Introduction to Algorithms and Data Structures

Greedy Algorithms: Dijkstra's algorithm for shortest paths

Going to the EICC

What is the fastest way to go from the School of Informatics to the EICC?

Shortest Paths in Graphs

Shortest Paths in Graphs

- **Input:** A directed graph $G = (V, E)$, and a designated node s in V. We also assume that every node u in V is reachable from s. We are also given a length ℓ_e for every edge e in E.

Shortest Paths in Graphs

- **Input:** A directed graph $G = (V, E)$, and a designated node s in V. We also assume that every node u in V is reachable from s. We are also given a length ℓ_e for every edge e in E.

- **Output:** For every node u in V, a shortest path s~u from s to u.

Shortest Paths in Graphs

- **Input:** A directed graph $G = (V, E)$, and a designated node s in V. We also assume that every node u in V is reachable from s. We are also given a length ℓ_e for every edge e in E.

- **Output:** For every node u in V, a shortest path s~u from s to u.

    - To be more precise, a list of paths:

Shortest Paths in Graphs

- **Input:** A directed graph $G = (V, E)$, and a designated node s in V. We also assume that every node u in V is reachable from s. We are also given a length ℓ_e for every edge e in E.

- **Output:** For every node u in V, a shortest path $s \sim u$ from s to u.

  - To be more precise, a list of paths:

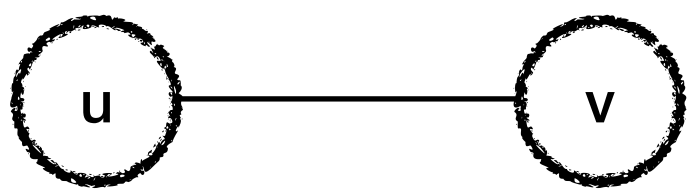
$P(s)$	$P(u_1)$	$P(u_2)$	$P(u_3)$	...

What about undirected graphs?

- **Input:** A directed graph $G = (V, E)$, …

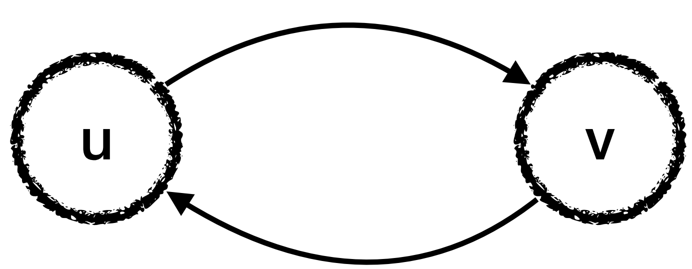
What about undirected graphs?

- **Input:** A directed graph $G = (V, E)$, …

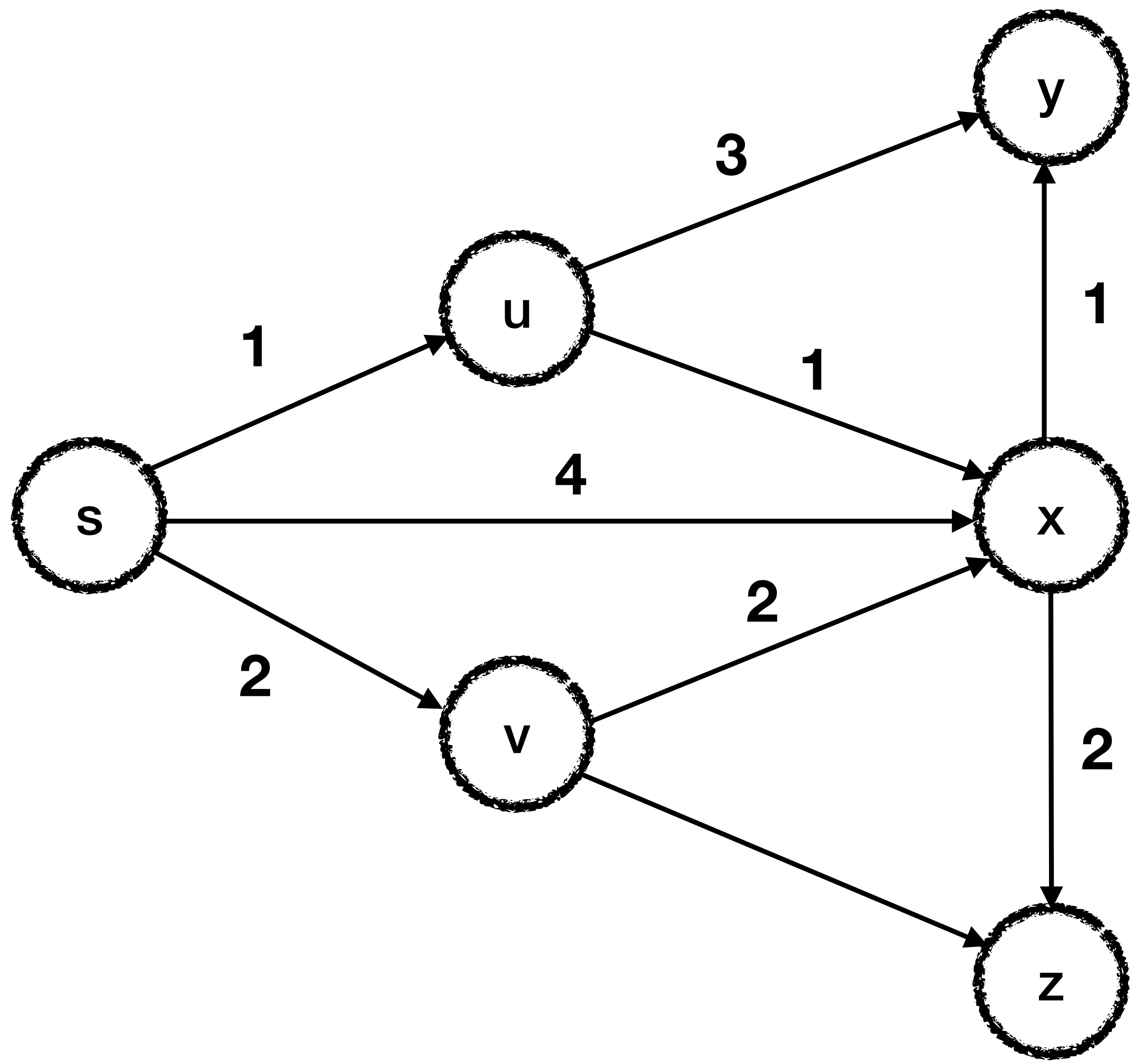


What about undirected graphs?

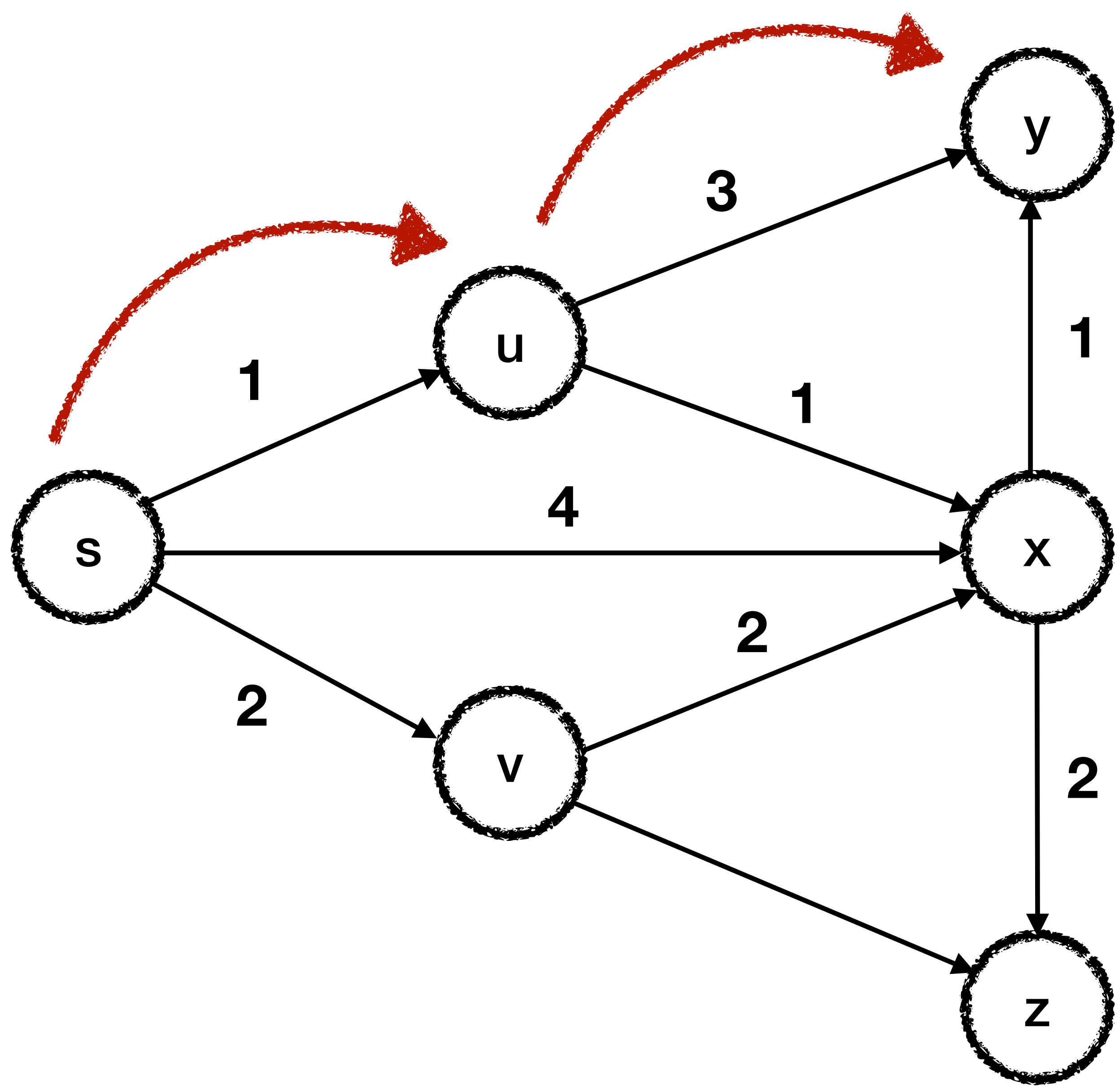
- **Input:** A directed graph $G = (V, E)$, …



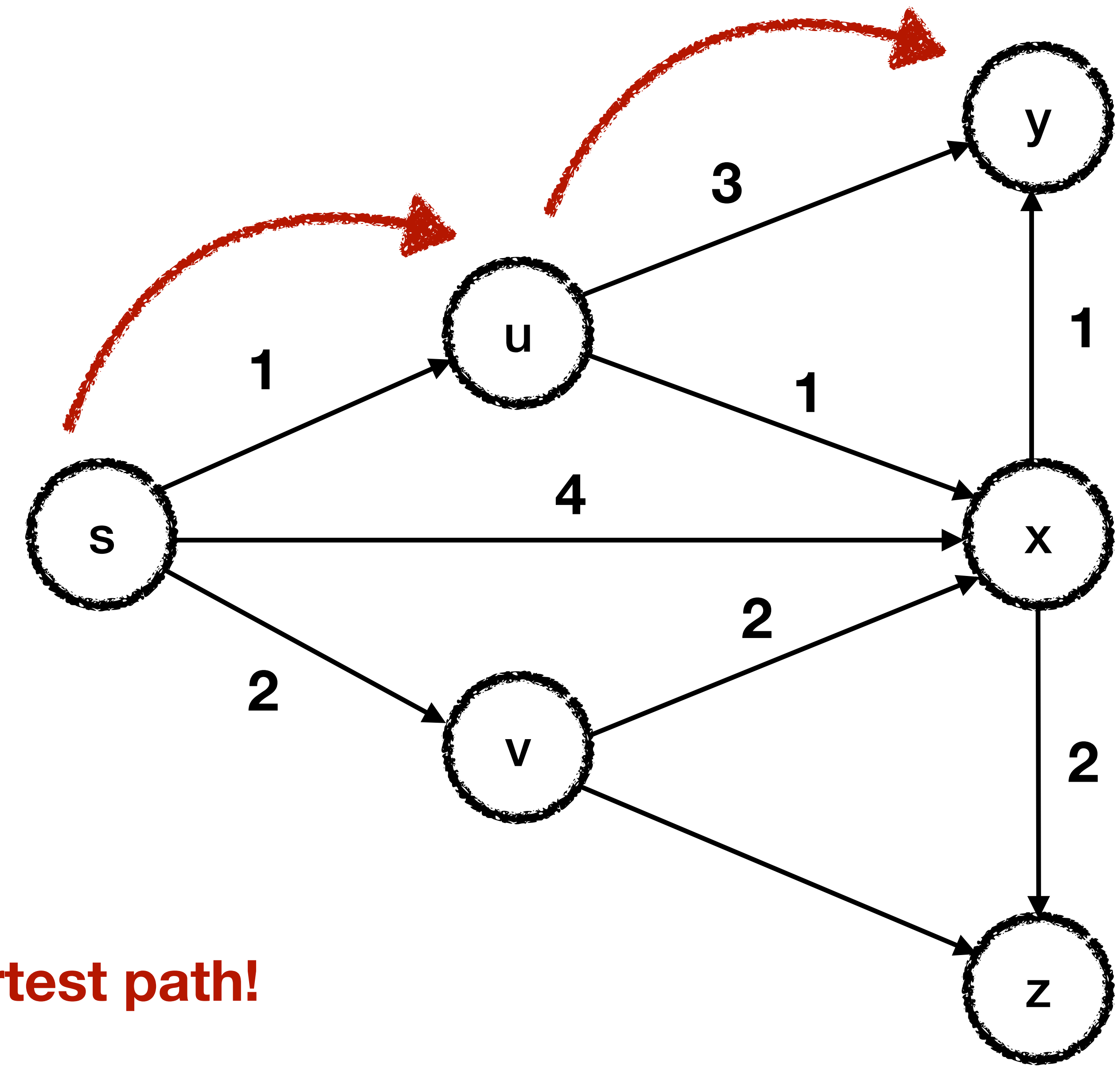
Running Example (KT Figure 5.7)



Running Example (KT Figure 5.7)

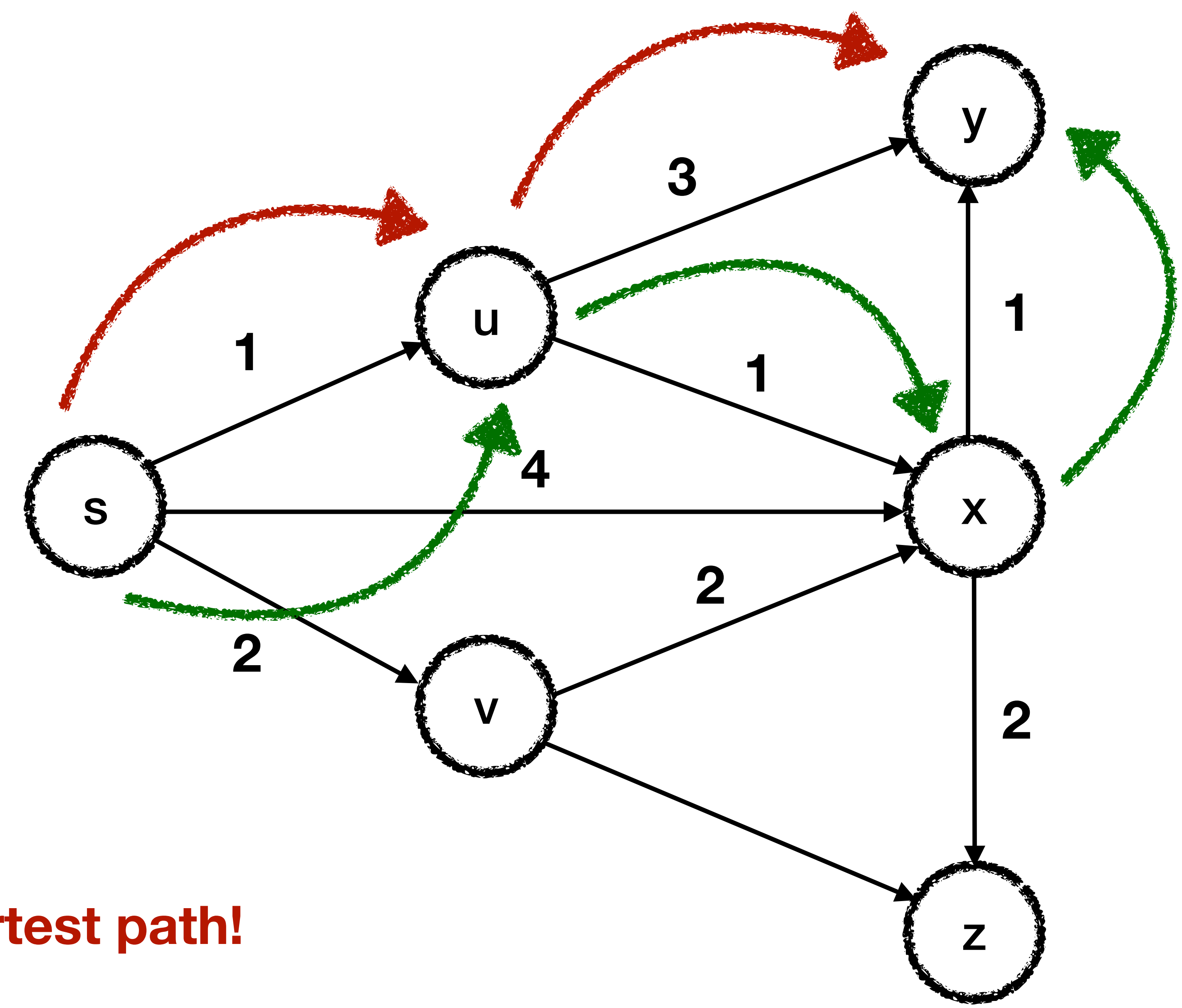


Running Example (KT Figure 5.7)



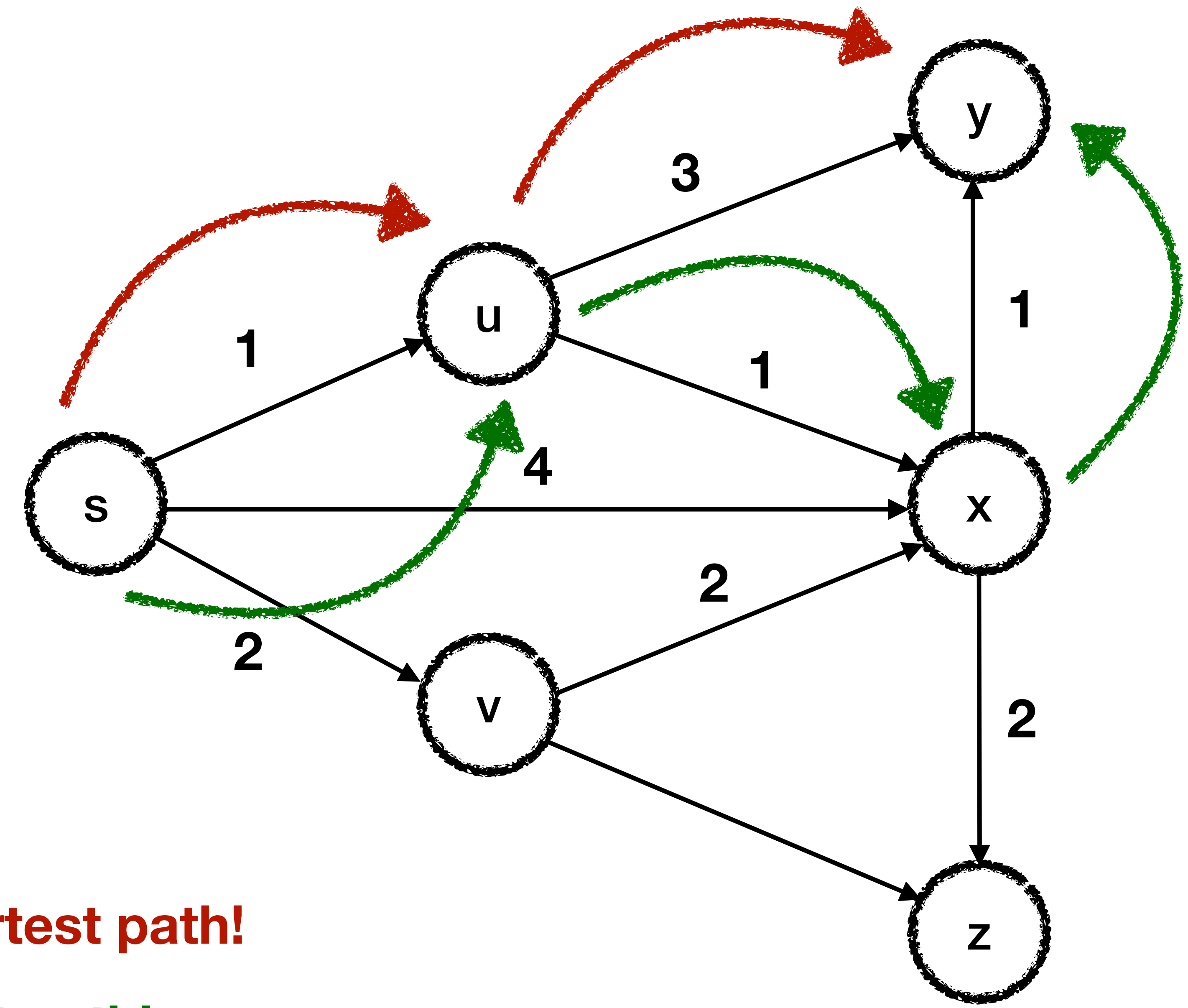
not a shortest path!

Running Example (KT Figure 5.7)



not a shortest path!

Running Example (KT Figure 5.7)



not a shortest path!

a shortest path!

Dijkstra’s Algorithm

Dijkstra’s Algorithm

- Proposed by Edsger Dijkstra in 1959.

Dijkstra’s Algorithm

- Proposed by Edsger Dijkstra in 1959.

- **Idea:** Maintain a set S of nodes u for which we have found the shortest path distance $d(u)$ from s.

Dijkstra's Algorithm

- Proposed by Edsger Dijkstra in 1959.
- **Idea:** Maintain a set S of nodes u for which we have found the shortest path distance $d(u)$ from s.
    - We may refer to S as the *explored* part of the graph.

Dijkstra's Algorithm

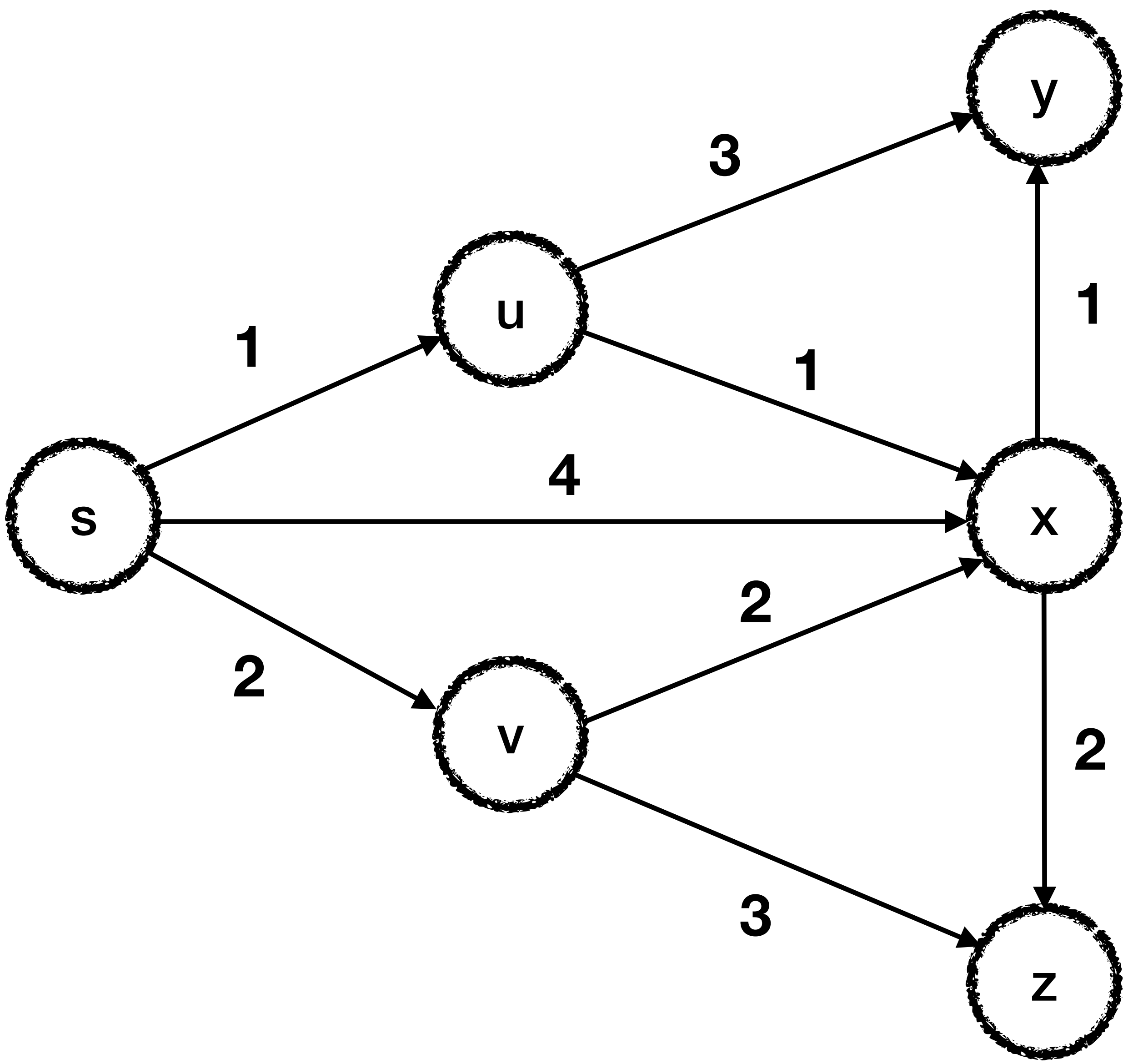
- Proposed by Edsger Dijkstra in 1959.

- **Idea:** Maintain a set S of nodes u for which we have found the shortest path distance $d(u)$ from s.

    - We may refer to S as the *explored* part of the graph.

    - Initially, $S = \{s\}$ and $d(s) = 0$.

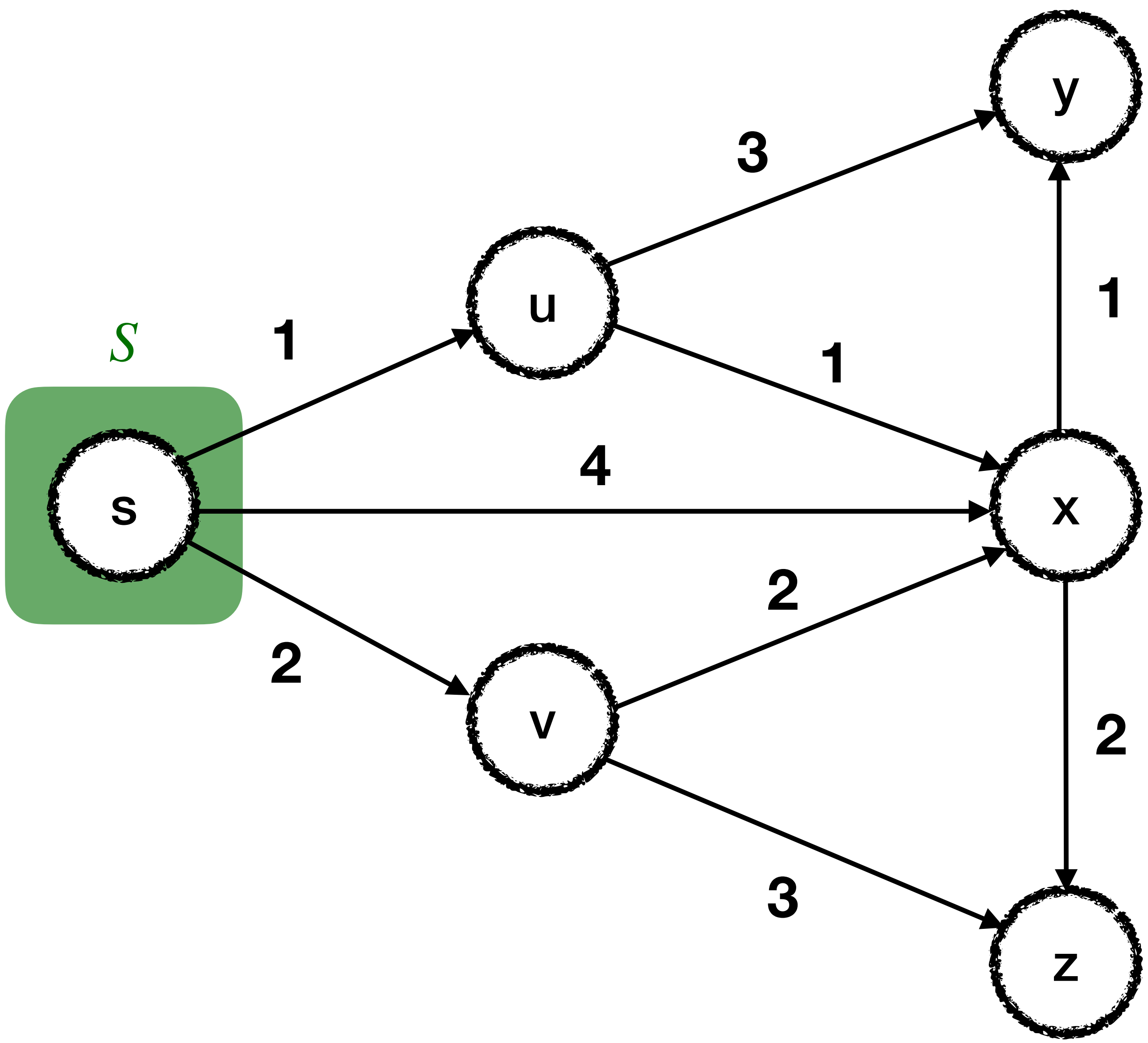
Running Example (KT Figure 5.7)

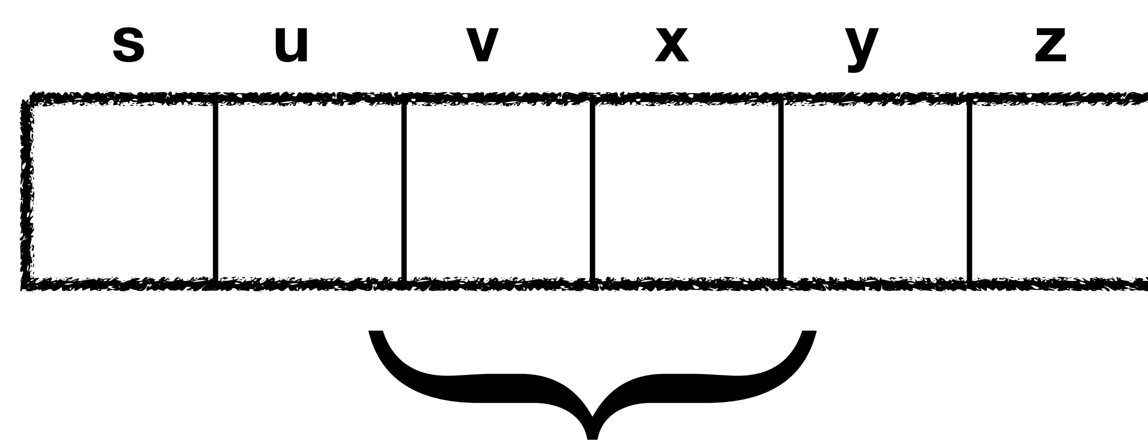


s	u	v	x	y	z

shortest path distances from s

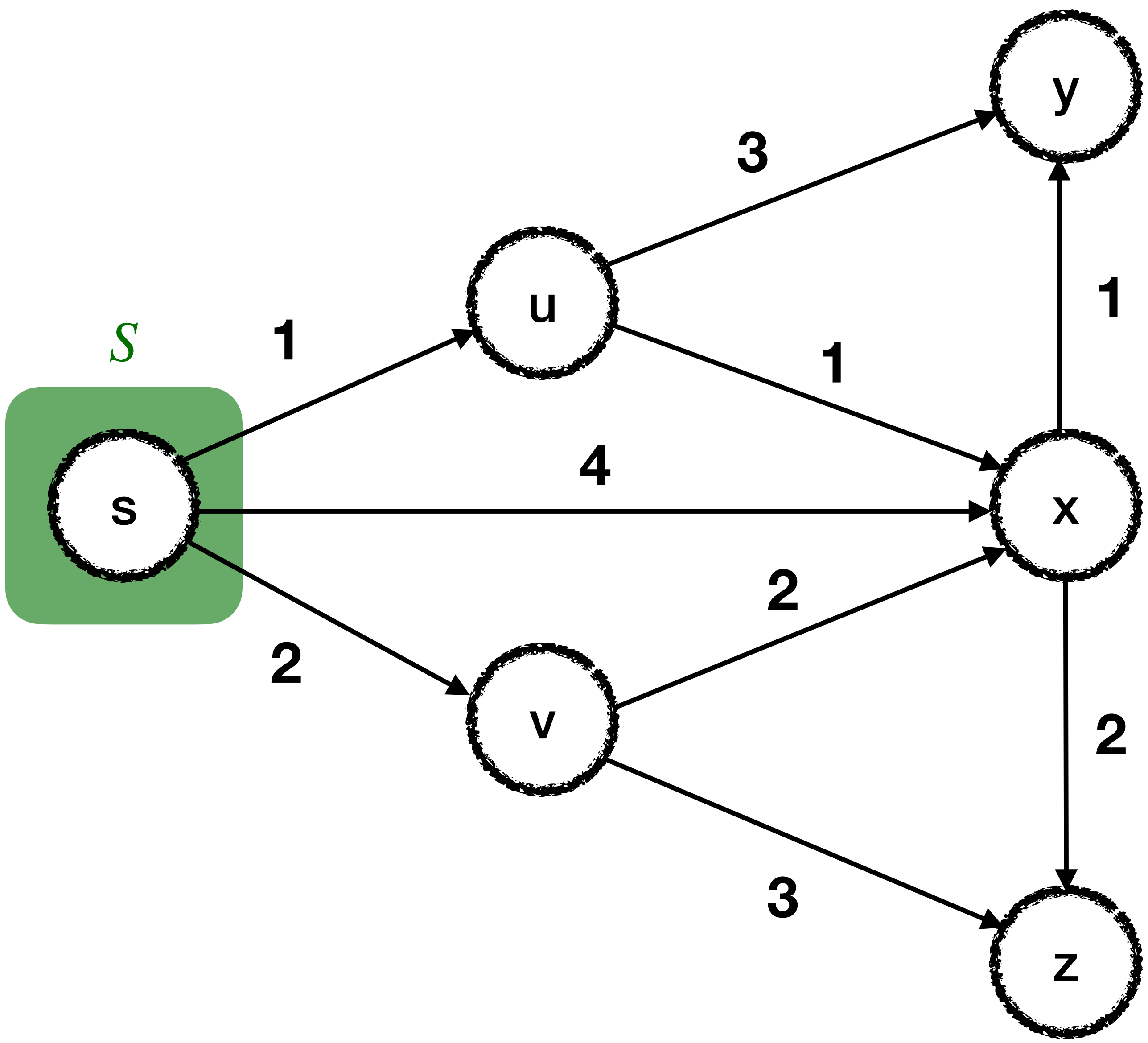
Running Example (KT Figure 5.7)





shortest path distances from s

Running Example (KT Figure 5.7)



s	u	v	x	y	z
0					

$\underbrace{\qquad\qquad\qquad}_{\text{shortest path distances from } s}$

Dijkstra's Algorithm

- Proposed by Edsger Dijkstra in 1959.
- **Idea:** Maintain a set S of nodes u for which we have found the shortest path distance $d(u)$ from s.
  - We may refer to S as the *explored* part of the graph.
  - Initially, $S = \{s\}$ and $d(s) = 0$.

Dijkstra's Algorithm

- Proposed by Edsger Dijkstra in 1959.
- **Idea:** Maintain a set S of nodes u for which we have found the shortest path distance $d(u)$ from s.
  - We may refer to S as the *explored* part of the graph.
  - Initially, $S = \{s\}$ and $d(s) = 0$.
  - For every node $v \in V - S$, we determine the shortest path that can be constructed by traveling along a path $s \sim u$ for $u \in S$, followed by (u, v).

Dijkstra’s Algorithm

- For every node $v \in V - S$, we determine the shortest path that can be constructed by traveling along a path $s \sim u$ for $u \in S$, followed by (u, v).

Dijkstra's Algorithm

- For every node $v \in V - S$, we determine the shortest path that can be constructed by traveling along a path $s \sim u$ for $u \in S$, followed by (u, v).

