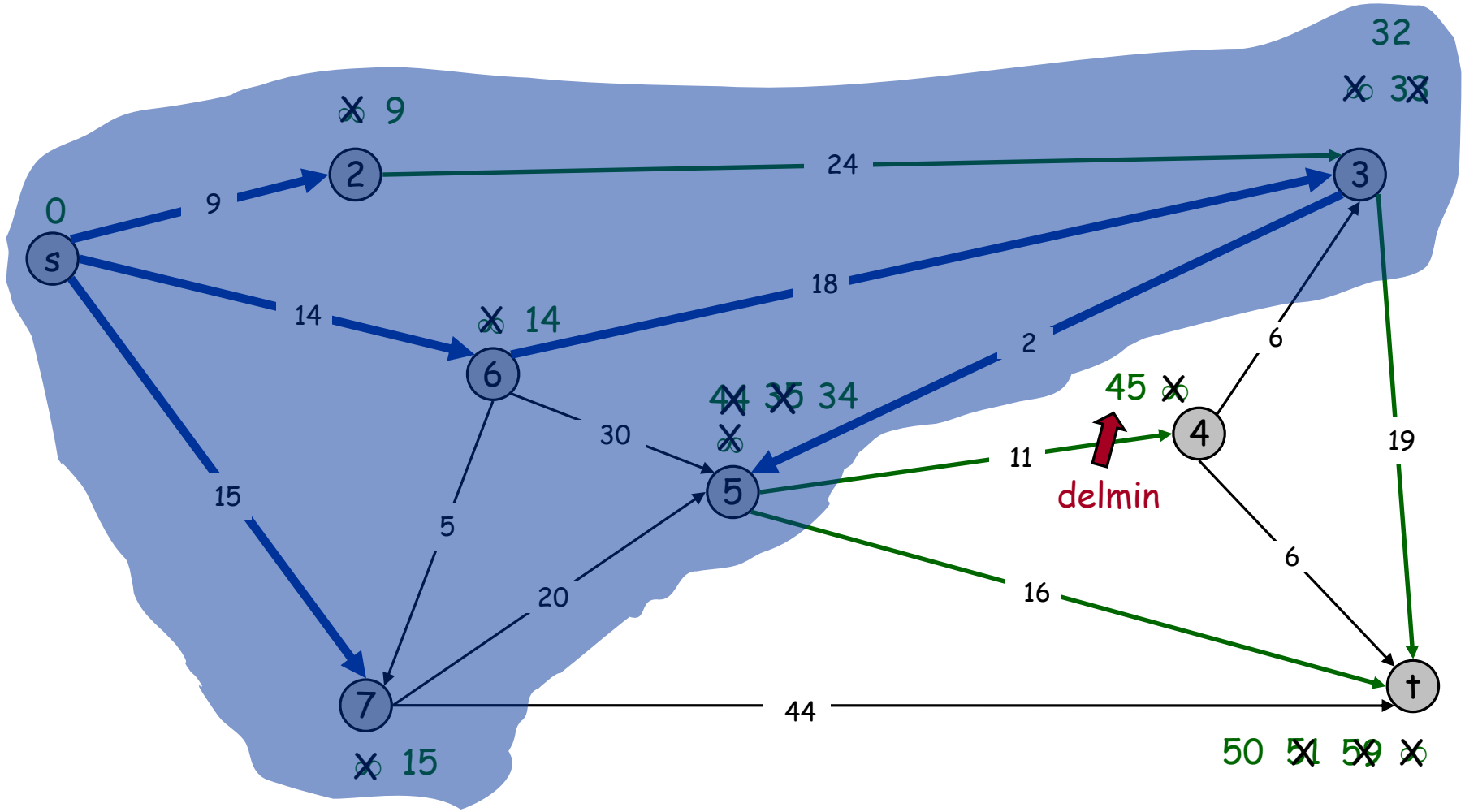
Dijkstra's Shortest Path Algorithm

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



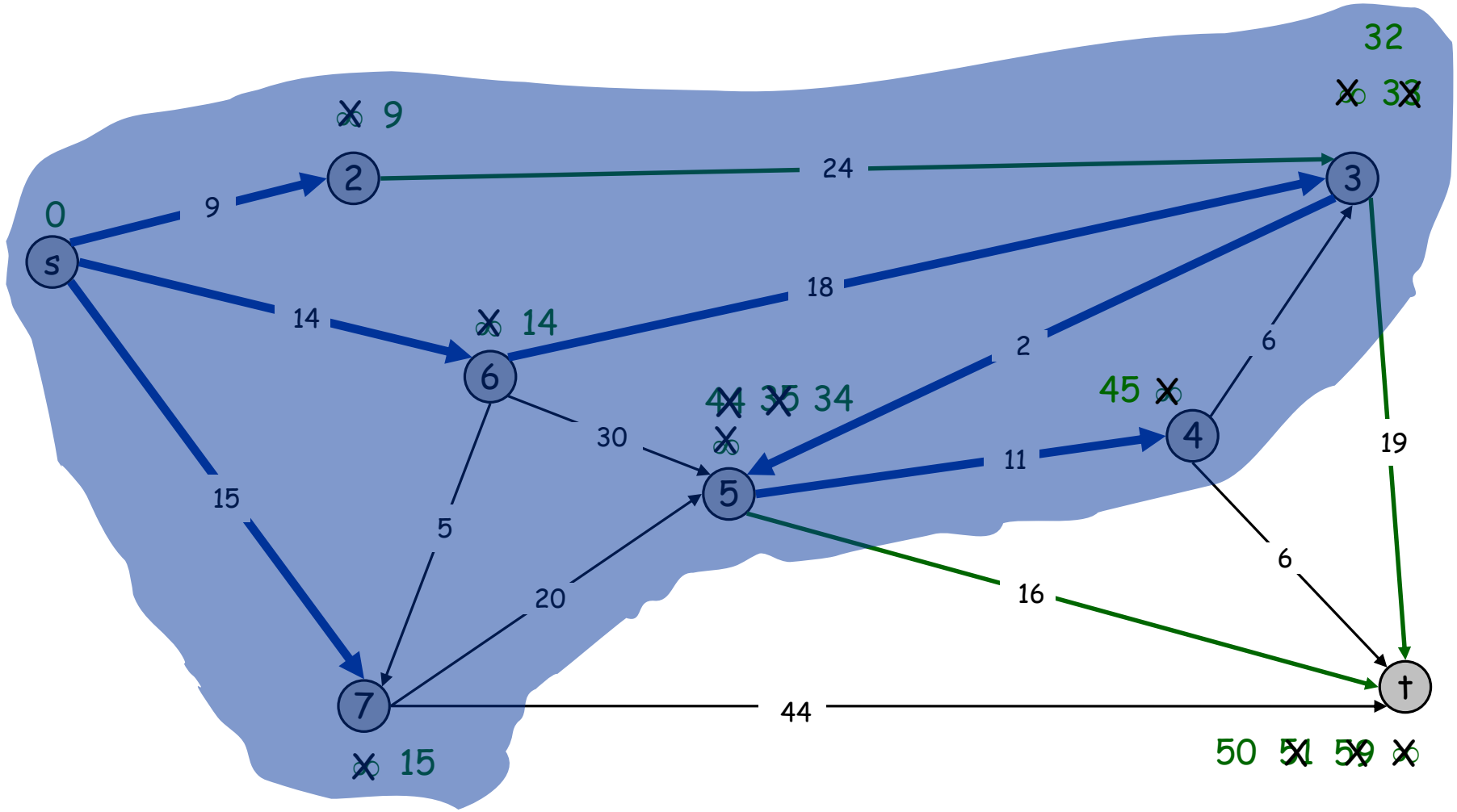
