Lecture 6: Speeding up Dijkstra

Overview

- Single-source single-target Dijkstra
- Bidirectional search
- Goal directed search - potentials and landmarks

Dijkstra's Algorithm

For each edge $(u, v) \epsilon E$, assume $w(u, v) \geq 0$, maintain a set S of vertices whose final shortest path weights have been determined. Repeatedly select $u \epsilon V - S$ with <u>minimum</u> shortest path estimate, add u to S, relax all edges out of u.

Pseudo-code

Dijkstra (G, W, s) //uses priority queue Q

    Initialize (G, s)
    $S \leftarrow \phi$
    $Q \leftarrow V[G]$ //Insert into Q
    while $Q \neq \phi$
        do $u \leftarrow$ EXTRACT-MIN(Q) //deletes u from Q
        $S = S \cup \{u\}$
        for each vertex $v \epsilon$ Adj$[u]$
            do RELAX (u, v, w) $\leftarrow$ this is an implicit DECREASE_KEY operation

DIJKSTRA single-source, single-target

Initialize()

$Q \leftarrow V[G]$

while $Q \neq \phi$

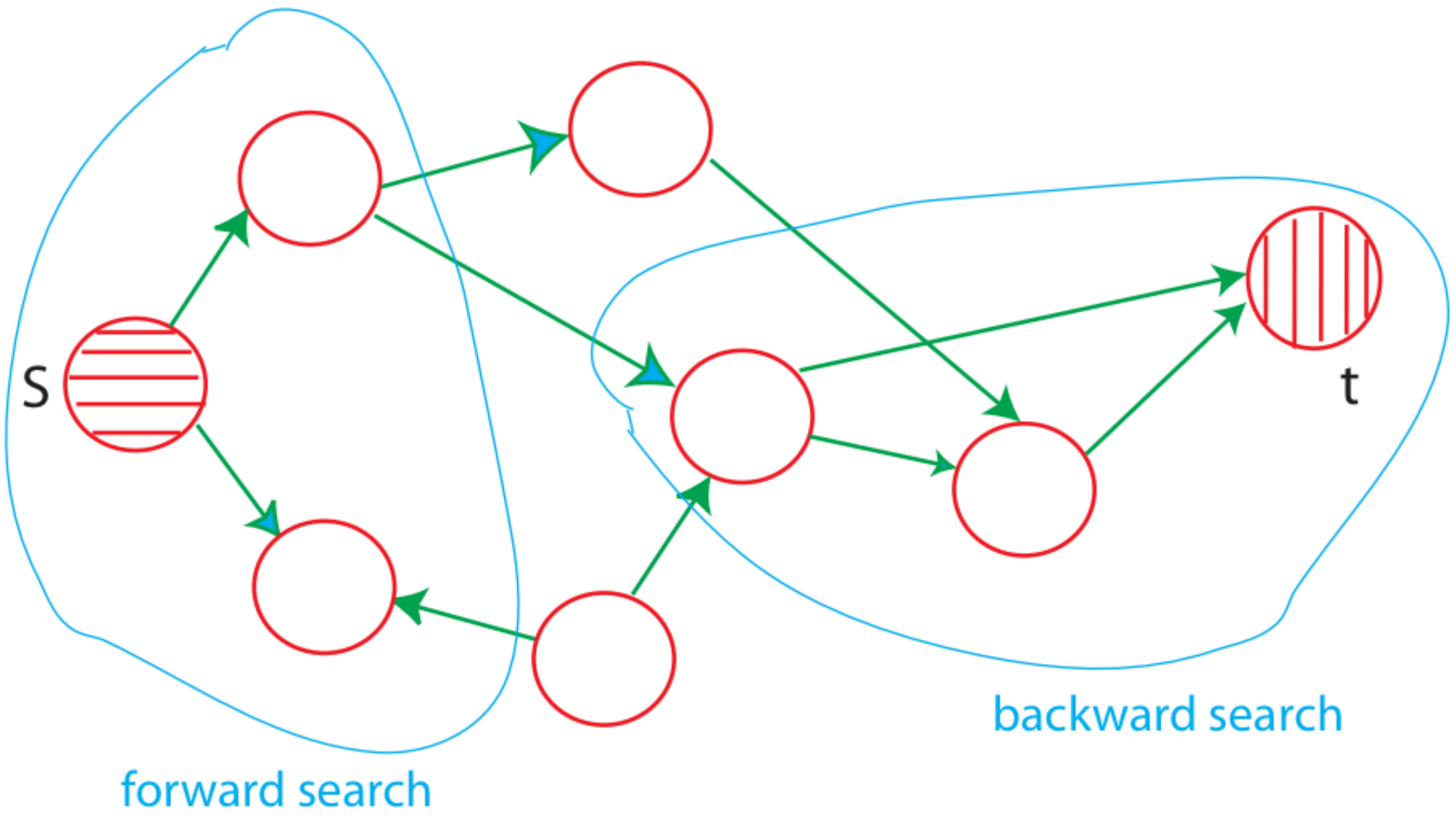
  do $u \leftarrow$ EXTRACT_MIN(Q) (stop if $u = t$!)