- In other words, we choose node $v \in V - S$ such that

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

Dijkstra's Algorithm

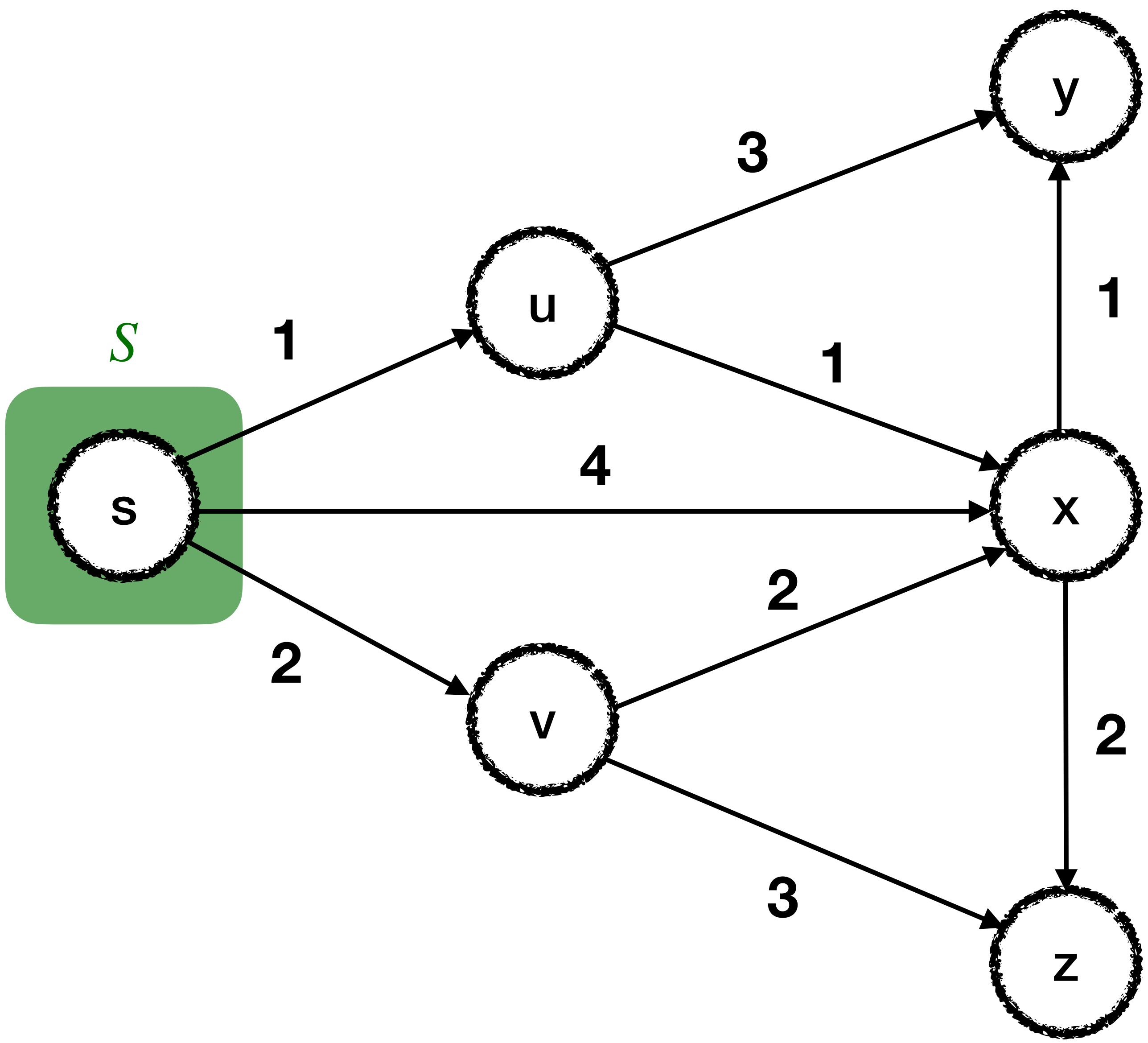
- For every node $v \in V - S$, we determine the shortest path that can be constructed by traveling along a path $s \sim u$ for $u \in S$, followed by (u, v).

- In other words, we choose node $v \in V - S$ such that

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

- Add v to S and define $d(v) = d'(v)$.

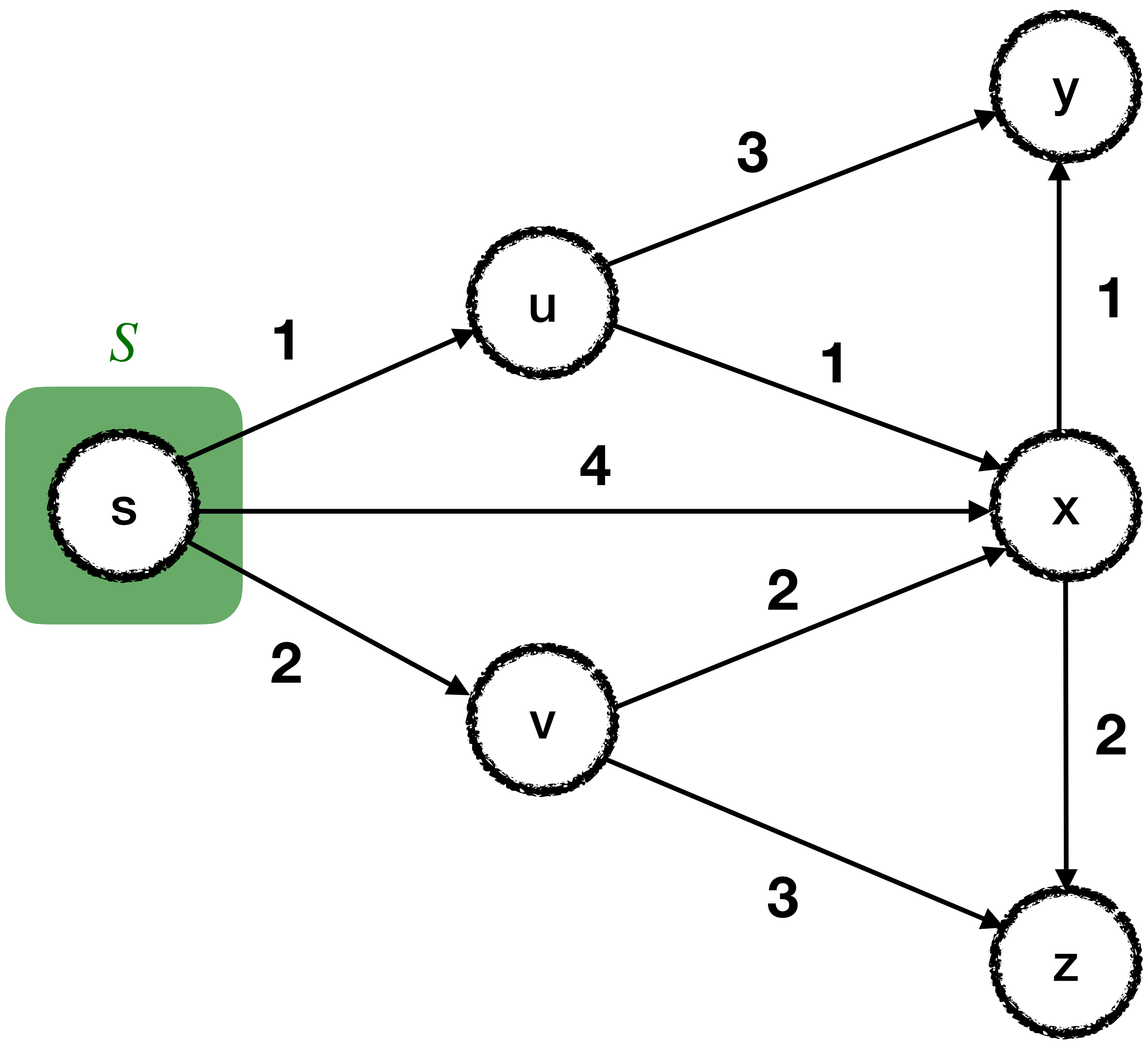
Running Example (KT Figure 5.7)



s	u	v	x	y	z
0					

shortest path distances from s

Running Example (KT Figure 5.7)



s	u	v	x	y	z
0					

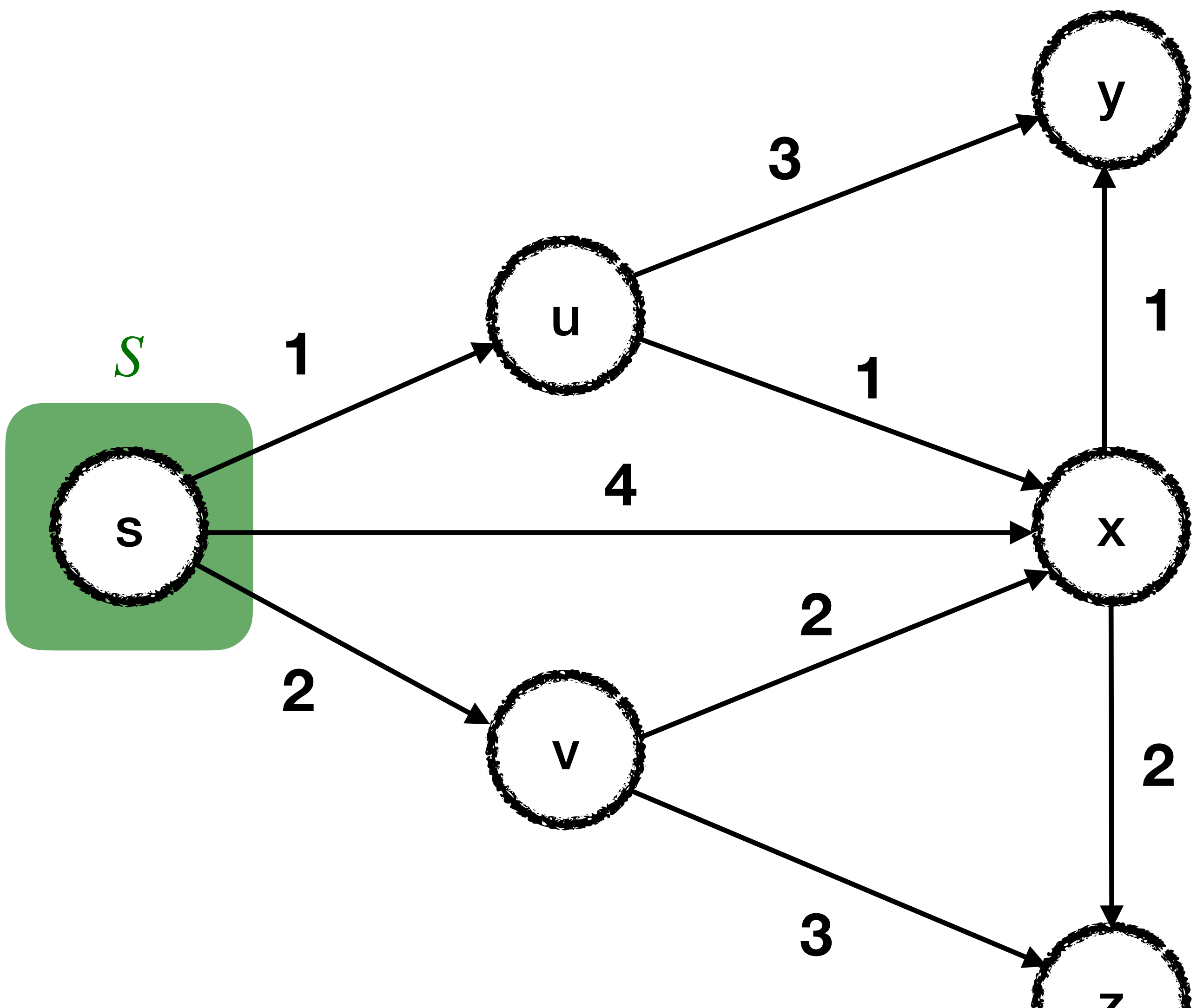
shortest path distances from s

In other words, we choose node $v \in V - S$ such that

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

Add v to S and define $d(v) = d'(v)$.

Running Example (KT Figure 5.7)



Iteration 1

s	u	v	x	y	z
0					

shortest path distances from s

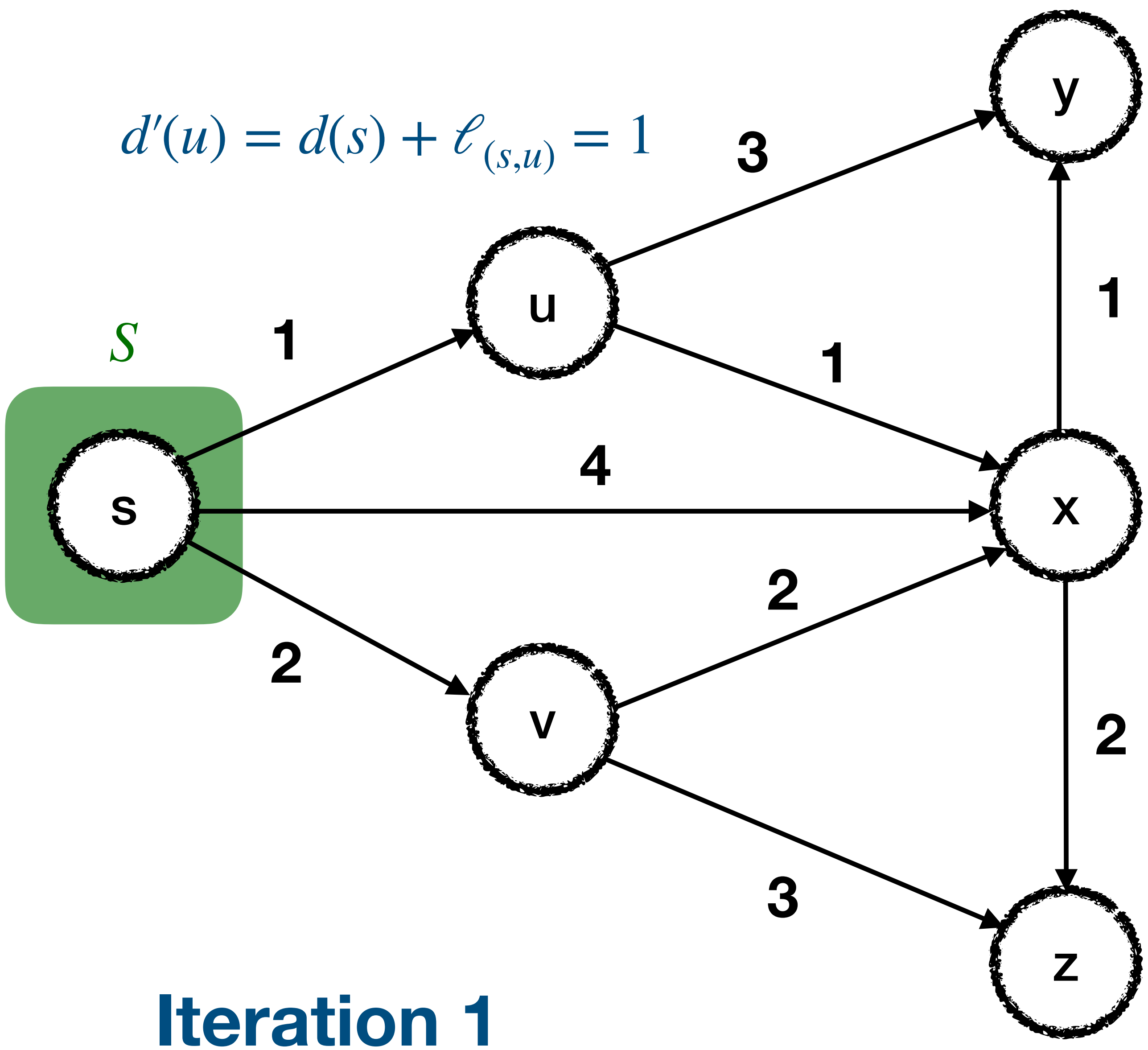
In other words, we choose node $v \in V - S$ such that

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

Add v to S and define $d(v) = d'(v)$.

Running Example (KT Figure 5.7)

$d'(u) = d(s) + \ell_{(s,u)} = 1$



Iteration 1

s	u	v	x	y	z
0					

$\underbrace{\hspace{6em}}$

shortest path distances from s

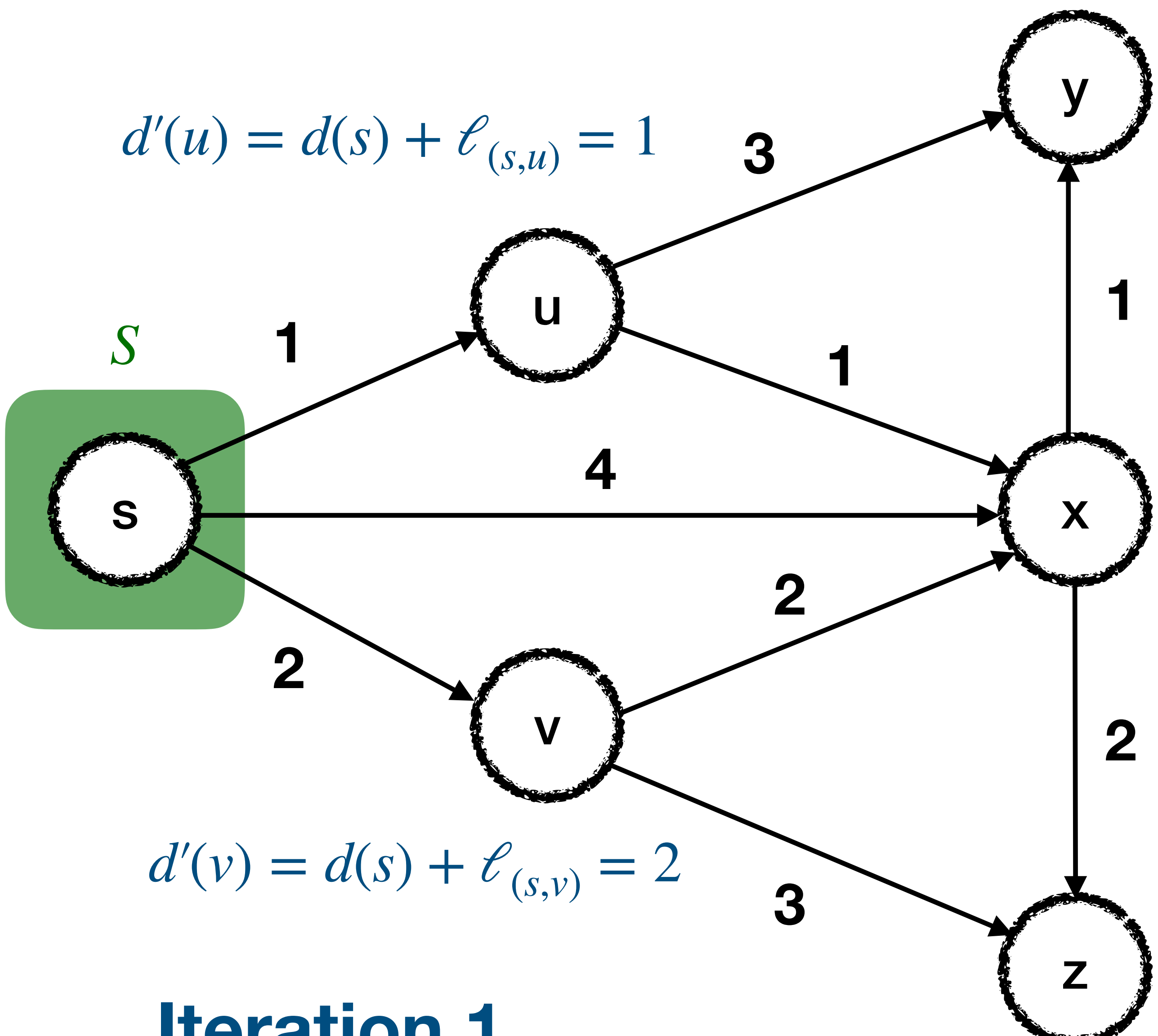
In other words, we choose node $v \in V - S$ such that

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

Add v to S and define $d(v) = d'(v)$.

Running Example (KT Figure 5.7)

$d'(u) = d(s) + \ell_{(s,u)} = 1$



$d'(v) = d(s) + \ell_{(s,v)} = 2$

Iteration 1

s	u	v	x	y	z
0					

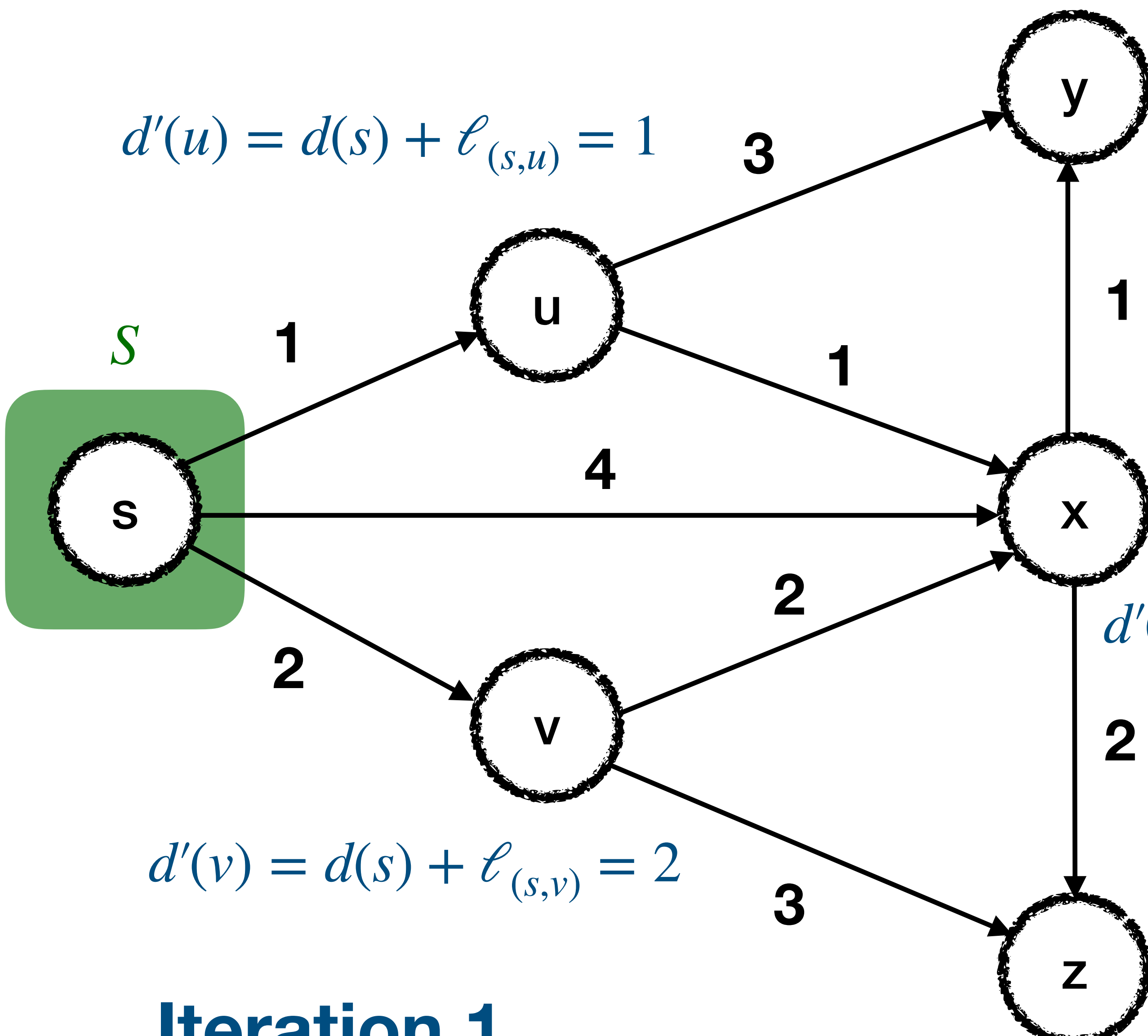
shortest path distances from s

In other words, we choose node $v \in V - S$ such that

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

Add v to S and define $d(v) = d'(v)$.

Running Example (KT Figure 5.7)



$d'(u) = d(s) + \ell_{(s,u)} = 1$

$d'(x) = d(s) + \ell_{(s,x)} = 4$

$d'(v) = d(s) + \ell_{(s,v)} = 2$

Iteration 1

s	u	v	x	y	z
0					

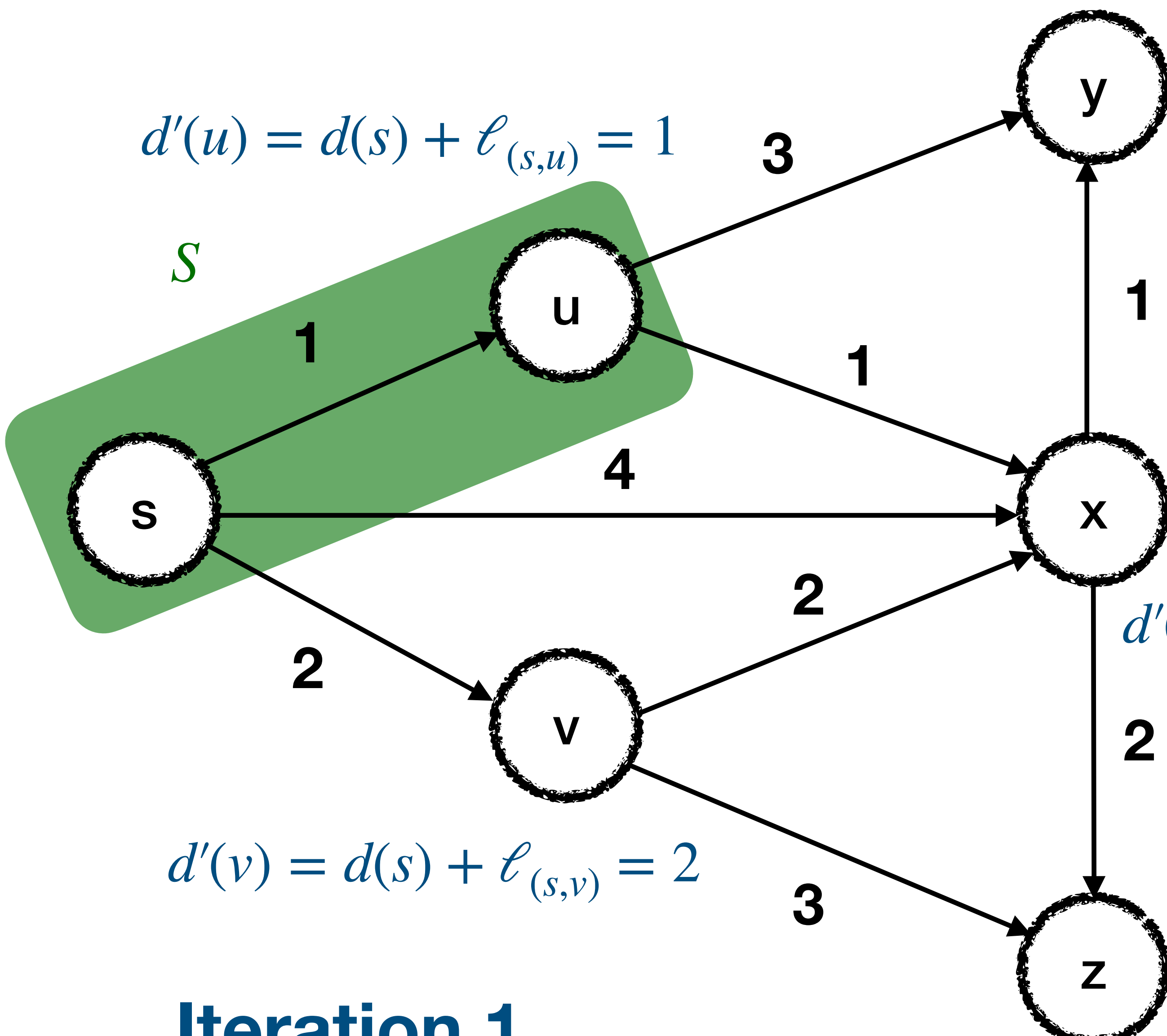
shortest path distances from s

In other words, we choose node $v \in V - S$ such that

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

Add v to S and define $d(v) = d'(v)$.

Running Example (KT Figure 5.7)



s	u	v	x	y	z
0					

shortest path distances from s

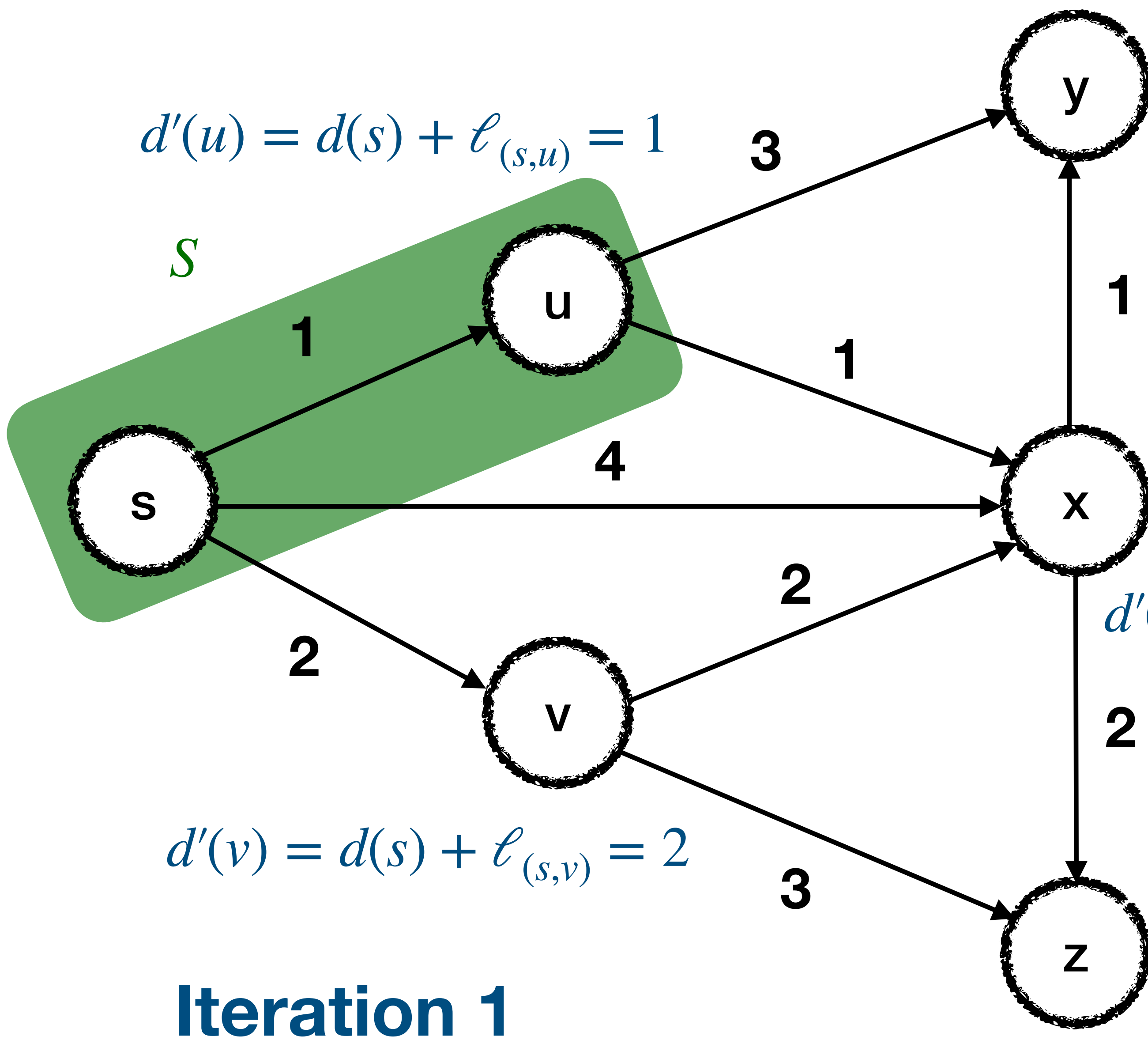
Iteration 1

In other words, we choose node $v \in V - S$ such that

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

Add v to S and define $d(v) = d'(v)$.

Running Example (KT Figure 5.7)



Iteration 1

s	u	v	x	y	z
0	1				

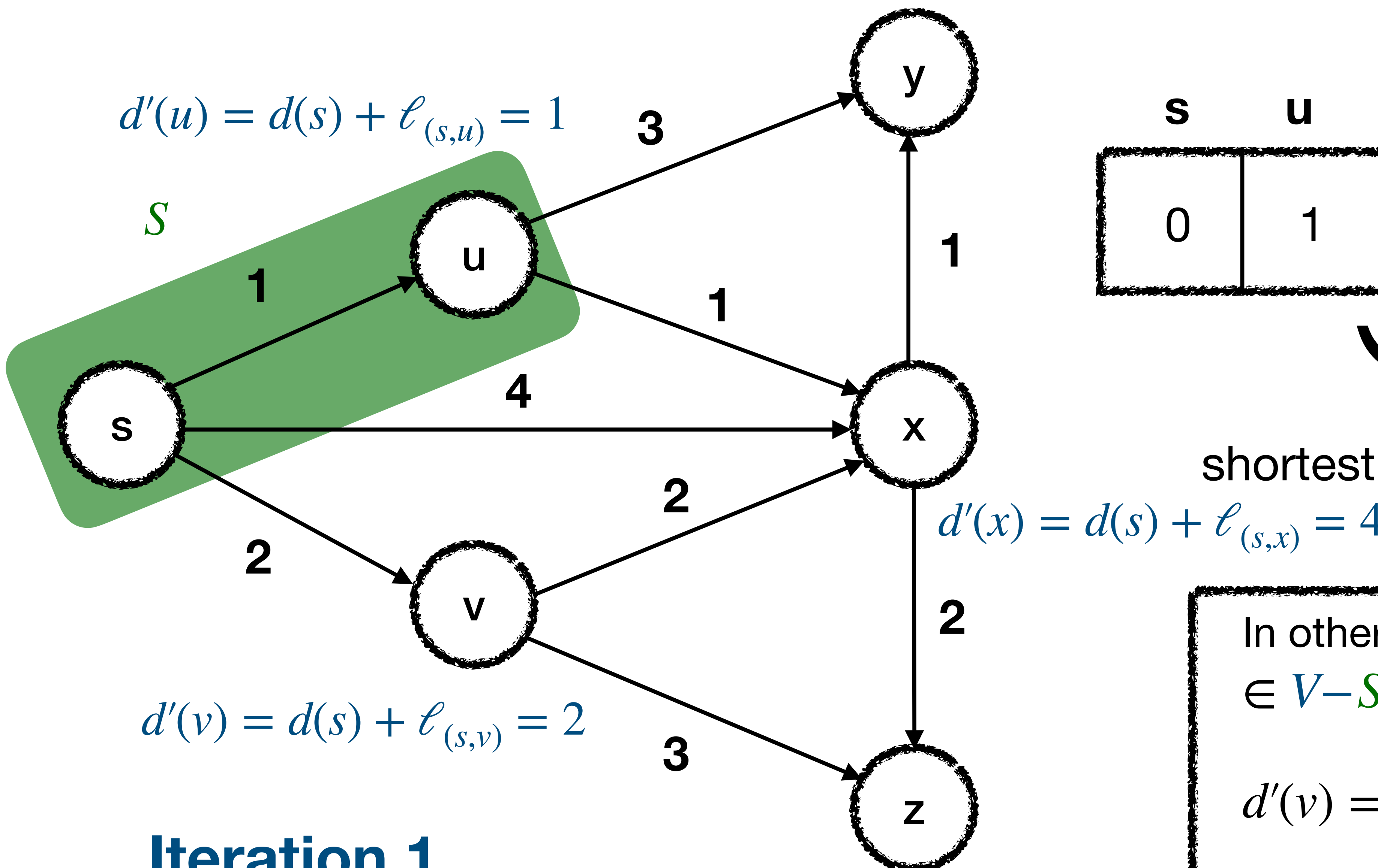
shortest path distances from s

In other words, we choose node $v \in V - S$ such that

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

Add v to S and define $d(v) = d'(v)$.

