Lecture 29

Greedy: Activity-Selection Problem (contd.), MST

Greedy Algorithm for Activity-Selection

Greedy Algorithm for Activity-Selection

i	0	1	2	3	4	5	6	7	8	9	10	11	12
s_i	0	1	3	0	5	3	5	6	7	8	2	12	16
f_i	0	4	5	6	7	9	9	10	11	12	14	16	16

Greedy Algorithm for Activity-Selection

i	0	1	2	3	4	5	6	7	8	9	10	11	12
s_i	0	1	3	0	5	3	5	6	7	8	2	12	16
f_i	0	4	5	6	7	9	9	10	11	12	14	16	16

Let's try to find $A_{0,12}$ using greedy choices!

Greedy Algorithm for Activity-Selection

$A_{0,12}$ ↓

i	0	1	2	3	4	5	6	7	8	9	10	11	12
s_i	0	1	3	0	5	3	5	6	7	8	2	12	16
f_i	0	4	5	6	7	9	9	10	11	12	14	16	16

Greedy Algorithm for Activity-Selection

$A_{0,12}$

↓

$A_{0,1} \cup \{a_1\} \cup A_{1,12}$

i	0	1	2	3	4	5	6	7	8	9	10	11	12
s_i	0	1	3	0	5	3	5	6	7	8	2	12	16
f_i	0	4	5	6	7	9	9	10	11	12	14	16	16

Greedy Algorithm for Activity-Selection

$$A_{0,12}$$

$\downarrow$

$$A_{0,1} \cup \{a_1\} \cup A_{1,12}$$

$\downarrow$

$$\varnothing$$

i	0	1	2	3	4	5	6	7	8	9	10	11	12
s_i	0	1	3	0	5	3	5	6	7	8	2	12	16
f_i	0	4	5	6	7	9	9	10	11	12	14	16	16

Greedy Algorithm for Activity-Selection

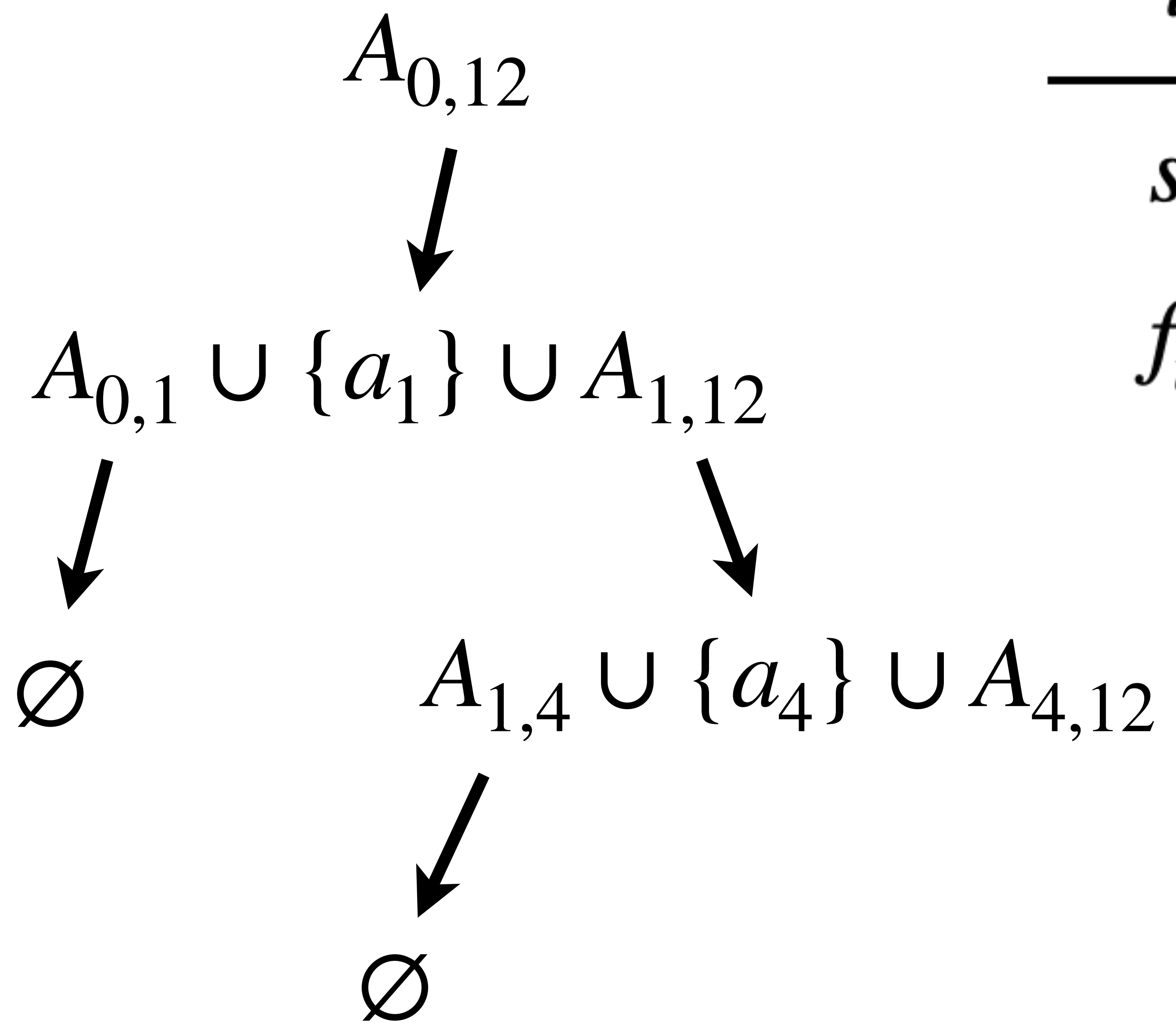
$A_{0,12}$

$A_{0,1} \cup \{a_1\} \cup A_{1,12}$

$\emptyset$ $\quad$ $A_{1,4} \cup \{a_4\} \cup A_{4,12}$

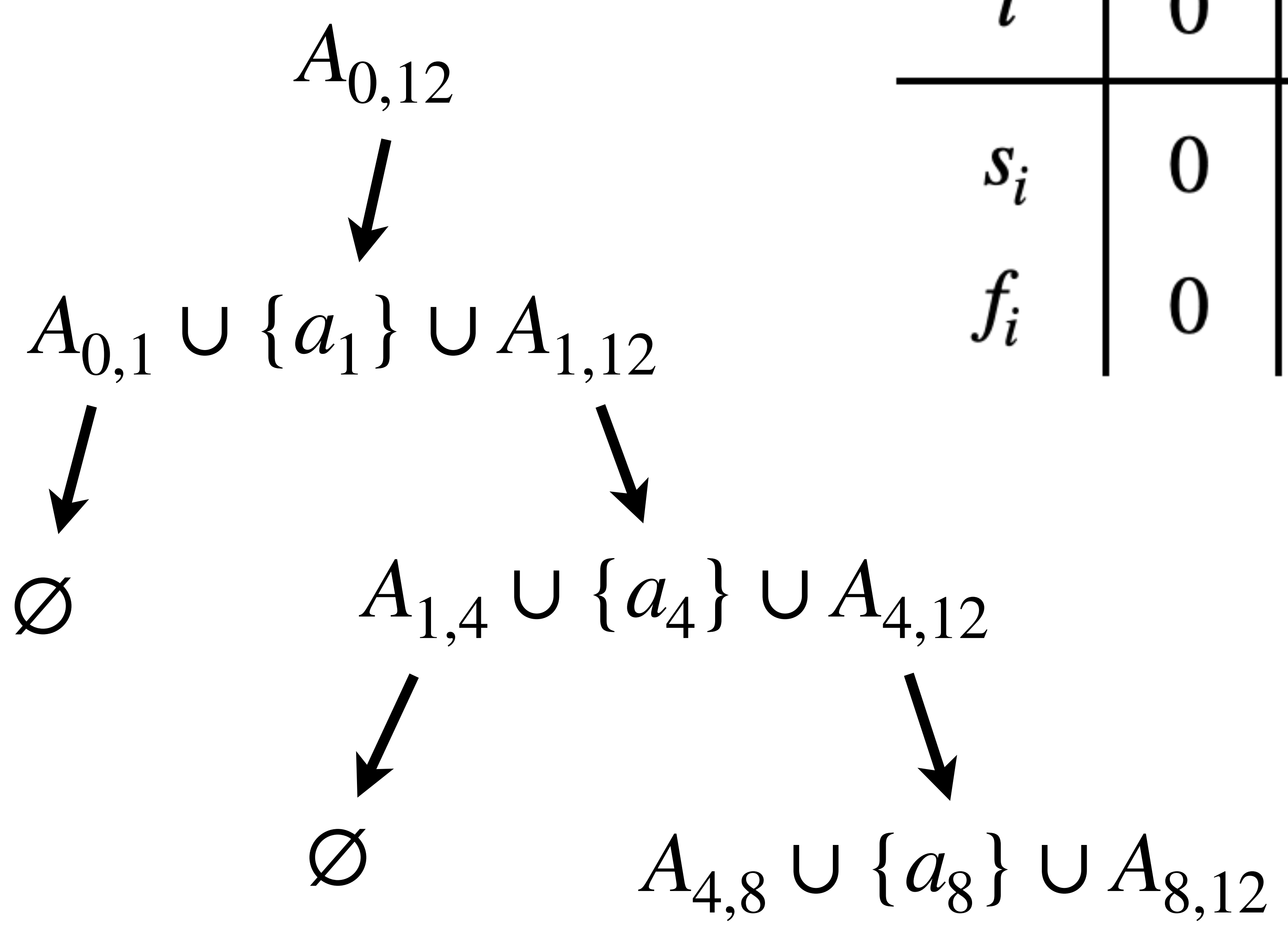
i	0	1	2	3	4	5	6	7	8	9	10	11	12
s_i	0	1	3	0	5	3	5	6	7	8	2	12	16
f_i	0	4	5	6	7	9	9	10	11	12	14	16	16

Greedy Algorithm for Activity-Selection



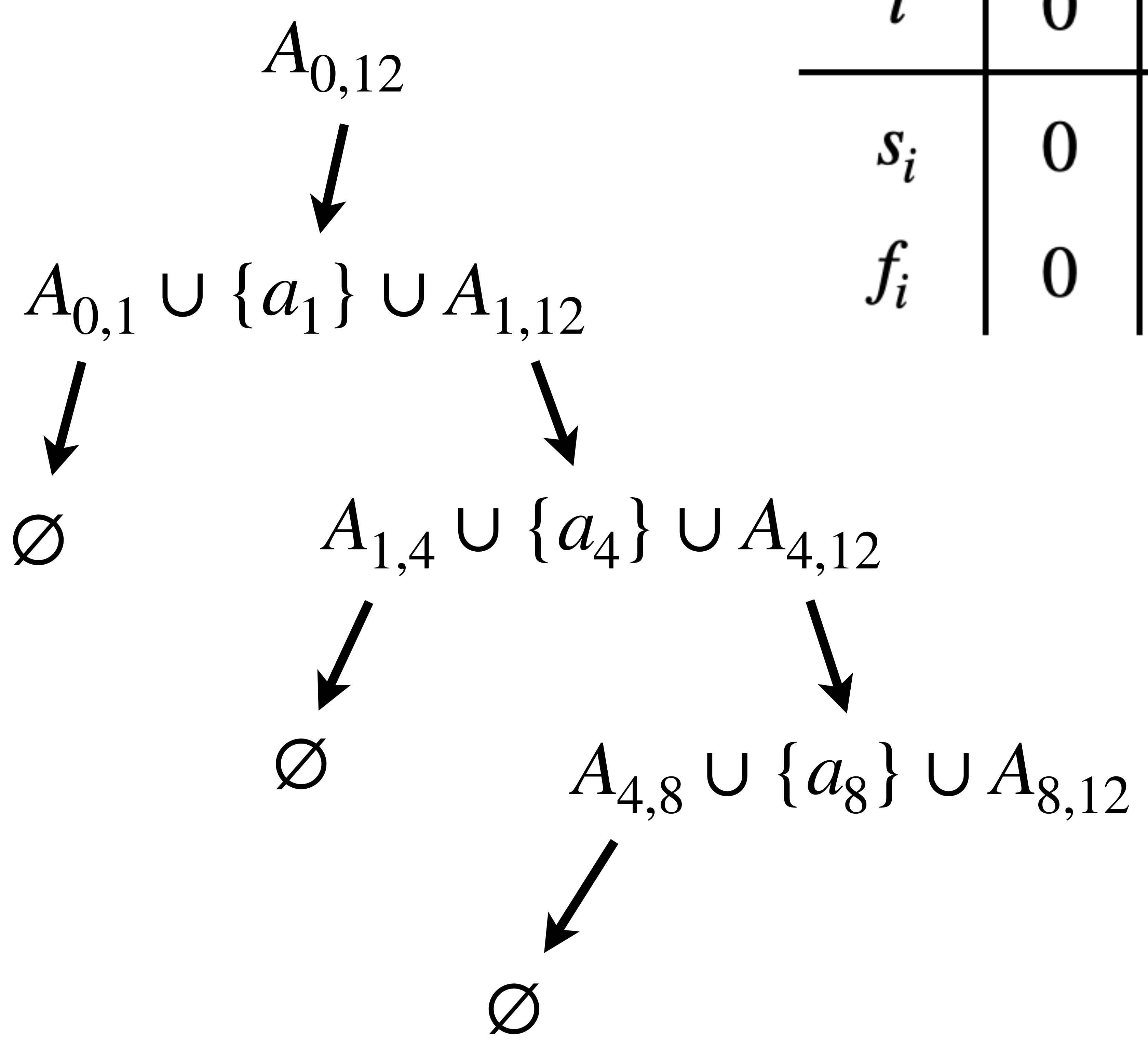
i	0	1	2	3	4	5	6	7	8	9	10	11	12
s_i	0	1	3	0	5	3	5	6	7	8	2	12	16
f_i	0	4	5	6	7	9	9	10	11	12	14	16	16