  for each vertex $v \in$ Adj$[u]$

    do RELAX(u, v, w)

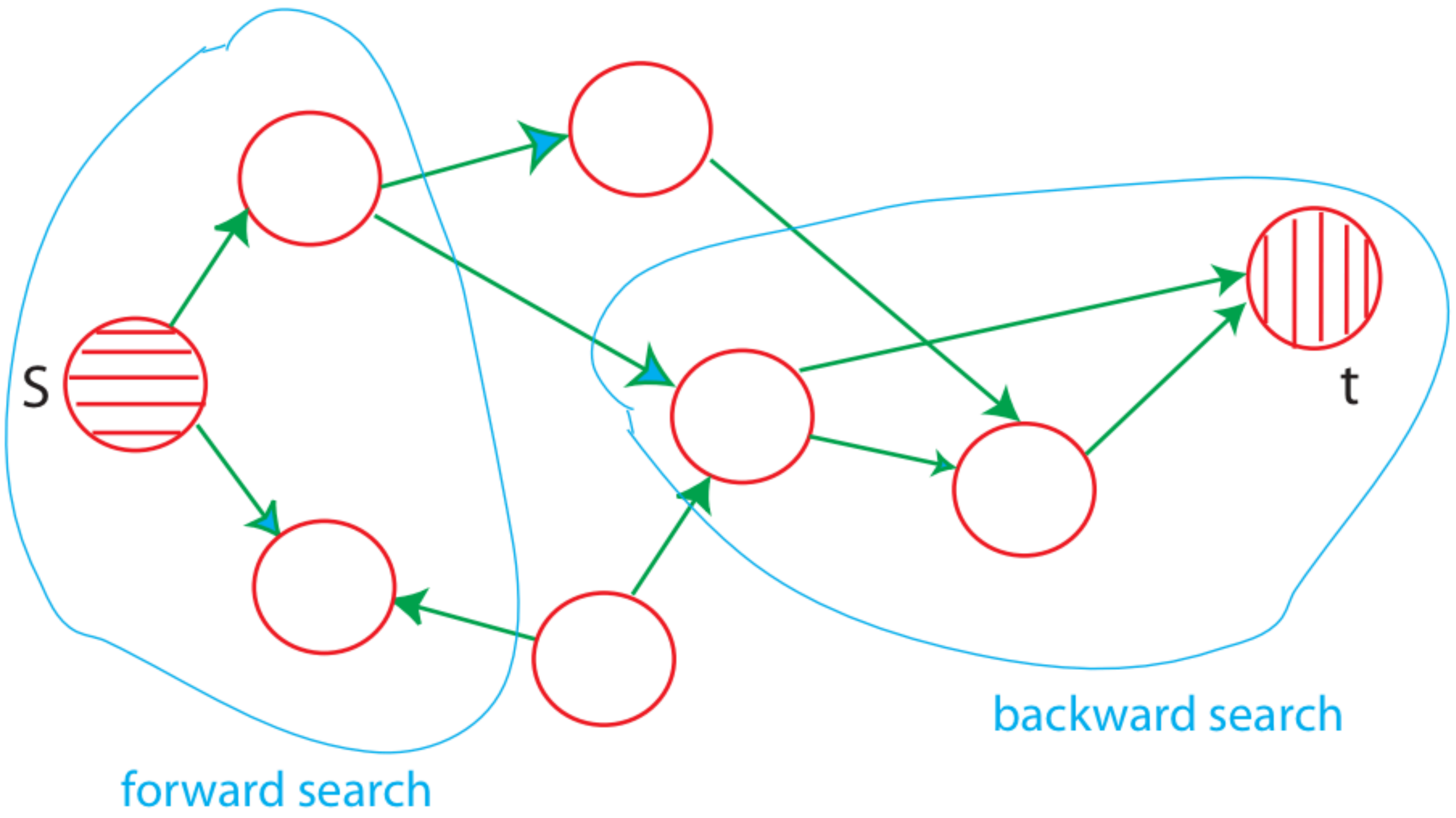
Observation: If only shortest path from s to t is required, stop when t is removed from Q, i.e., when u = t