Running Example (KT Figure 5.7)



s	u	v	x	y	z
0	1				

shortest path distances from s

In other words, we choose node $v \in V - S$ such that

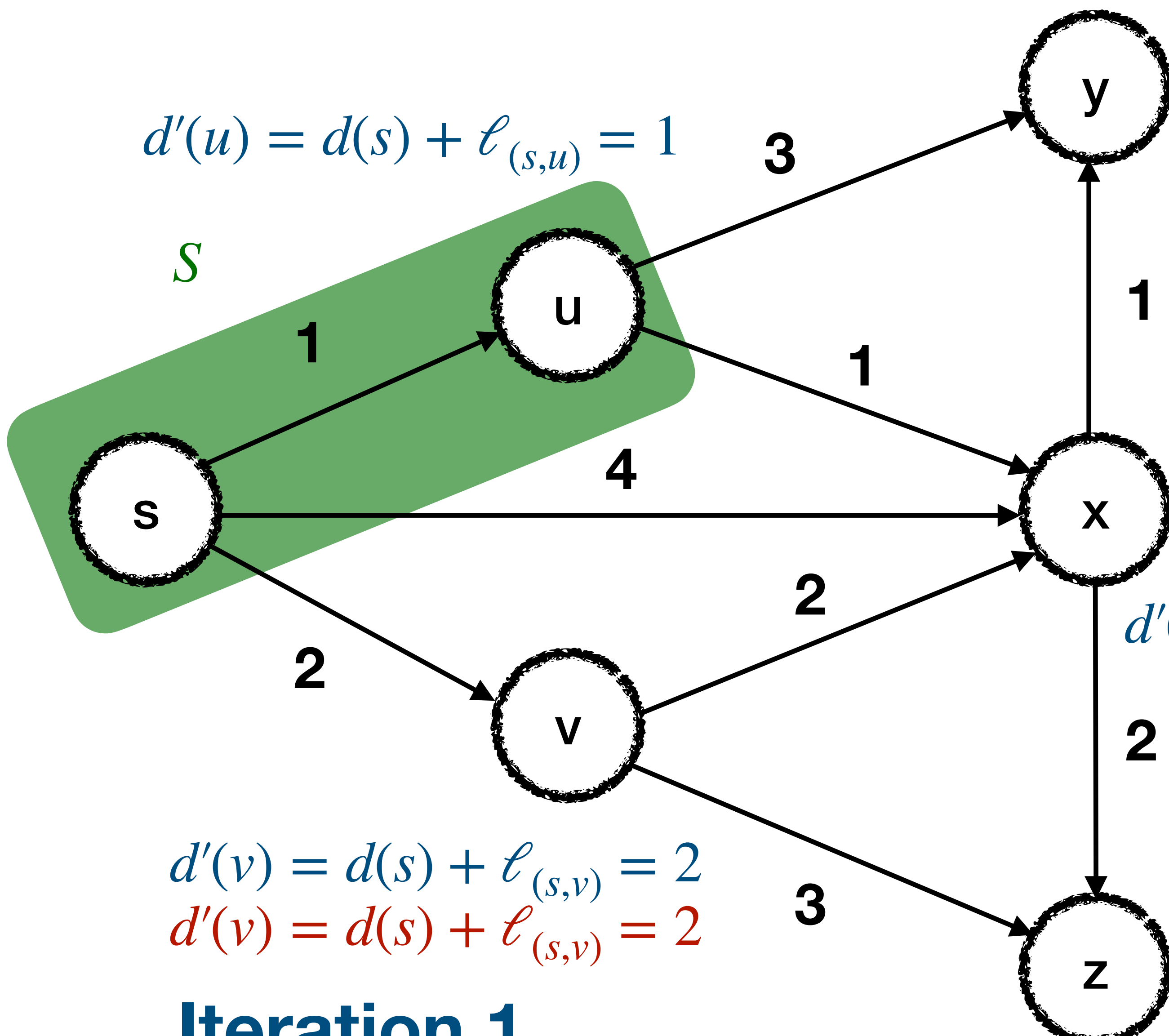
$$d'(v) = \min_{e=(u,v):u \in S} d(u) + \ell_e$$

Add v to S and define $d(v) = d'(v)$.

Iteration 1

Iteration 2

Running Example (KT Figure 5.7)



$d'(u) = d(s) + \ell_{(s,u)} = 1$

$d'(x) = d(s) + \ell_{(s,x)} = 4$

$d'(v) = d(s) + \ell_{(s,v)} = 2$

$d'(v) = d(s) + \ell_{(s,v)} = 2$

Iteration 1

Iteration 2

s	u	v	x	y	z
0	1				

$\underbrace{\qquad\qquad}$ shortest path distances from s

In other words, we choose node $v \in V - S$ such that

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

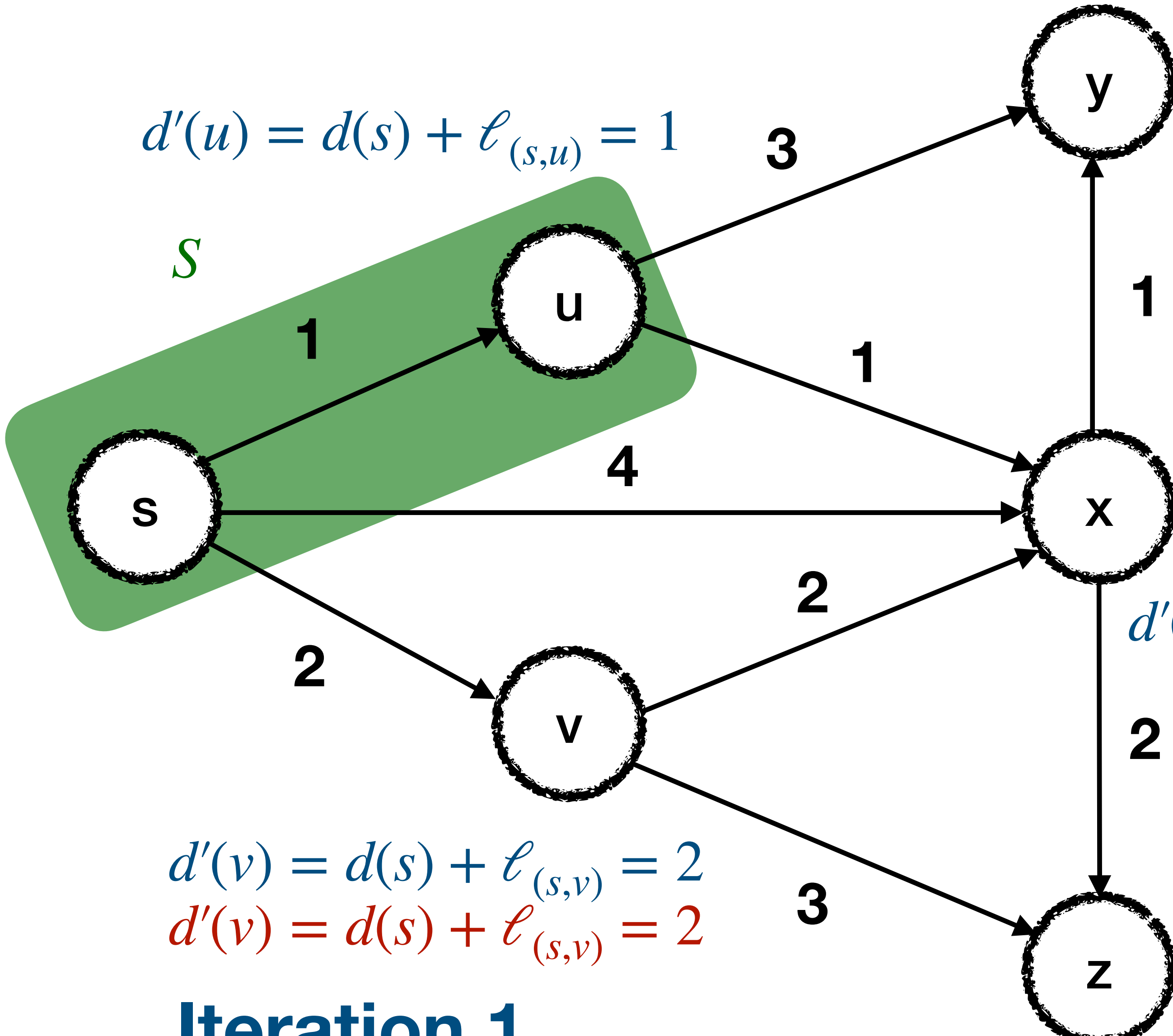
Add v to S and define $d(v) = d'(v)$.

Running Example (KT Figure 5.7)

$d'(y) = d(u) + \ell_{(u,y)} = 4$

$d'(u) = d(s) + \ell_{(s,u)} = 1$

S



$d'(x) = d(s) + \ell_{(s,x)} = 4$

$d'(v) = d(s) + \ell_{(s,v)} = 2$

$d'(v) = d(s) + \ell_{(s,v)} = 2$

Iteration 1

Iteration 2

s	u	v	x	y	z
0	1				

shortest path distances from s

In other words, we choose node v $\in V - S$ such that

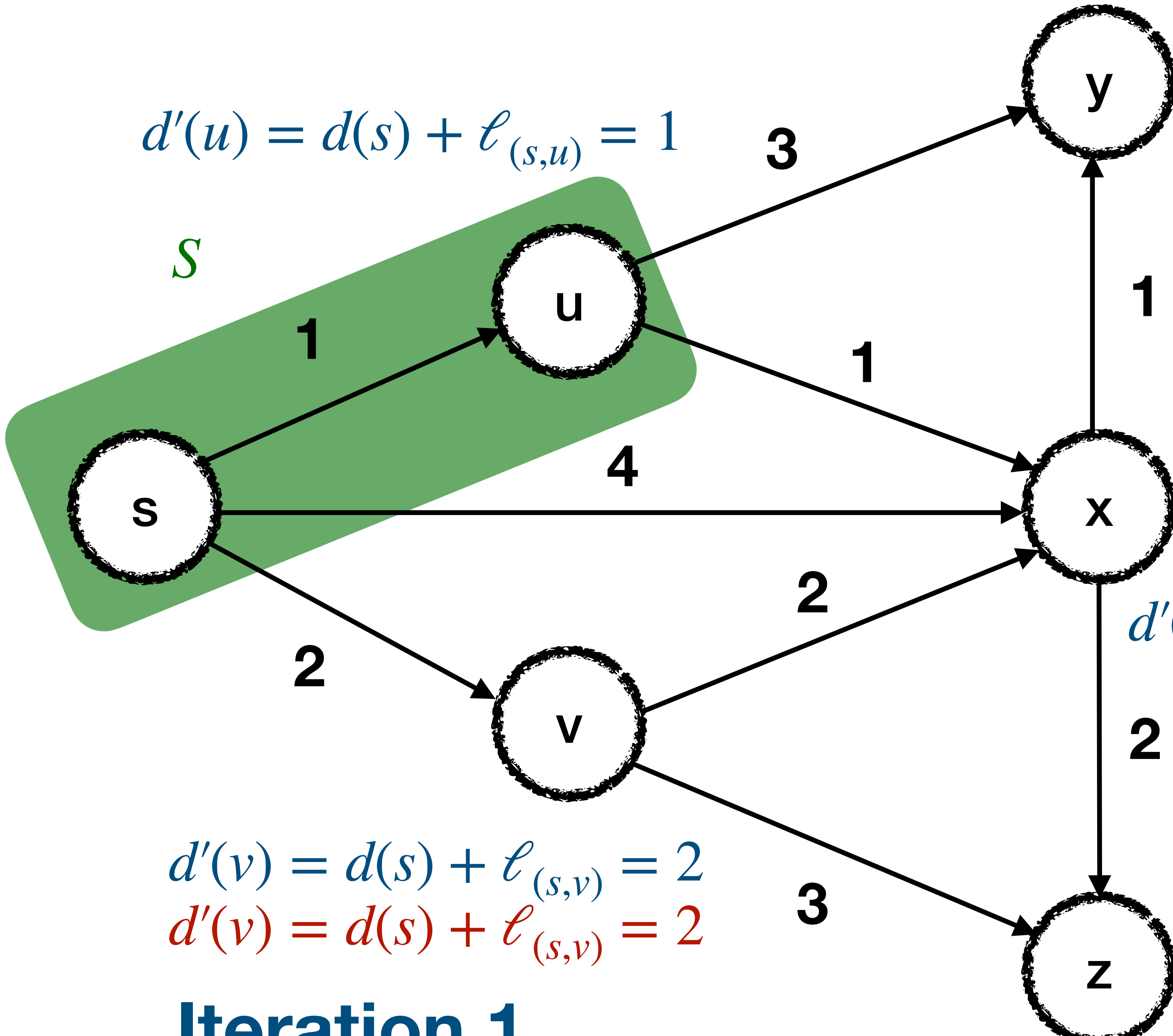
$$d'(v) = \min_{e=(u,v):u \in S} d(u) + \ell_e$$

Add v to S and define $d(v) = d'(v)$.

Running Example (KT Figure 5.7)

$d'(y) = d(u) + \ell_{(u,y)} = 4$

$d'(u) = d(s) + \ell_{(s,u)} = 1$



$d'(v) = d(s) + \ell_{(s,v)} = 2$

$d'(v) = d(s) + \ell_{(s,v)} = 2$

Iteration 1

Iteration 2

s	u	v	x	y	z
0	1				

shortest path distances from s

$d'(x) = d(s) + \ell_{(s,x)} = 4$ $d'(x) = d(u) + \ell_{(u,x)} = 2$

In other words, we choose node $v \in V-S$ such that

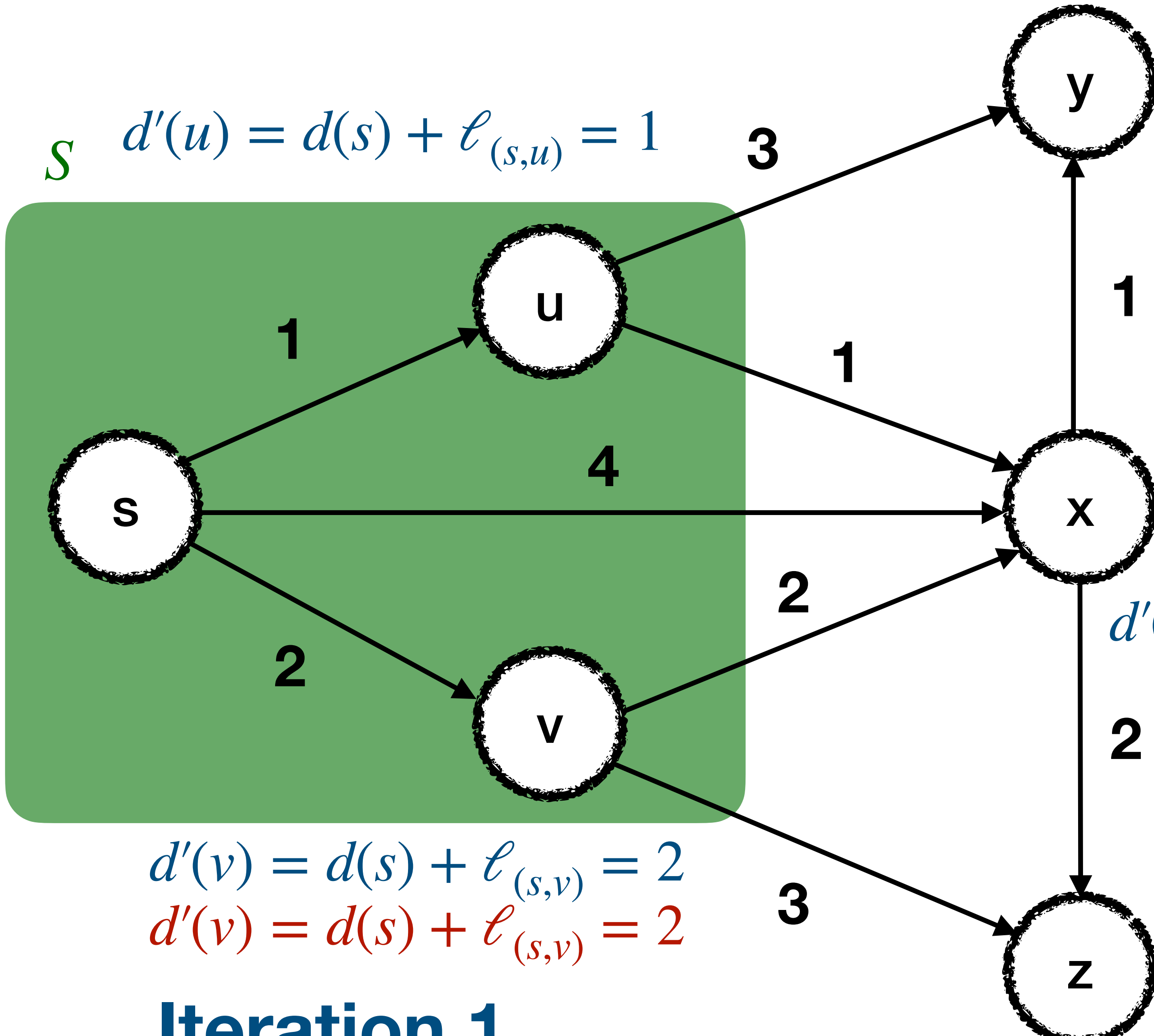
$$d'(v) = \min_{e=(u,v):u \in S} d(u) + \ell_e$$

Add v to S and define $d(v) = d'(v)$.

Running Example (KT Figure 5.7)

$d'(y) = d(u) + \ell_{(u,y)} = 4$

$d'(u) = d(s) + \ell_{(s,u)} = 1$



s	u	v	x	y	z
0	1				

shortest path distances from s

$d'(x) = d(s) + \ell_{(s,x)} = 4$ $\quad d'(x) = d(u) + \ell_{(u,x)} = 2$

$d'(v) = d(s) + \ell_{(s,v)} = 2$

$d'(v) = d(s) + \ell_{(s,v)} = 2$

Iteration 1

Iteration 2

In other words, we choose node $v \in V - S$ such that

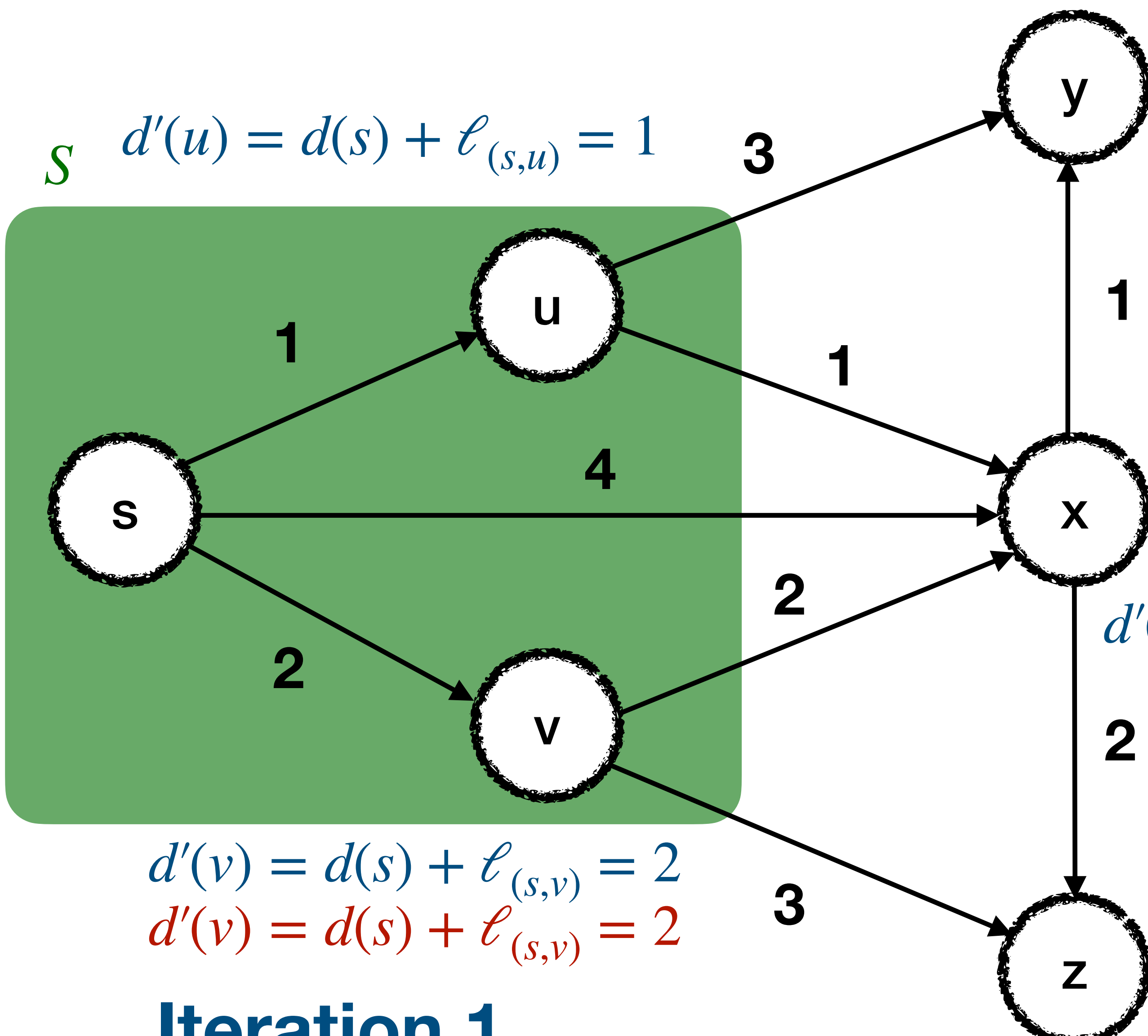
$$d'(v) = \min_{e=(u,v):u\in S} d(u) + \ell_e$$

Add v to S and define $d(v) = d'(v)$.

Running Example (KT Figure 5.7)

$d'(y) = d(u) + \ell_{(u,y)} = 4$

S $d'(u) = d(s) + \ell_{(s,u)} = 1$



$d'(v) = d(s) + \ell_{(s,v)} = 2$

$d'(v) = d(s) + \ell_{(s,v)} = 2$

s	u	v	x	y	z
0	1	2			

shortest path distances from s

$d'(x) = d(s) + \ell_{(s,x)} = 4$ $d'(x) = d(u) + \ell_{(u,x)} = 2$

Iteration 1

Iteration 2

In other words, we choose node $v \in V - S$ such that

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

Add v to S and define $d(v) = d'(v)$.

Dijkstra's Algorithm (Pseudocode)

<u>Dijsktra (G, ℓ)</u>

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$

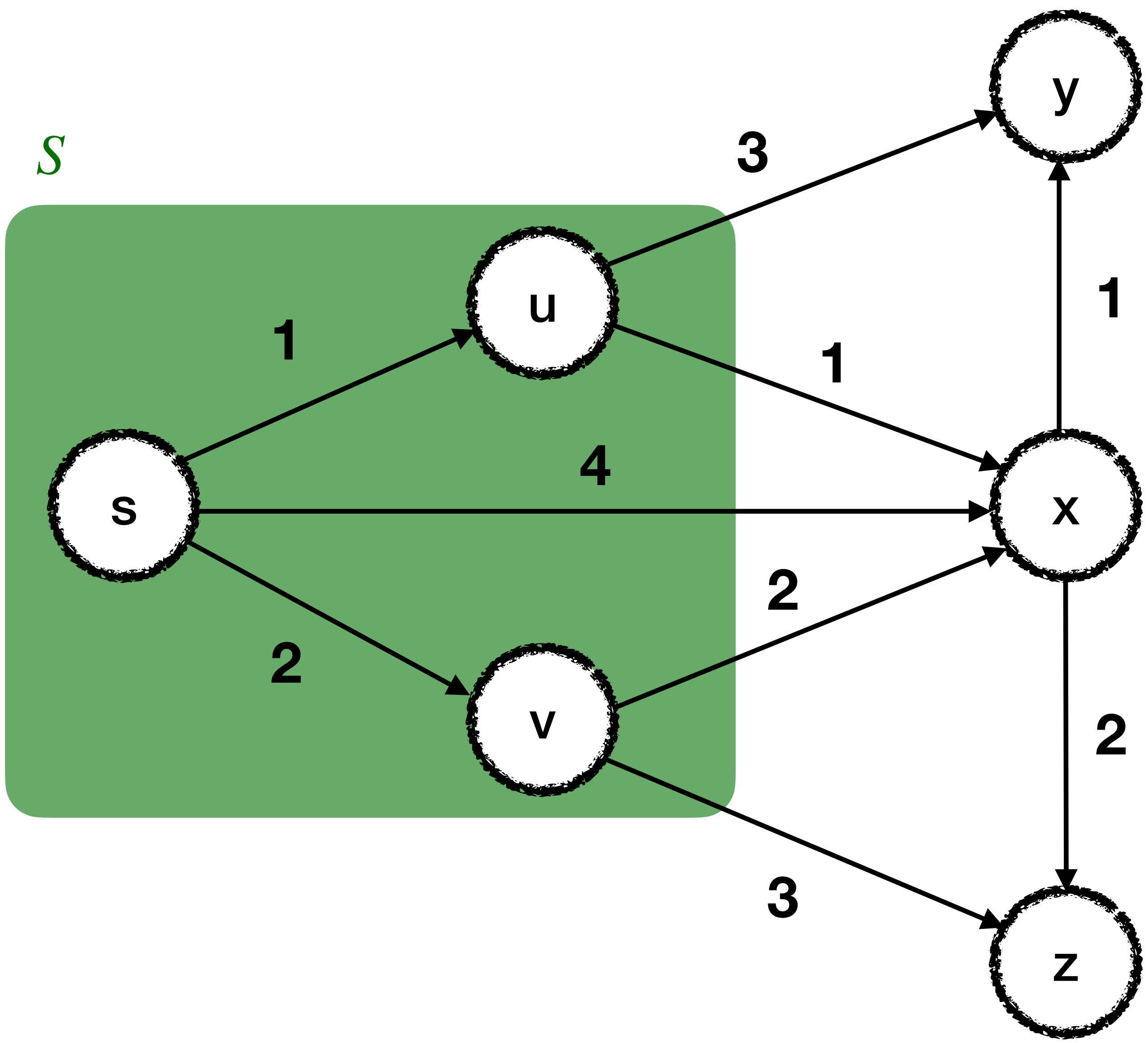
Select a node $v \in V - S$ connected via an edge with at least one node in S for which

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Running Example (KT Figure 5.7)



s	u	v	x	y	z
0	1	2			

shortest path distances from s

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$

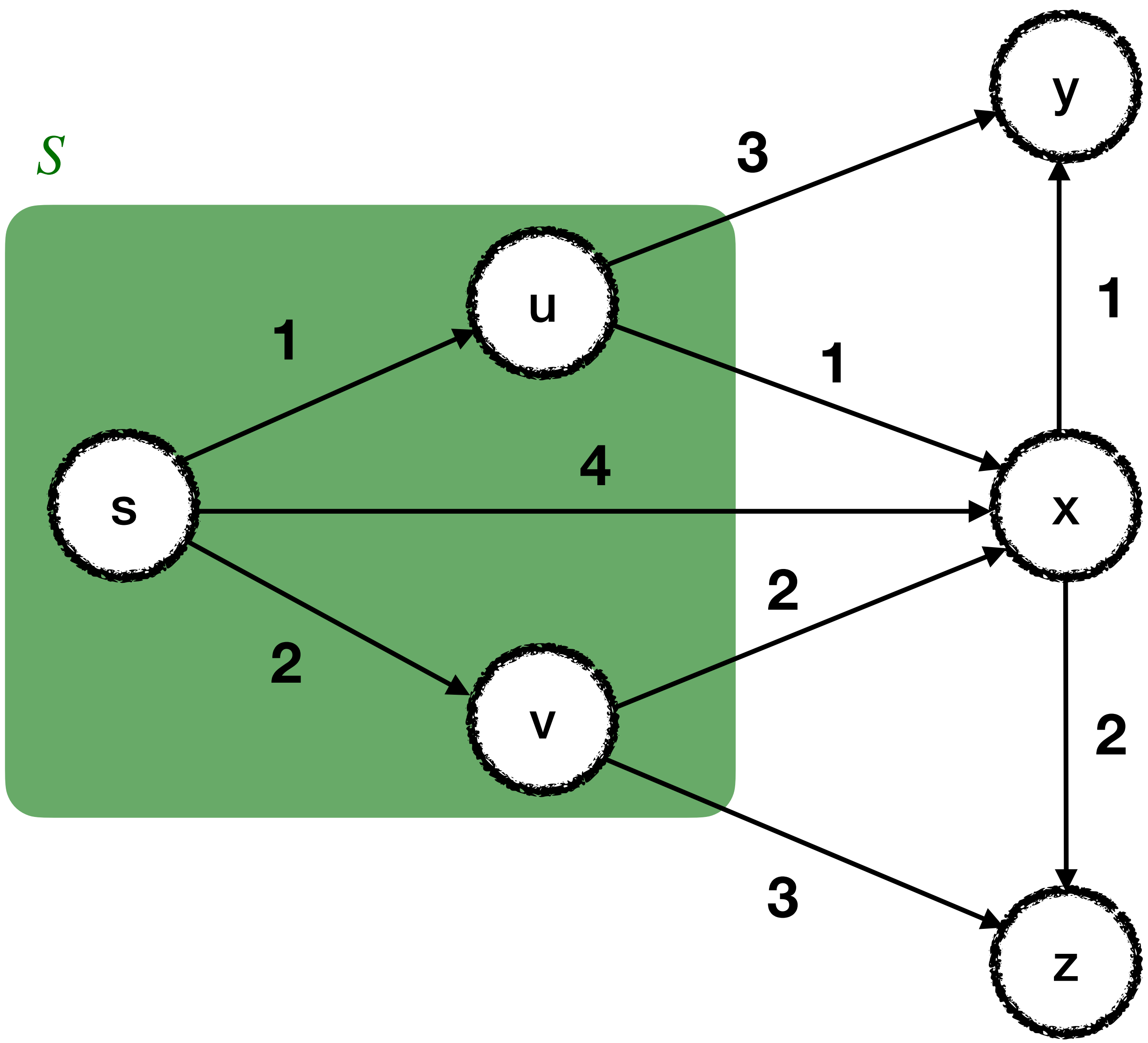
Select a node $v \in V - S$ connected via an edge with at least one node in S for which

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Running Example (KT Figure 5.7)



s	u	v	x	y	z
0	1	2			

shortest path distances from s

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$

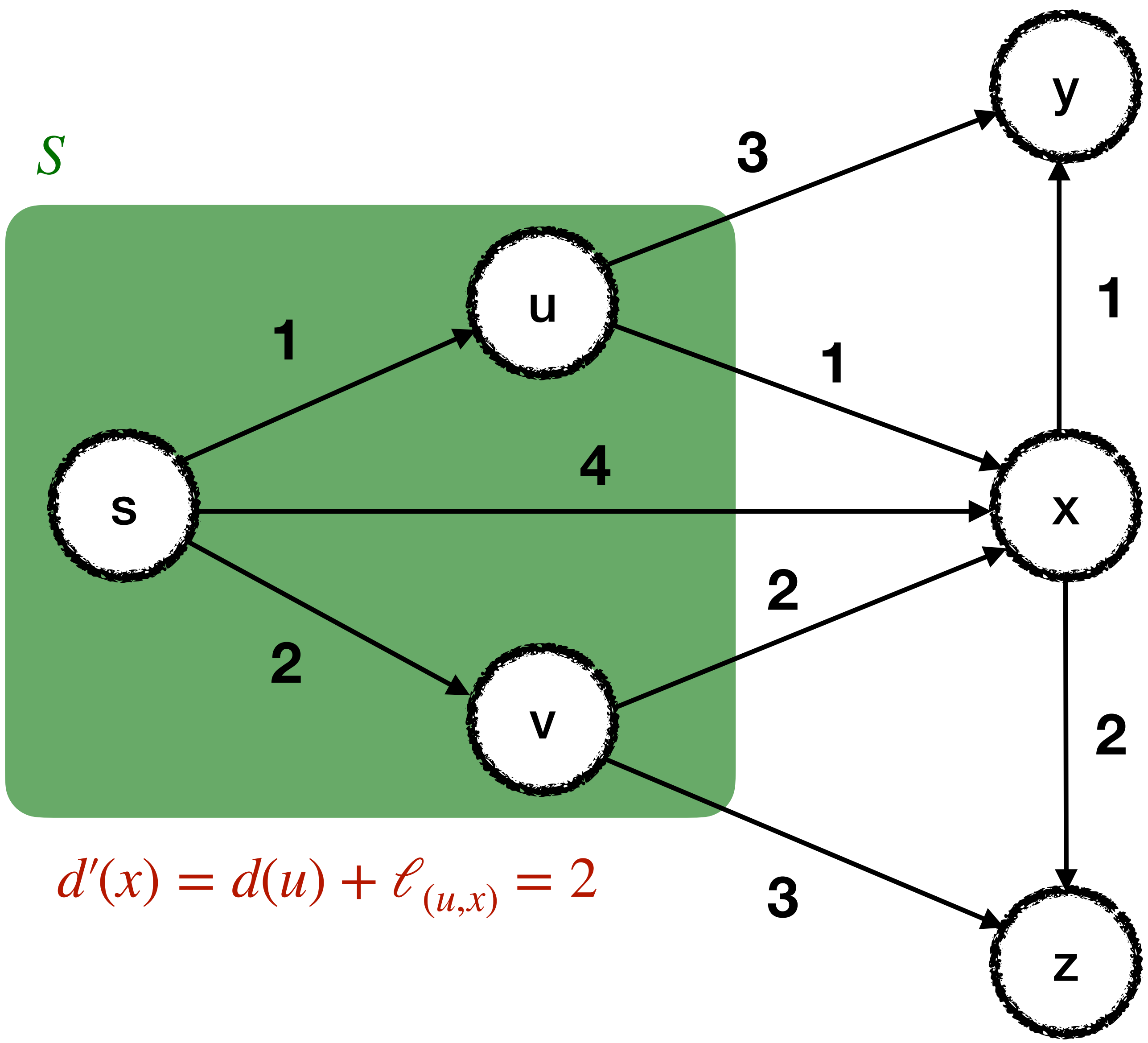
Select a node $v \in V - S$ connected via an edge with at least one node in S for which

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Running Example (KT Figure 5.7)



$d'(x) = d(u) + \ell_{(u,x)} = 2$

s	u	v	x	y	z
0	1	2			

shortest path distances from s

Dijsktra (G, ℓ)

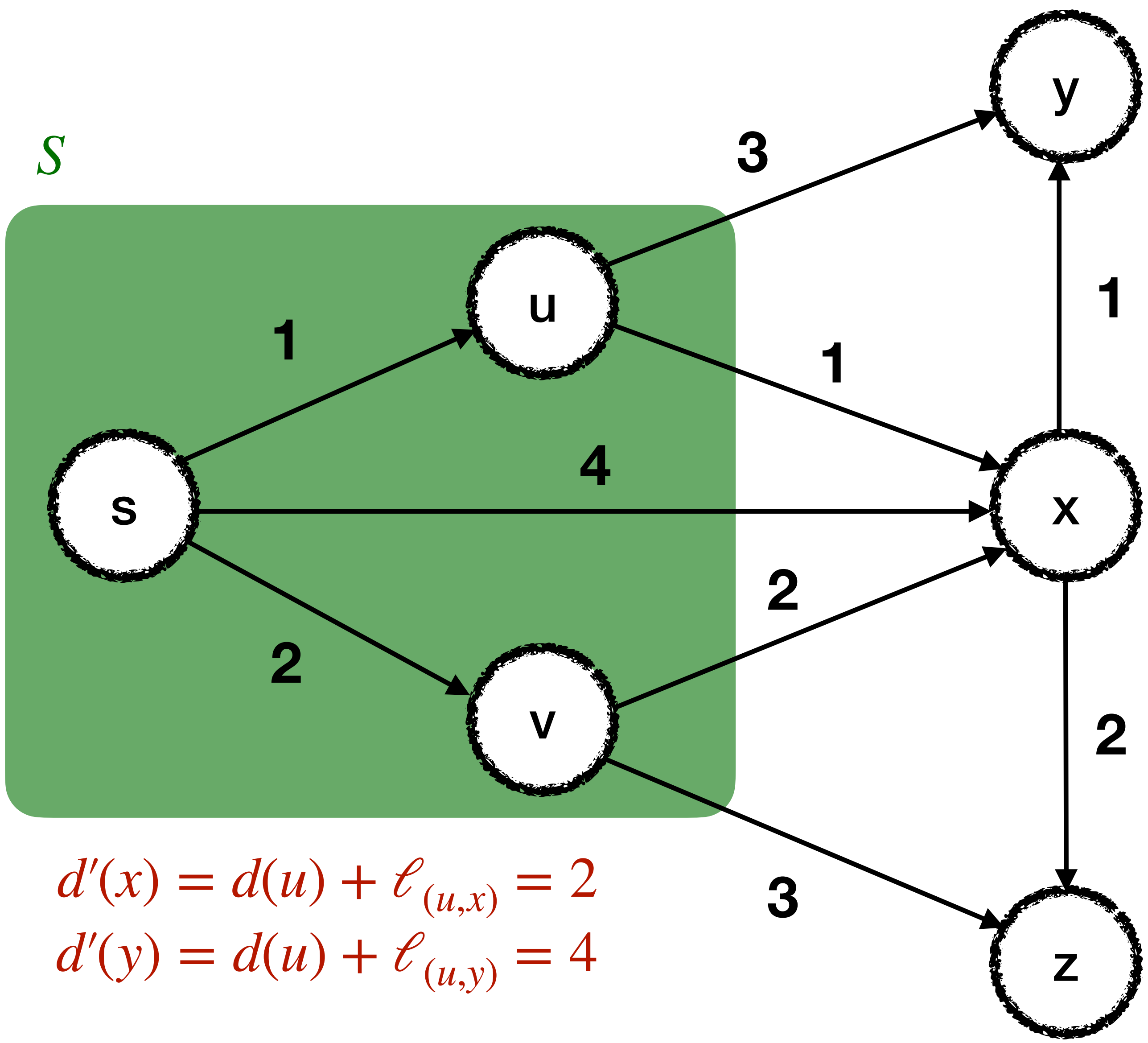
Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$ ✓