Greedy Algorithm for Activity-Selection



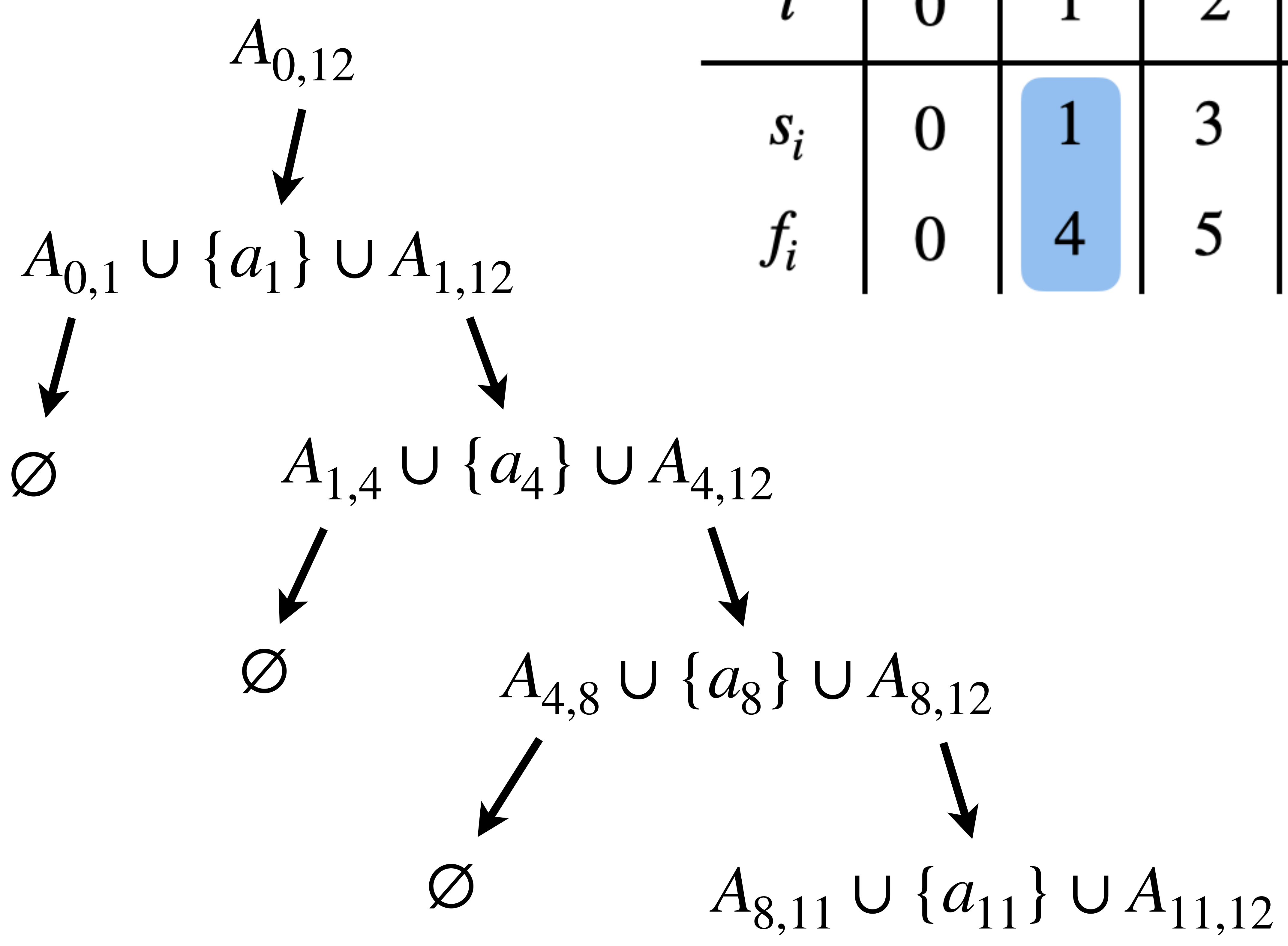
i	0	1	2	3	4	5	6	7	8	9	10	11	12
s_i	0	1	3	0	5	3	5	6	7	8	2	12	16
f_i	0	4	5	6	7	9	9	10	11	12	14	16	16

Greedy Algorithm for Activity-Selection



i	0	1	2	3	4	5	6	7	8	9	10	11	12
s_i	0	1	3	0	5	3	5	6	7	8	2	12	16
f_i	0	4	5	6	7	9	9	10	11	12	14	16	16

Greedy Algorithm for Activity-Selection



i	0	1	2	3	4	5	6	7	8	9	10	11	12
s_i	0	1	3	0	5	3	5	6	7	8	2	12	16
f_i	0	4	5	6	7	9	9	10	11	12	14	16	16

Greedy Algorithm for Activity-Selection

$A_{0,12}$

$A_{0,1} \cup \{a_1\} \cup A_{1,12}$

$\varnothing$

$A_{1,4} \cup \{a_4\} \cup A_{4,12}$

$\varnothing$

$A_{4,8} \cup \{a_8\} \cup A_{8,12}$

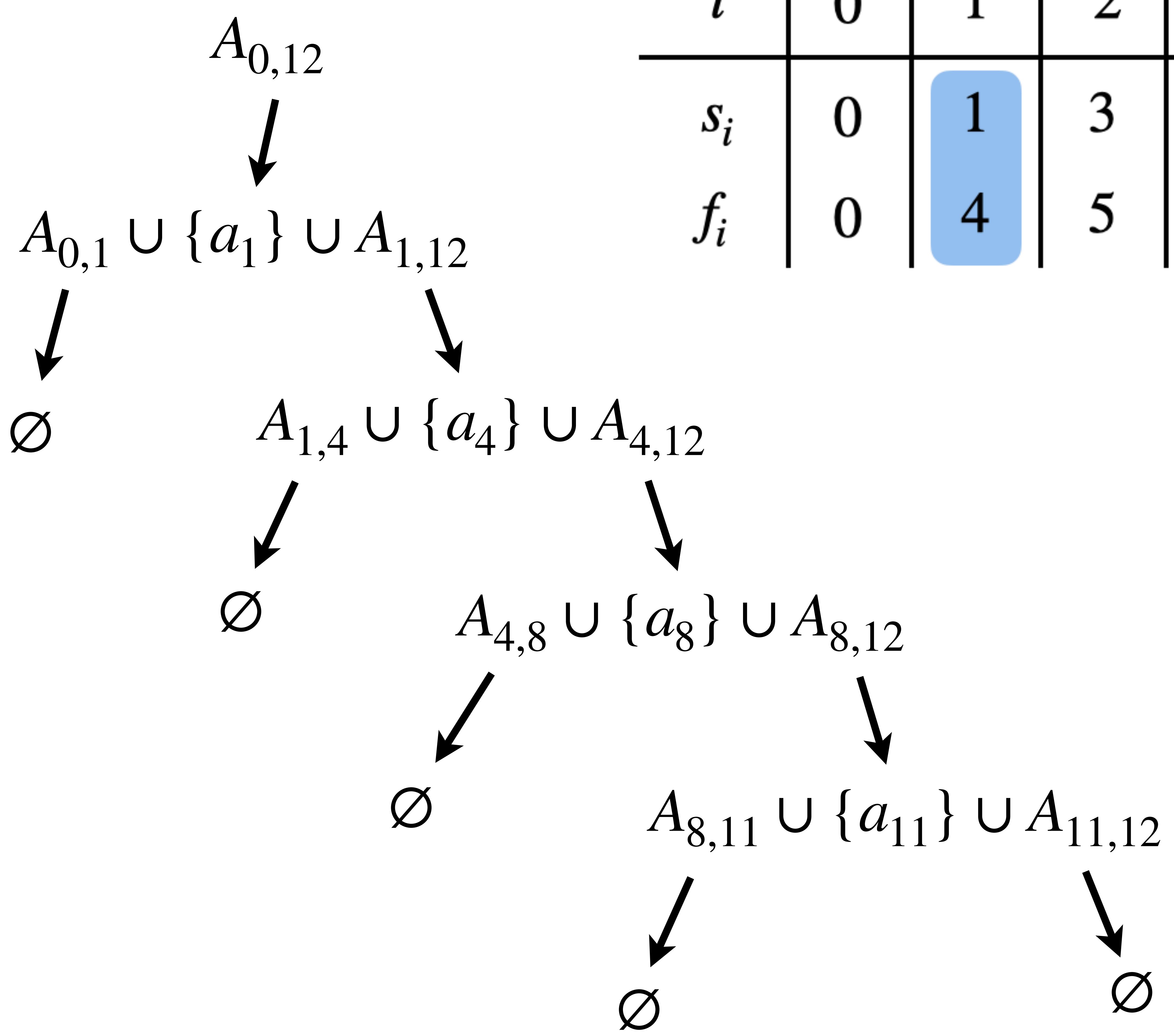
$\varnothing$

$A_{8,11} \cup \{a_{11}\} \cup A_{11,12}$

$\varnothing$

i	0	1	2	3	4	5	6	7	8	9	10	11	12
s_i	0	1	3	0	5	3	5	6	7	8	2	12	16
f_i	0	4	5	6	7	9	9	10	11	12	14	16	16

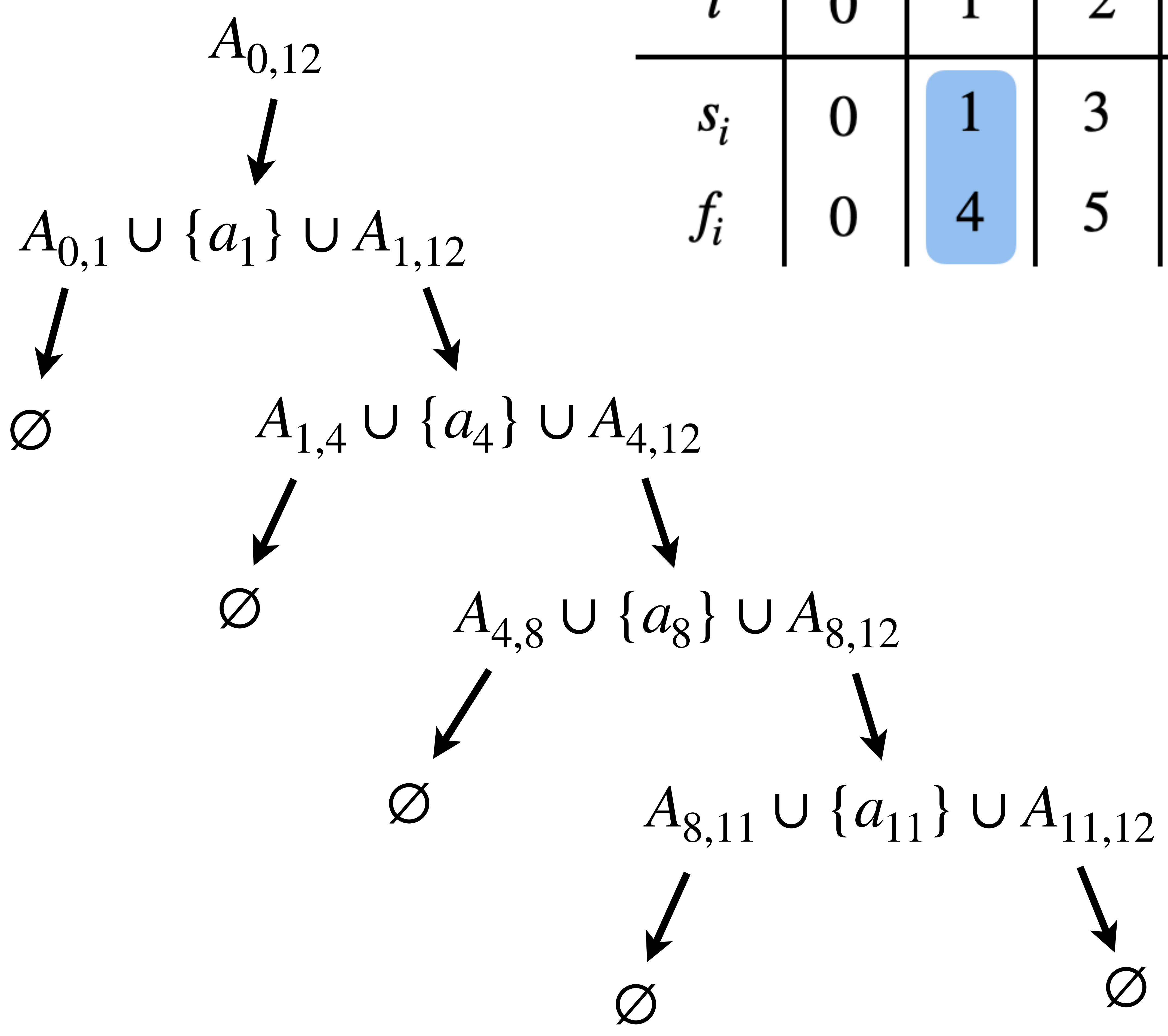
Greedy Algorithm for Activity-Selection



i	0	1	2	3	4	5	6	7	8	9	10	11	12
s_i	0	1	3	0	5	3	5	6	7	8	2	12	16
f_i	0	4	5	6	7	9	9	10	11	12	14	16	16