Dijkstra's Shortest Path Algorithm

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



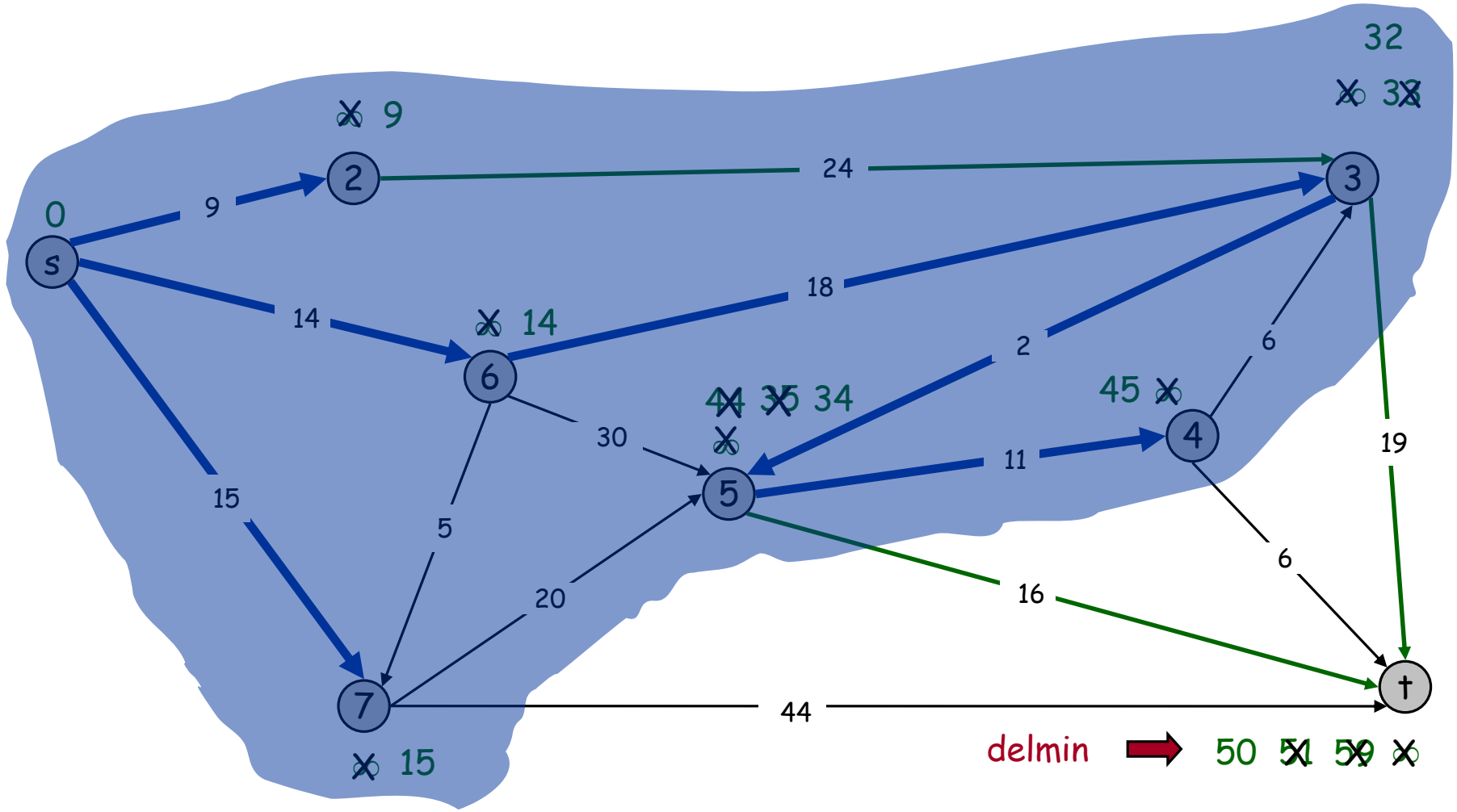