> Select a node $v \in V - S$ connected via an edge with at least one node in S for which
>
> $$d'(v) = \min_{e=(u,v):u \in S} d(u) + \ell_e$$
>
> Add v to S and define $d(v) = d'(v)$
>
> Let $A[v] = d(v)$

Running Example (KT Figure 5.7)



$d'(x) = d(u) + \ell_{(u,x)} = 2$

$d'(y) = d(u) + \ell_{(u,y)} = 4$

s	u	v	x	y	z
0	1	2			

shortest path distances from s

<u>Dijsktra (G, ℓ)</u>

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$

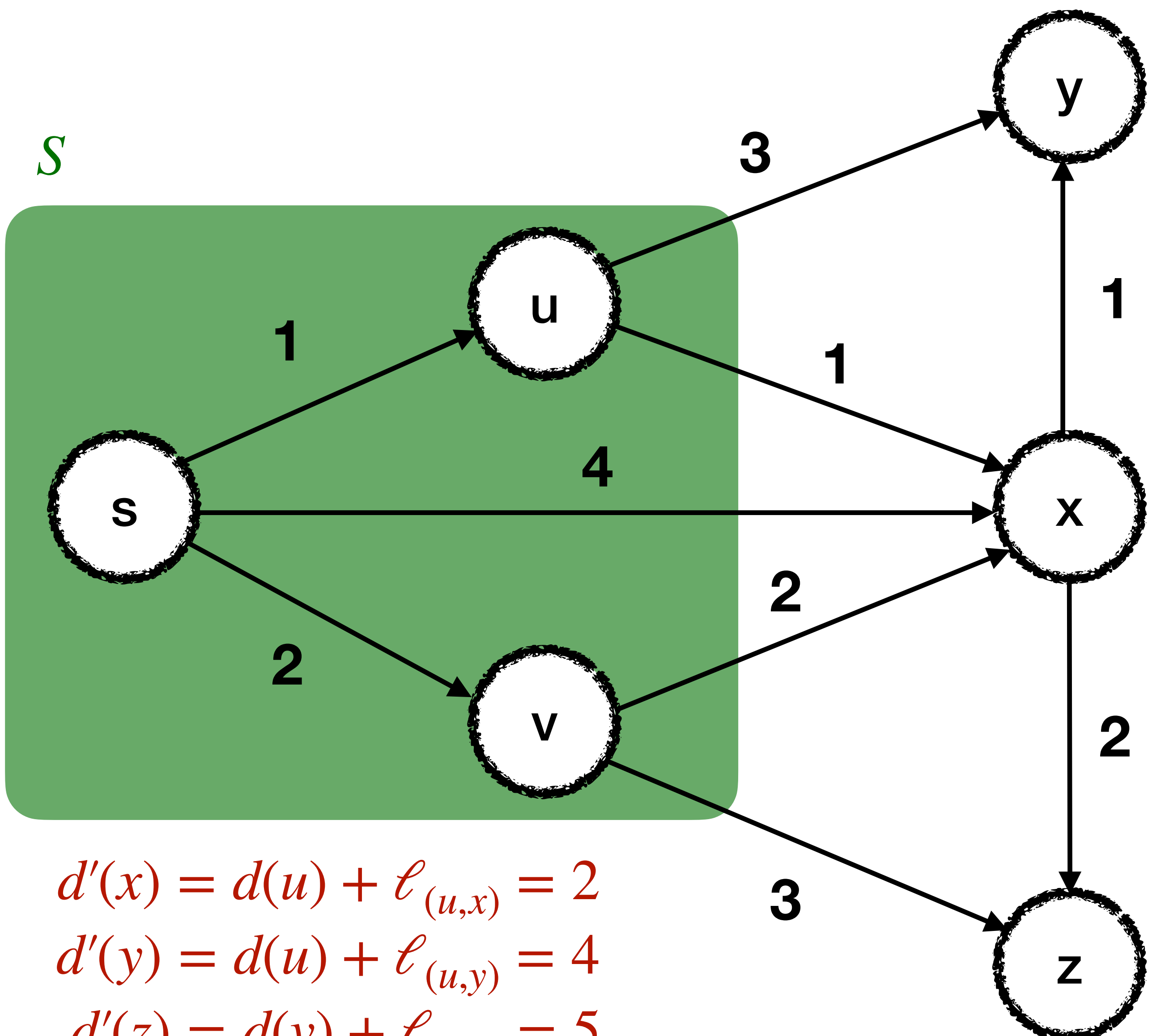
Select a node $v \in V - S$ connected via an edge with at least one node in S for which

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Running Example (KT Figure 5.7)



$d'(x) = d(u) + \ell_{(u,x)} = 2$

$d'(y) = d(u) + \ell_{(u,y)} = 4$

$d'(z) = d(v) + \ell_{(v,z)} = 5$

s	u	v	x	y	z
0	1	2			

shortest path distances from s

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$

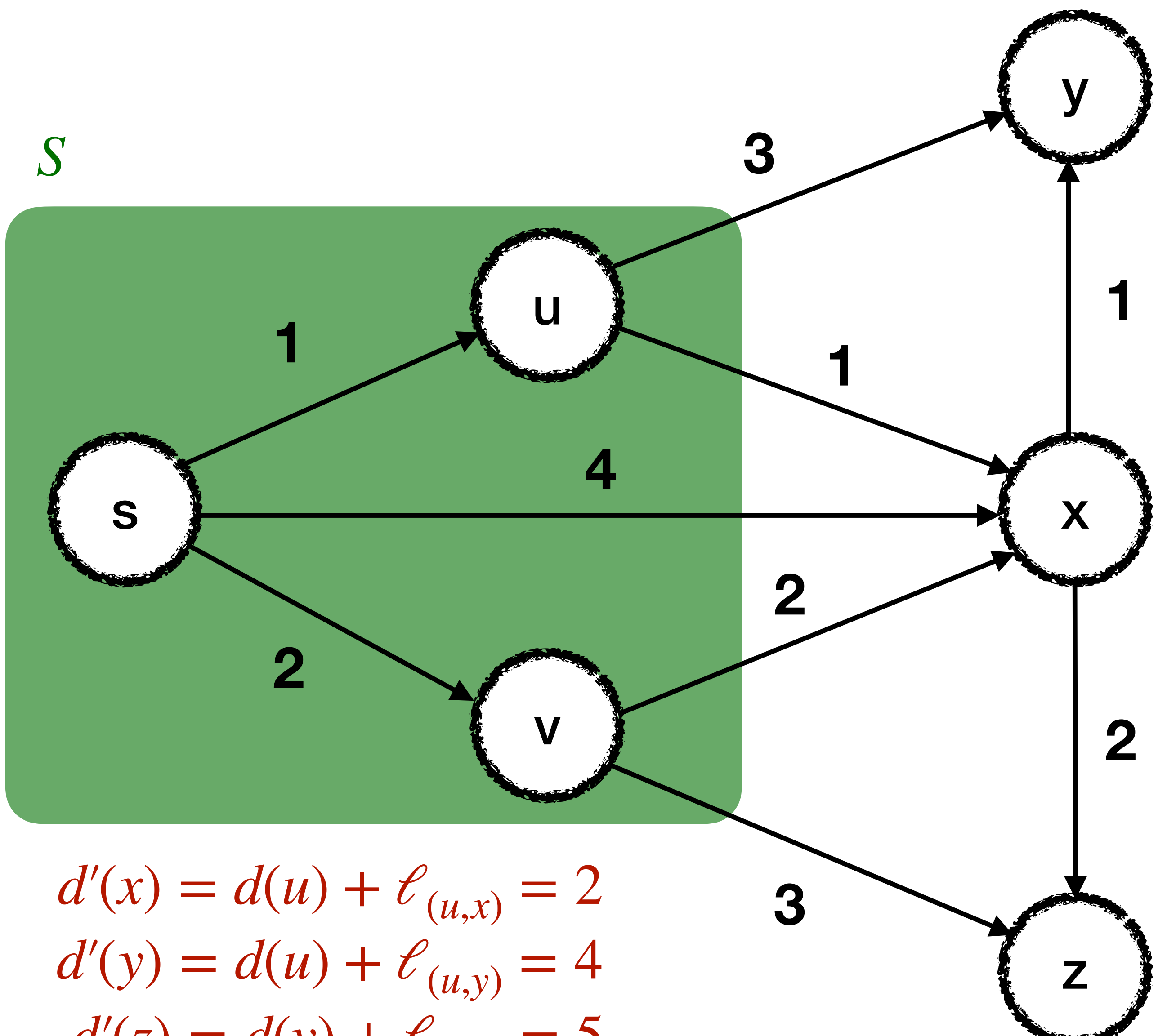
Select a node $v \in V - S$ connected via an edge with at least one node in S for which

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Running Example (KT Figure 5.7)



$$d'(x) = d(u) + \ell_{(u,x)} = 2$$
$$d'(y) = d(u) + \ell_{(u,y)} = 4$$
$$d'(z) = d(v) + \ell_{(v,z)} = 5$$

s	u	v	x	y	z
0	1	2			

shortest path distances from s

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$

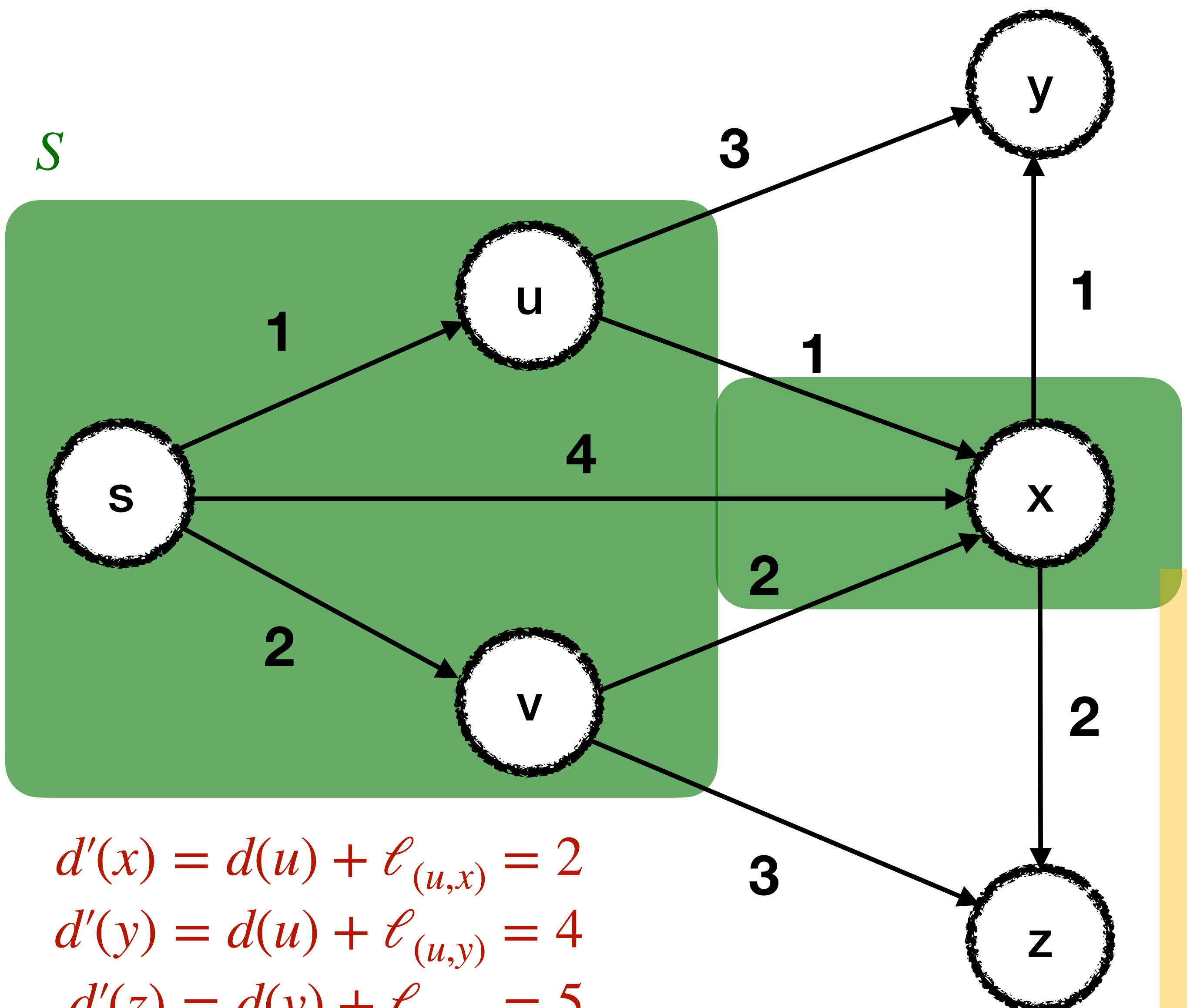
Select a node $v \in V - S$ connected via an edge with at least one node in S for which

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Running Example (KT Figure 5.7)



$d'(x) = d(u) + \ell_{(u,x)} = 2$
$d'(y) = d(u) + \ell_{(u,y)} = 4$
$d'(z) = d(v) + \ell_{(v,z)} = 5$

s	u	v	x	y	z
0	1	2			

shortest path distances from s

<u>Dijsktra (G, ℓ)</u>

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$

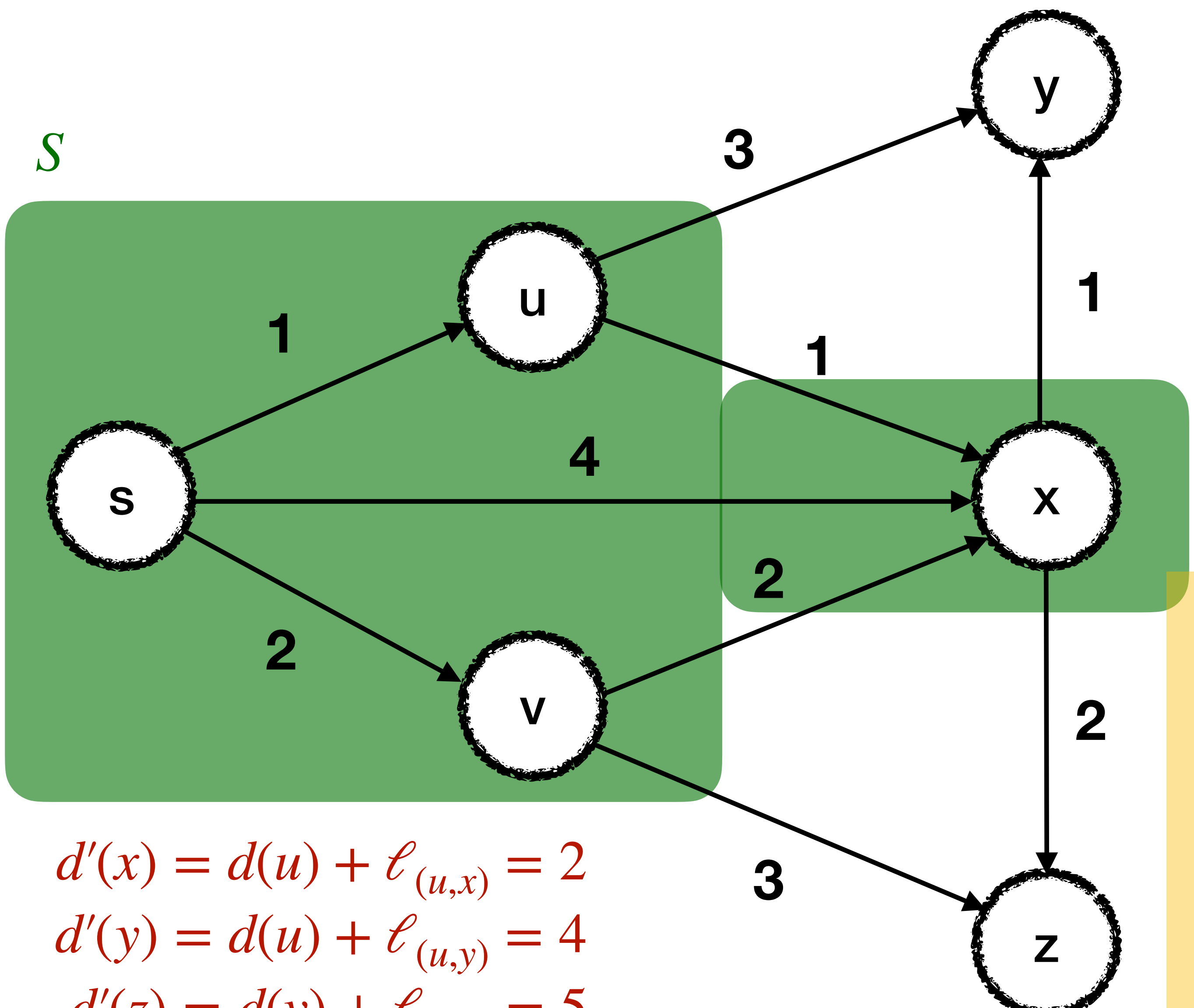
Select a node $v \in V - S$ connected via an edge with at least one node in S for which

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Running Example (KT Figure 5.7)



$$d'(x) = d(u) + \ell_{(u,x)} = 2$$
$$d'(y) = d(u) + \ell_{(u,y)} = 4$$
$$d'(z) = d(v) + \ell_{(v,z)} = 5$$

s	u	v	x	y	z
0	1	2			

$\underbrace{\qquad\qquad}_{\text{shortest path distances from } s}$

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$

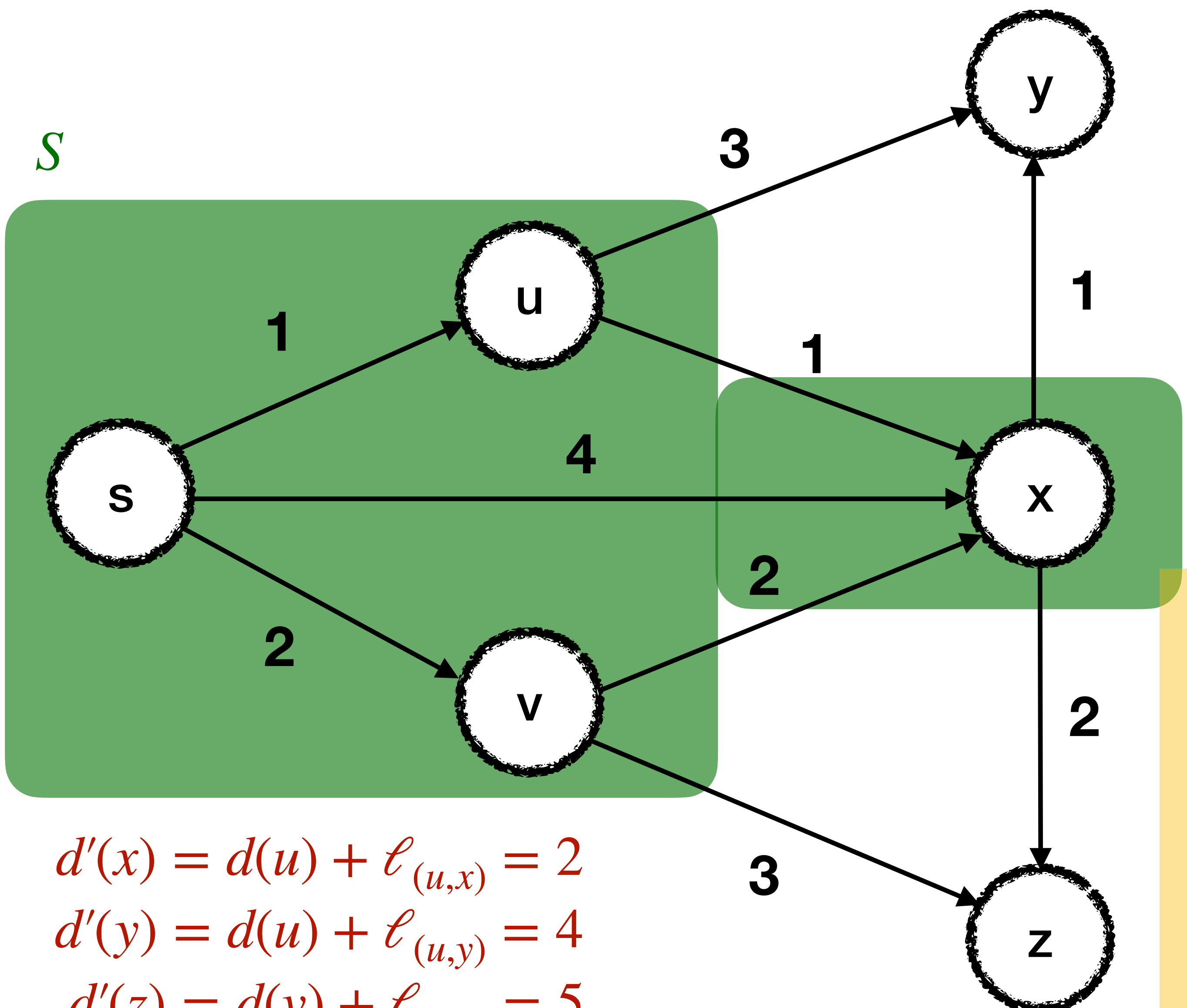
Select a node $v \in V - S$ connected via an edge with at least one node in S for which

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$ $\quad A[x] = 2$

Running Example (KT Figure 5.7)



s	u	v	x	y	z
0	1	2	2		

shortest path distances from s

$$d'(x) = d(u) + \ell_{(u,x)} = 2$$
$$d'(y) = d(u) + \ell_{(u,y)} = 4$$
$$d'(z) = d(v) + \ell_{(v,z)} = 5$$

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$

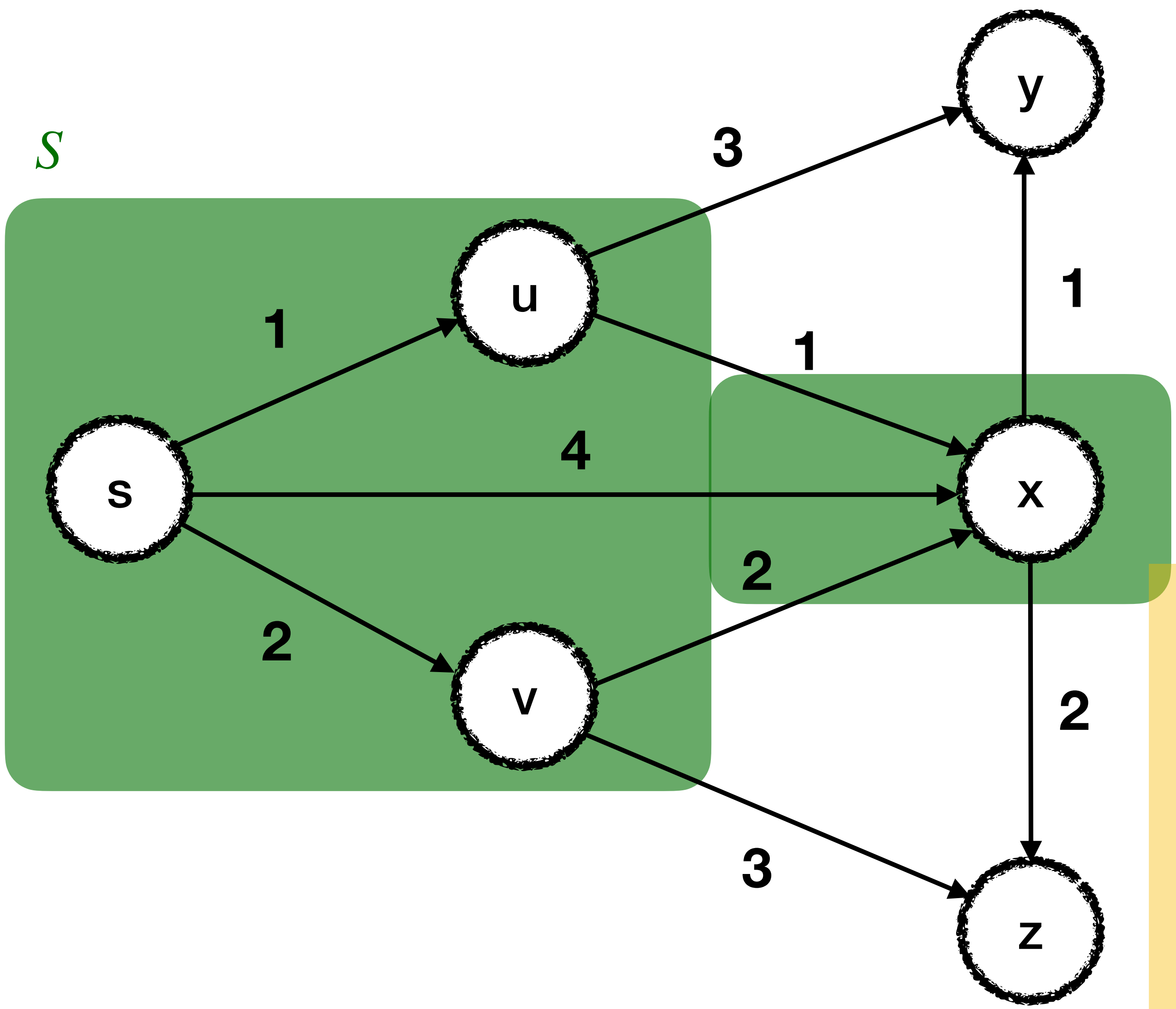
Select a node $v \in V - S$ connected via an edge with at least one node in S for which

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$ $\quad A[x] = 2$

Running Example (KT Figure 5.7)



s	u	v	x	y	z
0	1	2	2		

shortest path distances from s

<u>Dijsktra (G, ℓ)</u>

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$

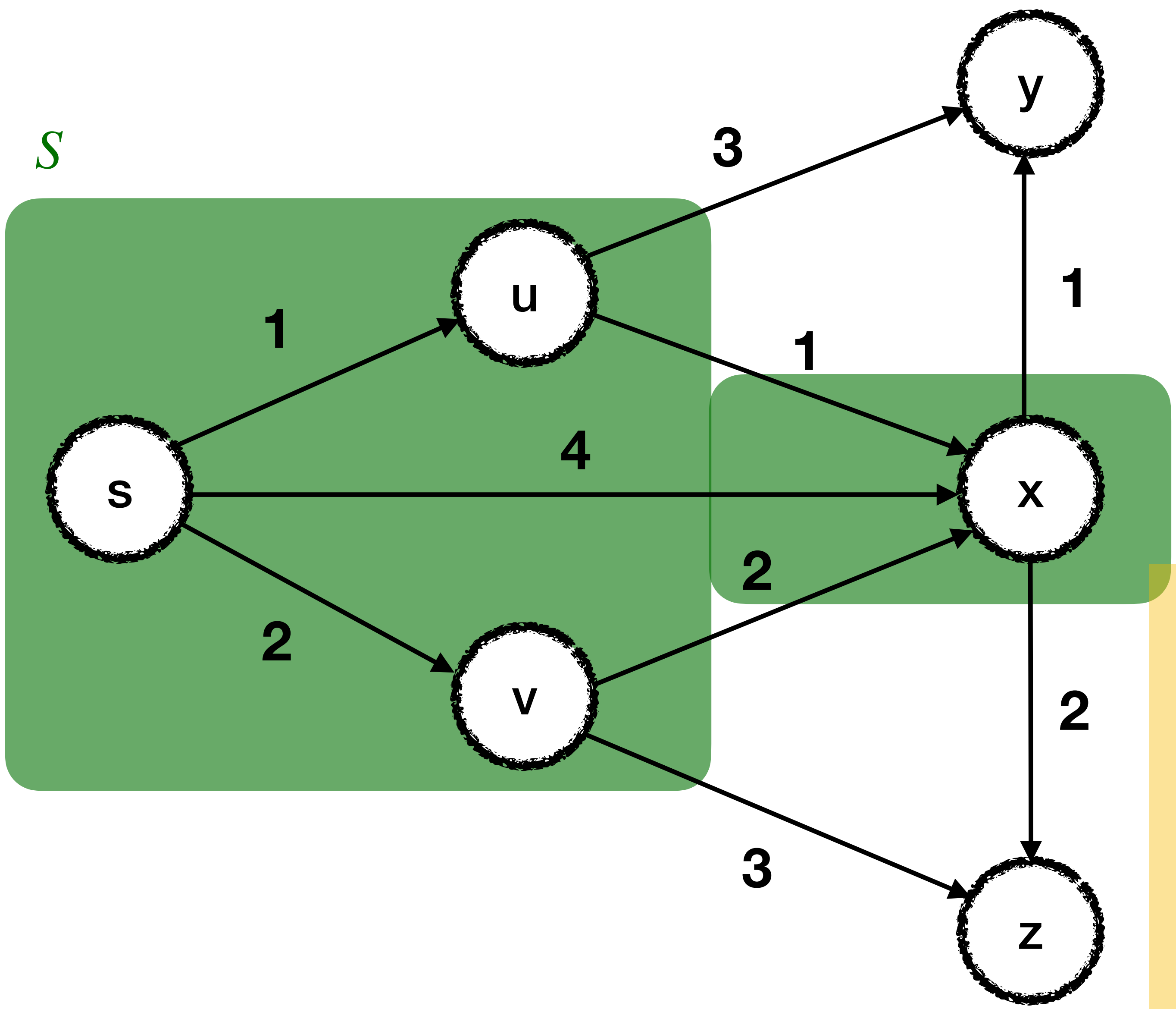
Select a node $v \in V - S$ connected via an edge with at least one node in S for which

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Running Example (KT Figure 5.7)



s	u	v	x	y	z
0	1	2	2		

shortest path distances from s

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$

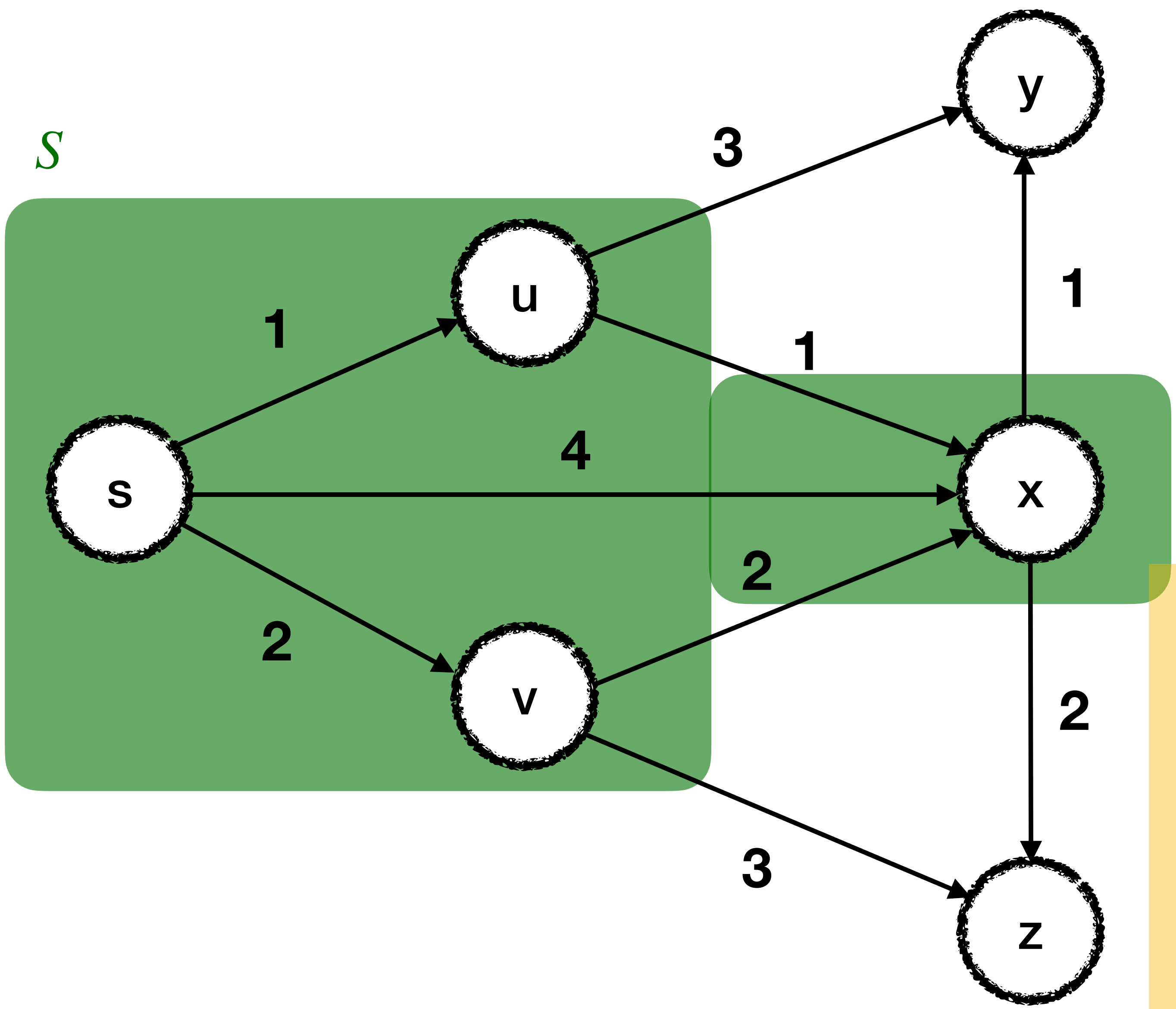
Select a node $v \in V - S$ connected via an edge with at least one node in S for which

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Running Example (KT Figure 5.7)



s	u	v	x	y	z
0	1	2	2		

$\underbrace{\qquad\qquad}$ shortest path distances from s

$$d'(y) = d(x) + \ell_{(x,y)} = 3$$

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$

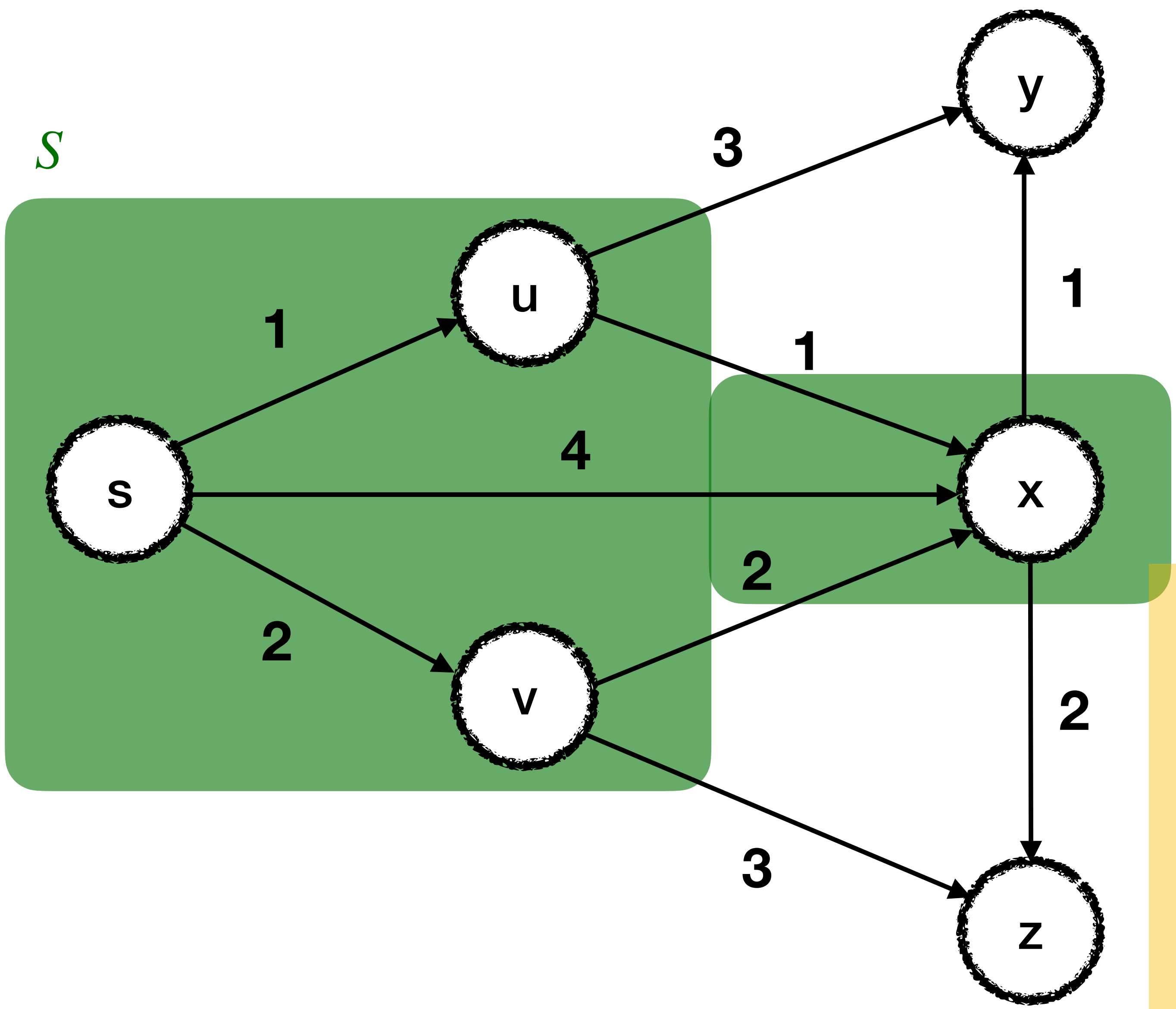
Select a node $v \in V - S$ connected via an edge with at least one node in S for which