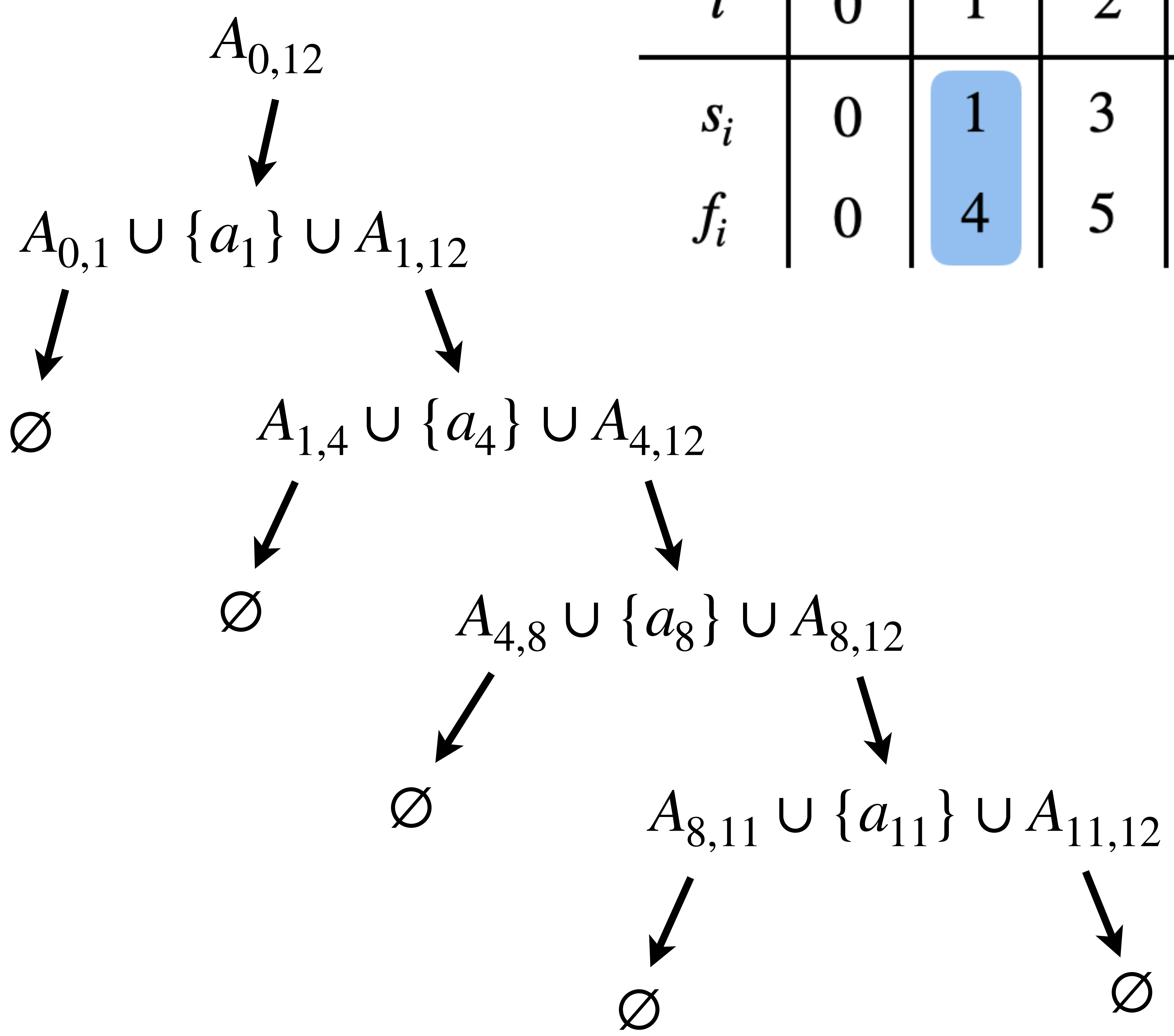
Greedy Algorithm for Activity-Selection

i	0	1	2	3	4	5	6	7	8	9	10	11	12
s_i	0	1	3	0	5	3	5	6	7	8	2	12	16
f_i	0	4	5	6	7	9	9	10	11	12	14	16	16



Why $A_{4,8}$ (or $A_{0,1}$, $A_{1,4}$, $A_{8,11}$) must be $\emptyset$?

Greedy Algorithm for Activity-Selection



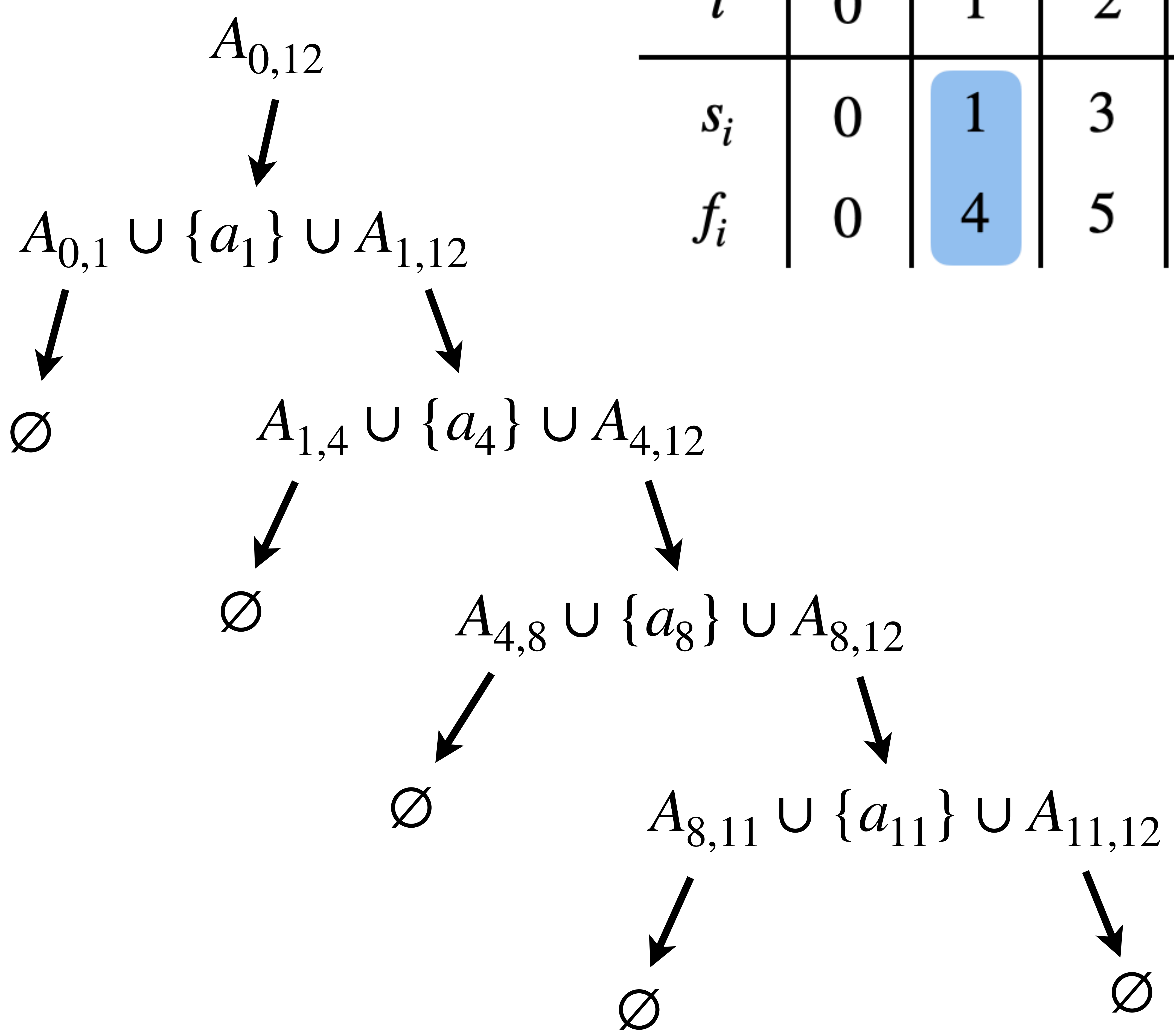
i	0	1	2	3	4	5	6	7	8	9	10	11	12
s_i	0	1	3	0	5	3	5	6	7	8	2	12	16
f_i	0	4	5	6	7	9	9	10	11	12	14	16	16

Why $A_{4,8}$ (or $A_{0,1}$, $A_{1,4}$, $A_{8,11}$) must be $\varnothing$?

If $A_{4,8}$ contains some activity, then we would

Greedy Algorithm for Activity-Selection

i	0	1	2	3	4	5	6	7	8	9	10	11	12
s_i	0	1	3	0	5	3	5	6	7	8	2	12	16
f_i	0	4	5	6	7	9	9	10	11	12	14	16	16



Why $A_{4,8}$ (or $A_{0,1}$, $A_{1,4}$, $A_{8,11}$) must be $\varnothing$?

If $A_{4,8}$ contains some activity, then we would have picked that activity instead of a_8 for $A_{4,12}$.

Greedy Algorithm for Activity-Selection

$$A_{0,12}$$

$$A_{0,1} \cup \{a_1\} \cup A_{1,12}$$

$$\emptyset \qquad A_{1,4} \cup \{a_4\} \cup A_{4,12}$$

$$\emptyset \qquad A_{4,8} \cup \{a_8\} \cup A_{8,12}$$

$$\emptyset \qquad A_{8,11} \cup \{a_{11}\} \cup A_{11,12}$$

$$\emptyset \qquad \emptyset$$

Greedy Algorithm for Activity-Selection

Greedy Strategy:

$$A_{0,12}$$

$$A_{0,1} \cup \{a_1\} \cup A_{1,12}$$

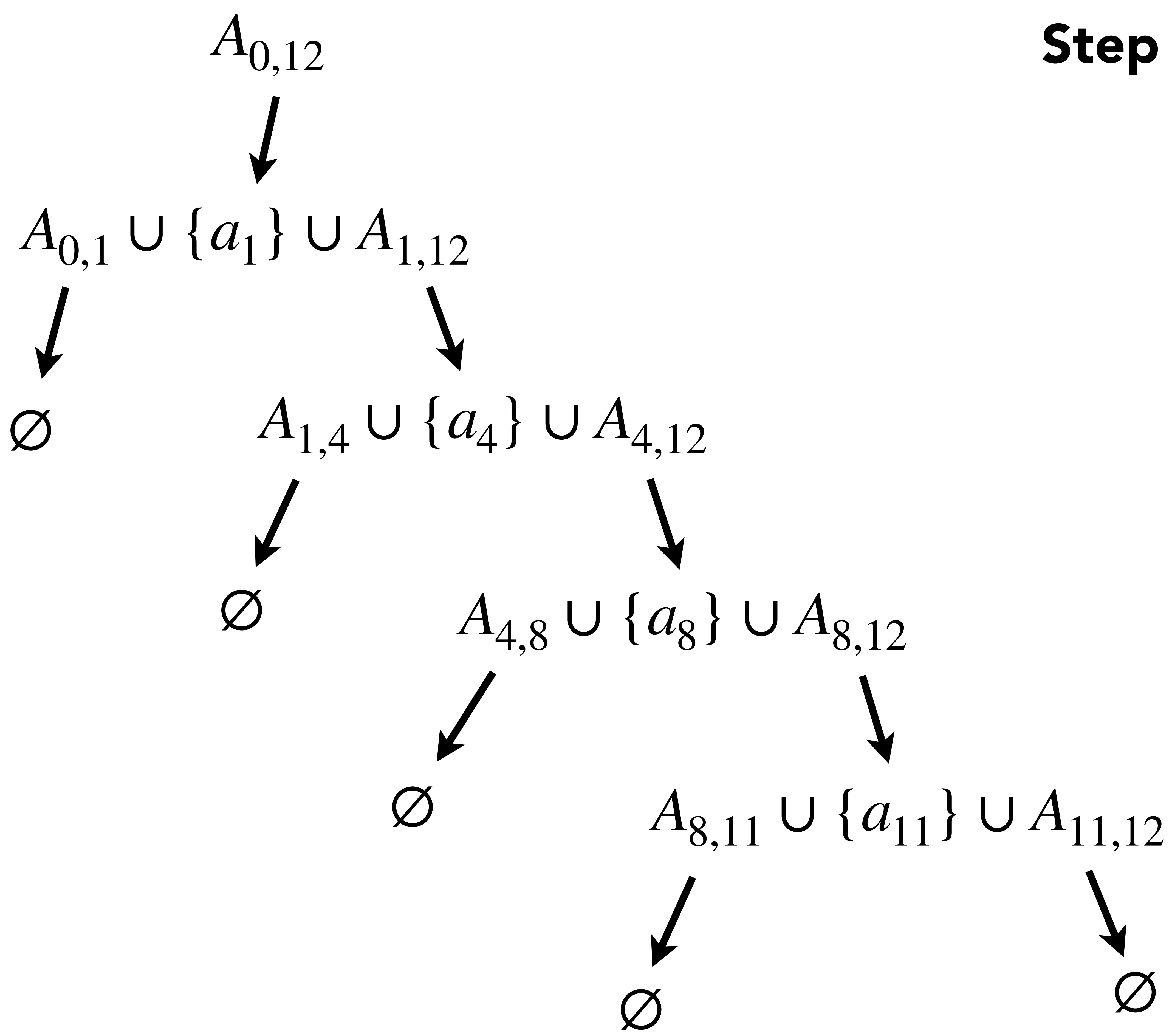
$$\emptyset \qquad A_{1,4} \cup \{a_4\} \cup A_{4,12}$$