Bi-Directional Search: Idea



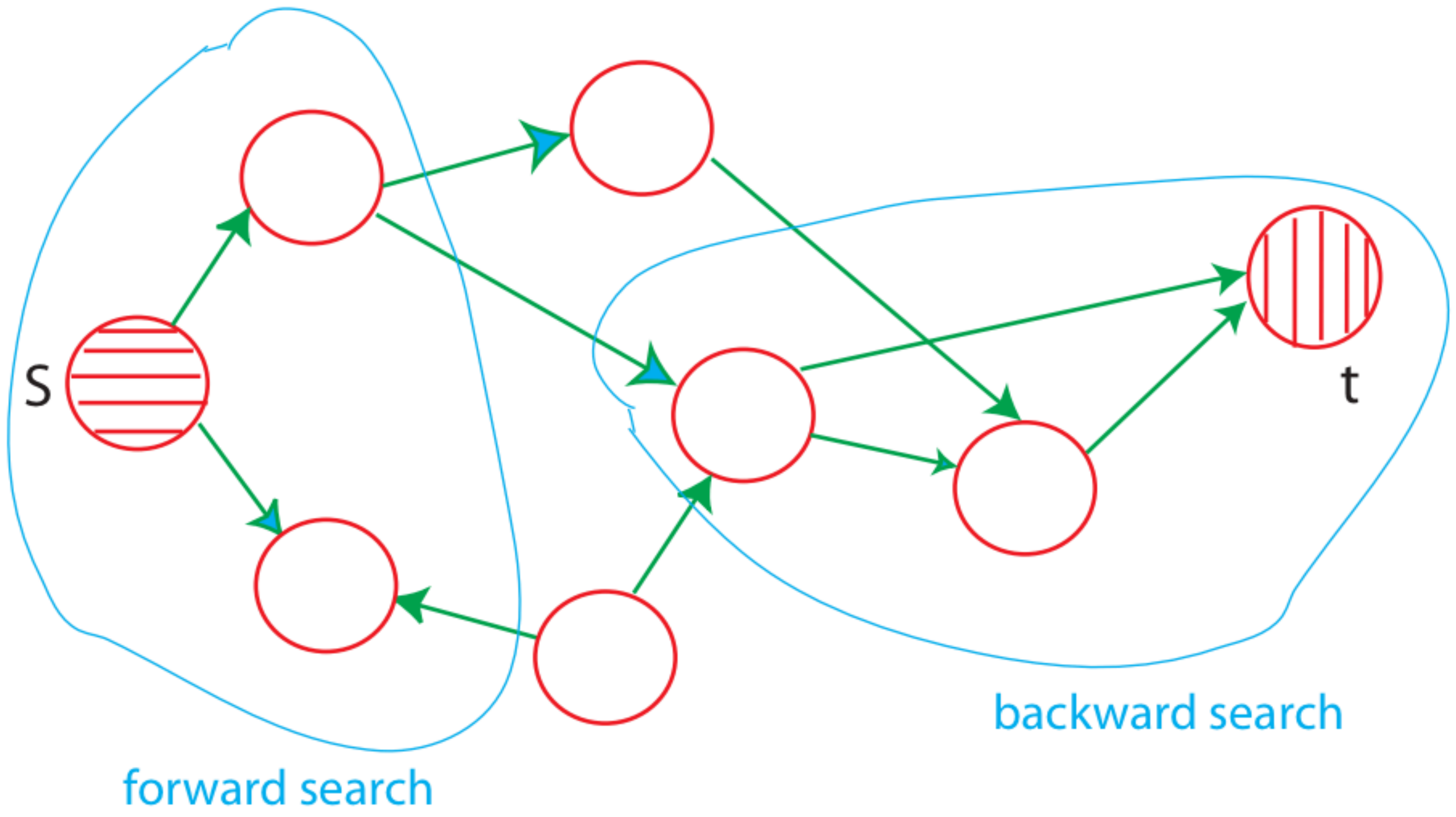
Note: Speedup techniques covered here do not change worst-case behaviour, but reduce the number of visited vertices in practice.