Dijkstra's Shortest Path Algorithm

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



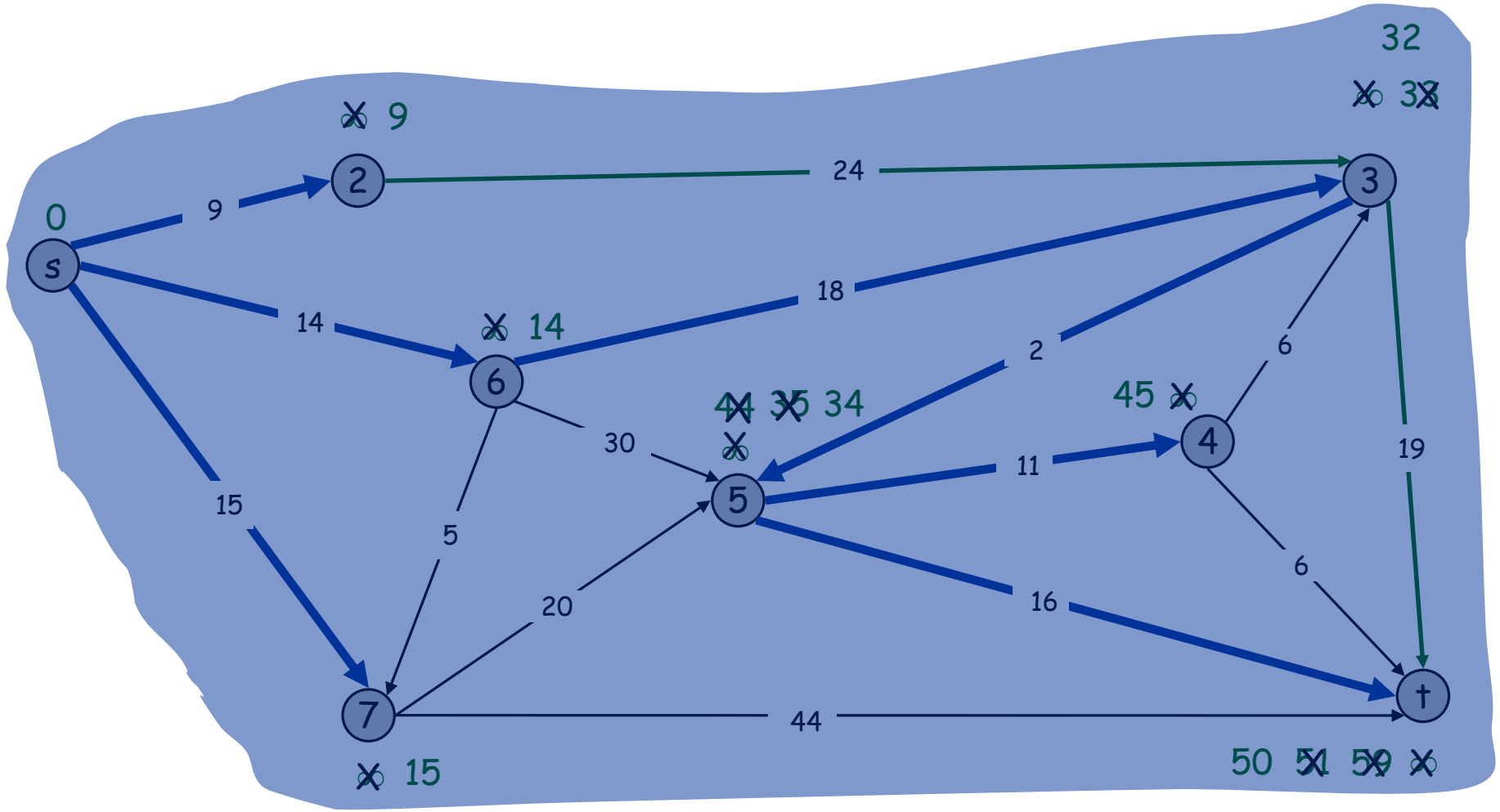
Dijkstra's Shortest Path Algorithm