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Running Example (KT Figure 5.7)



s	u	v	x	y	z
0	1	2	2		

shortest path distances from s

$$d'(y) = d(x) + \ell_{(x,y)} = 3$$
$$d'(z) = d(x) + \ell_{(x,z)} = 4$$

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$

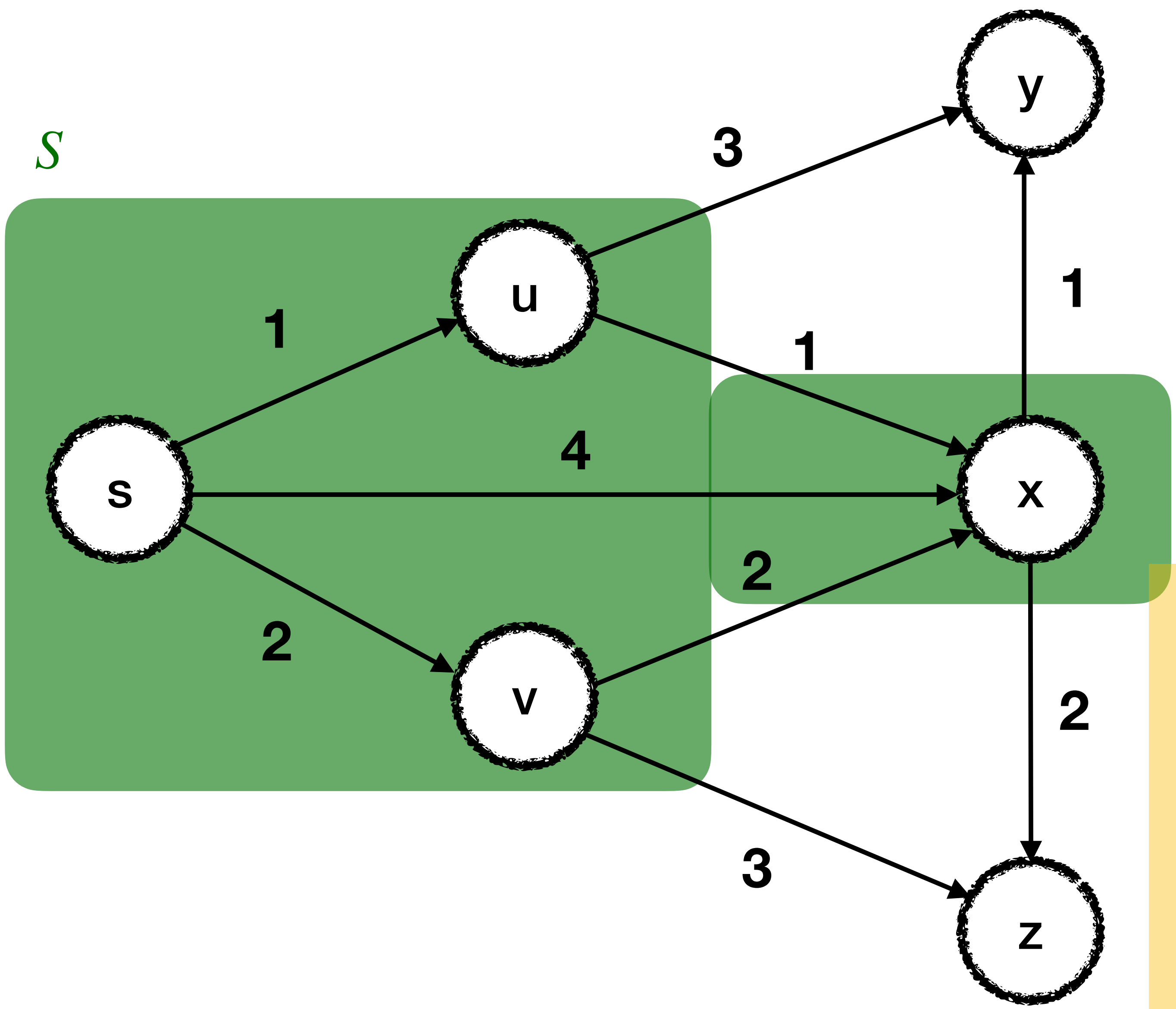
Select a node $v \in V - S$ connected via an edge with at least one node in S for which

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Running Example (KT Figure 5.7)



s	u	v	x	y	z
0	1	2	2		

$\underbrace{\hphantom{xxxxxx}}_{\text{shortest path distances from } s}$

$$d'(y) = d(x) + \ell_{(x,y)} = 3$$

$$d'(z) = d(x) + \ell_{(x,z)} = 4$$

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$

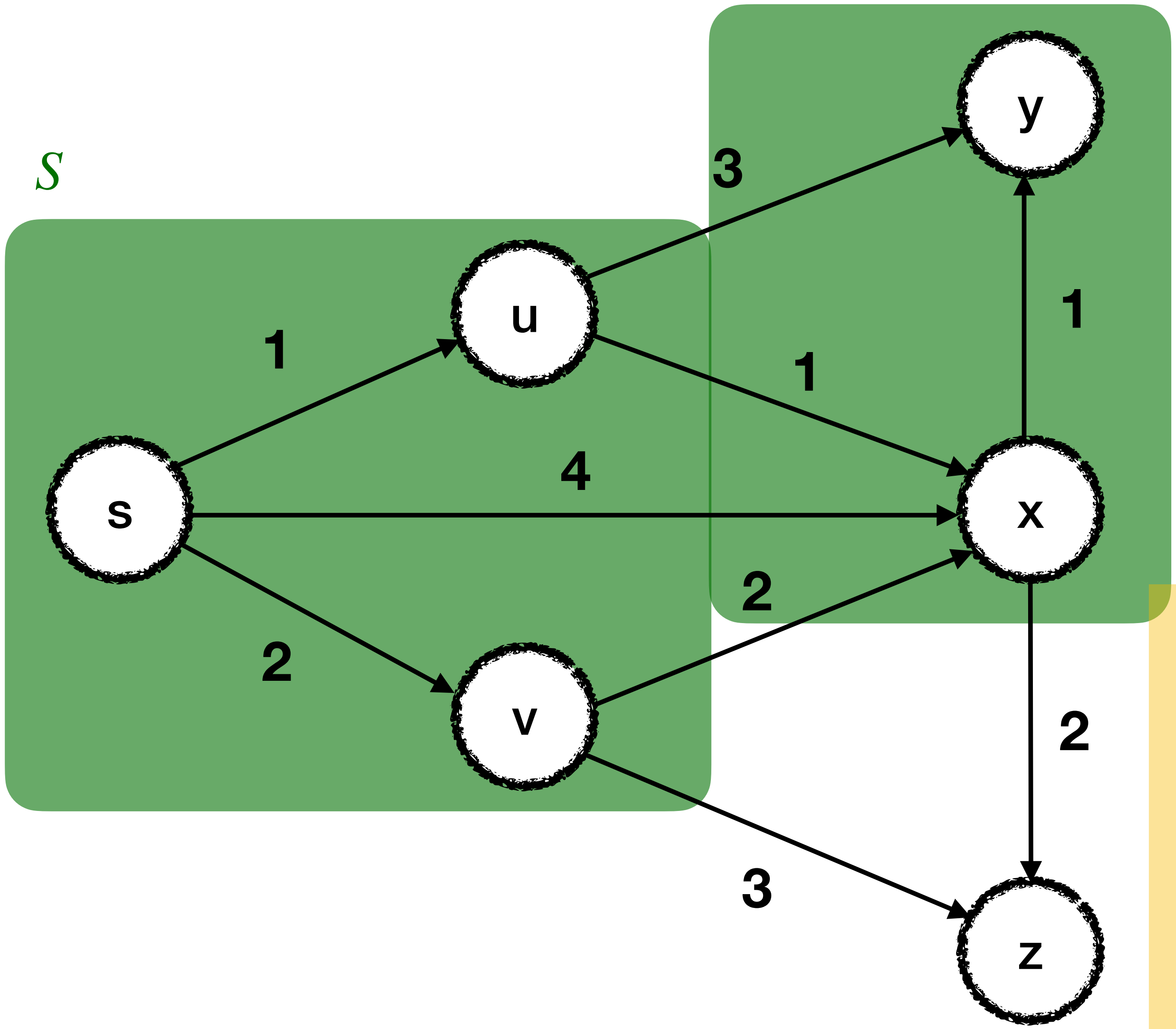
Select a node $v \in V - S$ connected via an edge with at least one node in S for which

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Running Example (KT Figure 5.7)



s	u	v	x	y	z
0	1	2	2		

$\underbrace{\hspace{6em}}$ shortest path distances from s

$d'(y) = d(x) + \ell_{(x,y)} = 3$

$d'(z) = d(x) + \ell_{(x,z)} = 4$

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$

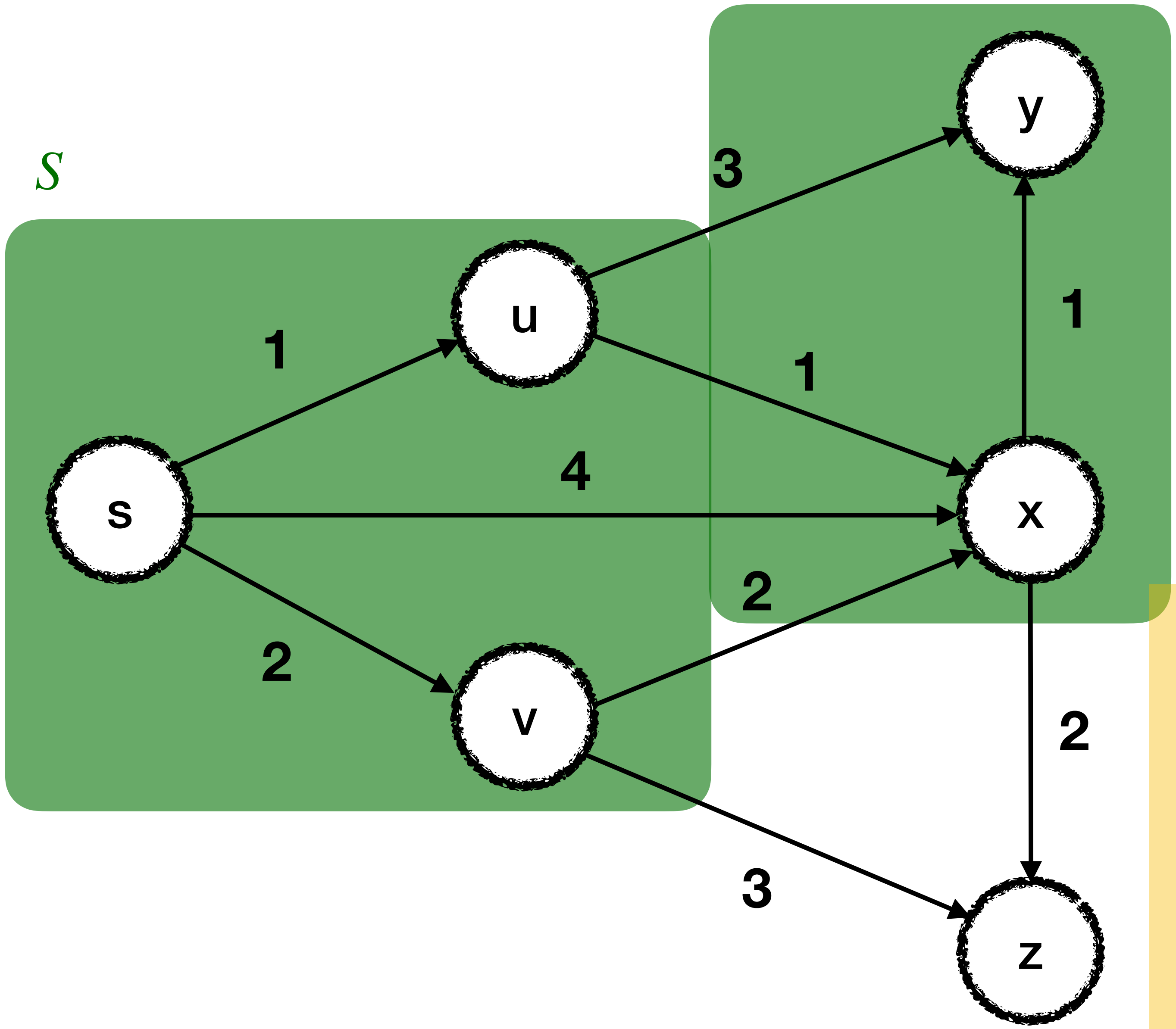
Select a node $v \in V - S$ connected via an edge with at least one node in S for which

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Running Example (KT Figure 5.7)



s	u	v	x	y	z
0	1	2	2		

shortest path distances from s

$$d'(y) = d(x) + \ell_{(x,y)} = 3$$
$$d'(z) = d(x) + \ell_{(x,z)} = 4$$

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$

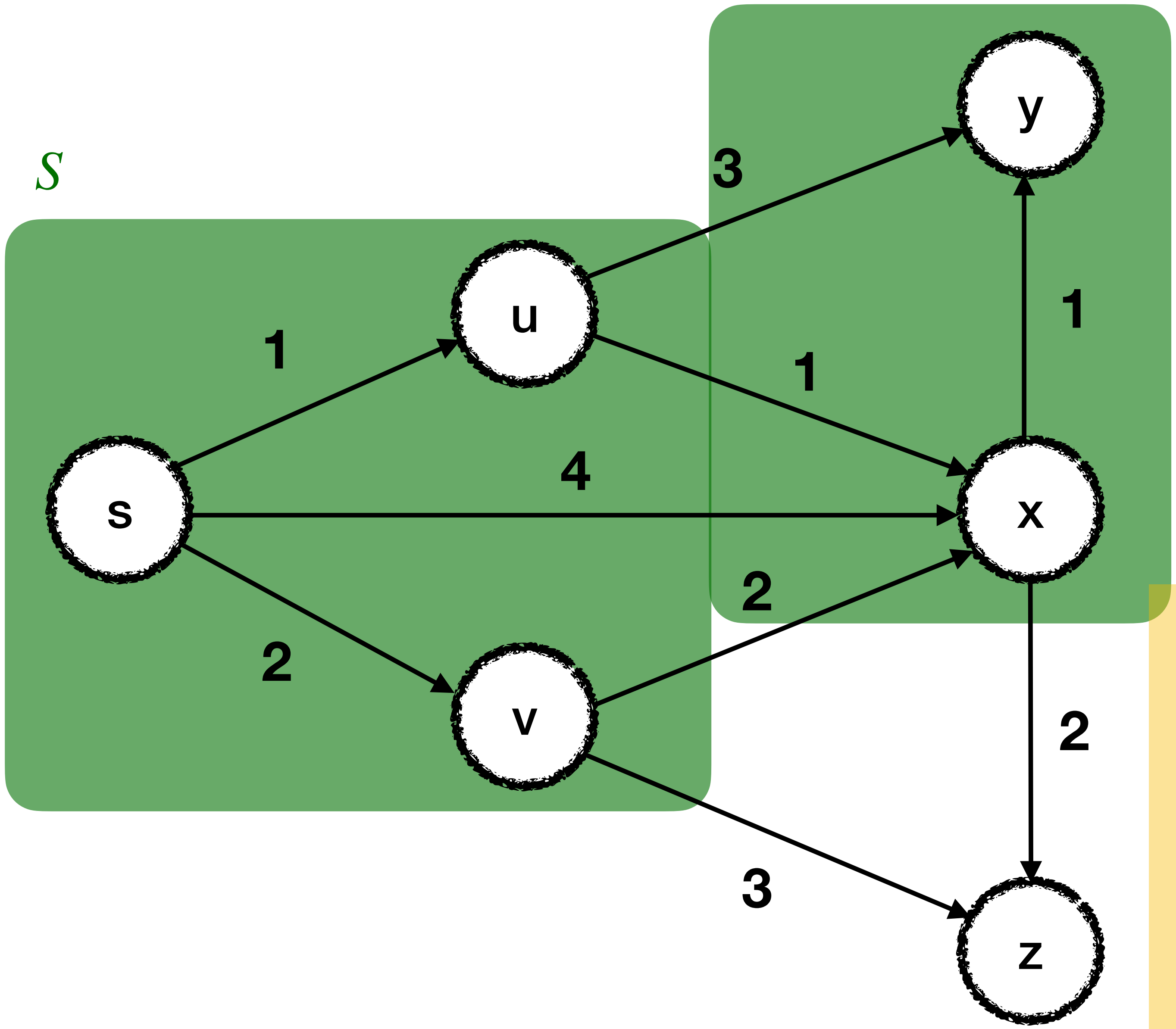
Select a node $v \in V - S$ connected via an edge with at least one node in S for which

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$ $\qquad A[y] = 3$

Running Example (KT Figure 5.7)



s	u	v	x	y	z
0	1	2	2	3	

$\underbrace{\qquad\qquad\qquad}_{}$ shortest path distances from s

$$d'(y) = d(x) + \ell_{(x,y)} = 3$$
$$d'(z) = d(x) + \ell_{(x,z)} = 4$$

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$

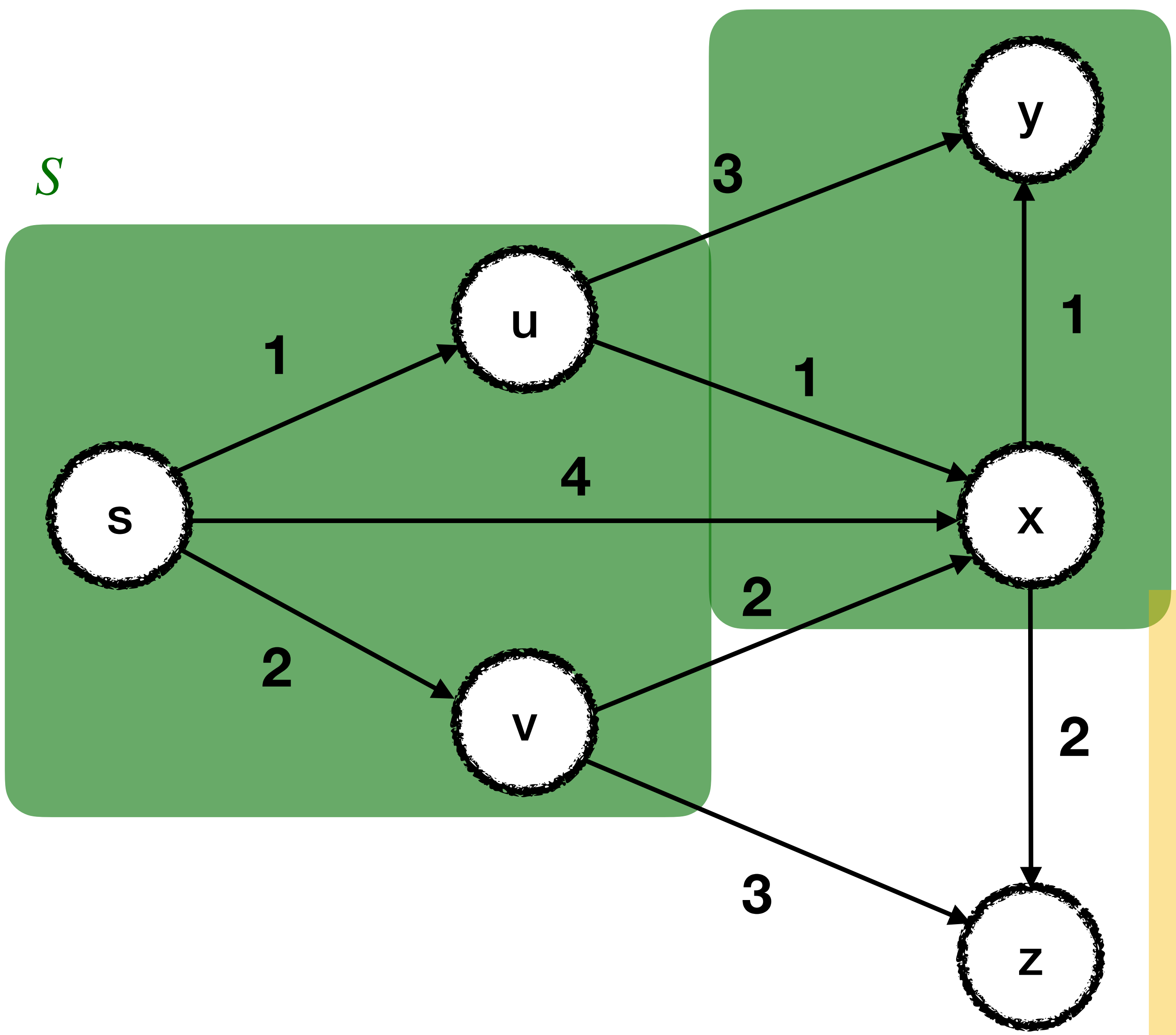
Select a node $v \in V - S$ connected via an edge with at least one node in S for which

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$ $\qquad A[y] = 3$

Running Example (KT Figure 5.7)



s	u	v	x	y	z
0	1	2	2	3	

shortest path distances from s

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$

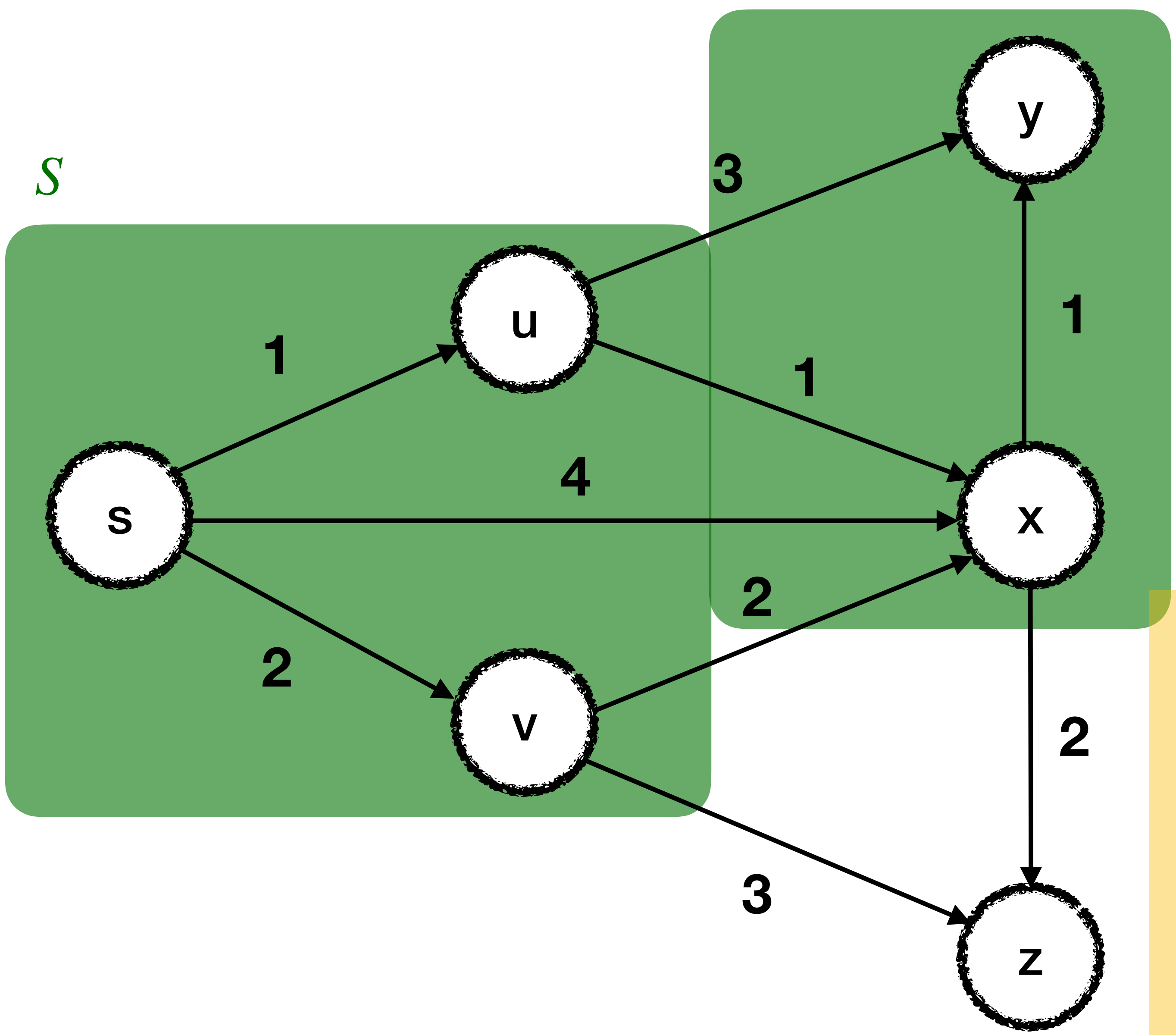
Select a node $v \in V - S$ connected via an edge with at least one node in S for which

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Running Example (KT Figure 5.7)



s	u	v	x	y	z
0	1	2	2	3	

shortest path distances from s

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$

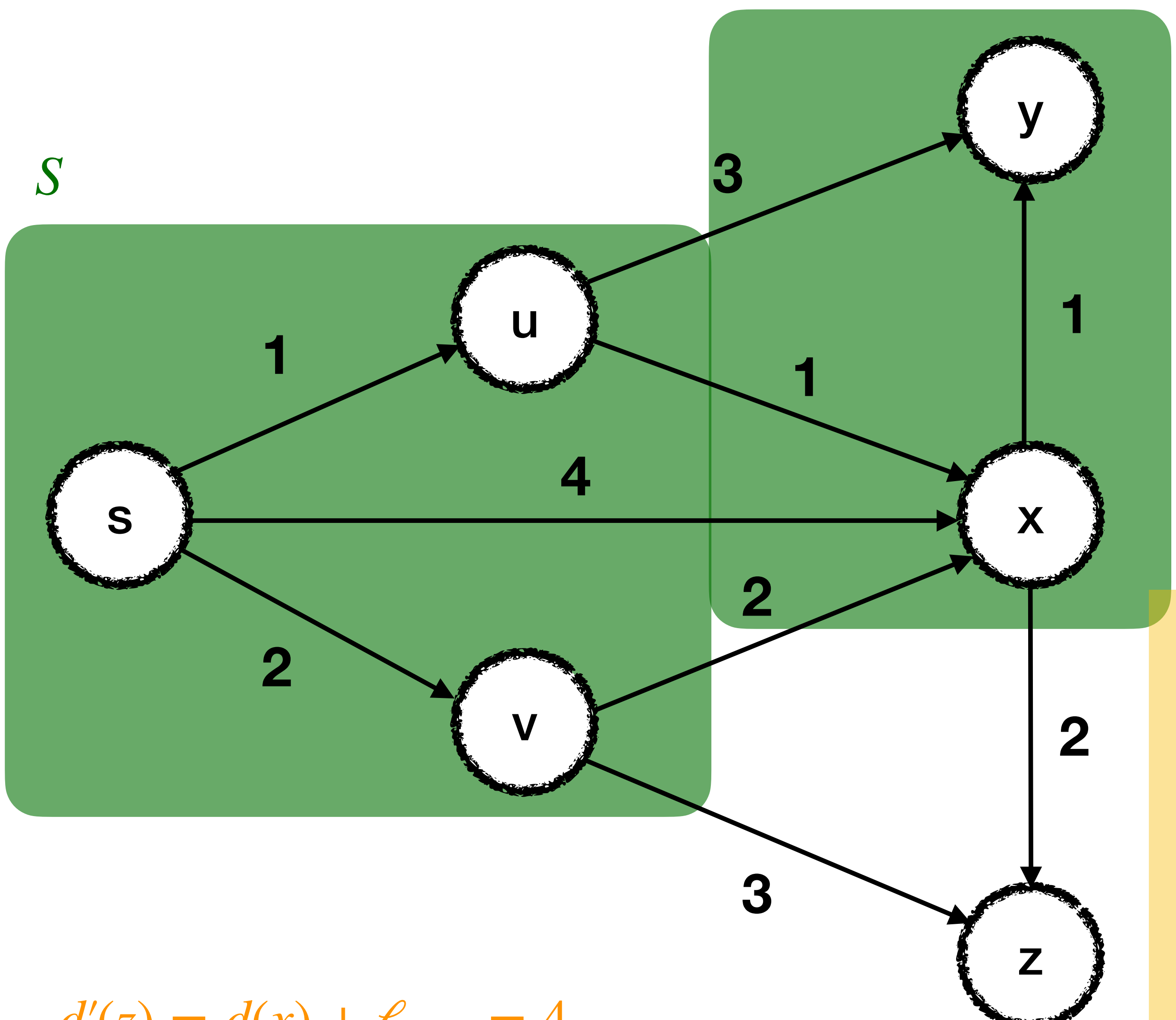
Select a node $v \in V - S$ connected via an edge with at least one node in S for which

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Running Example (KT Figure 5.7)



$d'(z) = d(x) + \ell_{(x,z)} = 4$

s	u	v	x	y	z
0	1	2	2	3	

shortest path distances from s

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$

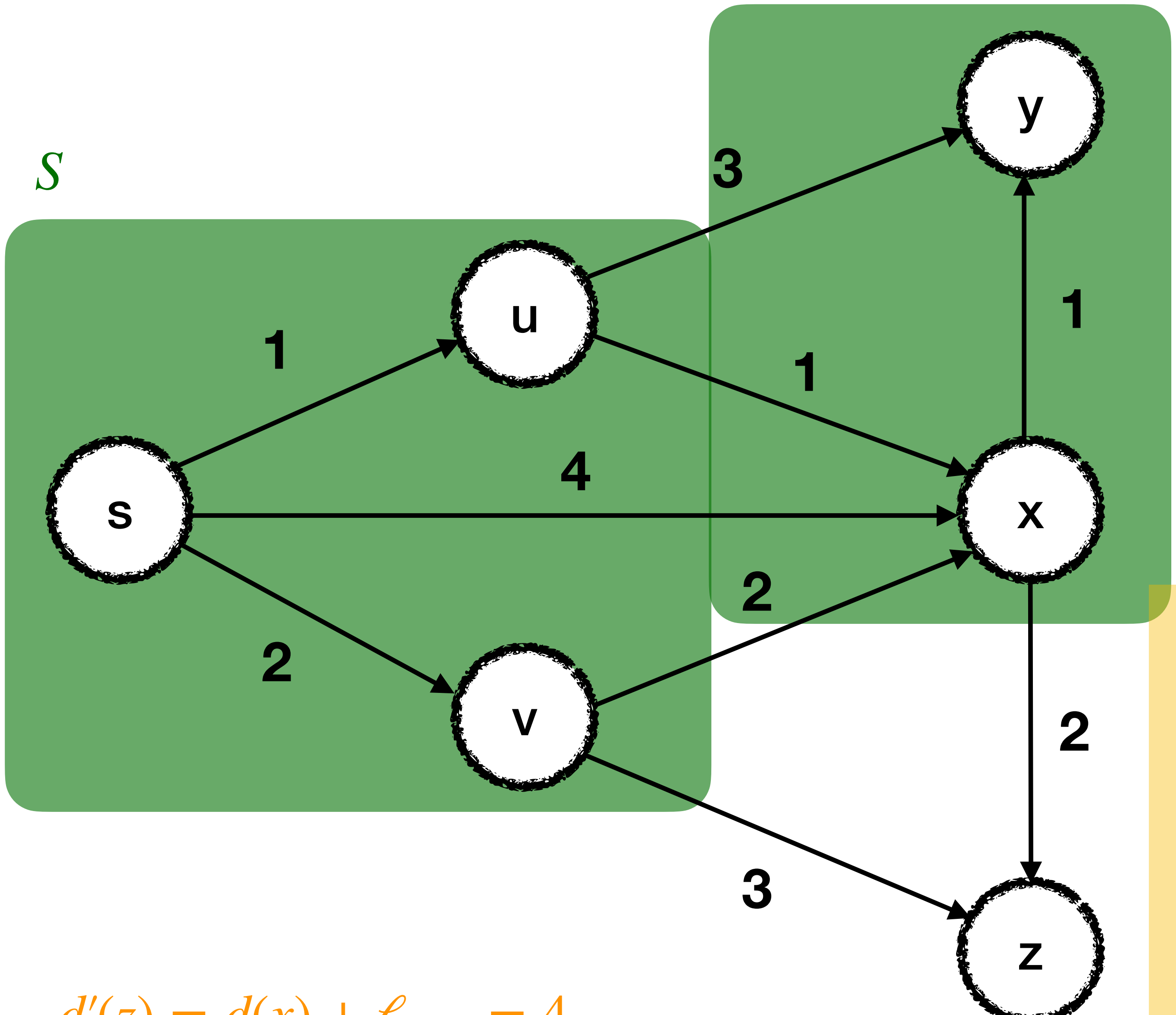
Select a node $v \in V - S$ connected via an edge with at least one node in S for which

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Running Example (KT Figure 5.7)



$d'(z) = d(x) + \ell_{(x,z)} = 4$

s	u	v	x	y	z
0	1	2	2	3	

shortest path distances from s

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$

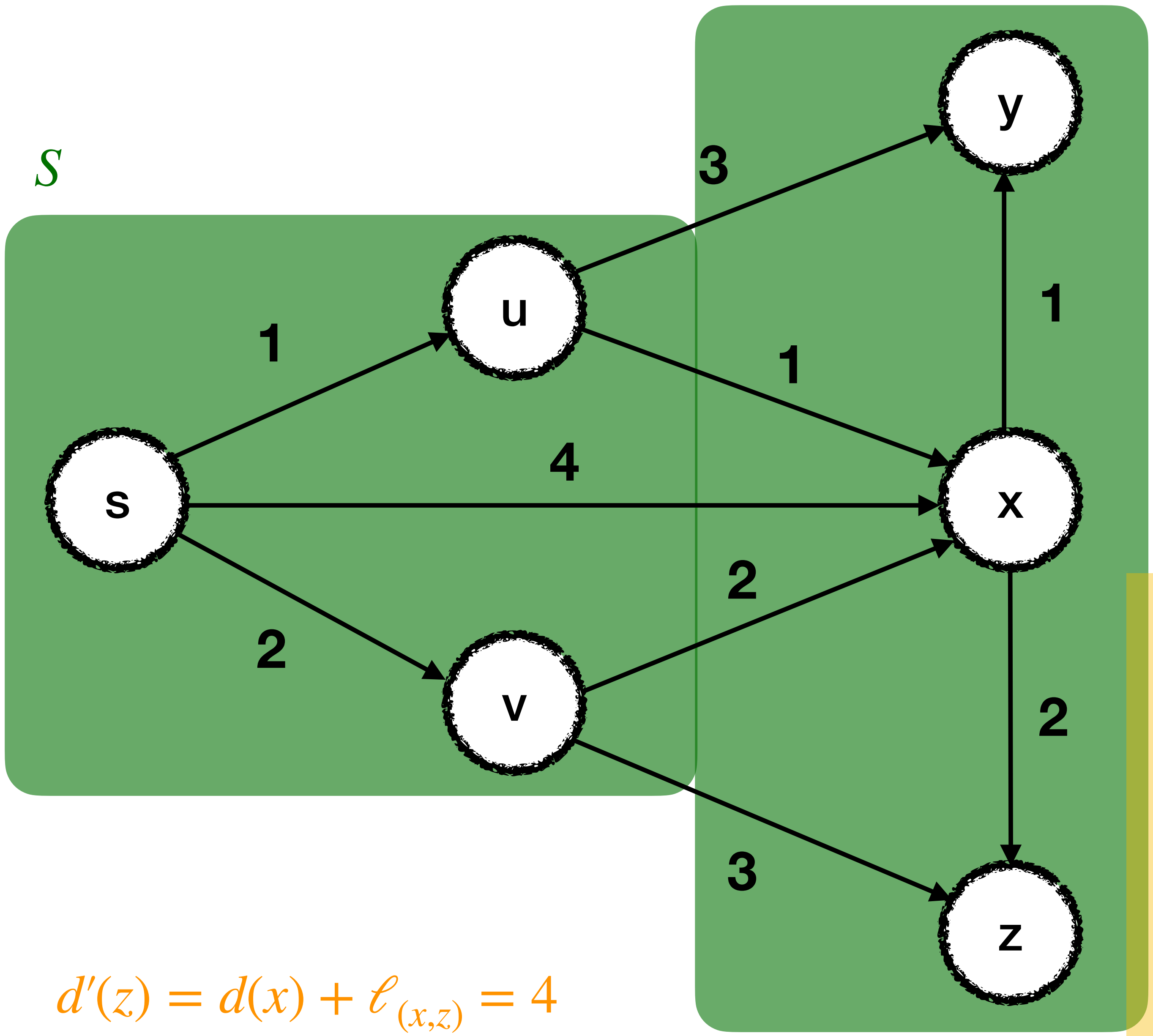
Select a node $v \in V - S$ connected via an edge with at least one node in S for which