Bi-Directional Search: Idea

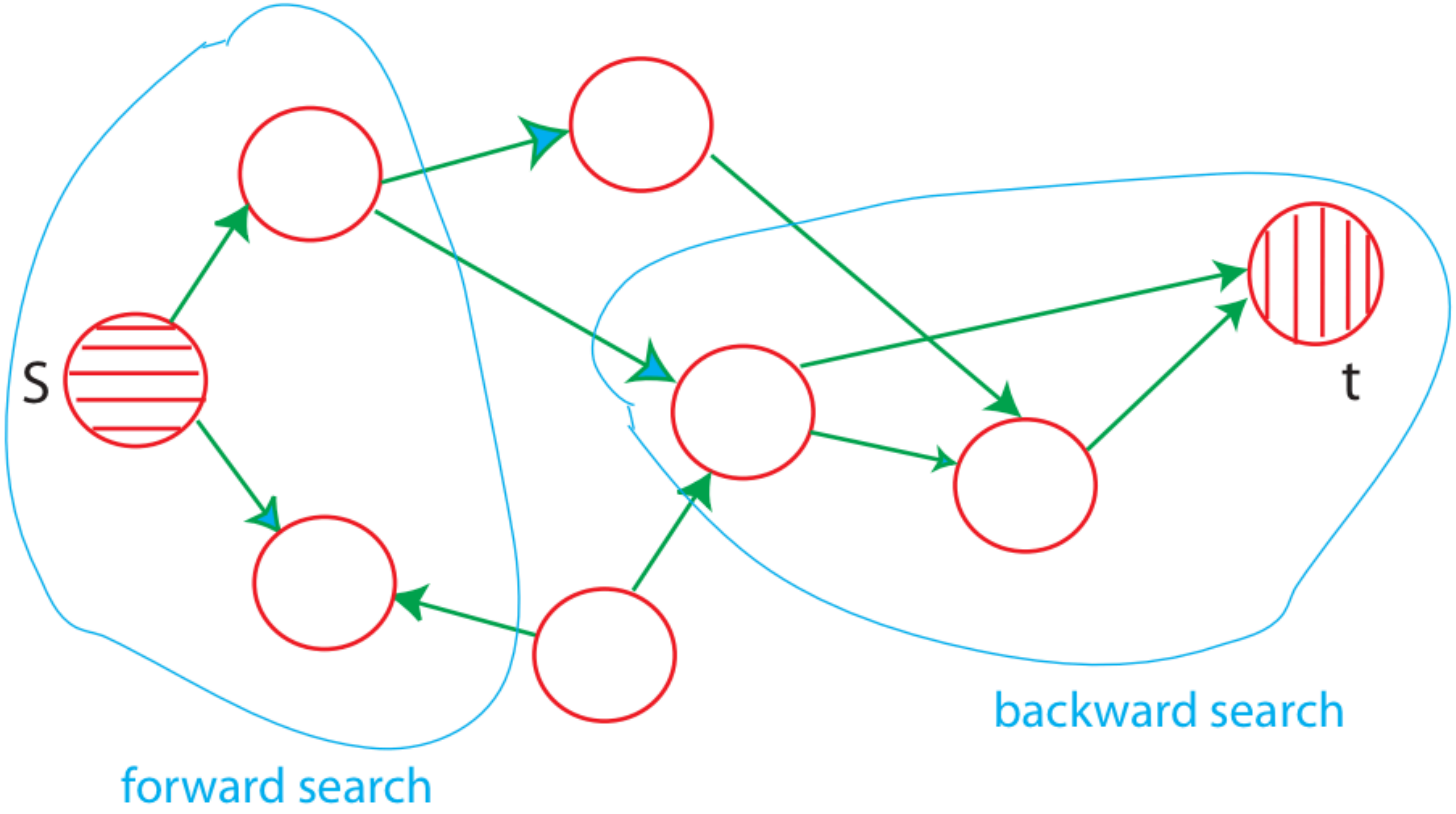


- Alternate forward search from s
- backward search from t (follow edges backward)
- $d_f(u)$ distances for forward search
- $d_b(u)$ distances for backward search

When should the algorithm stop?



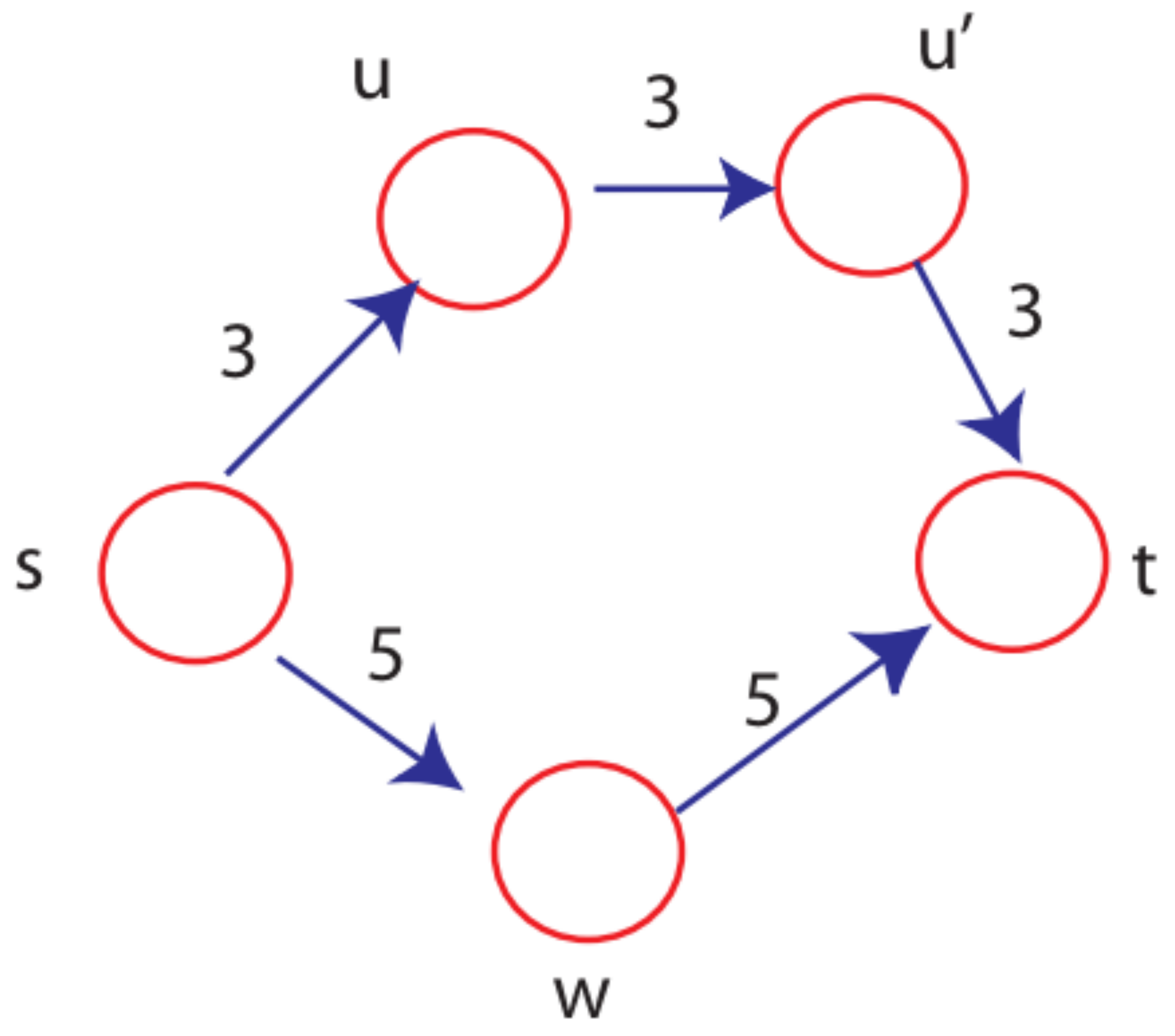
When should the algorithm stop?