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



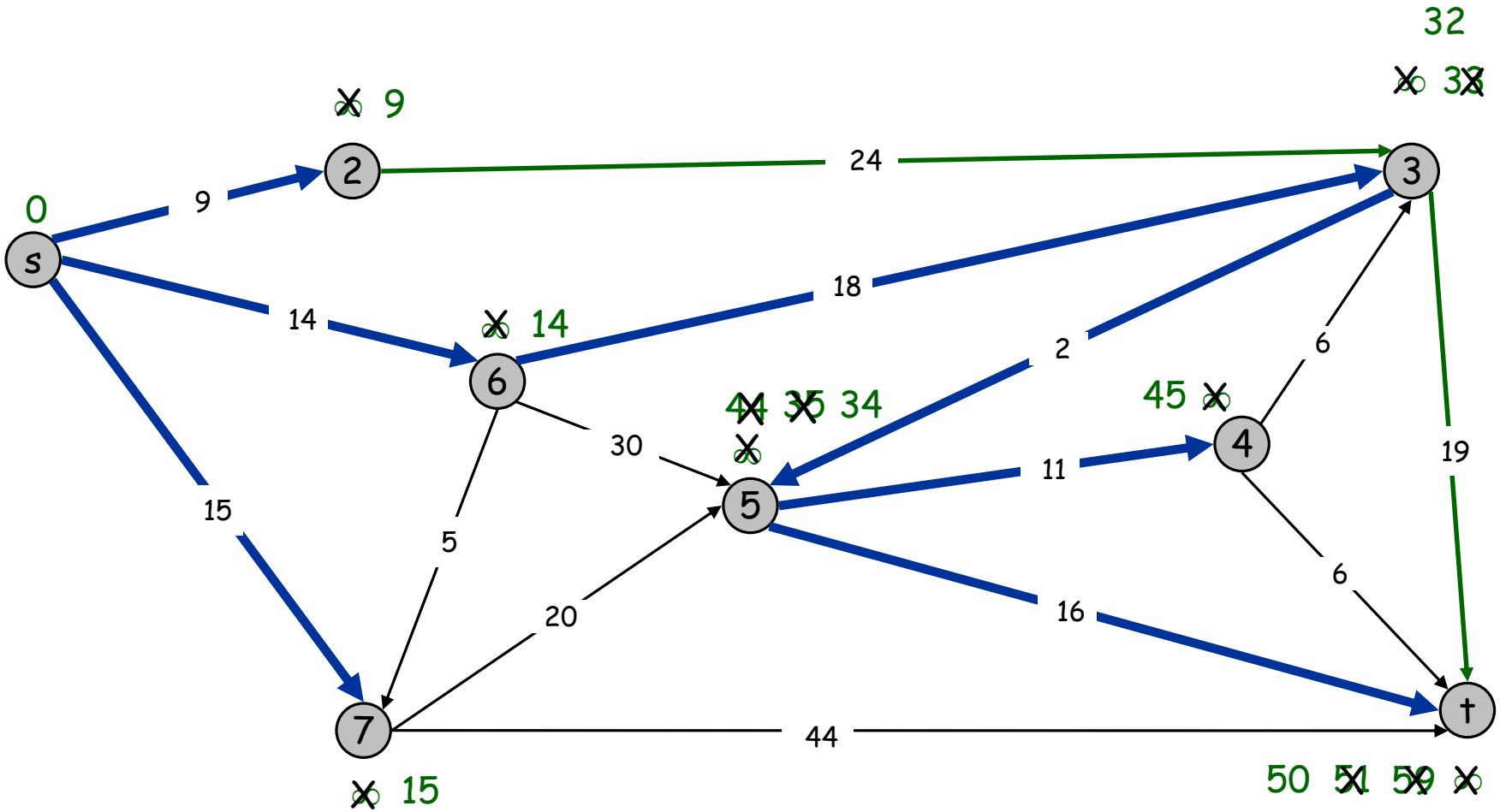
Dijkstra's Shortest Path Algorithm

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



Dijkstra's Shortest Path Algorithm

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



课堂问题

- 问题:
- 如何找到无向图中s到t的最短路径
- 如果存在负数权重又该怎么处理