$$\emptyset \qquad A_{4,8} \cup \{a_8\} \cup A_{8,12}$$

$$\emptyset \qquad A_{8,11} \cup \{a_{11}\} \cup A_{11,12}$$

$$\emptyset \qquad \emptyset$$

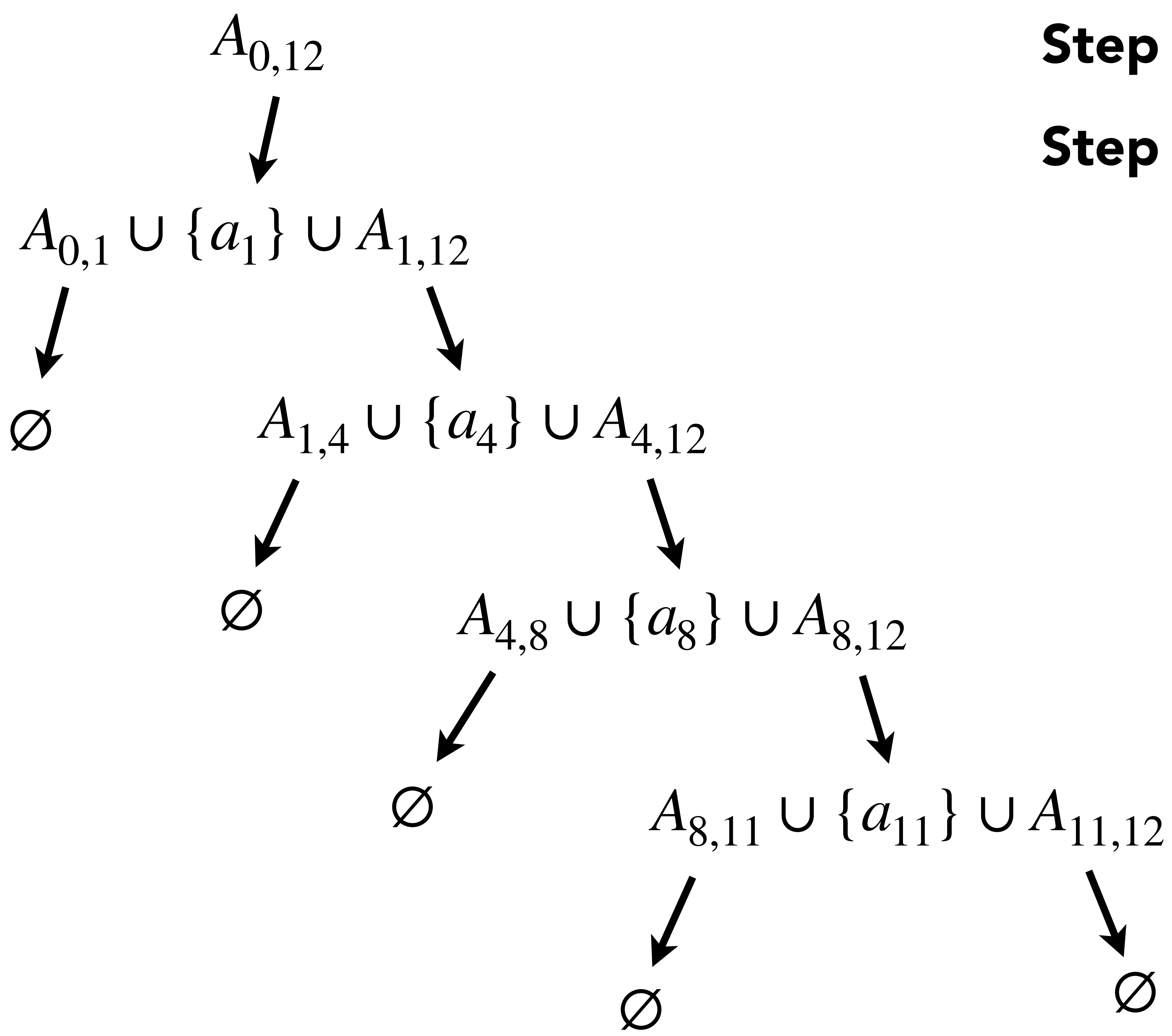
Greedy Algorithm for Activity-Selection



Greedy Strategy:

Step 1: Pick a_1.

Greedy Algorithm for Activity-Selection

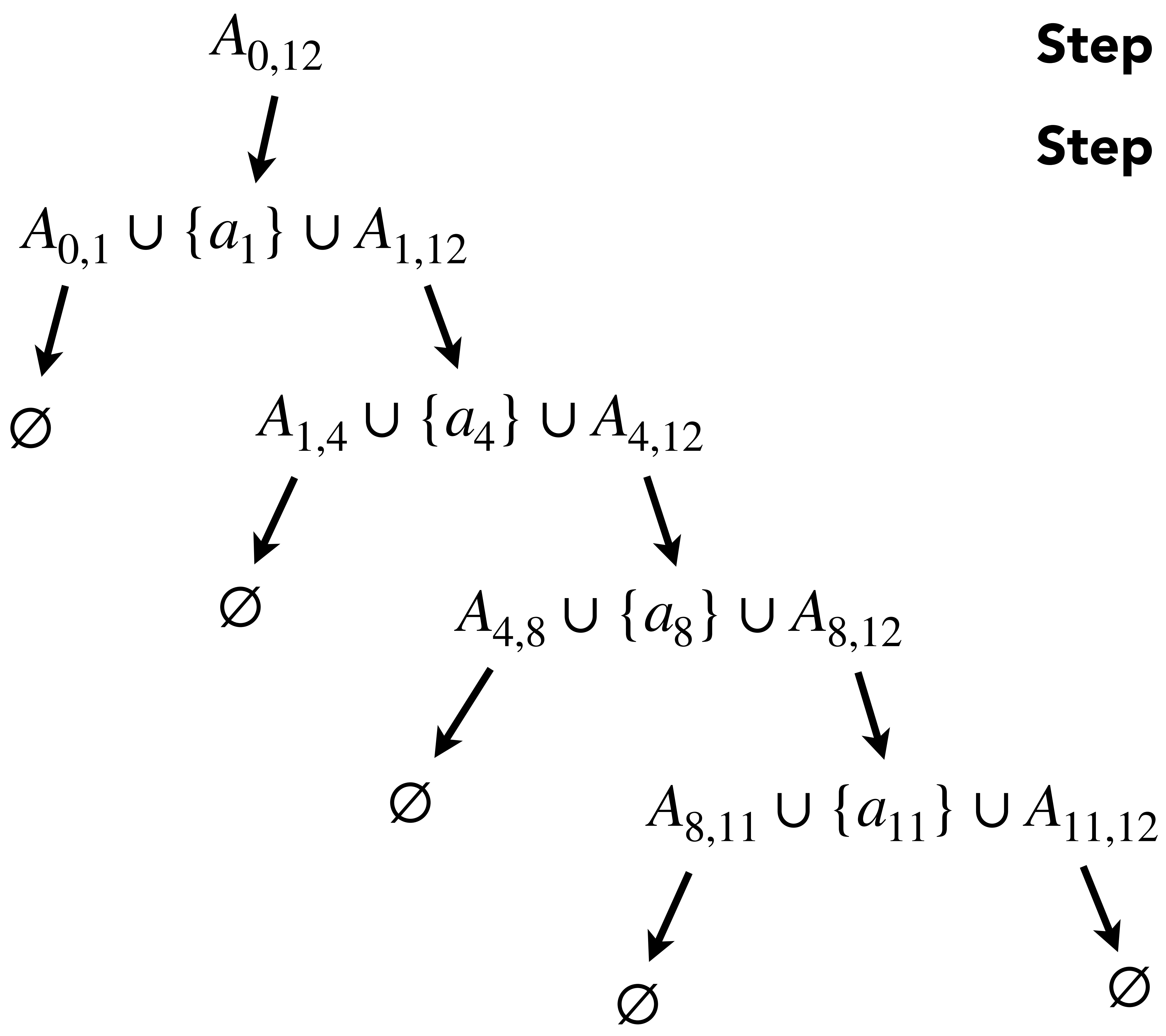


Greedy Strategy:

Step 1: Pick a_1.

Step 2: Let a_k was the last picked activity. Then, pick the

Greedy Algorithm for Activity-Selection

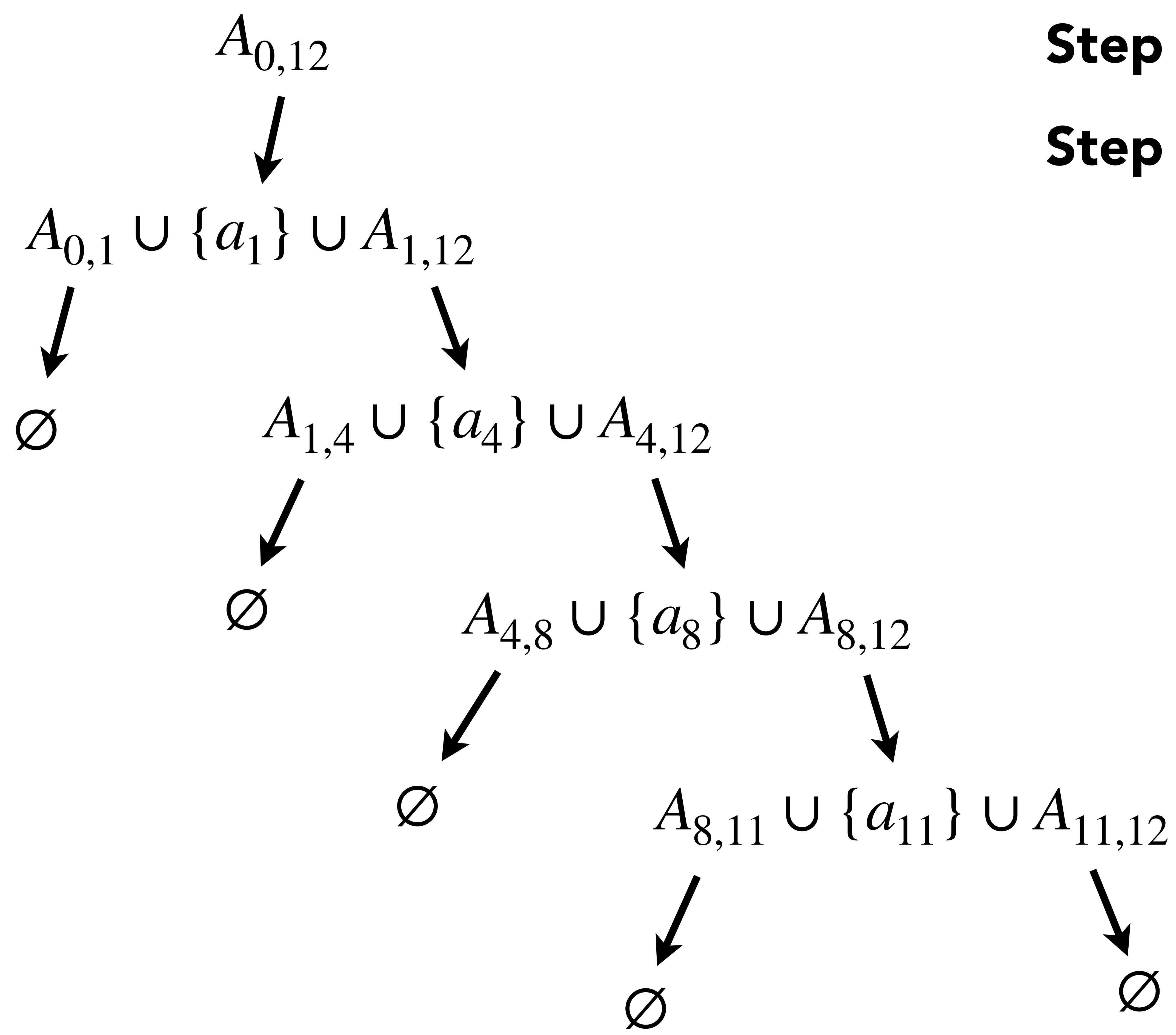


Greedy Strategy:

Step 1: Pick a_1.

Step 2: Let a_k was the last picked activity. Then, pick the earliest finishing activity in $S_{k,n+1}$.