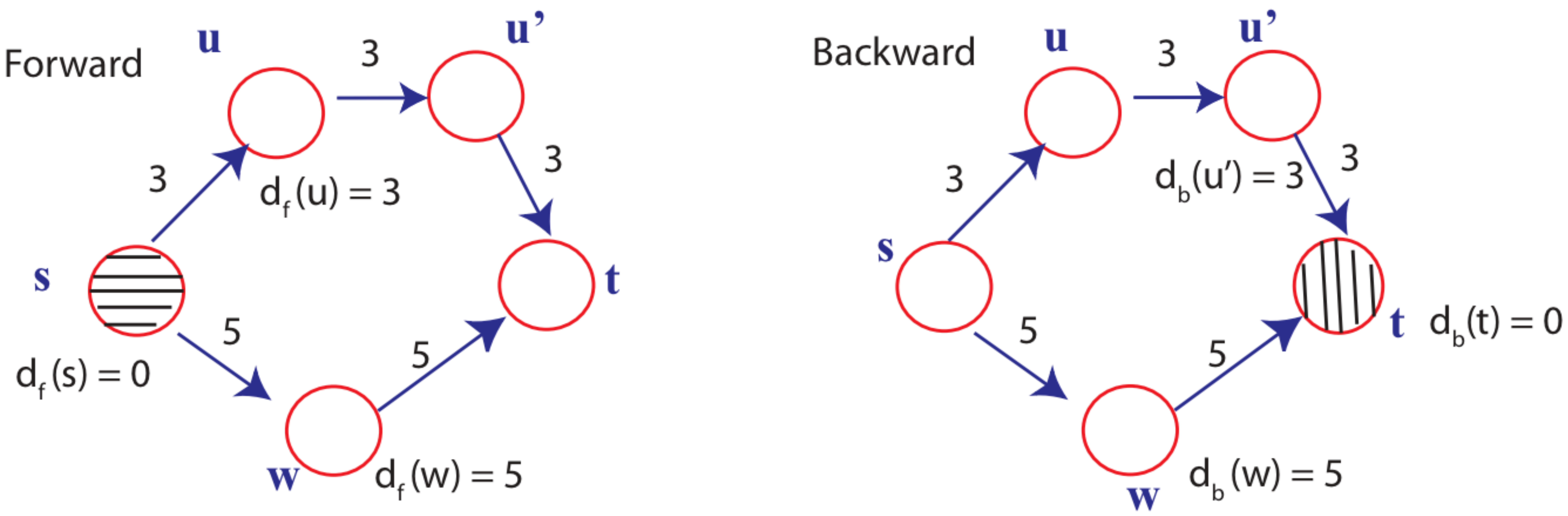
Algorithm terminates when some vertex ***w*** has been processed, i.e., deleted from the queue of both searches, Qf and Qb

Note: the shortest path may not include ***w*** !