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Running Example (KT Figure 5.7)



$d'(z) = d(x) + \ell_{(x,z)} = 4$

s	u	v	x	y	z
0	1	2	2	3	

$\underbrace{\hphantom{xxxxxxxx}}_{\text{shortest path distances from } s}$

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$

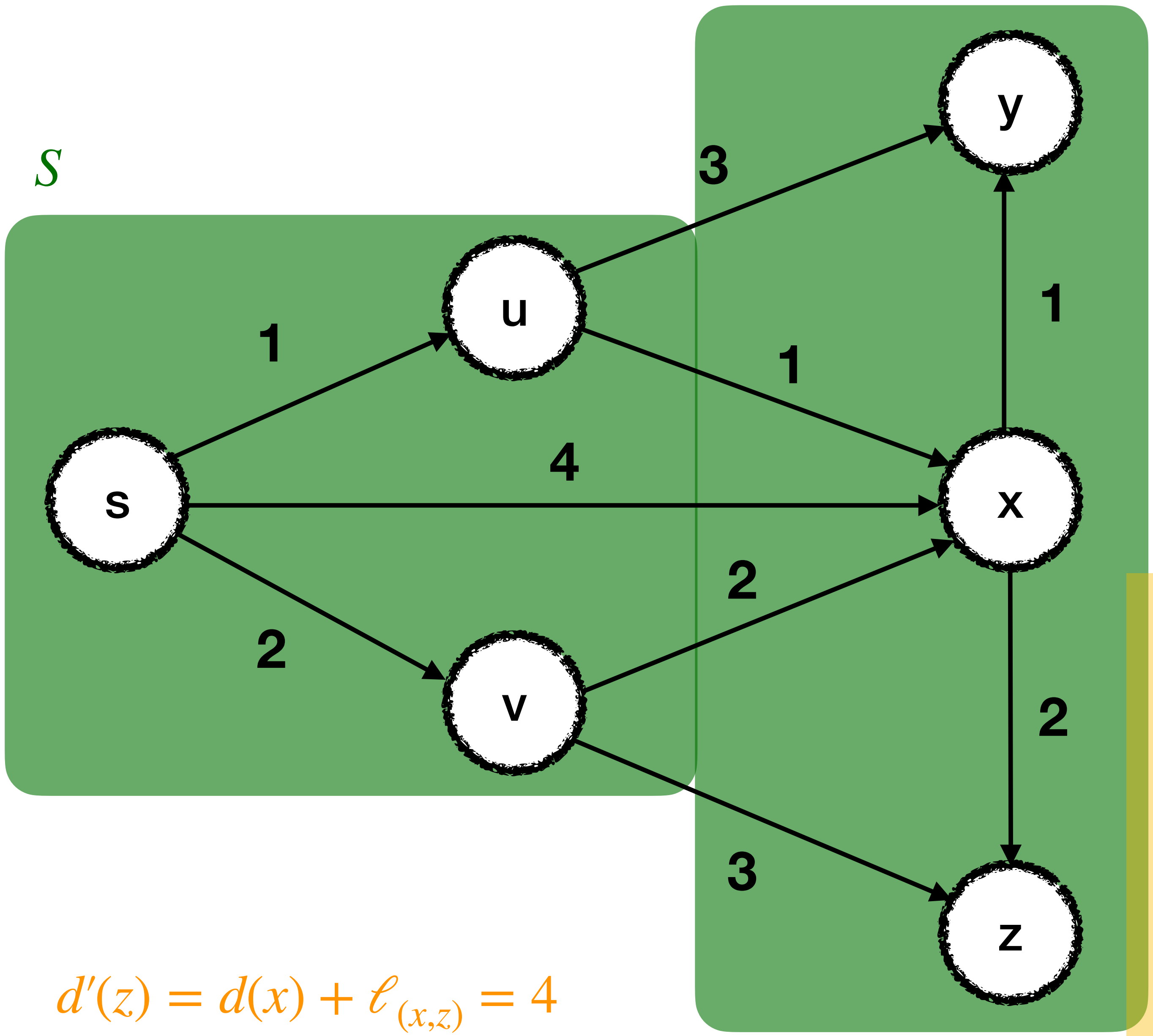
Select a node $v \in V - S$ connected via an edge with at least one node in S for which

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Running Example (KT Figure 5.7)



$d'(z) = d(x) + \ell_{(x,z)} = 4$

s	u	v	x	y	z
0	1	2	2	3	

$\underbrace{\hphantom{0\quad 1\quad 2\quad 2\quad 3\quad}}$

shortest path distances from s

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$ ✓ ✓ ✓

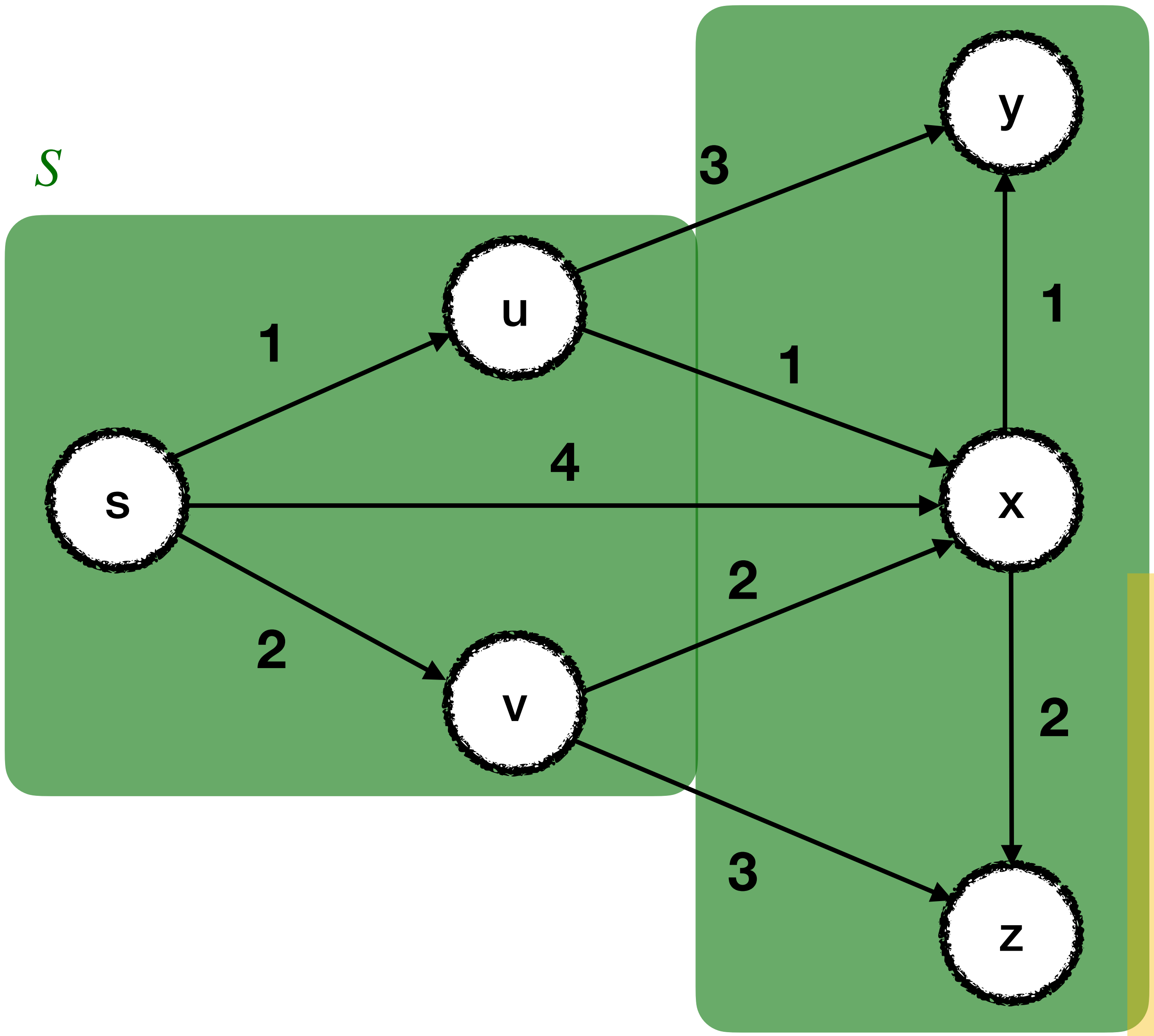
Select a node $v \in V - S$ connected via an edge with at least one node in S for which

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$ $\qquad A[z] = 4$

Running Example (KT Figure 5.7)



s	u	v	x	y	z
0	1	2	2	3	

shortest path distances from s

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$

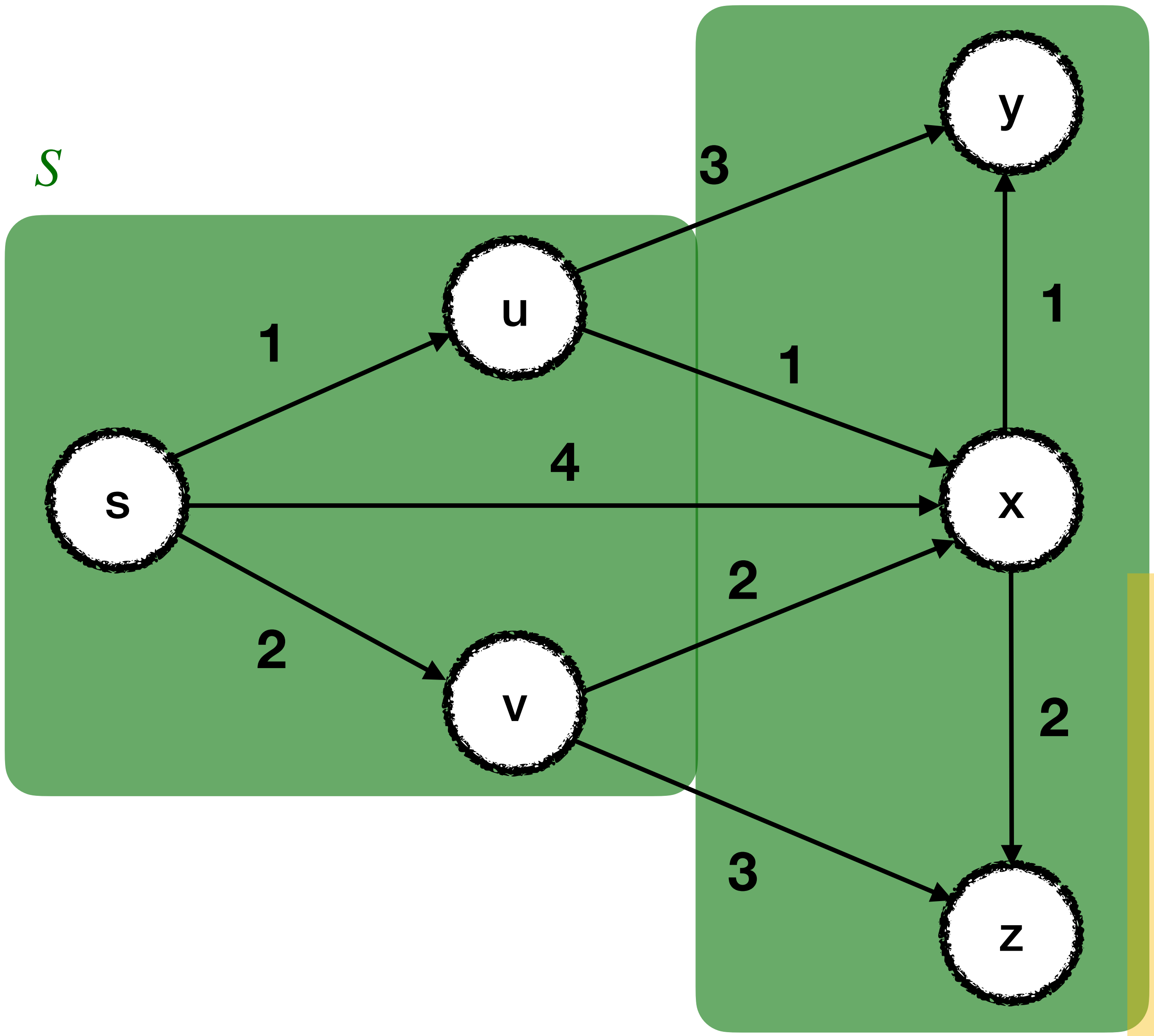
Select a node $v \in V - S$ connected via an edge with at least one node in S for which

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Running Example (KT Figure 5.7)



s	u	v	x	y	z
0	1	2	2	3	4

shortest path distances from s

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$

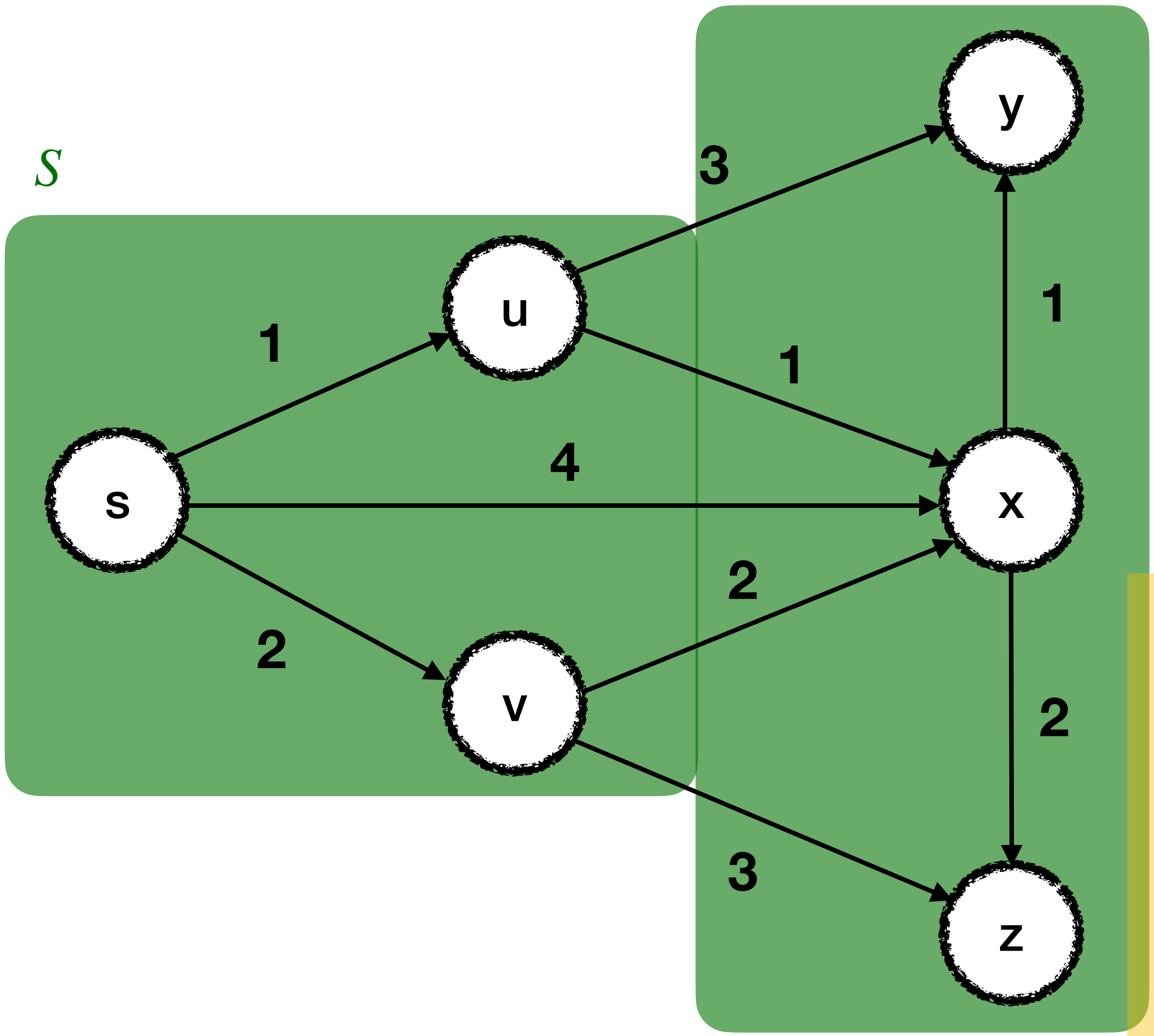
Select a node $v \in V - S$ connected via an edge with at least one node in S for which

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Running Example (KT Figure 5.7)



s	u	v	x	y	z
0	1	2	2	3	4

shortest path distances from s

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$

Select a node $v \in V - S$ connected via an edge with at least one node in S for which

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Are we done?

Are we done?

- **Output:** For every node u in V, a shortest path s~u from s to u.

Are we done?

- **Output:** For every node u in V, a shortest path $s{\sim}u$ from s to u.
  - To be more precise, a list of paths:

Are we done?

- **Output:** For every node u in V, a shortest path s~u from s to u.
  - To be more precise, a list of paths:

$P(s)$	$P(u_1)$	$P(u_2)$	$P(u_3)$	...

Are we done?

- **Output:** For every node u in V, a shortest path $s \sim u$ from s to u.

  - To be more precise, a list of paths:

$P(s)$	$P(u_1)$	$P(u_2)$	$P(u_3)$	...

s	u	v	x	y	z
0	1	2	2	3	4

Are we done?

- **Output:** For every node u in V, a shortest path s~u from s to u.
  - To be more precise, a list of paths:

$P(s)$	$P(u_1)$	$P(u_2)$	$P(u_3)$	...

s	u	v	x	y	z
0	1	2	2	3	4

This is only a list of shortest path lengths, not the paths themselves!

From lengths to paths

From lengths to paths

- When we add a node v to S, we record the edge (u,v) that led us to explore v.

From lengths to paths

- When we add a node v to S, we record the edge (u,v) that led us to explore v.
- This is enough to recursively recover the path P_v : P_v is just P_u + (u,v). In turn, P_u is P_w + (w,u), where w is the node from which we explored u, and so on.

Correctness

Theorem: Consider the set S at any point in the execution of the algorithm. For each $u \in S$, the path P_u is a shortest s-u path.

Correctness

Theorem: Consider the set S at any point in the execution of the algorithm. For each $u \in S$, the path P_u is a shortest s-u path.

This is enough to prove correctness. Why?

Correctness

Theorem: Consider the set S at any point in the execution of the algorithm. For each $u \in S$, the path P_u is a shortest s-u path.

Correctness

Theorem: Consider the set S at any point in the execution of the algorithm. For each $u \in S$, the path P_u is a shortest s-u path.

Proof by induction on the size of S.

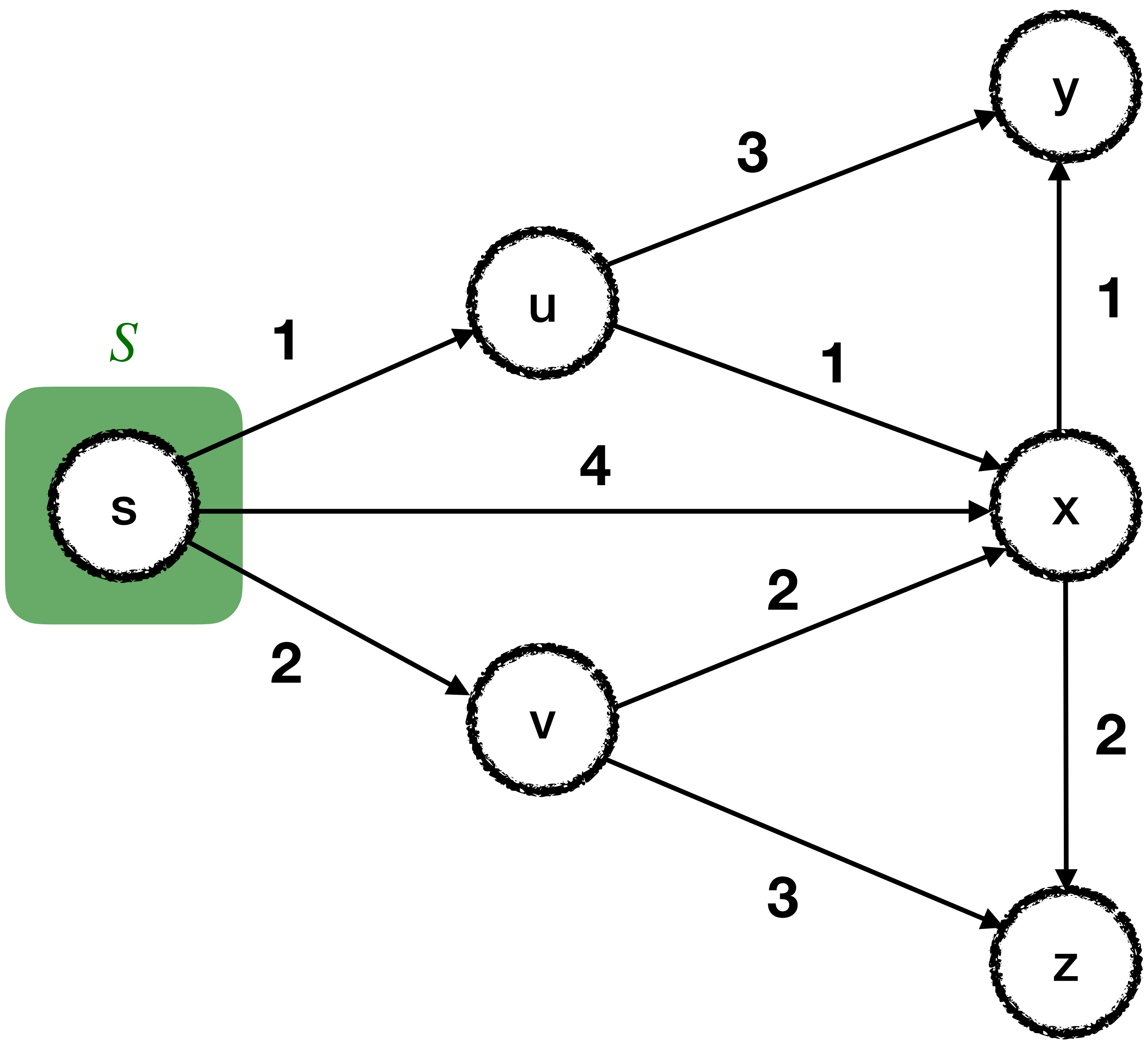
Correctness

Theorem: Consider the set S at any point in the execution of the algorithm. For each $u \in S$, the path P_u is a shortest s-u path.

Proof by induction on the size of S.

Base Case: $|S| = 1$

Running Example (KT Figure 5.7)



s	u	v	x	y	z
0					

$\underbrace{\hspace{6em}}$

shortest path distances from s

Correctness

Theorem: Consider the set S at any point in the execution of the algorithm. For each $u \in S$, the path P_u is a shortest s-u path.

Proof by induction on the size of S.

Base Case: $|S| = 1$, $S = \{s\}$, $d(s) = 0$, trivially shortest path.

Correctness

Theorem: Consider the set S at any point in the execution of the algorithm. For each $u \in S$, the path P_u is a shortest s-u path.

Proof by induction on the size of S.

Base Case: $|S| = 1$, $S = \{s\}, d(s) = 0$, trivially shortest path.

Induction Hypothesis: Assume that it holds for $|S| = k$ for some $k \geq 1$.

Correctness

Theorem: Consider the set S at any point in the execution of the algorithm. For each $u \in S$, the path P_u is a shortest s-u path.

Proof by induction on the size of S.

Base Case: $|S| = 1$, $S = \{s\}, d(s) = 0$, trivially shortest path.

Induction Hypothesis: Assume that it holds for $|S| = k$ for some $k \geq 1$.

Induction Step: We will prove that it holds for $|S| = k + 1$

Adding a node

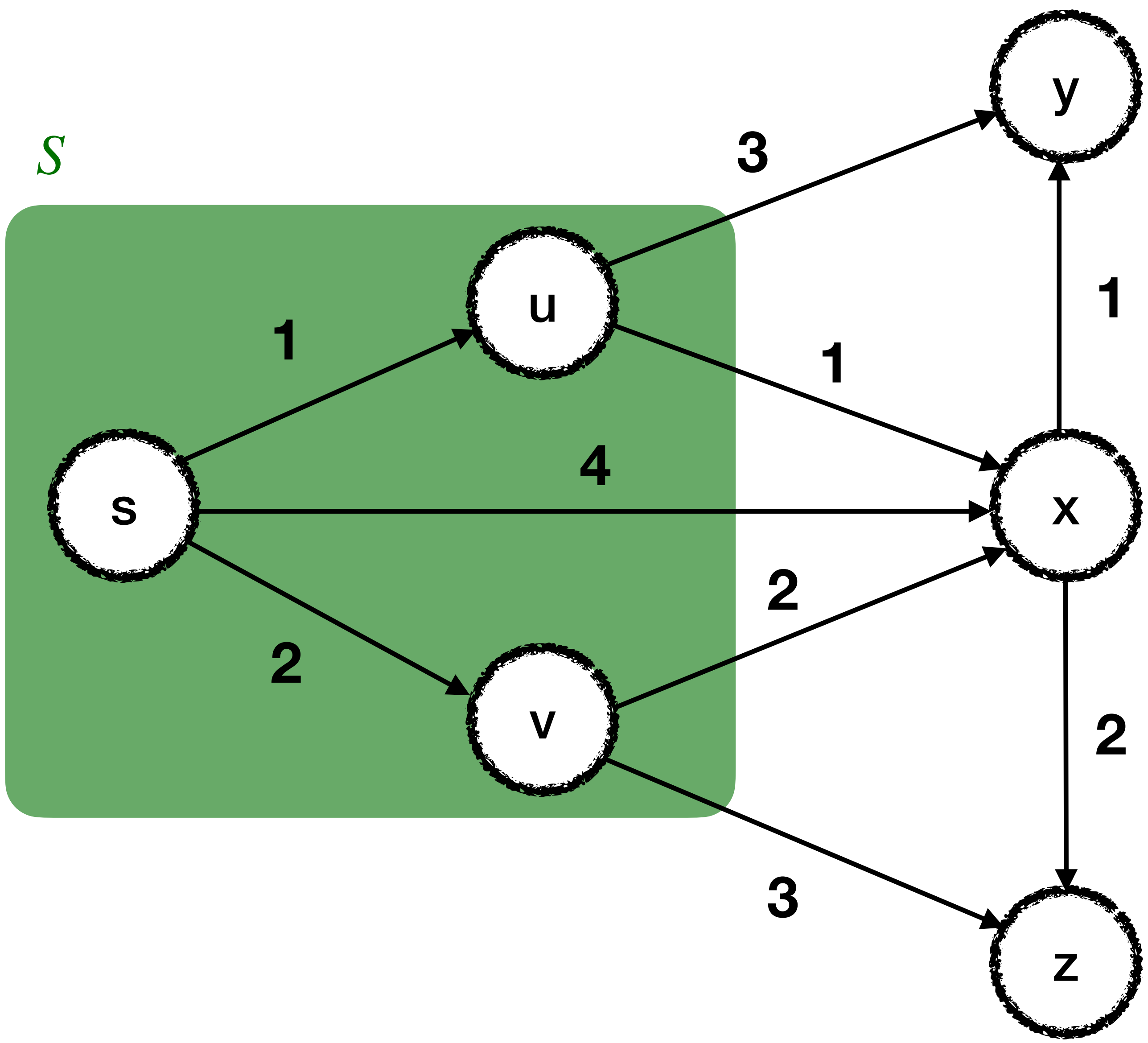
Induction Step: We will prove that it holds for $|S| = k + 1$

Adding a node

Induction Step: We will prove that it holds for $|S| = k + 1$

Let v^* be the node added to S. Let (u^*, v^*) be the final edge on the path P_{v^*}.

Running Example (KT Figure 5.7)

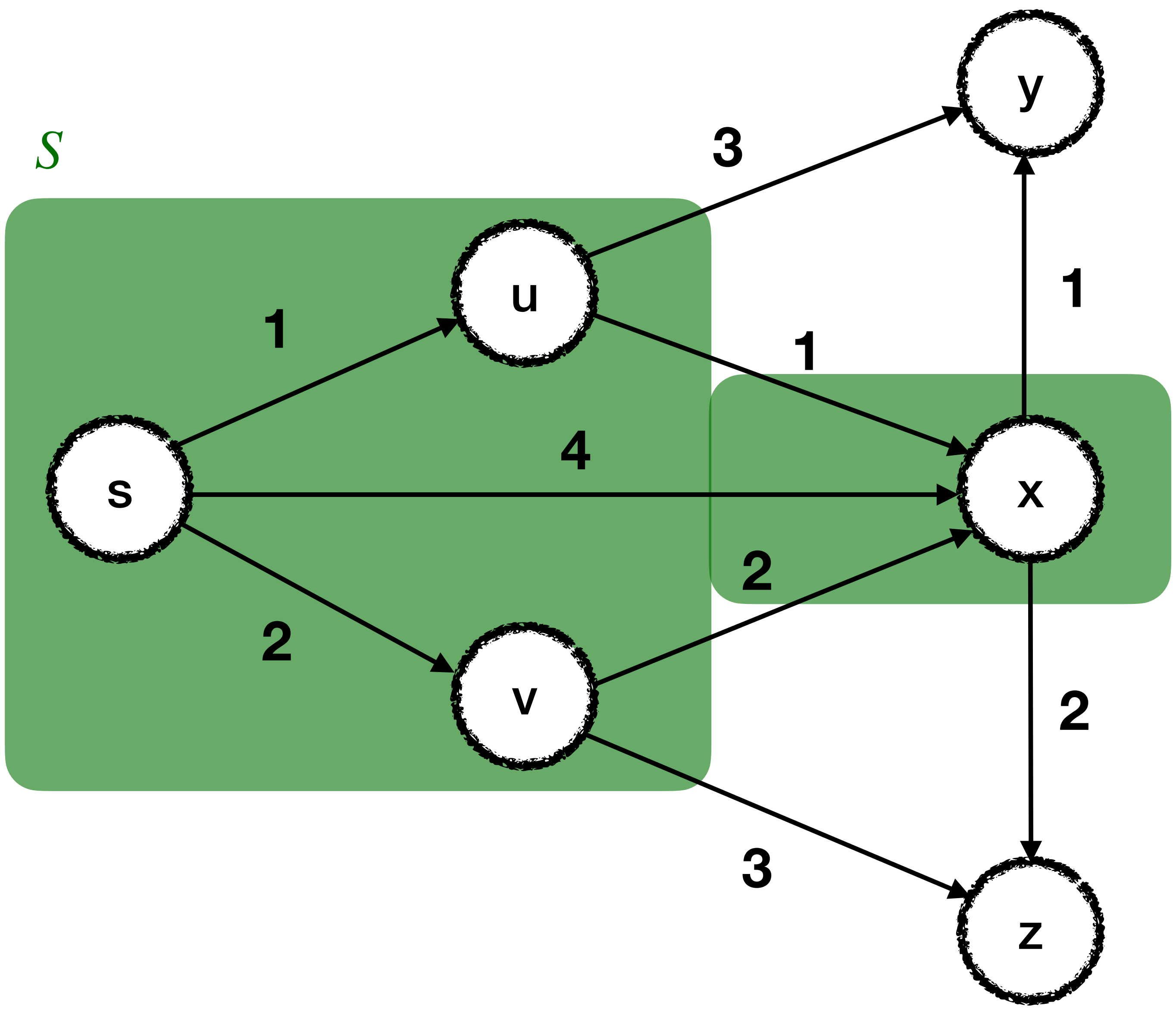


s	u	v	x	y	z
0	1	2			

shortest path distances from s

Let v^* be the node added to S. Let (u^*, v^*) be the final edge on the path P_{v^*}.

Running Example (KT Figure 5.7)



s	u	v	x	y	z
0	1	2			

shortest path distances from s

Let v^* be the node added to S. Let (u^*, v^*) be the final edge on the path P_{v^*}.

Adding a node

Induction Step: We will prove that it holds for $|S| = k + 1$

Let v^* be the node added to S. Let (u^*, v^*) be the final edge on the path P_{v^*}.

Adding a node

Induction Step: We will prove that it holds for $|S| = k + 1$

Let v^* be the node added to S. Let (u^*, v^*) be the final edge on the path P_{v^*}.

Assume by contradiction that P_{v^*} is *not* a shortest s-v^* path.

Adding a node

Induction Step: We will prove that it holds for $|S| = k + 1$

Let v^* be the node added to S. Let (u^*, v^*) be the final edge on the path P_{v^*}.

Assume by contradiction that P_{v^*} is *not* a shortest s-v^* path.

That means that there exists some other path P that is shorter than P_{v^*}.

Adding a node

Assume by contradiction that P_{v*} is *not* a shortest s-$v*$ path.

That means that there exists some other path P that is shorter than P_{v*}.

Adding a node

Assume by contradiction that P_{v*} is *not* a shortest s-$v*$ path.

That means that there exists some other path P that is shorter than P_{v*}.

Fact: P starts in S and must leave S at some point. Why?

Adding a node

Assume by contradiction that P_{v*} is *not* a shortest s-$v*$ path.

That means that there exists some other path P that is shorter than P_{v*}.

Fact: P starts in S and must leave S at some point. Why?

Let $y*$ be the first node of P that is not in S.

Adding a node

Assume by contradiction that P_{v*} is *not* a shortest s-$v*$ path.

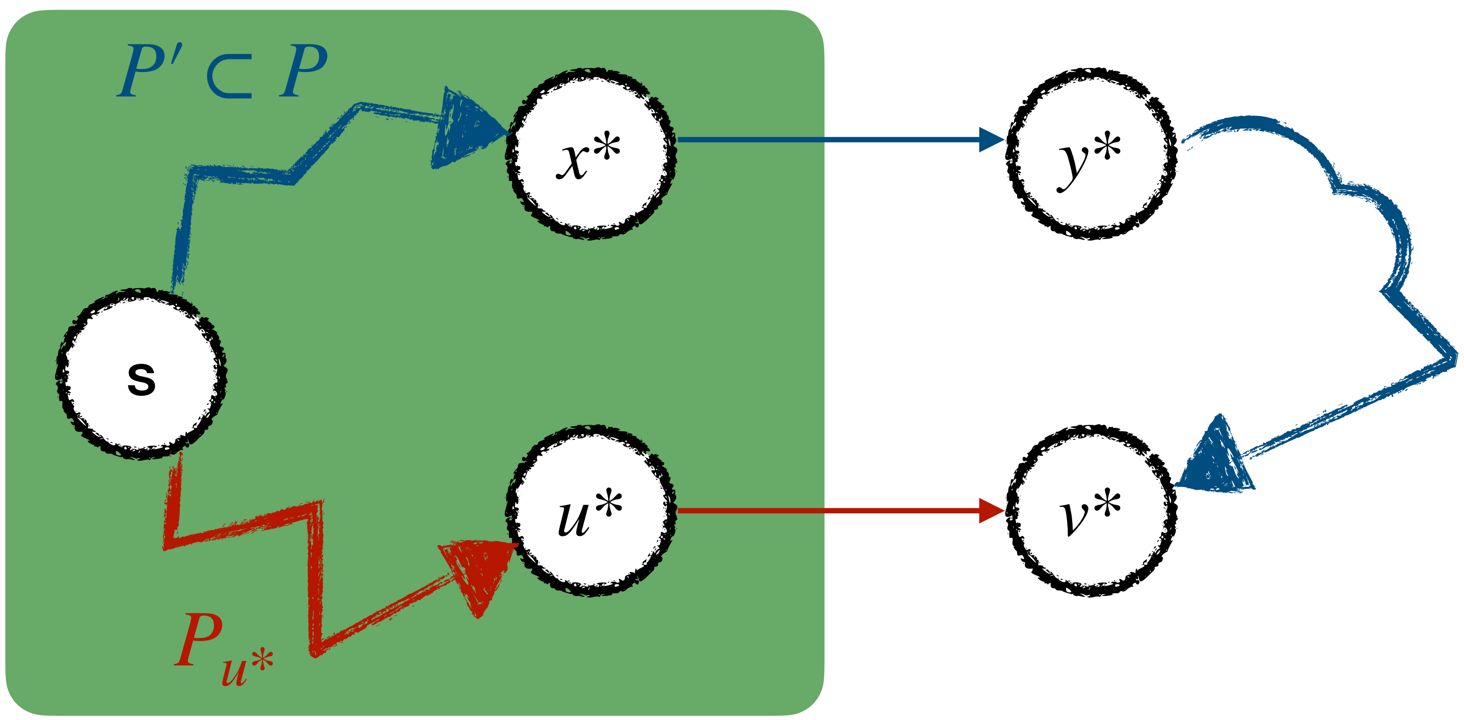
That means that there exists some other path P that is shorter than P_{v*}.

Fact: P starts in S and must leave S at some point. Why?

Let $y*$ be the first node of P that is not in S.

Let $x*$ be the node “just before” $y*$, i.e., the last node of P before it leaves S.