Greedy Algorithm for Activity-Selection

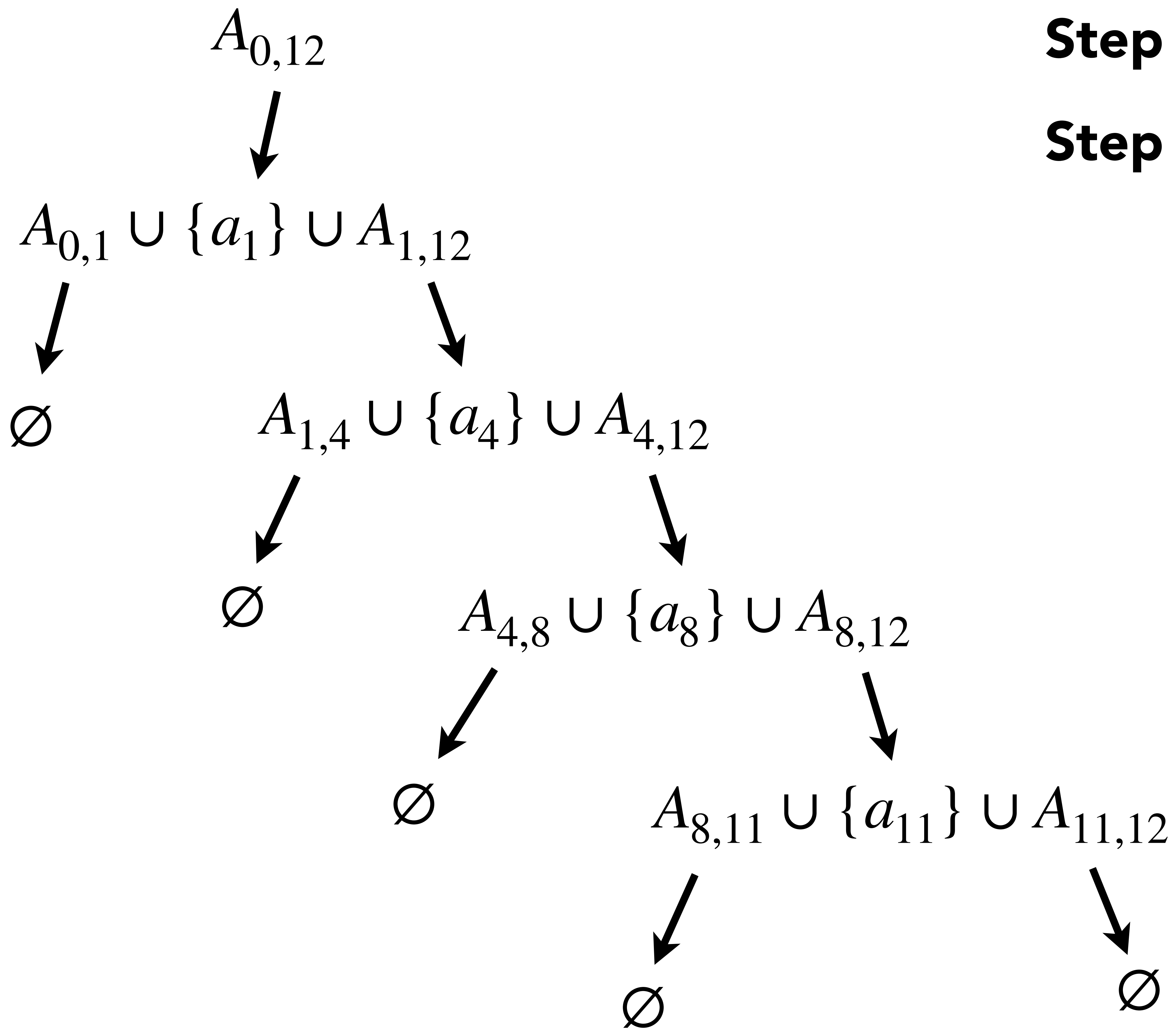


Greedy Strategy:

Step 1**:** Pick a_1.

Step 2**:** Let a_k was the last picked activity. Then, pick the earliest finishing activity in $S_{k,n+1}$. That is, the first

Greedy Algorithm for Activity-Selection

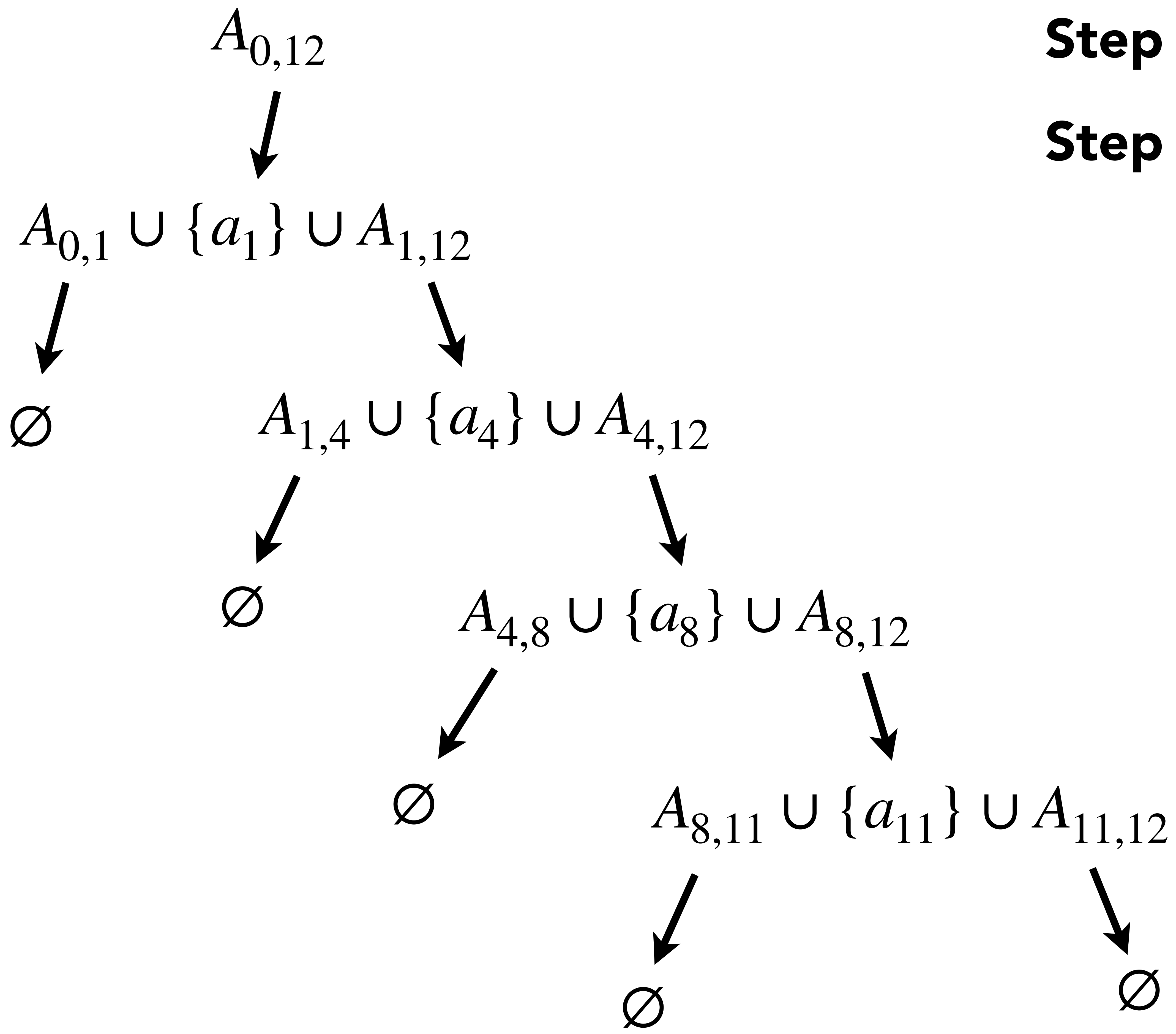


Greedy Strategy:

Step 1**:** Pick a_1.

Step 2**:** Let a_k was the last picked activity. Then, pick the earliest finishing activity in $S_{k,n+1}$. That is, the first activity after a_k, say a_i, so that:

Greedy Algorithm for Activity-Selection



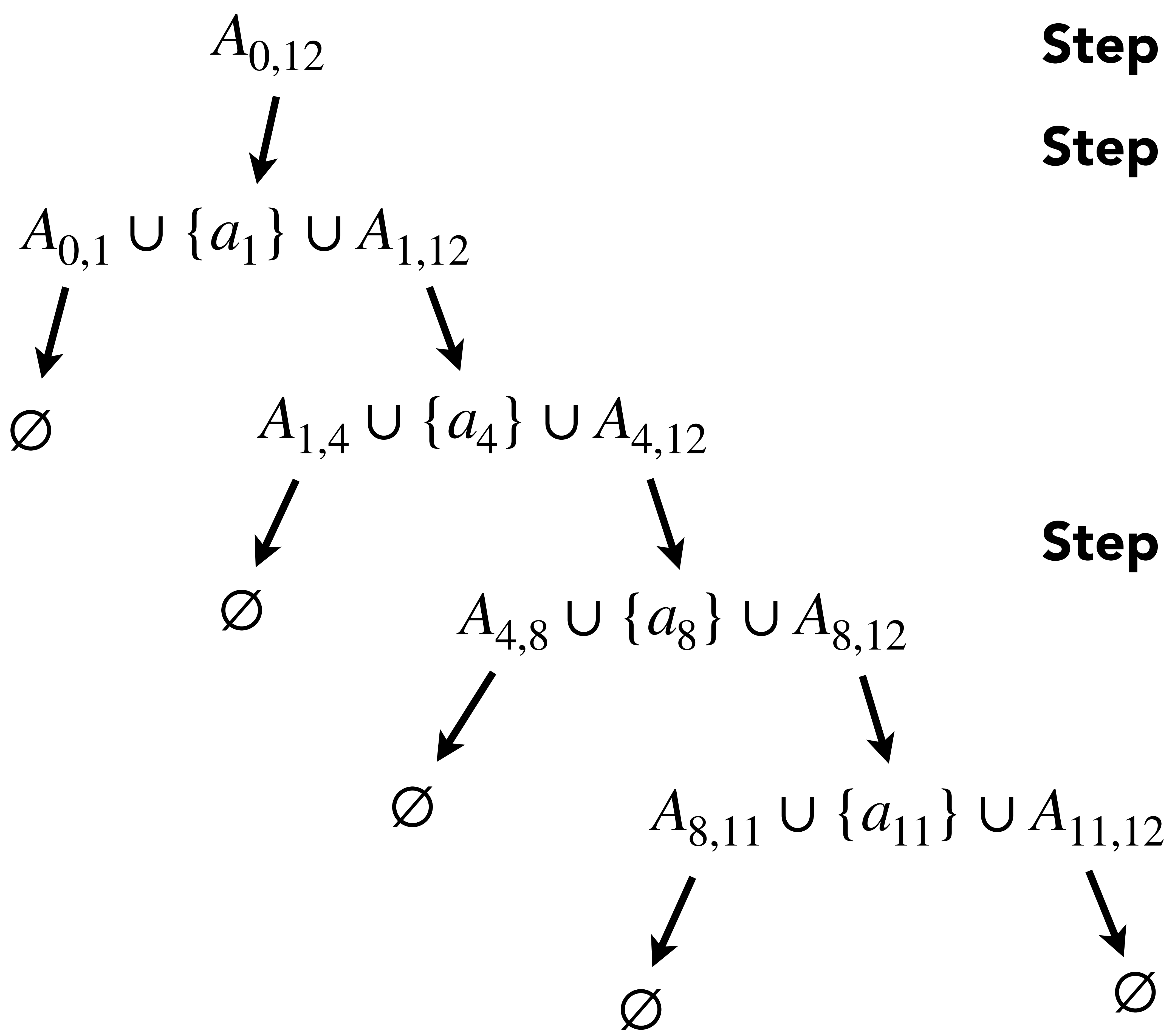
Greedy Strategy:

Step 1: Pick a_1.

Step 2: Let a_k was the last picked activity. Then, pick the earliest finishing activity in $S_{k,n+1}$. That is, the first activity after a_k, say a_i, so that:

$$a_k.finish \leq a_i.start \text{ and } a_i.finish \leq a_{n+1}.start$$

Greedy Algorithm for Activity-Selection



Greedy Strategy:

Step 1**:** Pick a_1.

Step 2**:** Let a_k was the last picked activity. Then, pick the earliest finishing activity in $S_{k,n+1}$. That is, the first activity after a_k, say a_i, so that:

$$a_k.finish \le a_i.start \text{ and } a_i.finish \le a_{n+1}.start$$

Step 3**:** Go to **Step** 2, if you can.