Example

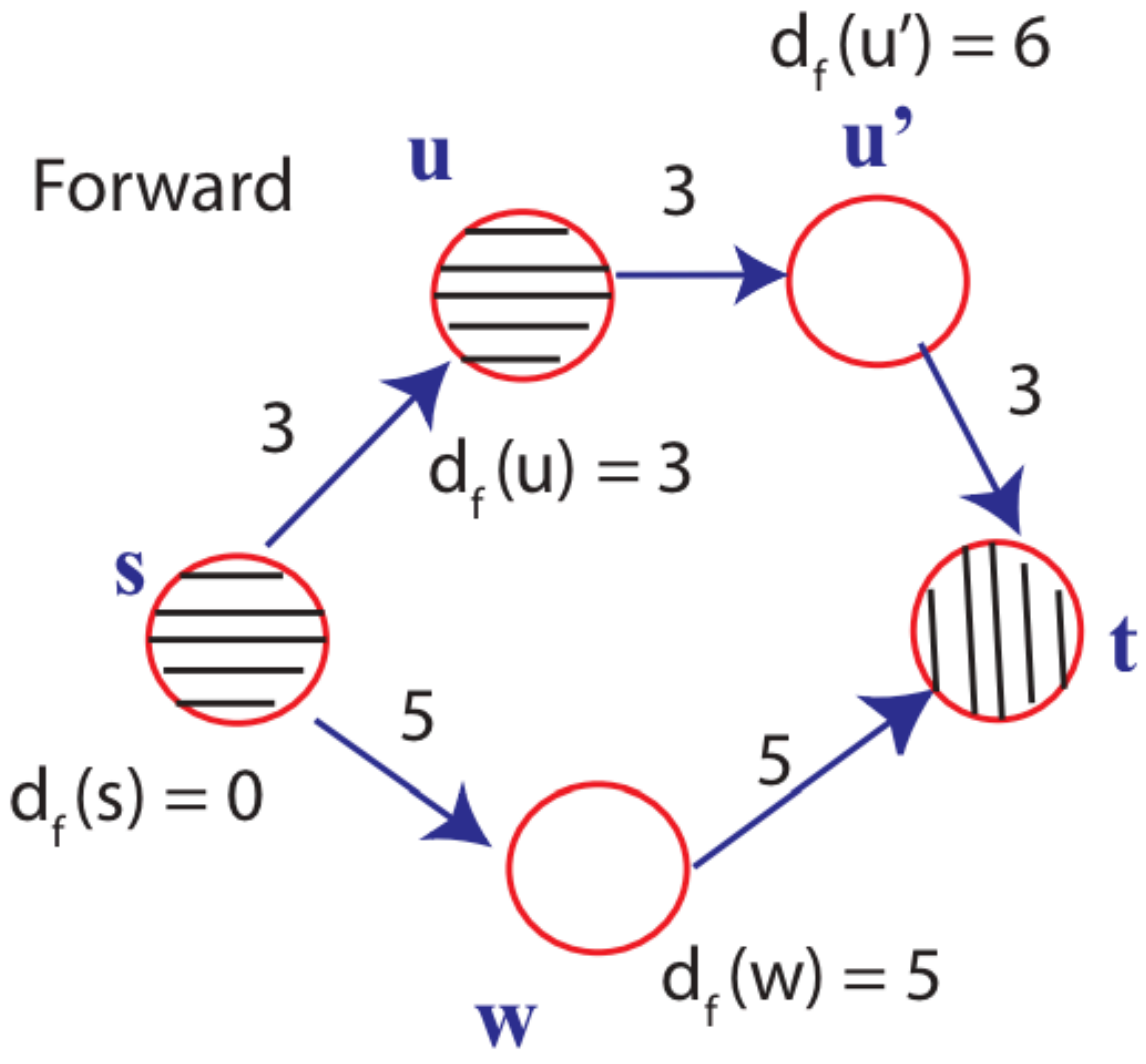


Example

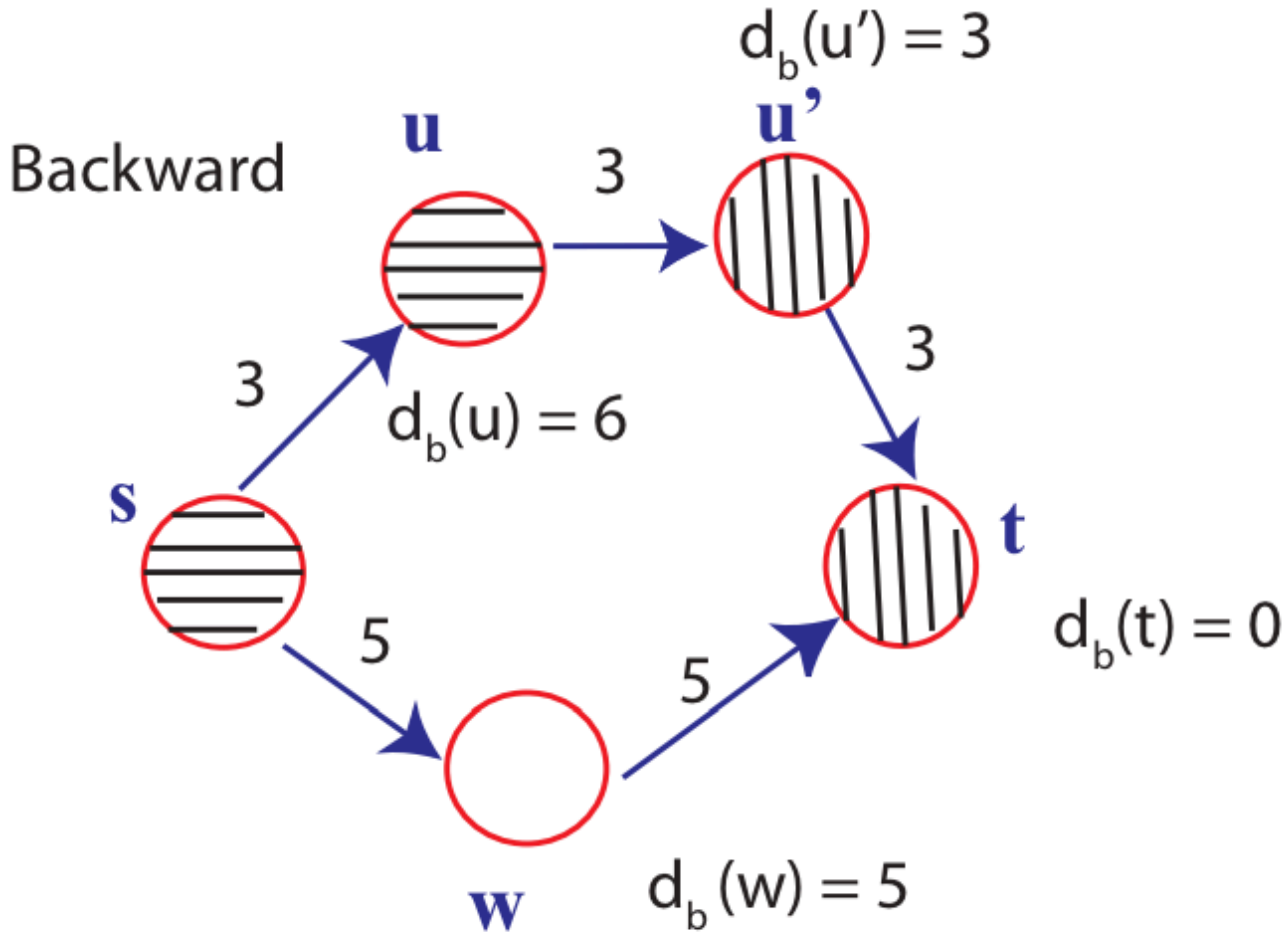


Forward
u
u'
3
3
3
5
5
s
t
w
d_f(u) = 3
d_f(s) = 0
d_f(w) = 5
Backward
u
u'
3