Pictorially

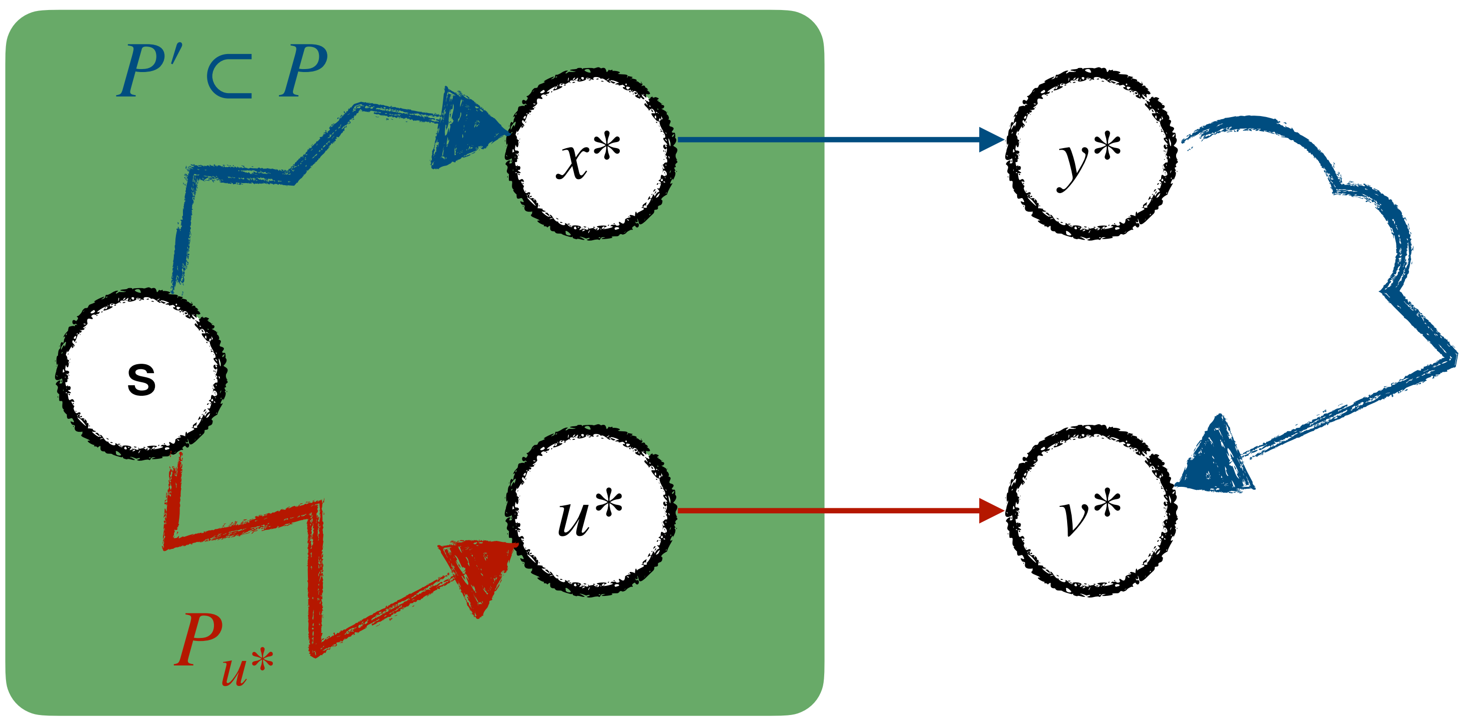


Let v^* be the node added to S. Let (u^*, v^*) be the final edge on the path P_{v^*}.

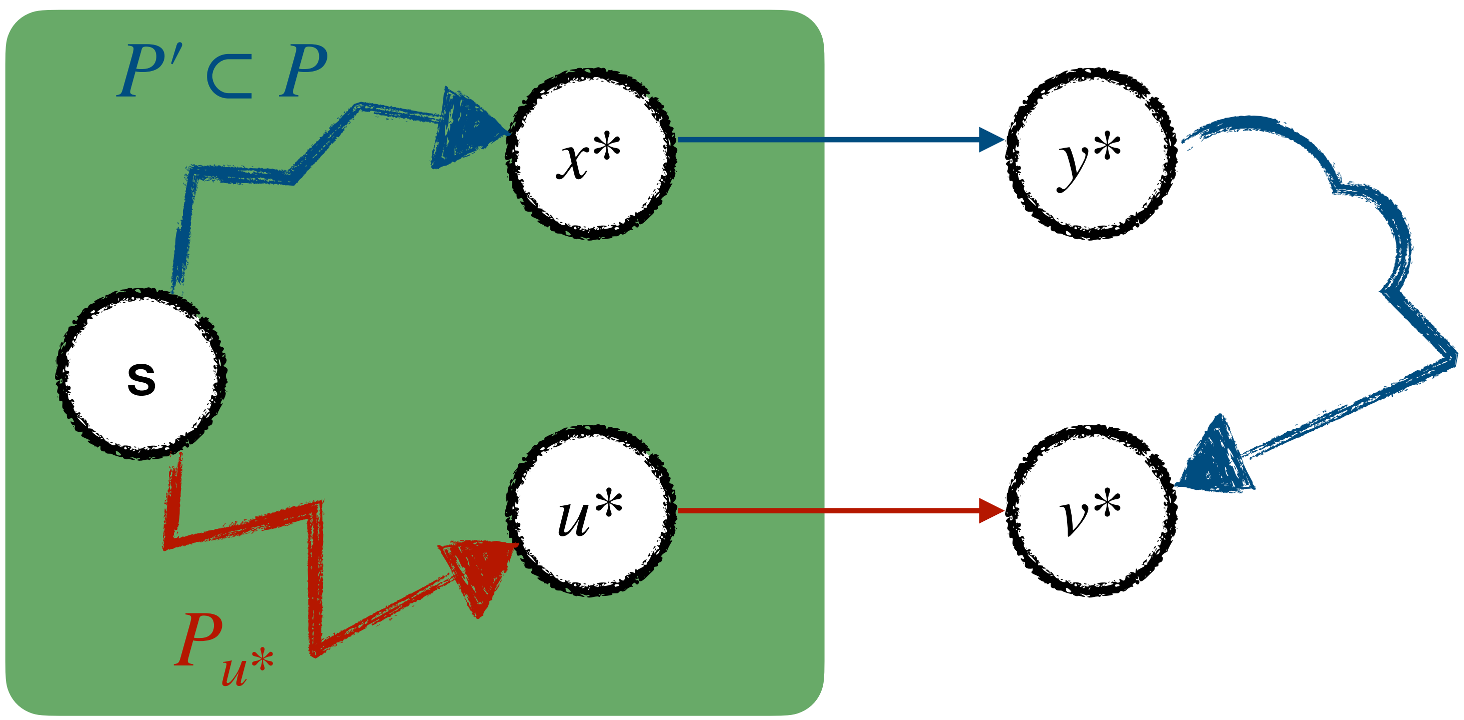
Let y^* be the first node of P that is not in S.

Let x^* be the node "just before" y^*, i.e., the last node of P before it leaves S.

Correctness

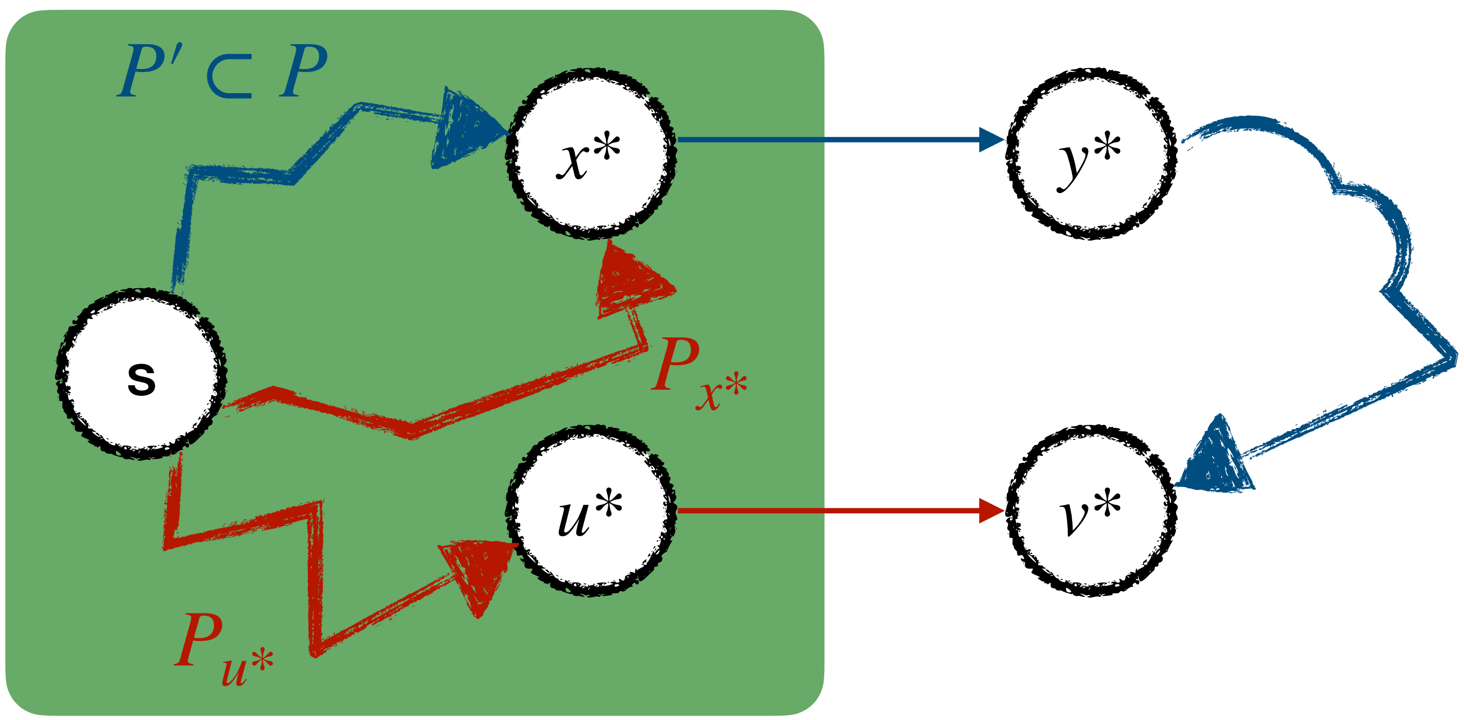


Correctness



We know that the s-x^* path P_{x^*} is a shortest path. Why?

Correctness



We know that the s-x^* path P_{x^*} is a shortest path. Why?

Correctness

Theorem: Consider the set S at any point in the execution of the algorithm. For each $u \in S$, the path P_u is a shortest s-u path.

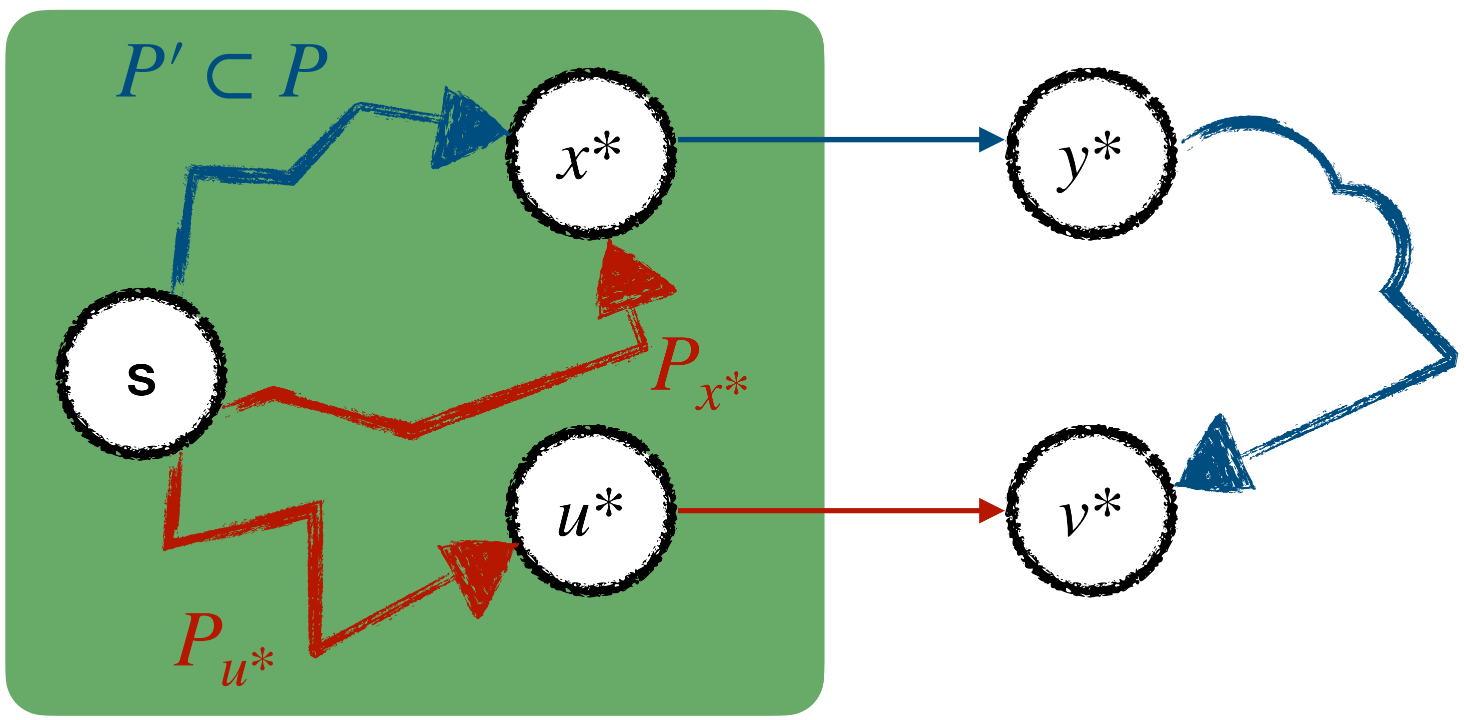
Proof by induction on the size of S.

Base Case: $|S| = 1$, $S = \{s\}, d(s) = 0$, trivially shortest path.

Induction Hypothesis: Assume that it holds for $|S| = k$ for some $k \geq 1$.

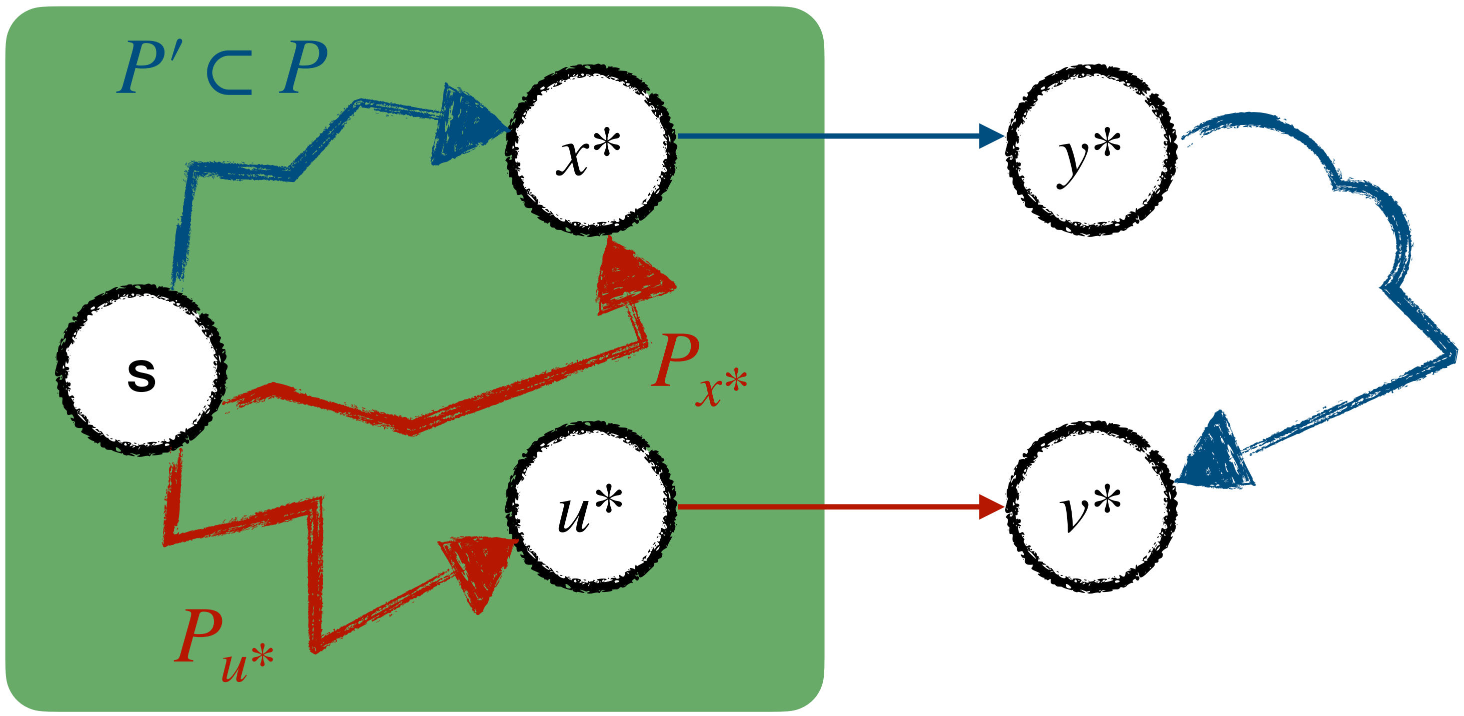
Induction Step: We will prove that it holds for $|S| = k + 1$

Correctness



We now that the s-x^* path P_{x^*} is a shortest path. Why?
By the induction hypothesis.

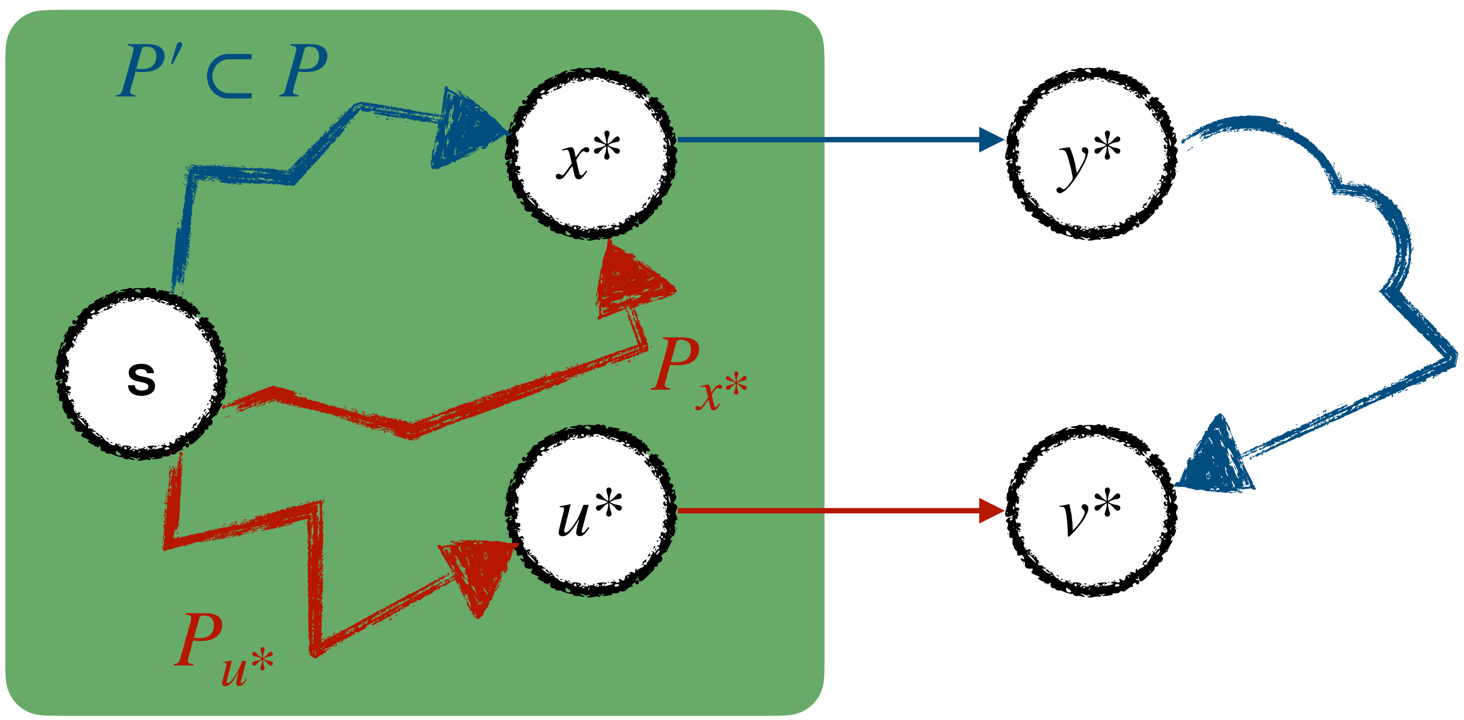
Correctness



We now that the s-x^* path P_{x^*} is a shortest path. Why?
By the induction hypothesis.

This means that $\ell(P') \geq \ell(P_{x^*}) = d(x^*)$

Correctness

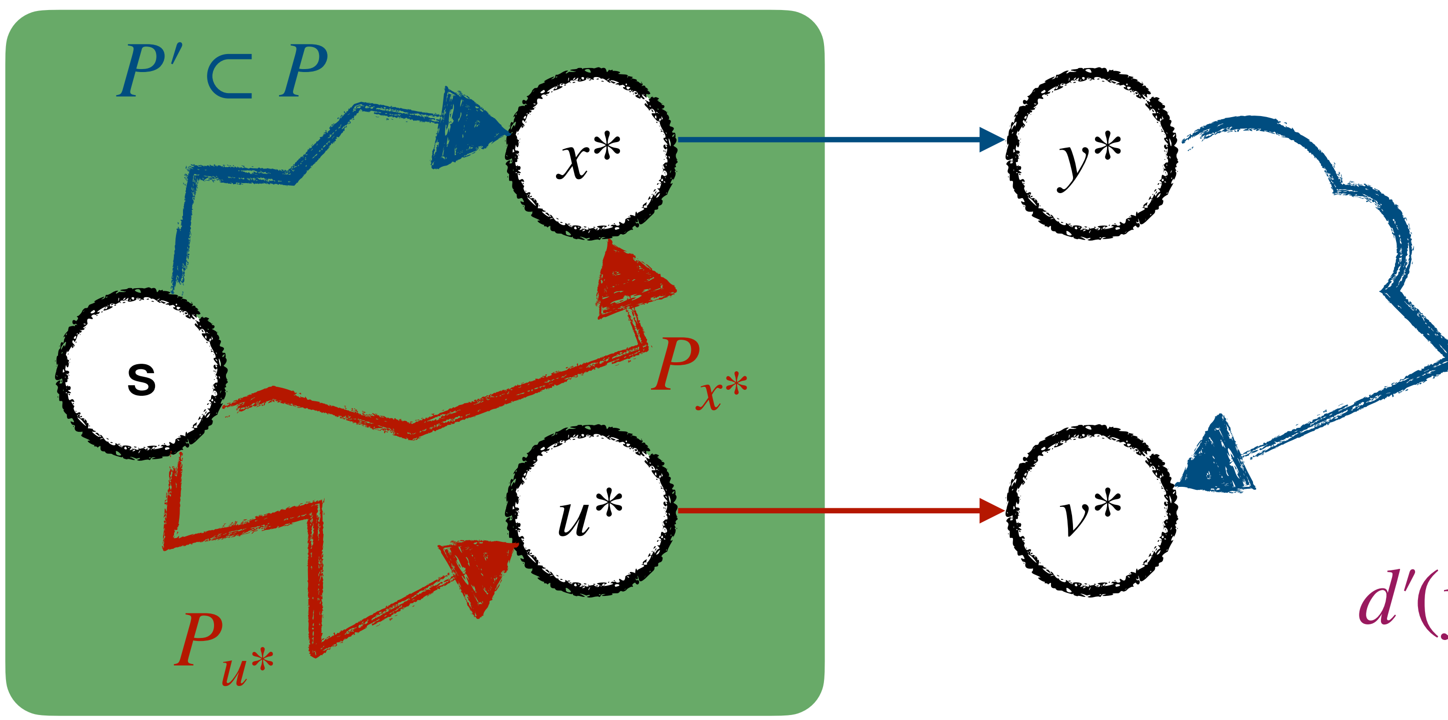


We now that the s-x^* path P_{x^*} is a shortest path. Why?
By the induction hypothesis.

This means that $\ell(P') \geq \ell(P_{x^*}) = d(x^*)$

Therefore $\ell(P) \geq \ell(P') + \ell(x^*, y^*) \geq \ell(x^*, y^*) + d(x^*) \geq d'(y^*)$

Correctness



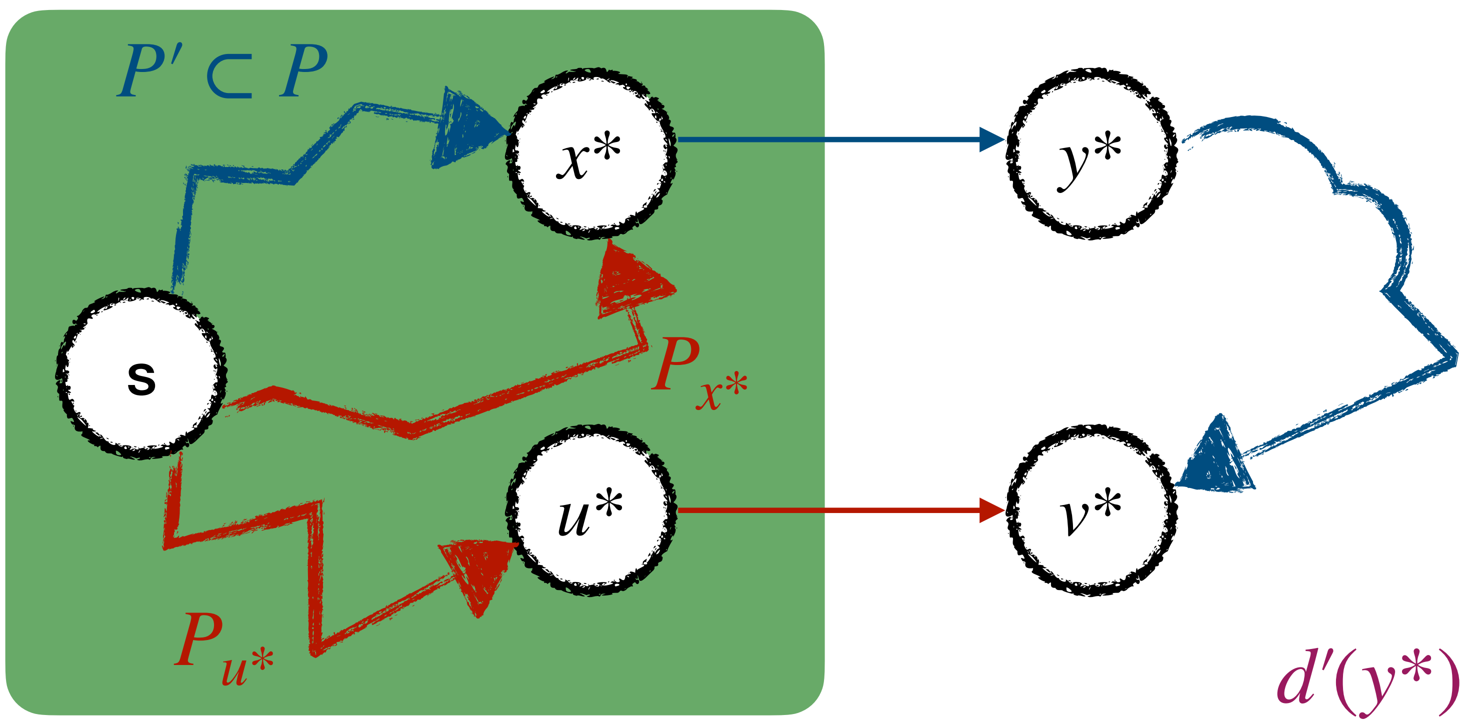
$$d'(y^*) = \min_{e=(u,y^*):u \in S} d(u) + \ell_e$$

We now that the s-x^* path P_{x^*} is a shortest path. Why?
By the induction hypothesis.

This means that $\ell(P') \geq \ell (P_{x^*}) = d(x^*)$

Therefore $\ell(P) \geq \ell (P') + \ell(x^*, y^*) \geq \ell(x^*, y^*) + d(x^*) \geq d'(y^*)$

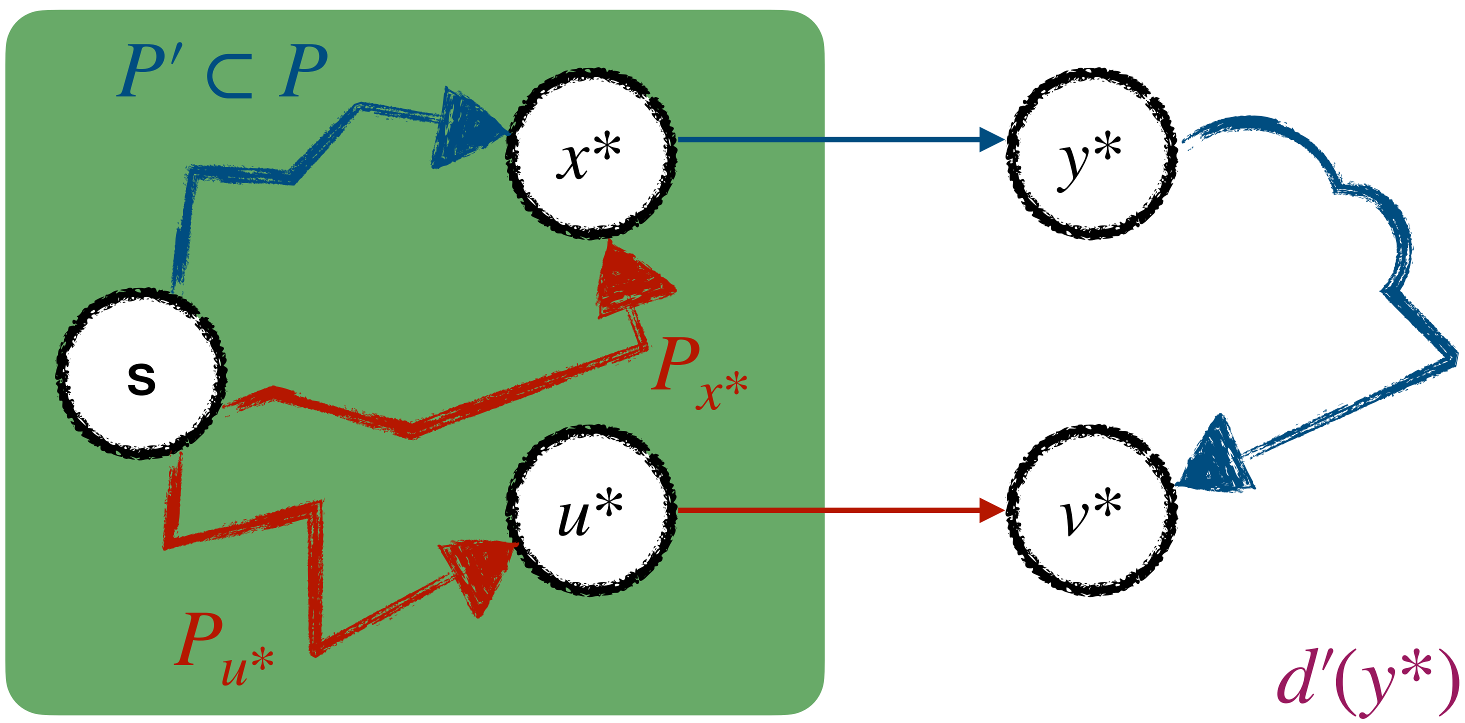
Correctness



$$d'(y^*) = \min_{e=(u,y^*):u\in S} d(u) + \ell_e$$

Therefore $\ell(P) \geq d'(y^*)$

Correctness



$$d'(y^*) = \min_{e=(u,y^*):u \in S} d(u) + \ell_e$$

Therefore $\ell(\,P\,) \geq d'(\,y^*\,)$

At the same time, we know that $d'(\,y^*\,) \geq d'(\,v^*\,) = \ell(\,P_{v^*}\,)$.
Why?

Dijkstra's Algorithm (Pseudocode)

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$

Select a node $v \in V - S$ connected via an edge with at least one node in S such that

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Dijkstra's Algorithm (Pseudocode)

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$

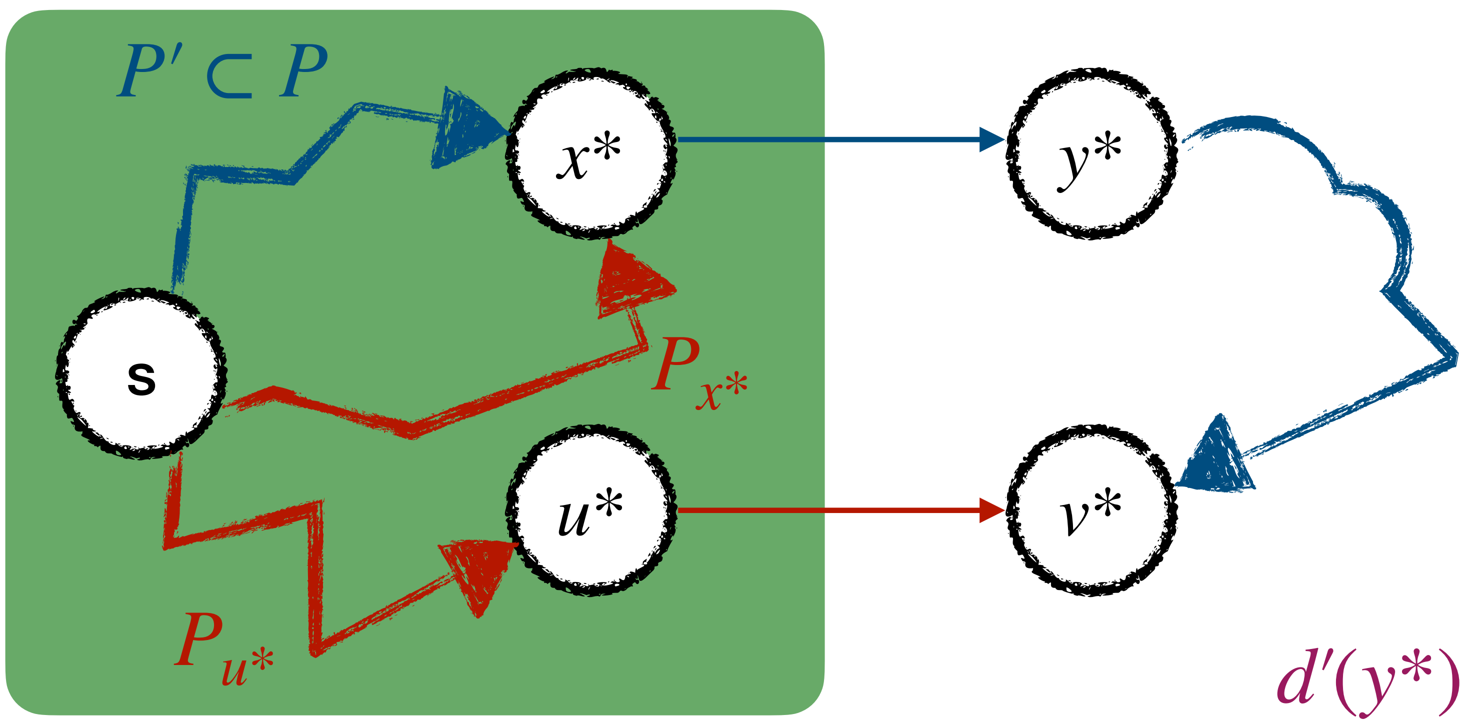
Select a node $v \in V - S$ connected via an edge with at least one node in S such that

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Correctness



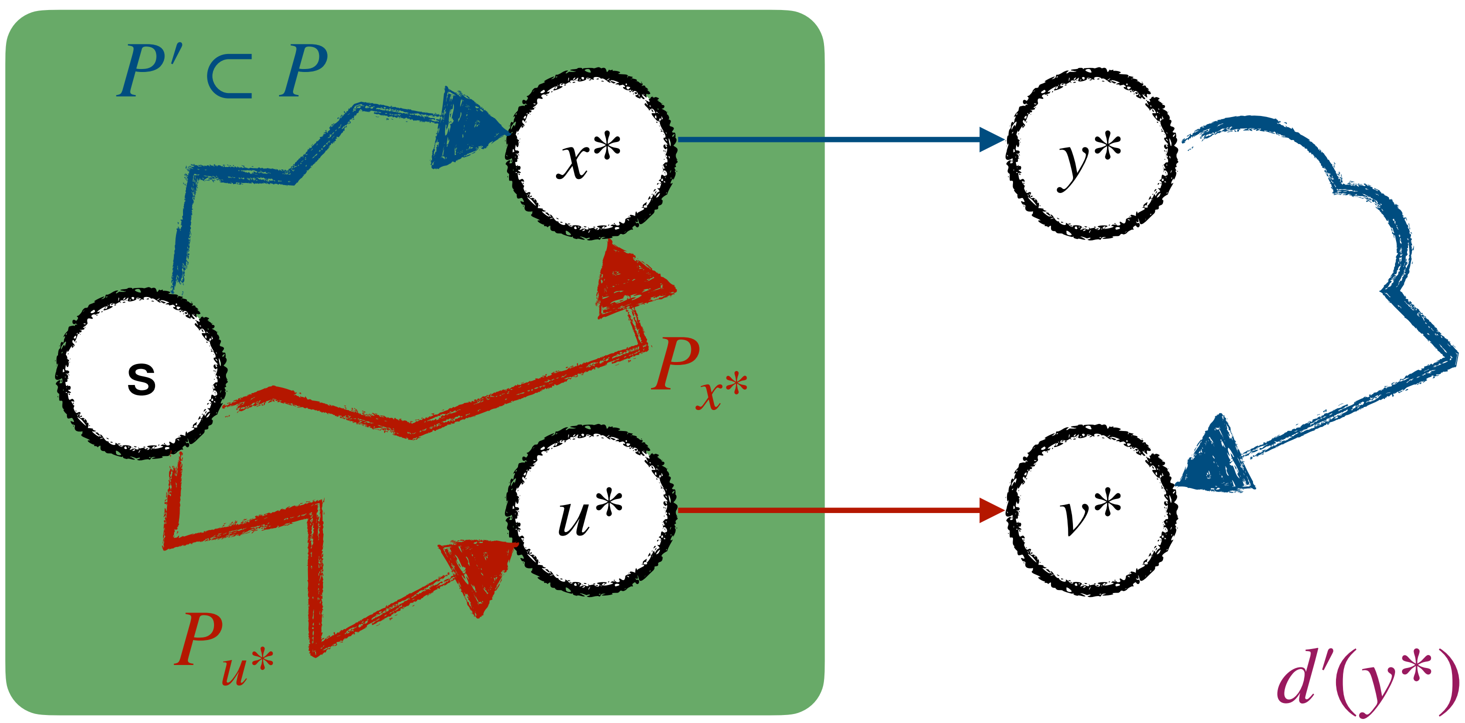
$$d'(y^*) = \min_{e=(u,y^*):u \in S} d(u) + \ell_e$$

Therefore $\ell(\ P\ ) \geq d'(\ y^*\ )$

At the same time, we know that $d'(\ y^*\ ) \geq d'(\ v^*\ ) = \ell(\ P_{v^*}\ )$.