Greedy Algorithm for Activity-Selection

Greedy Algorithm for Activity-Selection

Activity-Selection$(s, f, n+2)$

Greedy Algorithm for Activity-Selection

Start and finish time of $n+2$ activities (with dummy activities a_0 and a_{n+1})

Activity-Selection$(s, f, n+2)$

Greedy Algorithm for Activity-Selection

Start and finish time of $n + 2$ activities (with dummy activities a_0 and a_{n+1})

Activity-Selection$(s, f, n + 2)$

1. $A = \{a_1\}$

Greedy Algorithm for Activity-Selection

Start and finish time of $n + 2$ activities (with dummy activities a_0 and a_{n+1})

Activity-Selection$(s, f, n + 2)$

1. $A = \{a_1\}$
2. $k = 1$ *// k is the index of the last picked activity*

Greedy Algorithm for Activity-Selection

Start and finish time of $n + 2$ activities (with dummy activities a_0 and a_{n+1})

Activity-Selection$(s, f, n + 2)$

1. $A = \{a_1\}$
2. $k = 1$ *// k is the index of the last picked activity*
3. **for** $i = 2$ **to** n

Greedy Algorithm for Activity-Selection

Start and finish time of $n + 2$ activities (with dummy activities a_0 and a_{n+1})

Activity-Selection$(s, f, n + 2)$

1. $A = \{a_1\}$
2. $k = 1$ *// k is the index of the last picked activity*
3. **for** $i = 2$ **to** n
4. **if** $f[k] \leq s[i]$ and $f[i] \leq s[n + 1]$

Greedy Algorithm for Activity-Selection

Start and finish time of $n+2$ activities (with dummy activities a_0 and a_{n+1})

Activity-Selection$(s, f, n+2)$

$A = \{a_1\}$

$k = 1$ *// k is the index of the last picked activity*

for $i = 2$ **to** n

  if $f[k] \leq s[i]$ and $f[i] \leq s[n+1]$

    $A = A \cup \{a_i\}$

Greedy Algorithm for Activity-Selection

Start and finish time of $n + 2$ activities (with dummy activities a_0 and a_{n+1})

Activity-Selection$(s, f, n + 2)$

$A = \{a_1\}$
$k = 1$ *// k is the index of the last picked activity*
for $i = 2$ **to** n
    if $f[k] \leq s[i]$ and $f[i] \leq s[n + 1]$
        $A = A \cup \{a_i\}$
        $k = i$ *// resetting the index of the last picked activity*

Greedy Algorithm for Activity-Selection

Start and finish time of $n+2$ activities (with dummy activities a_0 and a_{n+1})

Activity-Selection$(s, f, n+2)$

$A = \{a_1\}$

$k = 1$ *// k is the index of the last picked activity*

for $i = 2$ **to** n

    if $f[k] \le s[i]$ and $f[i] \le s[n+1]$

        $A = A \cup \{a_i\}$

        $k = i$ *// resetting the index of the last picked activity*

return A

Greedy Algorithm for Activity-Selection

Start and finish time of $n + 2$ activities (with dummy activities a_0 and a_{n+1})

Activity-Selection$(s, f, n + 2)$

1. $A = \{a_1\}$
2. $k = 1$ *// k is the index of the last picked activity*
3. **for** $i = 2$ **to** n
4. **if** $f[k] \leq s[i]$ **and** $f[i] \leq s[n + 1]$
5. $A = A \cup \{a_i\}$
6. $k = i$ *// resetting the index of the last picked activity*
7. **return** A

Will always be true. Hence, can be skipped.

Greedy Algorithm for Activity-Selection

Start and finish time of $n + 2$ activities (with dummy activities a_0 and a_{n+1})

Activity-Selection$(s, f, n + 2)$

$A = \{a_1\}$

$k = 1$ *// k is the index of the last picked activity*

for $i = 2$ **to** n

  if $f[k] \leq s[i]$ and $f[i] \leq s[n + 1]$

    $A = A \cup \{a_i\}$

    $k = i$ *// resetting the index of the last picked activity*

return A

Will always be true. Hence, can be skipped.

Time Complexity: $\Theta(n)$

When to Use Greedy?

When to Use Greedy?

Greedy is typically used in optimization problems with the following two properties:

When to Use Greedy?

Greedy is typically used in optimization problems with the following two properties:

Optimal Substructure: Optimal solution to the problem contains optimal solutions to

When to Use Greedy?

Greedy is typically used in optimization problems with the following two properties:

Optimal Substructure: Optimal solution to the problem contains optimal solutions to subproblems.

When to Use Greedy?

Greedy is typically used in optimization problems with the following two properties:

Optimal Substructure: Optimal solution to the problem contains optimal solutions to subproblems.

If $a_{i+1} \in A_{i,j}$, then $c_{i,j} = c_{i,i+1} + c_{i+1,j} + 1$

When to Use Greedy?

Greedy is typically used in optimization problems with the following two properties:

Optimal Substructure: Optimal solution to the problem contains optimal solutions to subproblems.

If $a_{i+1} \in A_{i,j}$, then $c_{i,j} = c_{i,i+1} + c_{i+1,j} + 1$

Greedy Choice Property: A globally optimal solution can be constructed by making locally optimal (greedy) choices.

When to Use Greedy?

Greedy is typically used in optimization problems with the following two properties:

Optimal Substructure: Optimal solution to the problem contains optimal solutions to subproblems.

If $a_{i+1} \in A_{i,j}$, then $c_{i,j} = c_{i,i+1} + c_{i+1,j} + 1$

Greedy Choice Property: A globally optimal solution can be constructed by making locally optimal (greedy) choices.

Earliest finishing activity in $S_{i,j}$ will be part of some $A_{i,j}$.

How to Use Greedy?

How to Use Greedy?

Solving a problem using Greedy usually takes five steps:

How to Use Greedy?

Solving a problem using Greedy usually takes five steps:

- Find the optimal substructure.

How to Use Greedy?

Solving a problem using Greedy usually takes five steps:

- Find the optimal substructure.
- Recursively define the value of optimal solution.

How to Use Greedy?

Solving a problem using Greedy usually takes five steps:

- Find the optimal substructure.
- Recursively define the value of optimal solution.
- Show that by making a greedy choice you don’t need to solve all the subproblems.

How to Use Greedy?

Solving a problem using Greedy usually takes five steps:

- Find the optimal substructure.
- Recursively define the value of optimal solution.
- Show that by making a greedy choice you don’t need to solve all the subproblems.
- Show that it is safe to make a greedy choice.

How to Use Greedy?

Solving a problem using Greedy usually takes five steps:

- Find the optimal substructure.
- Recursively define the value of optimal solution.
- Show that by making a greedy choice you don't need to solve all the subproblems.
- Show that it is safe to make a greedy choice.
- Develop the algorithm that implements the greedy strategy.

How to Use Greedy?

Solving a problem using Greedy usually takes five steps:

- Find the optimal substructure.
- Recursively define the value of optimal solution.
- Show that by making a greedy choice you don’t need to solve all the subproblems.
- Show that it is safe to make a greedy choice.
- Develop the algorithm that implements the greedy strategy.

Note: In practice, one can directly present a greedy algorithm, skipping the above steps..

Minimum Spanning Tree

Minimum Spanning Tree

MST:

Minimum Spanning Tree

MST:

Input: An undirected, weighted and connected graph $G = (V, E, w)$.

Minimum Spanning Tree

MST:

Input: An undirected, weighted and connected graph $G = (V, E, w)$.

Output: A spanning tree $G' = (V, E')$ of G, such that $w(E') = \sum_{(u,v)\in E'} w(u, v)$ is minimised.

Minimum Spanning Tree

MST:

Input: An undirected, weighted and connected graph $G = (V, E, w)$.

Output: A spanning tree $G' = (V, E')$ of G, such that $w(E') = \sum_{(u,v)\in E'} w(u, v)$ is minimised.

Example:

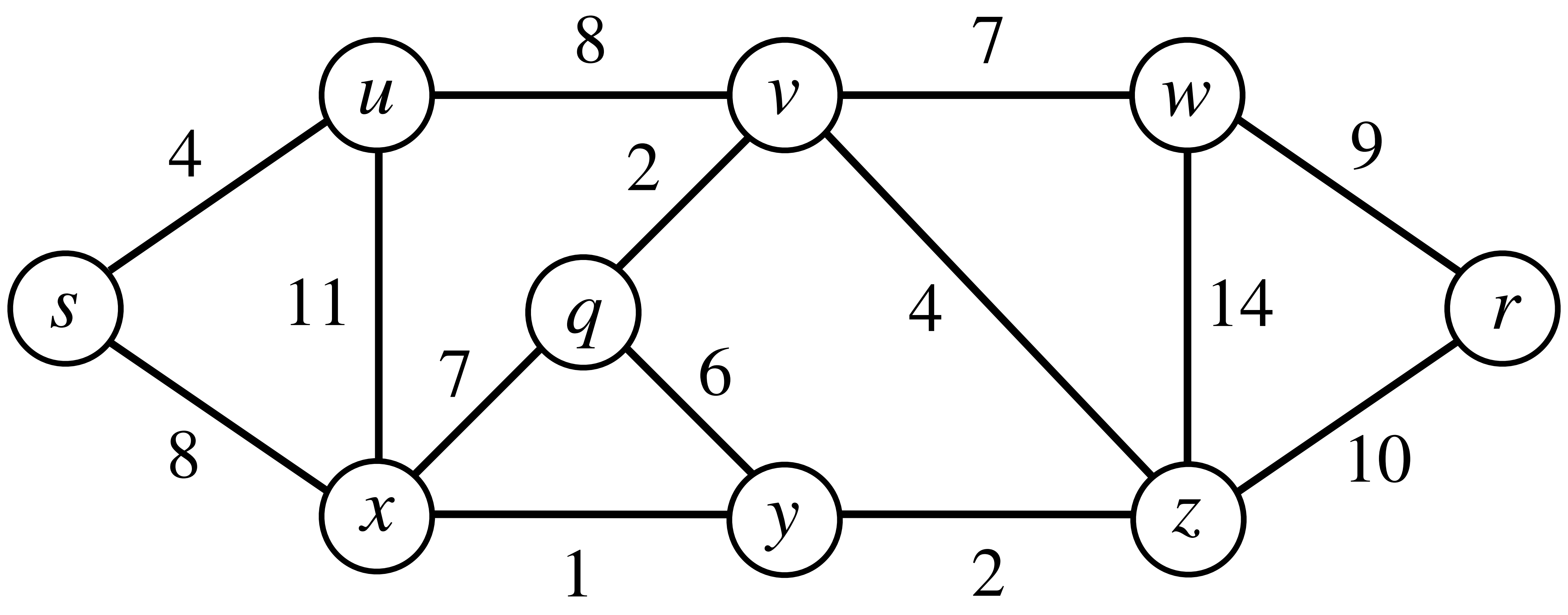
Minimum Spanning Tree

MST:

Input: An undirected, weighted and connected graph $G = (V, E, w)$.

Output: A spanning tree $G' = (V, E')$ of G, such that $w(E') = \sum_{(u,v)\in E'} w(u, v)$ is minimised.

Example:



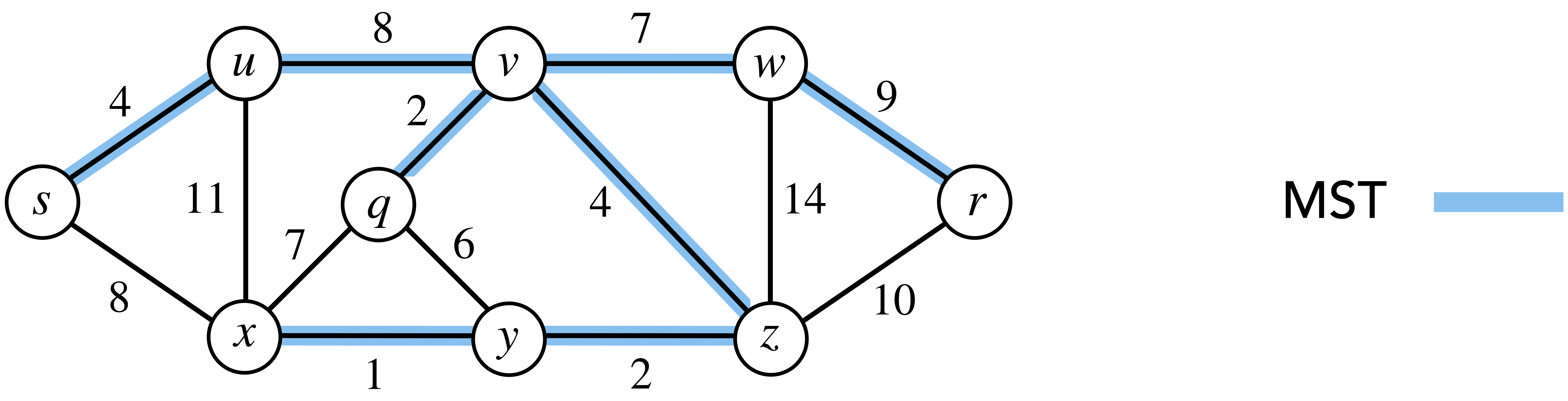
Minimum Spanning Tree

MST:

Input: An undirected, weighted and connected graph $G = (V, E, w)$.

Output: A spanning tree $G' = (V, E')$ of G, such that $w(E') = \sum_{(u,v)\in E'} w(u, v)$ is minimised.