3
3
5
5
s
t
w
d_b(u') = 3
d_b(t) = 0
d_b(w) = 5

Example

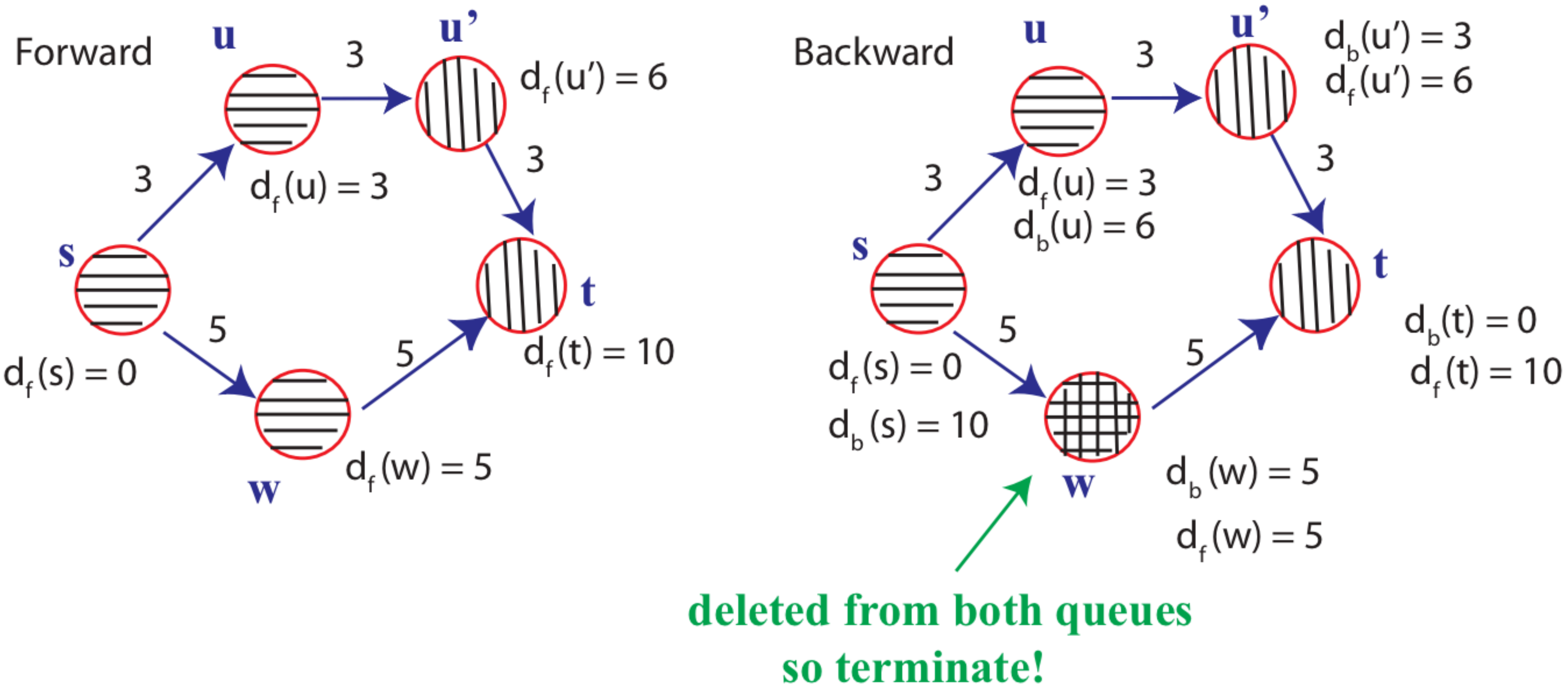


Forward
$d_f(u') = 6$
u
u'
3
3
3
$d_f(u) = 3$
s
t
5
5
$d_f(s) = 0$
w
$d_f(w) = 5$