Why? Because v^* was chosen by Dijkstra's Algorithm.

Putting it together



$$d'(y^*) = \min_{e=(u,y^*):u\in S} d(u) + \ell_e$$

Therefore $\ell(\,P\,) \geq d'(\,y^*\,)$

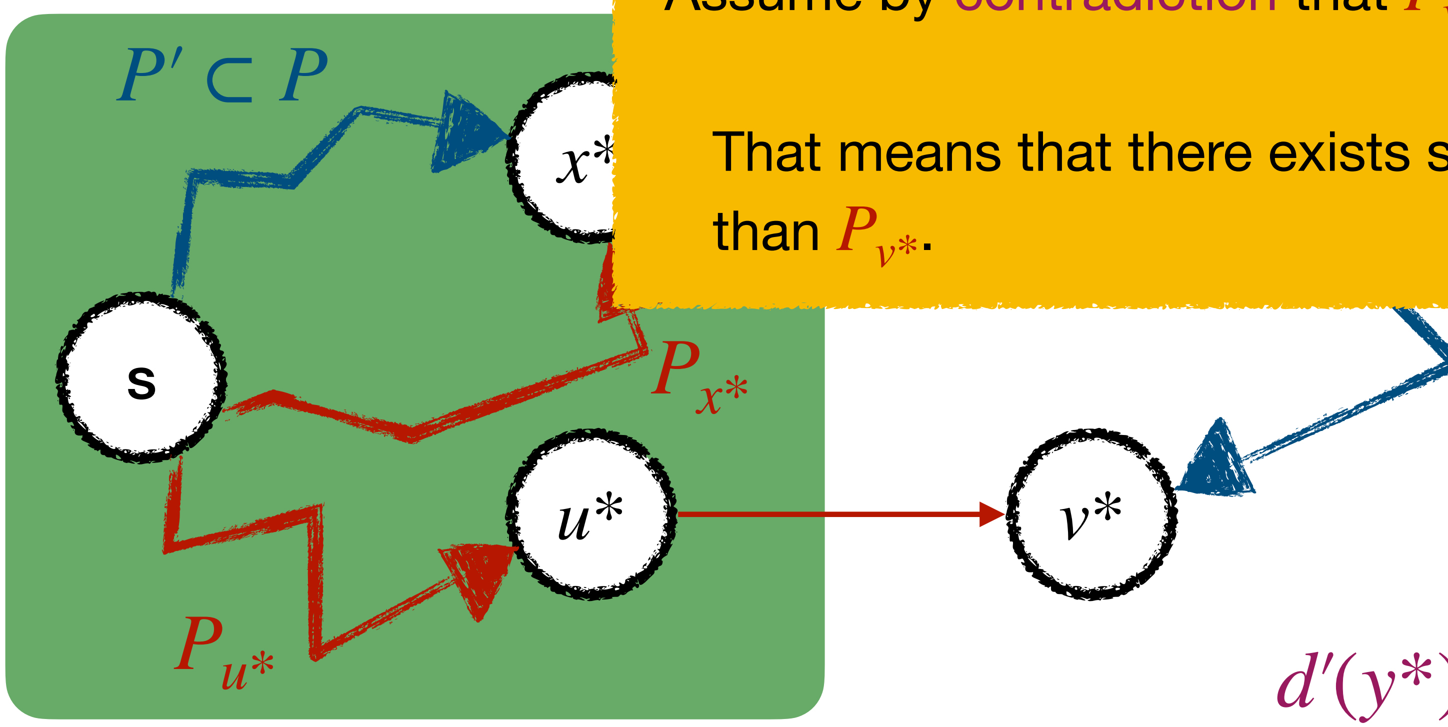
At the same time, we know that $d'(\,y^*\,) \geq \ell(\,P_{v^*}\,)$.

That implies that $\ell(\,P\,) \geq \ell(\,P_{v^*}\,)$, a contradiction.

Putting it together

Assume by contradiction that P_{v*} is *not* a shortest s-$v*$ path.

That means that there exists some other path P that is shorter than P_{v*}.



$$d'(y*) = \min_{e=(u,y*):u\in S} d(u) + \ell_e$$

Therefore $\ell(P) \geq d'(y*)$

At the same time, we know that $d'(y*) \geq \ell(P_{v*})$.

That implies that $\ell(P) \geq \ell(P_{v*})$, a contradiction.

Running Time

Lets look at the pseudocode.

Dijkstra's Algorithm (Pseudocode)

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$

Select a node $v \in V - S$ connected via an edge with at least one node in S such that

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Dijkstra's Algorithm (Pseudocode)

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$ **How many iterations here?**

Select a node $v \in V - S$ connected via an edge with at least one node in S such that

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Dijkstra's Algorithm (Pseudocode)

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$ **How many iterations here?** $|V| - 1 = n - 1$

Select a node $v \in V - S$ connected via an edge with at least one node in S such that

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Dijkstra's Algorithm (Pseudocode)

<u>**Dijsktra (G, ℓ)**</u>

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$ **How many iterations here?** $|V| - 1 = n - 1$

Select a node $v \in V - S$ connected via an edge with at least one node in S such that

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

Here, consider every node v outside S, and then consider all edges between S and v.

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Dijkstra's Algorithm (Pseudocode)

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$ **How many iterations here?** $|V| - 1 = n - 1$

Select a node $v \in V - S$ connected via an edge with at least one node in S such that

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Here, consider every node v outside S, and then consider all edges between S and v.

$|E| = m$

Dijkstra's Algorithm (Pseudocode)

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$ **How many iterations here?** $|V| - 1 = n - 1$

Select a node $v \in V - S$ connected via an edge with at least one node in S such that

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

Add v to S and define $d(v) = d'(v)$

Let $A[v] = d(v)$

Here, consider every node v outside S, and then consider all edges between S and v.

$|E| = m$

Overall: $O(nm)$.

Dijkstra's Algorithm (Pseudocode)

Dijsktra (G, ℓ)

Let S be the set of explored nodes, A be a list of distances.

Initially $S = \{s\}$ and $d(s) = 0$, $A[s] = d(s) = 0$

While $S \neq V$ **How many iterations here?** $|V| - 1 = n - 1$

Select a node $v \in V - S$ connected via an edge with at least one node in S such that

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

Here, consider every node v outside S, and then consider all edges between S and v.

Add v to S and define $d(v) = d'(v)$

$|E| = m$

Let $A[v] = d(v)$

Overall: $O(nm)$. **Not terrible, not great.**

Running Time

Lets look at the pseudocode.

That was somewhat naive. Can we do better?

Priority Queues

Tue 31 Oct	Lecture 12	Aris	**Lecture 12 - Heap operations and Priority Queues** **Lecture 12 - Heap operations and Priority Queues (.key file, better animations)** Recording from previous years (caution: different lecturer, different slides, presents all heap operations in Lecture 11 and Heapsort in Lecture 12).	CLRS 6.5 KT 2.5 (uses min-heap) Roughgarden 10.2, 10.5 (caution: RG uses min-heap and the terms "Heapify" for Build-Max-Heap and implements Min-Heapify as part of "ExtractMin").

- The values denote *priorities*.

- For Max-Priority Queues, the elements with the largest values are those with the highest priority.

Priority Queues

- Priority queue: A data structure that maintains
  - A set of elements S.
  - Each with an associated value, key(v).
  - The values denote *priorities*.
    - For Max-Priority Queues, the elements with the largest values are those with the highest priority.

Priority Queues

- Priority queue: A data structure that maintains
  - A set of elements S.
  - Each with an associated value, key(v).
  - The values denote *priorities*.
    - For Min-Priority Queues, the elements with the smallest values are those with the highest priority.

Priority Queue Operations

- Insert(Q, v) inserts a new item v in the priority queue.
- FindMin(Q) finds the element with the maximum priority (the smallest value) in the priority queue and returns it (but does not remove it).
- ExtractMin(Q) finds the element with the maximum priority (smallest value) in the priority queue, returns it, and deletes it from the queue.

Priority Queue Operations

- ExtractMin(Q) finds the element with the maximum priority (smallest value) in the priority queue, returns it, and deletes it from the queue.

- ChangeKey(Q, v, a) changes the key value of element v to key(v) $= a$.

Running Time

Lets look at the pseudocode.

That was somewhat naive. Can we do better?

Running Time

Lets look at the pseudocode.

That was somewhat naive. Can we do better?

1. Once we have computed $d'(v) = \min_{(u,v):u \in S} d(u) + \ell_e$, we store it, so we don't have to compute it again.

Running Time

Lets look at the pseudocode.

That was somewhat naive. Can we do better?

1. Once we have computed $d'(v) = \min_{(u,v):u\in S} d(u) + \ell_e$, we store it, so we don't have to compute it again.

2. Place the nodes in a priority queue, with $d'(v)$ as the key of v.

Running Time

1. Once we have computed $d'(v) = \min_{(u,v):u \in S} d(u) + \ell_e$, we store it, so we don't have to compute it again.

2. Place the nodes in a priority queue, with $d'(v)$ as the key of v.

Running Time

1. Once we have computed $d'(v) = \min_{(u,v):u \in S} d(u) + \ell_e$, we store it, so we don't have to compute it again.

2. Place the nodes in a priority queue, with $d'(v)$ as the key of v.

How do we get the node v to be added to S?

Running Time

1. Once we have computed $d'(v) = \min_{(u,v):u \in S} d(u) + \ell_e$, we store it, so we don't have to compute it again.

2. Place the nodes in a priority queue, with $d'(v)$ as the key of v.

How do we get the node v to be added to S?

ExtractMin(Q)

Running Time

1. Once we have computed $d'(v) = \min_{(u,v):u \in S} d(u) + \ell_e$, we store it, so we don't have to compute it again.

2. Place the nodes in a priority queue, with $d'(v)$ as the key of v.

How do we get the node v to be added to S?

ExtractMin(Q)

Issue: Once we add nodes to S, we need to update the values $d'(v)$, i.e., the keys in the priority queue.

Running Time

Issue: Once we add nodes to S, we need to update the values $d'(v)$, i.e., the keys in the priority queue.

Running Time

Issue: Once we add nodes to S, we need to update the values $d'(v)$, i.e., the keys in the priority queue.

Consider an iteration when v is added to S.

Running Time

Issue: Once we add nodes to S, we need to update the values $d'(v)$, i.e., the keys in the priority queue.

Consider an iteration when v is added to S.

Let $w \in V - S$ be a node that remains in the priority queue.

Running Time

Issue: Once we add nodes to S, we need to update the values $d'(v)$, i.e., the keys in the priority queue.

Consider an iteration when v is added to S.

Let $w \in V - S$ be a node that remains in the priority queue.

Case 1: $(v, w) \notin E$.

Running Time

Issue: Once we add nodes to S, we need to update the values $d'(v)$, i.e., the keys in the priority queue.

Consider an iteration when v is added to S.

Let $w \in V - S$ be a node that remains in the priority queue.

Case 1: $(v, w) \notin E$.

$\min_{e=(u,w):u \in S} d(u) + \ell_e$ has not changed!

Running Time

Issue: Once we add nodes to S, we need to update the values $d'(v)$, i.e., the keys in the priority queue.

Consider an iteration when v is added to S.

Let $w \in V - S$ be a node that remains in the priority queue.

Case 1: $(v, w) \notin E$.

$$\min_{e=(u,w):u \in S} d(u) + \ell_e \text{ has not changed!}$$

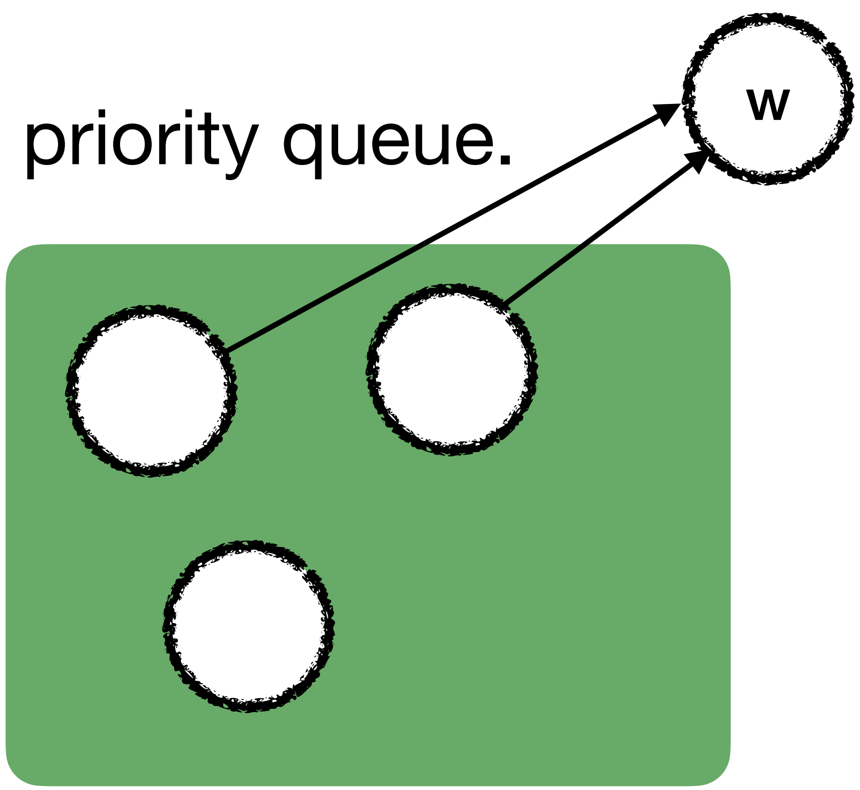
(this is because when we explore w in the future, it will not be via v).

Running Time

Issue: Once we add nodes to S, we need to update the values $d'(v)$, i.e., the keys in the priority queue.

Consider an iteration when v is added to S.

Let $w \in V - S$ be a node that remains in the priority queue.



Case 1: $(v, w) \notin E$.

$$\min_{e=(u,w):u \in S} d(u) + \ell_e$$ has not changed!

(this is because when we explore w in the future, it will not be via v).

Running Time

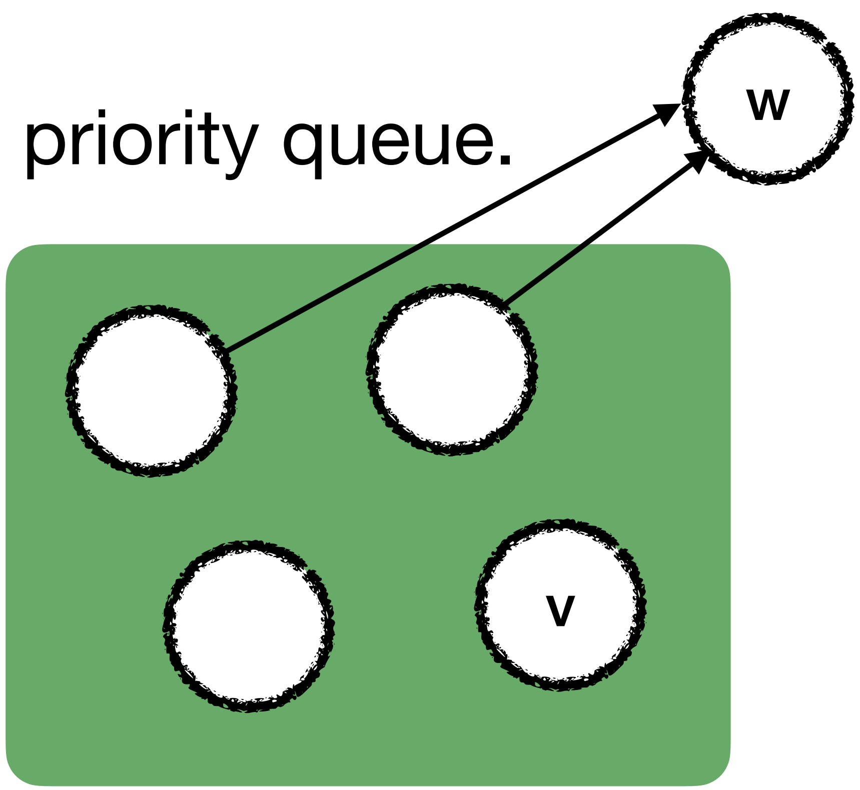
Issue: Once we add nodes to S, we need to update the values $d'(v)$, i.e., the keys in the priority queue.

Consider an iteration when v is added to S.

Let $w \in V - S$ be a node that remains in the priority queue.

Case 1: $(v, w) \notin E$.

$$\min_{e=(u,w):u\in S} d(u) + \ell_e$$ has not changed!



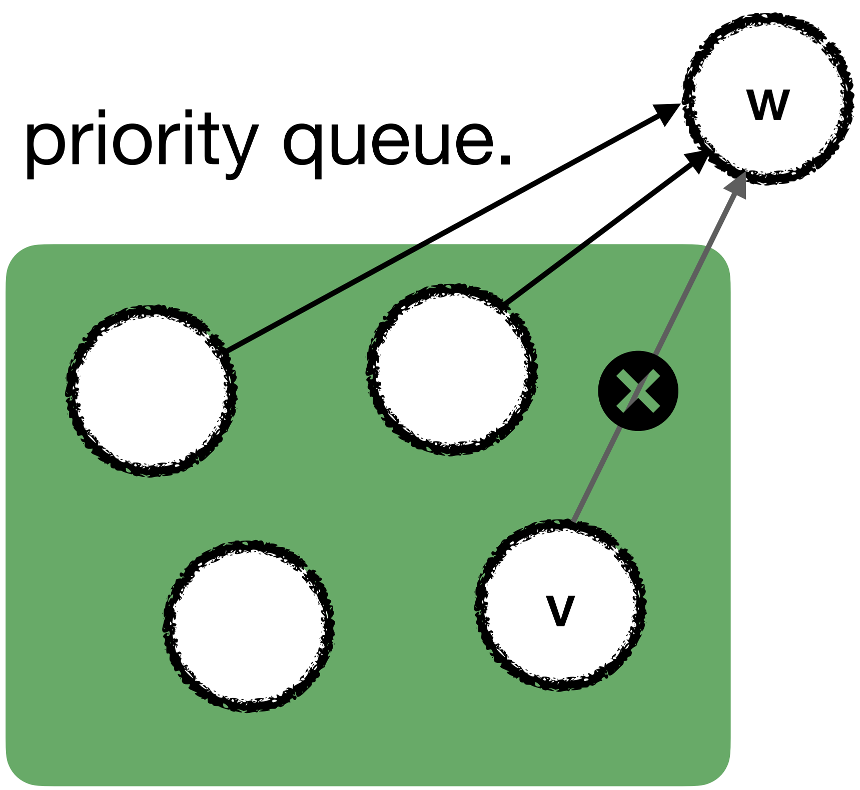
(this is because when we explore w in the future, it will not be via v).

Running Time

Issue: Once we add nodes to S, we need to update the values $d'(v)$, i.e., the keys in the priority queue.

Consider an iteration when v is added to S.

Let $w \in V - S$ be a node that remains in the priority queue.



Case 1: $(v, w) \notin E$.

$\min_{e=(u,w):u \in S} d(u) + \ell_e$ has not changed!

(this is because when we explore w in the future, it will not be via v).

Running Time

Issue: Once we add nodes to S, we need to update the values $d'(v)$, i.e., the keys in the priority queue.

Consider an iteration when v is added to S.

Let $w \in V - S$ be a node that remains in the priority queue.

Running Time

Issue: Once we add nodes to S, we need to update the values $d'(v)$, i.e., the keys in the priority queue.

Consider an iteration when v is added to S.

Let $w \in V - S$ be a node that remains in the priority queue.

Case 2: $(v, w) \in E$.

Running Time

Issue: Once we add nodes to S, we need to update the values $d'(v)$, i.e., the keys in the priority queue.

Consider an iteration when v is added to S.

Let $w \in V - S$ be a node that remains in the priority queue.

Case 2: $(v, w) \in E$.

$$d'(w) = \min(d'(w), d(v) + \ell_{(v,w)})$$

Running Time

Issue: Once we add nodes to S, we need to update the values $d'(v)$, i.e., the keys in the priority queue.

Consider an iteration when v is added to S.

Let $w \in V - S$ be a node that remains in the priority queue.

Case 2: $(v, w) \in E$.

$$d'(w) = \min(d'(w), d(v) + \ell_{(v,w)})$$

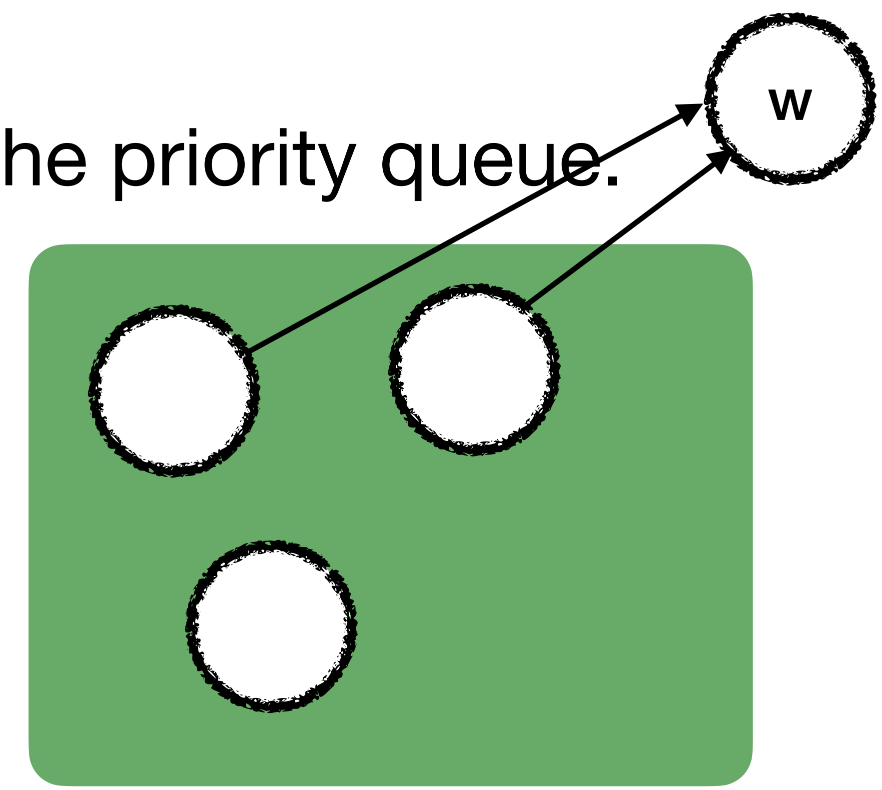
(the distance is as before, except if the path via v is shorter).

Running Time

Issue: Once we add nodes to S, we need to update the values $d'(v)$, i.e., the keys in the priority queue.

Consider an iteration when v is added to S.

Let $w \in V - S$ be a node that remains in the priority queue.



Case 2: $(v, w) \in E$.

$$d'(w) = \min(d'(w), d(v) + \ell_{(v,w)})$$

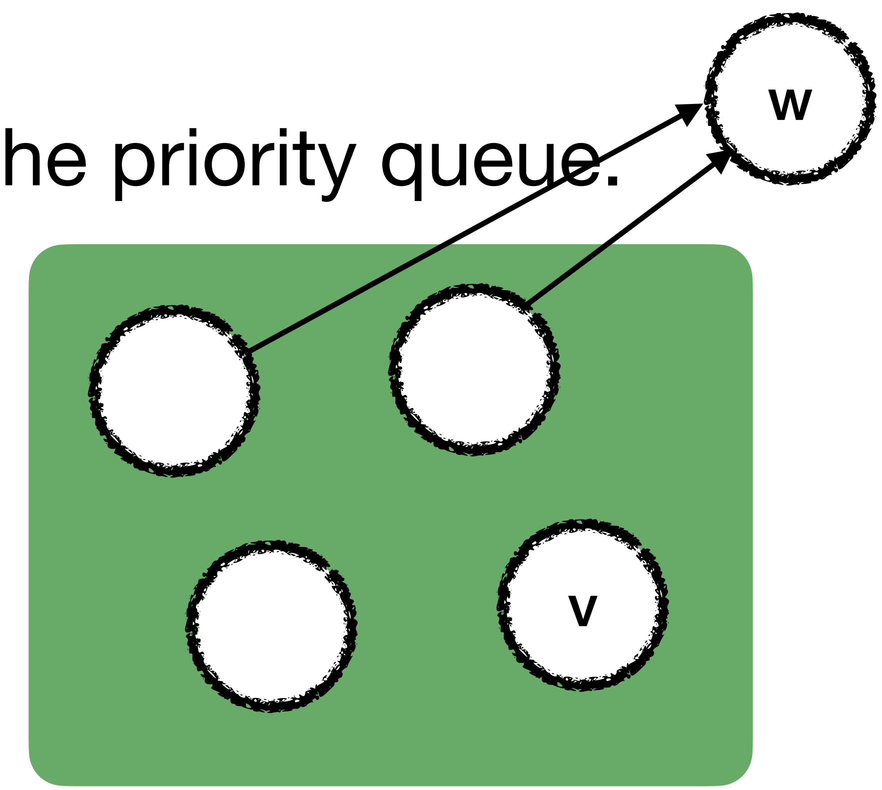
(the distance is as before, except if the path via v is shorter).

Running Time

Issue: Once we add nodes to S, we need to update the values $d'(v)$, i.e., the keys in the priority queue.

Consider an iteration when v is added to S.

Let $w \in V - S$ be a node that remains in the priority queue.



Case 2: $(v, w) \in E$.

$$d'(w) = \min(d'(w), d(v) + \ell_{(v,w)})$$

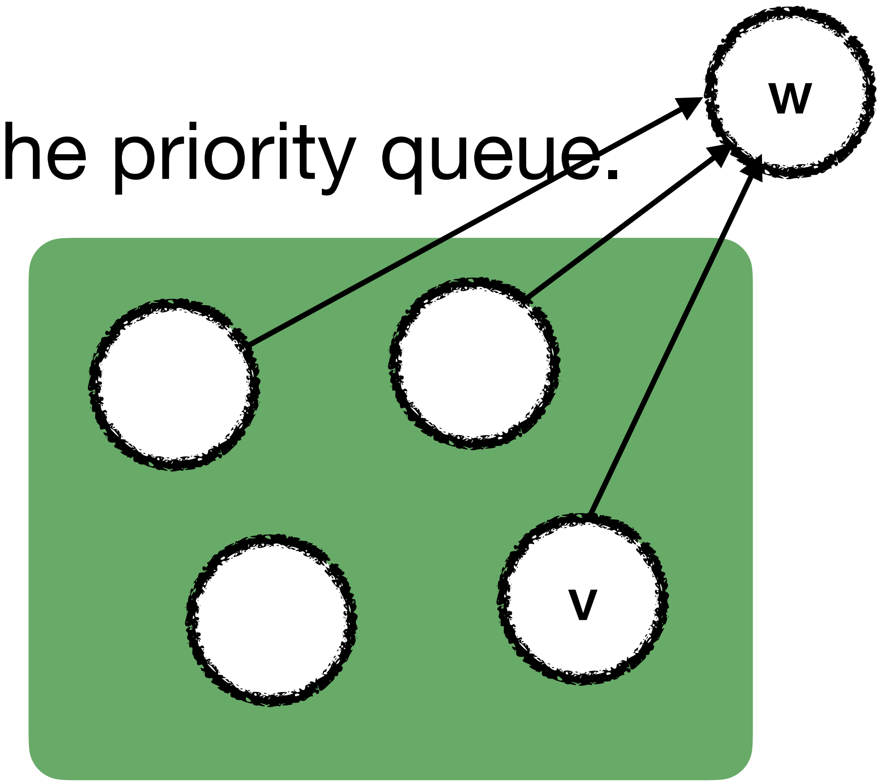
(the distance is as before, except if the path via v is shorter).

Running Time

Issue: Once we add nodes to S, we need to update the values $d'(v)$, i.e., the keys in the priority queue.

Consider an iteration when v is added to S.

Let $w \in V - S$ be a node that remains in the priority queue.



Case 2: $(v, w) \in E$.

$$d'(w) = \min(d'(w), d(v) + \ell_{(v,w)})$$

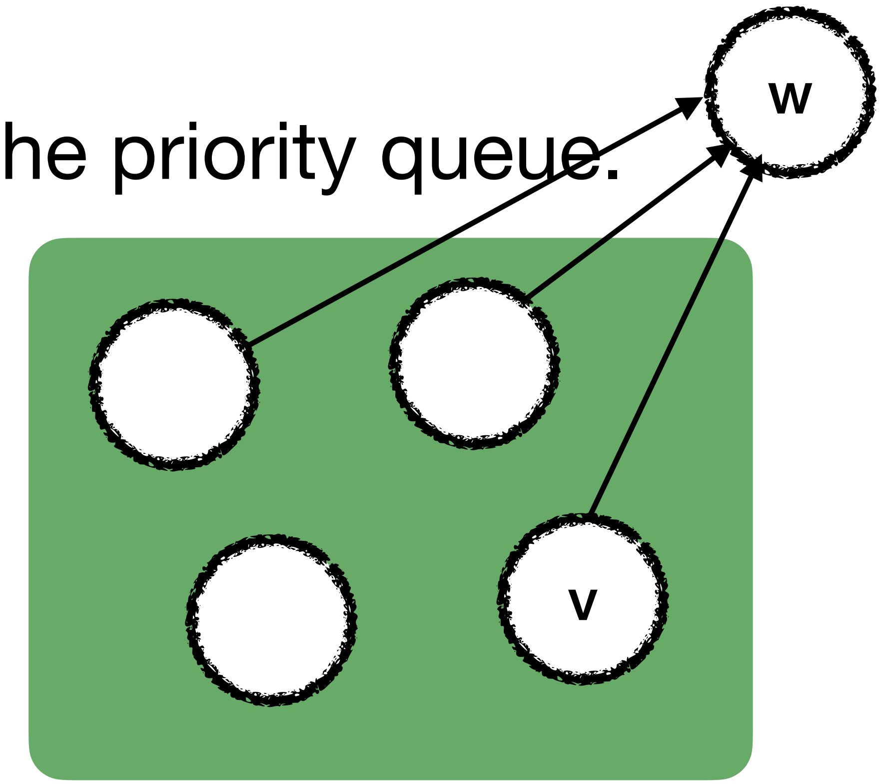
(the distance is as before, except if the path via v is shorter).

Running Time

Issue: Once we add nodes to S, we need to update the values $d'(v)$, i.e., the keys in the priority queue.

Consider an iteration when v is added to S.

Let $w \in V - S$ be a node that remains in the priority queue.



Case 2: $(v, w) \in E$.

$$d'(w) = \min(d'(w), d(v) + \ell_{(v,w)})$$

(the distance is as before, except if the path via v is shorter).

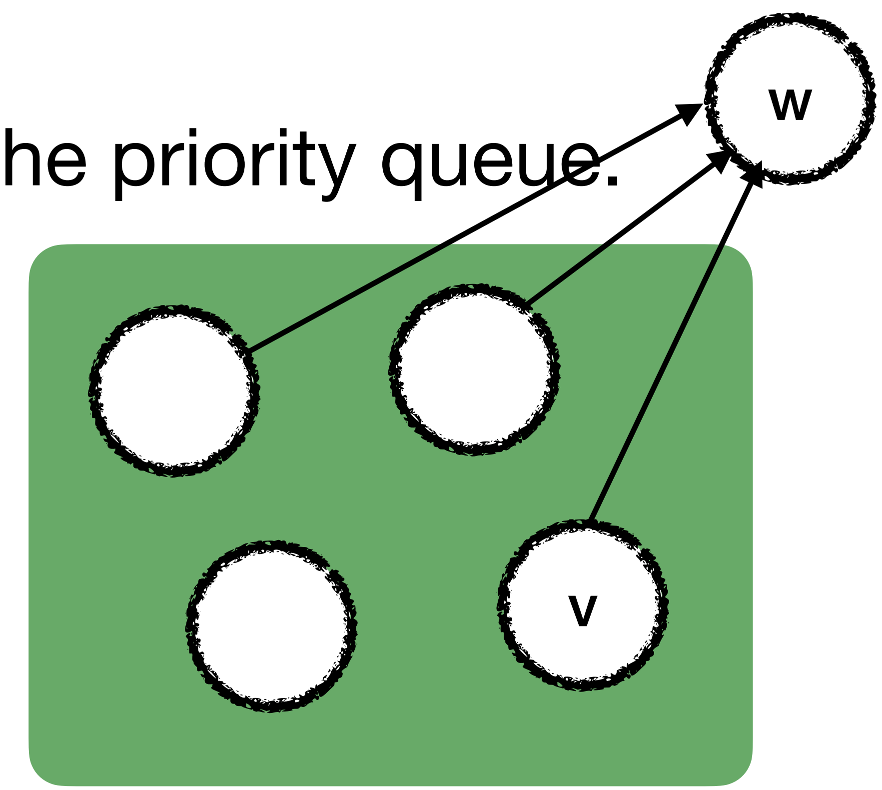
ChangeKey(Q, v, a)

Running Time

Issue: Once we add nodes to S, we need to update the values $d'(v)$, i.e., the keys in the priority queue.

Consider an iteration when v is added to S.

Let $w \in V - S$ be a node that remains in the priority queue.



Case 2: $(v, w) \in E$.

$$d'(w) = \min(d'(w), d(v) + \ell_{(v,w)})$$

(the distance is as before, except if the path via v is shorter).

ChangeKey(Q, v, a) **At most once per edge!**

Running Time

Lets look at the pseudocode.

That was somewhat naive. Can we do better?

Running Time

Lets look at the pseudocode.

That was somewhat naive. Can we do better?

We need n ExtractMin(Q) and m ChangeKey(Q, v, a) operations, plus $O(m)$ time for computing the distances.

Running Time

Lets look at the pseudocode.

That was somewhat naive. Can we do better?

We need n ExtractMin(Q) and m ChangeKey(Q, v, a) operations, plus $O(m)$ time for computing the distances.

How much time do we need for the priority queue operations?

Running Time

Lets look at the pseudocode.

That was somewhat naive. Can we do better?

We need n ExtractMin(Q) and m ChangeKey(Q, v, a) operations, plus $O(m)$ time for computing the distances.

How much time do we need for the priority queue operations?

$O(\log n)$

Running Time

Lets look at the pseudocode.

That was somewhat naive. Can we do better?

We need n ExtractMin(Q) and m ChangeKey(Q, v, a) operations, plus $O(m)$ time for computing the distances.

How much time do we need for the priority queue operations?

$O(\log n)$

Overall: $O(m \log n)$

Reading

Kleinberg and Tardos Chapter 5.4. (or 4.4. in the online weird version). Slides follow this religiously.

Roughgarden 9.2., 9.3.

CLRS 24.3.

You can also find visualisers online and play around with them, e.g., https://www.cs.usfca.edu/~galles/visualization/Dijkstra.html and the more general https://visualgo.net/en/sssp?slide=1