Example:



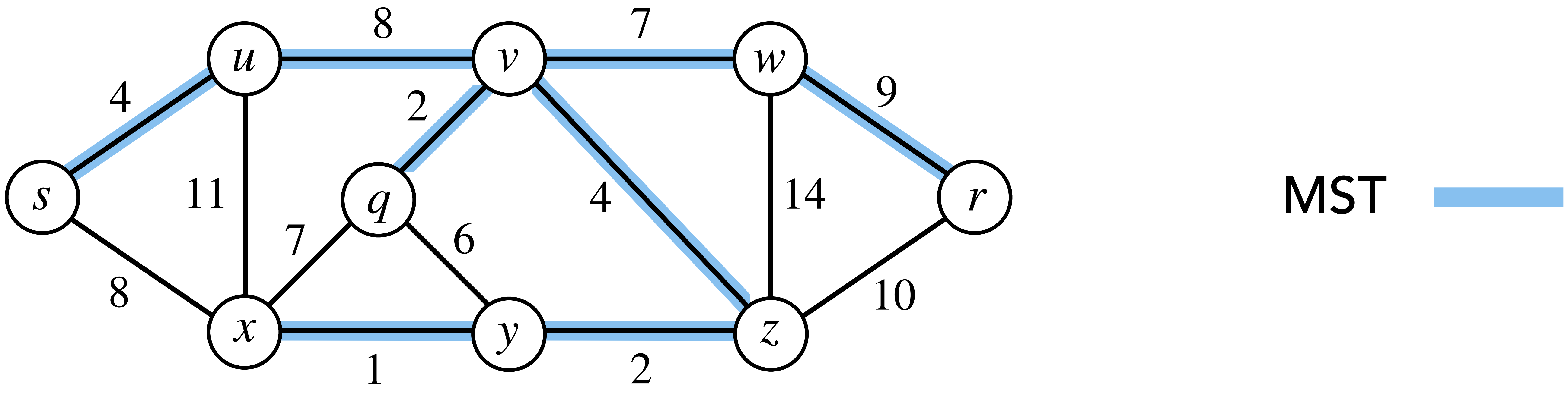
Minimum Spanning Tree

MST:

Input: An undirected, weighted and connected graph $G = (V, E, w)$.

Output: A spanning tree $G' = (V, E')$ of G, such that $w(E') = \sum_{(u,v) \in E'} w(u, v)$ is minimised.

Example:



Note: We will represent an MST as a set of edges.

Cut Connection of MST

Cut Connection of MST

Defn: A cut $C = (S, T)$ of a graph $G = (V, E)$

Cut Connection of MST

Defn: A cut $C = (S, T)$ of a graph $G = (V, E)$ is a partition of V in two subsets S and $T = V - S$.

Cut Connection of MST

Defn: A cut $C = (S, T)$ of a graph $G = (V, E)$ is a partition of V in two subsets S and $T = V - S$.

The cut-set of a cut $C = (S, T)$ is the set of edges that have one endpoint in S and other in T.

Cut Connection of MST

Defn: A cut $C = (S, T)$ of a graph $G = (V, E)$ is a partition of V in two subsets S and $T = V - S$.

The cut-set of a cut $C = (S, T)$ is the set of edges that have one endpoint in S and other in T.

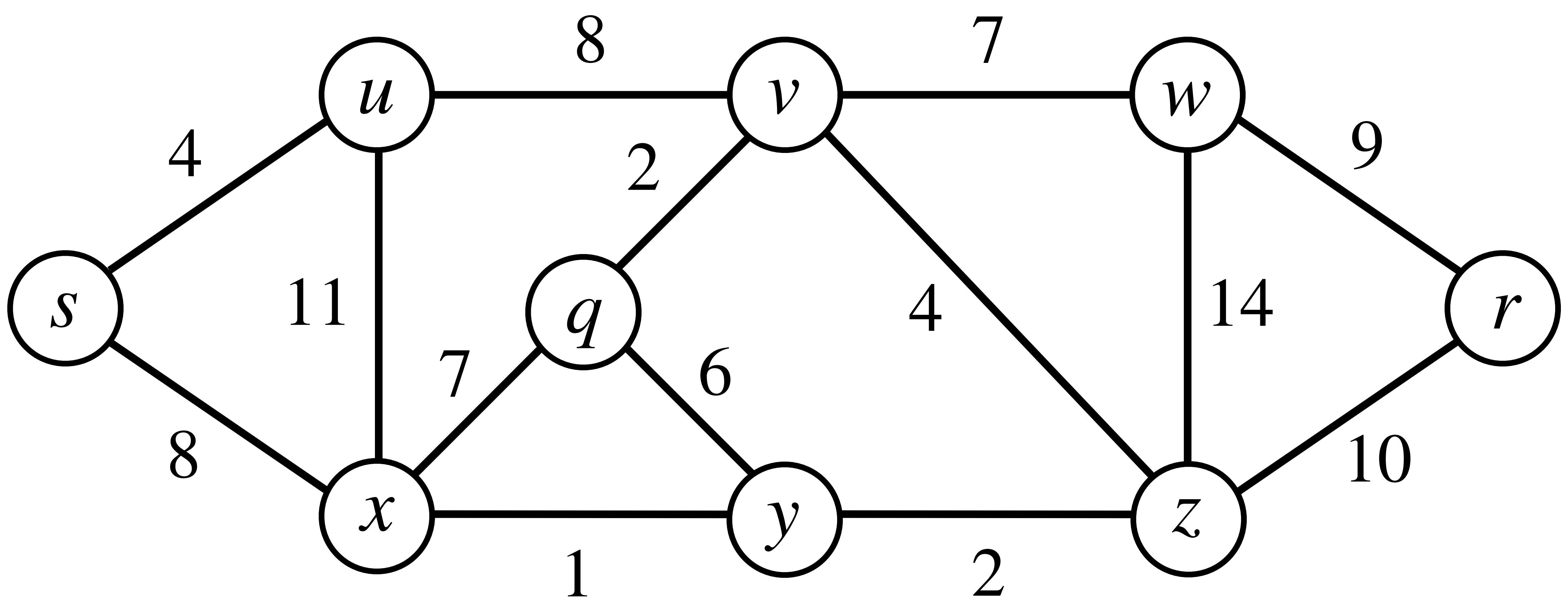
Example:

Cut Connection of MST

Defn: A cut $C = (S, T)$ of a graph $G = (V, E)$ is a partition of V in two subsets S and $T = V - S$.

The cut-set of a cut $C = (S, T)$ is the set of edges that have one endpoint in S and other in T.

Example:

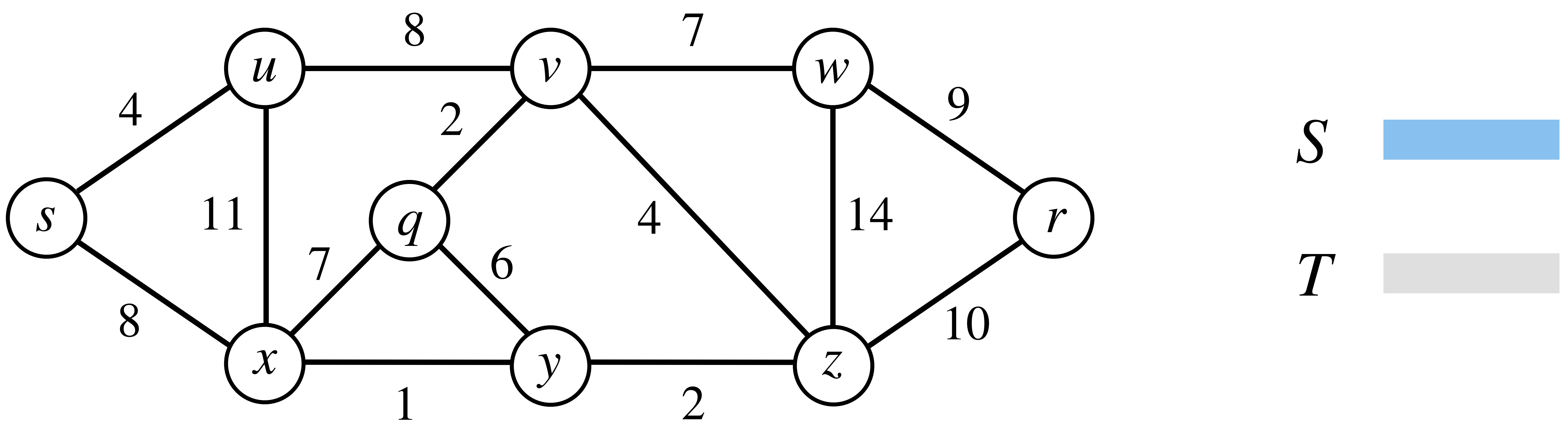


Cut Connection of MST

Defn: A cut $C = (S, T)$ of a graph $G = (V, E)$ is a partition of V in two subsets S and $T = V - S$.

The cut-set of a cut $C = (S, T)$ is the set of edges that have one endpoint in S and other in T.

Example:

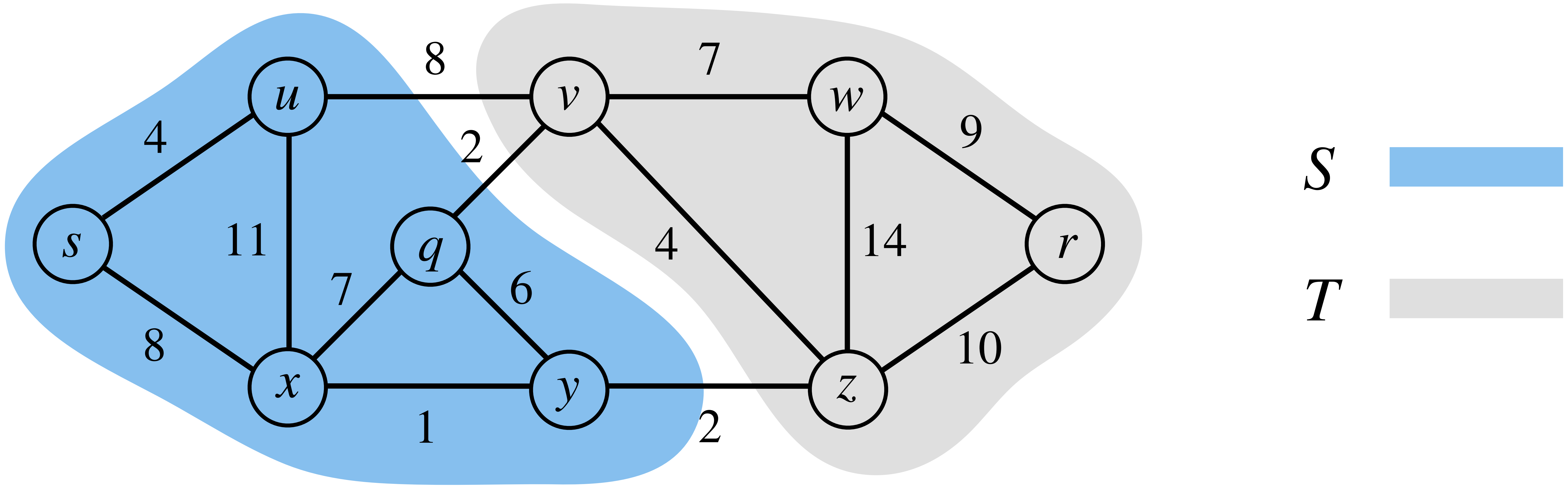


Cut Connection of MST

Defn: A cut $C = (S, T)$ of a graph $G = (V, E)$ is a partition of V in two subsets S and $T = V - S$.

The cut-set of a cut $C = (S, T)$ is the set of edges that have one endpoint in S and other in T.

Example:

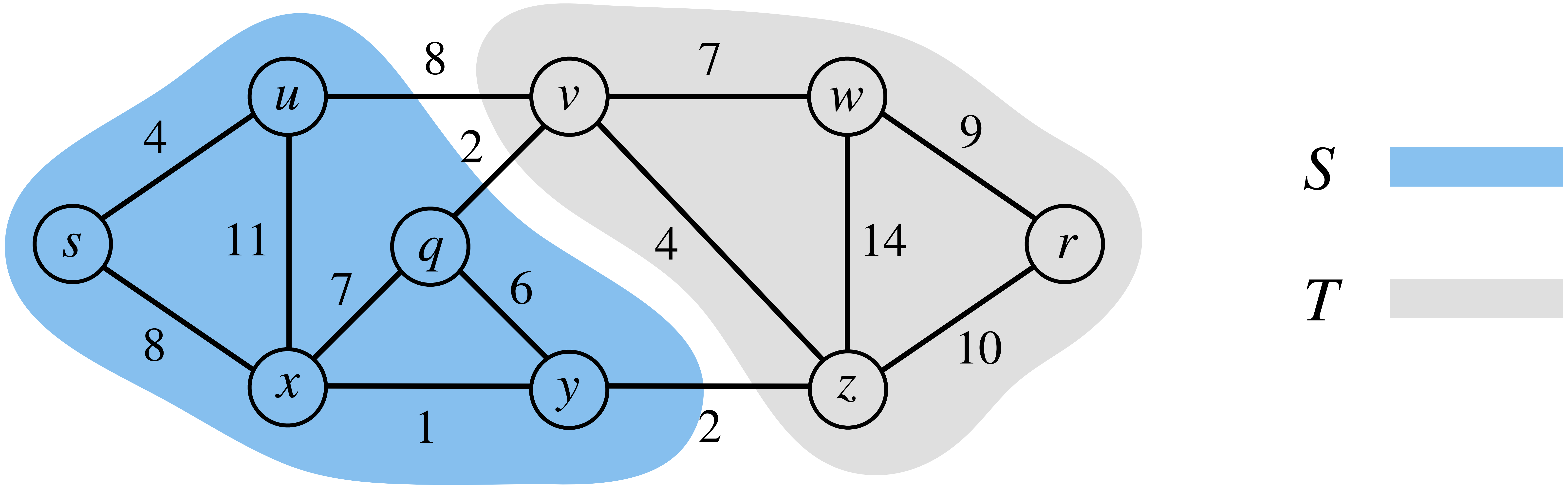


Cut Connection of MST

Defn: A cut $C = (S, T)$ of a graph $G = (V, E)$ is a partition of V in two subsets S and $T = V - S$.