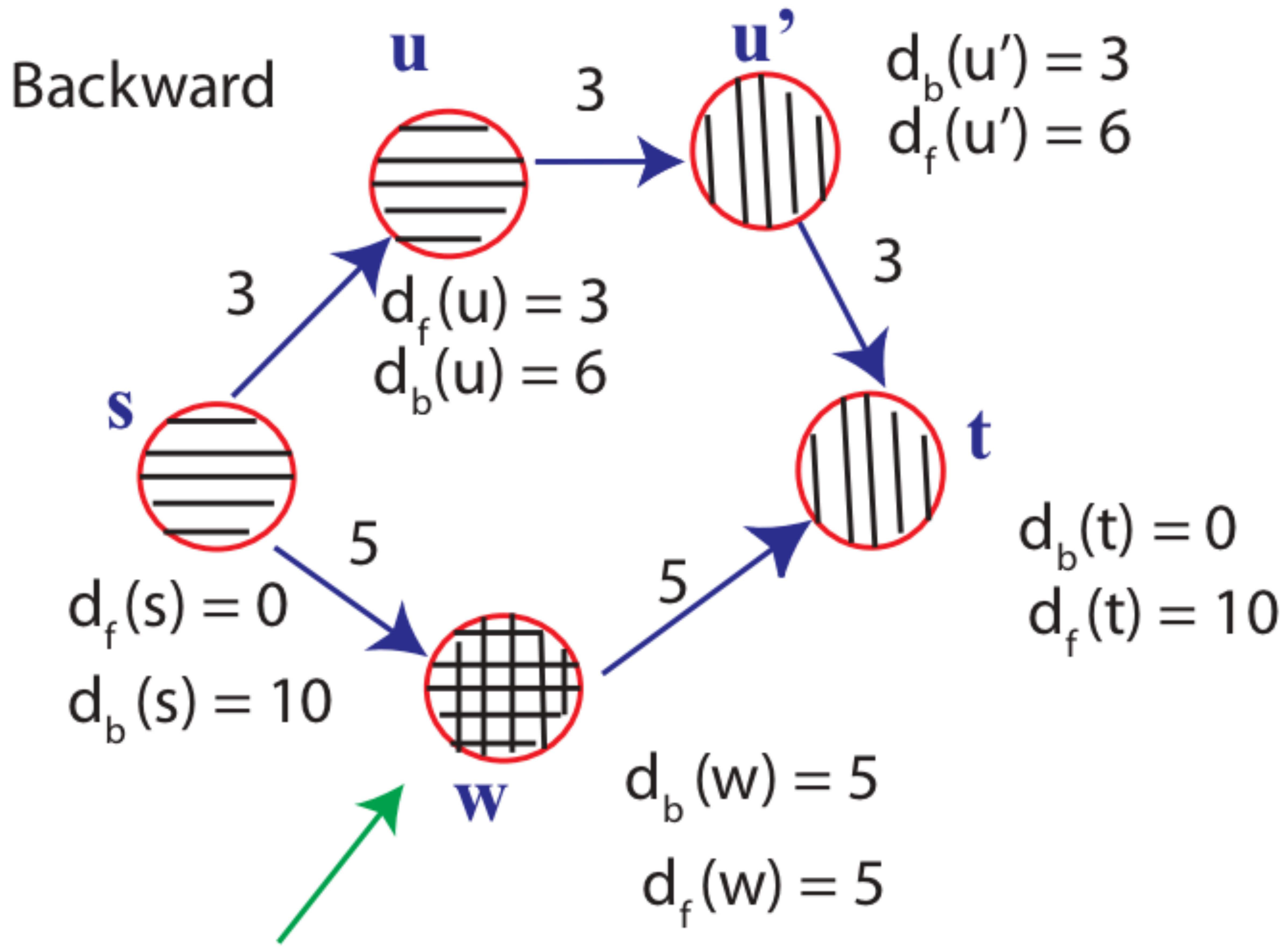


Backward
$d_b(u') = 3$
u
u'
3
3
3
$d_b(u) = 6$
s
t
5
5
$d_b(t) = 0$
w
$d_b(w) = 5$

Example


Forward
u
u'
s
t
w
3
3
3
5
5
$d_f(s) = 0$
$d_f(u) = 3$
$d_f(u') = 6$
$d_f(w) = 5$
$d_f(t) = 10$
Backward
u
u'
s
t
w
3
3
3
5
5
$d_b(u') = 3$
$d_f(u') = 6$
$d_f(u) = 3$
$d_b(u) = 6$
$d_b(t) = 0$
$d_f(t) = 10$
$d_f(s) = 0$
$d_b(s) = 10$
$d_b(w) = 5$
$d_f(w) = 5$
deleted from both queues
so terminate!

Example



$$d_f(u) + d_b(u) = 3 + 6 = 9$$

$$d_f(u') + d_b(u') = 6 + 3 = 9$$

$$d_f(w) + d_b(w) = 5 + 5 = 10$$

Bi-Directional Search: Idea

- Alternate forward search from s
- backward search from t (follow edges backward)
- $d_f(u)$ distances for forward search
- $d_b(u)$ distances for backward search
- Find minimum value for $d_f(x) + d_b(x)$ over all vertices that have been processed in at least one search!