The cut-set of a cut $C = (S, T)$ is the set of edges that have one endpoint in S and other in T.

Example:



The cut-set for cut (S, T) is $\{\{u, v\}, \{q, v\}, \{y, z\}\}$

Cut Connection of MST

Cut Connection of MST

Lemma: Let $C = (S, T)$ be a cut of an undirected, weighted and connected graph $G = (V, E, w)$.

Cut Connection of MST

Lemma: Let $C = (S, T)$ be a cut of an undirected, weighted and connected graph $G = (V, E, w)$.
If e is the least weight edge in the cut-set of C,

Cut Connection of MST

Lemma: Let $C = (S, T)$ be a cut of an undirected, weighted and connected graph $G = (V, E, w)$. If e is the least weight edge in the cut-set of C, then e is part of some MST of G.

Cut Connection of MST

Lemma: Let $C = (S, T)$ be a cut of an undirected, weighted and connected graph $G = (V, E, w)$. If e is the least weight edge in the cut-set of C, then e is part of some MST of G.

Proof: On the next slide.

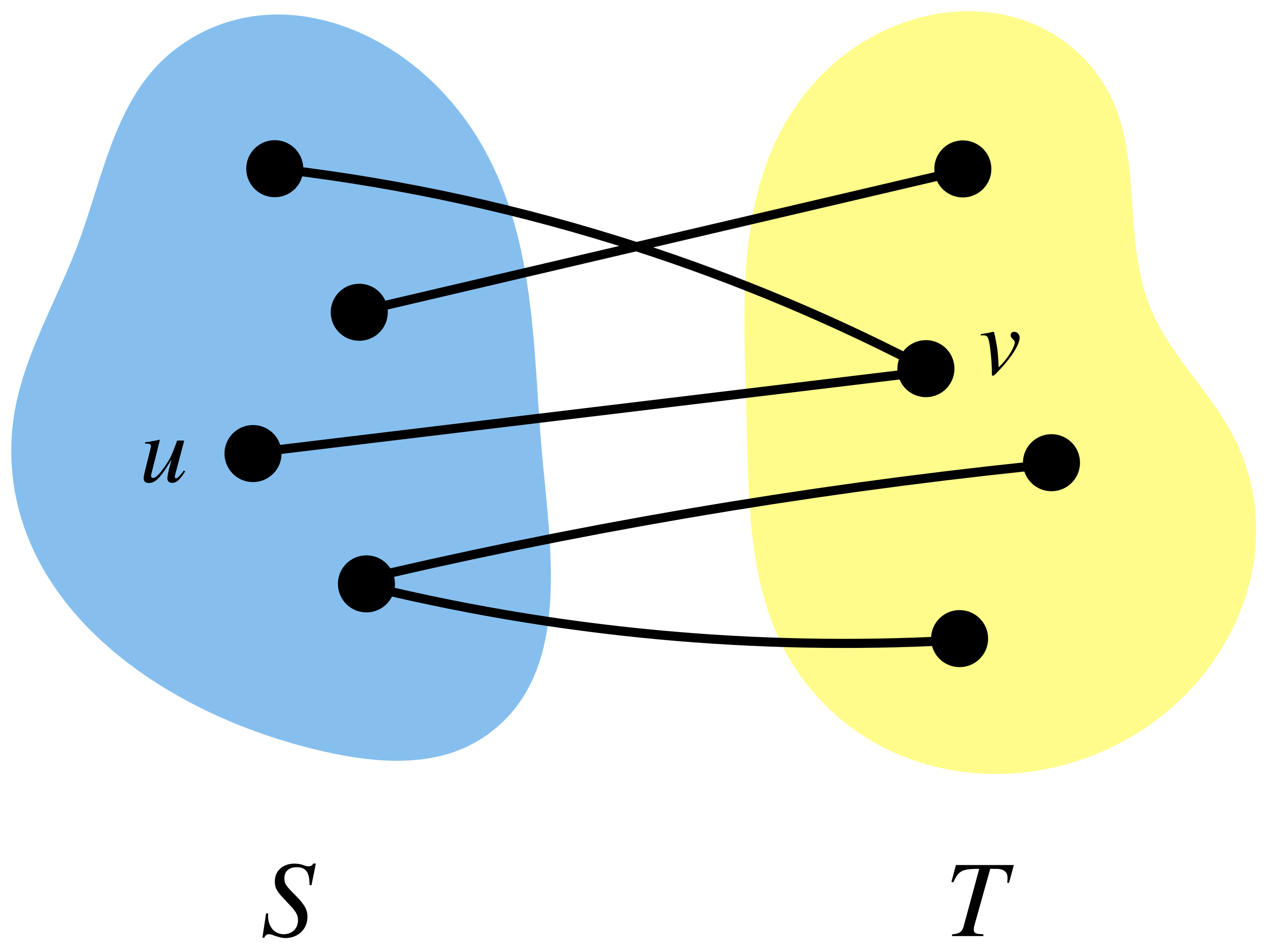
Cut Connection of MST

Cut Connection of MST

Proof:

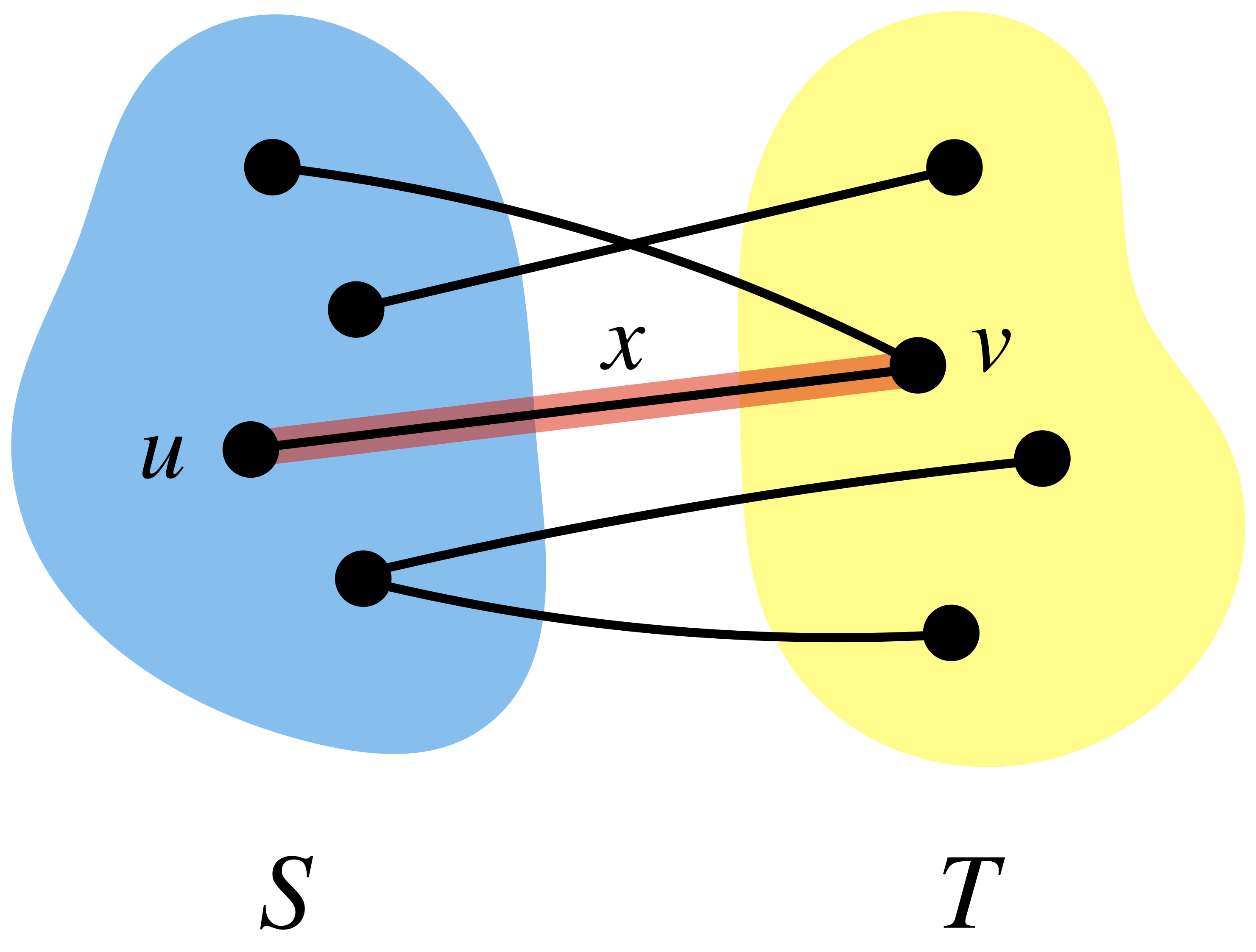
Cut Connection of MST

Proof:



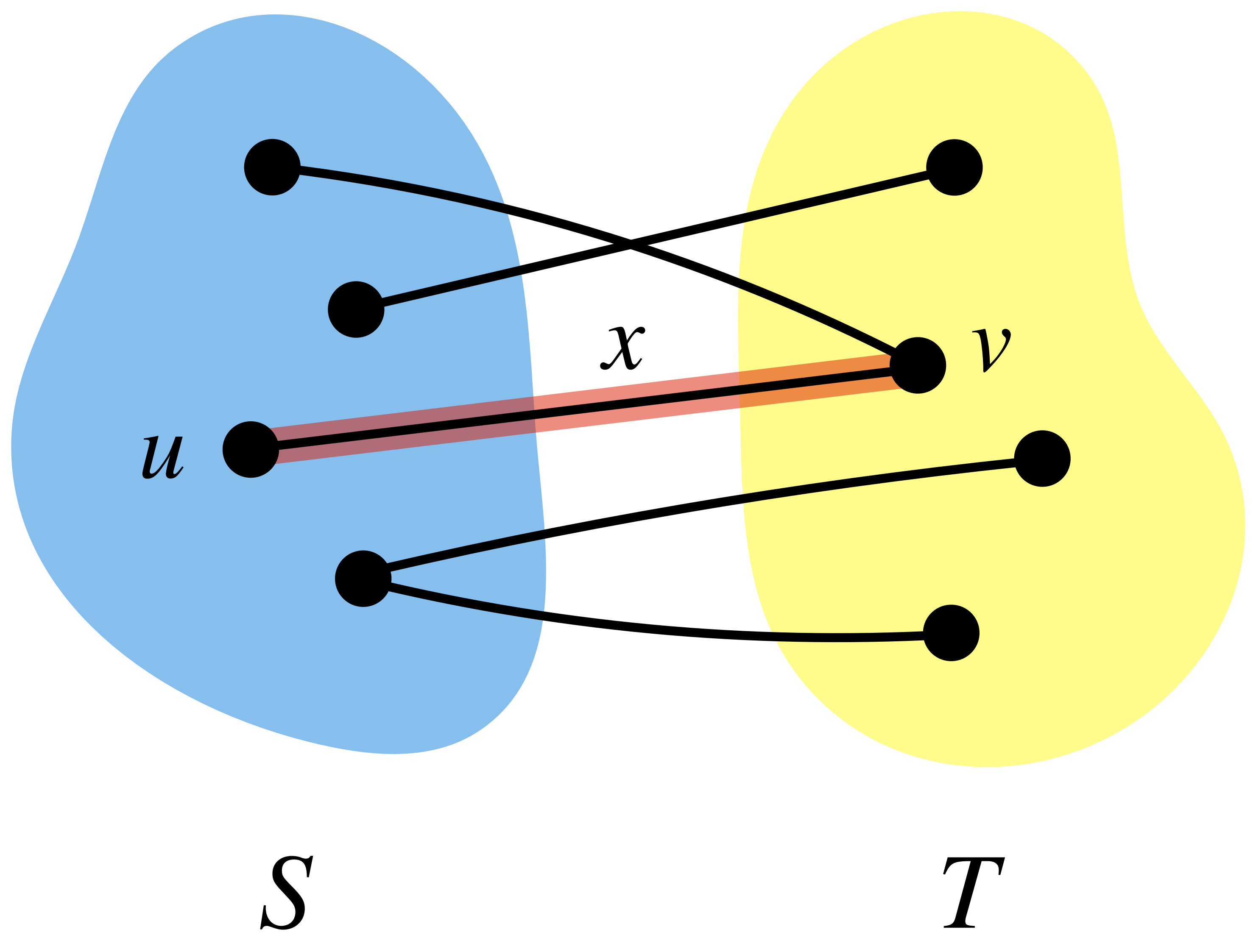
Cut Connection of MST

Proof:



Cut Connection of MST

Proof: Let $\{u, v\}$ be a least weight edge in the cut-set of C with weight x.



Cut Connection of MST

Proof: Let $\{u, v\}$ be a least weight edge in the cut-set of C with weight x.

Cut Connection of MST

Proof: Let $\{u, v\}$ be a least weight edge in the cut-set of C with weight x.

Let T be an MST that does not contain $\{u, v\}$.

Cut Connection of MST

Proof: Let $\{u, v\}$ be a least weight edge in the cut-set of C with weight x.

Let T be an MST that does not contain $\{u, v\}$. ← If we cannot pick such a T we are done.

Cut Connection of MST

Proof: Let $\{u, v\}$ be a least weight edge in the cut-set of C with weight x.

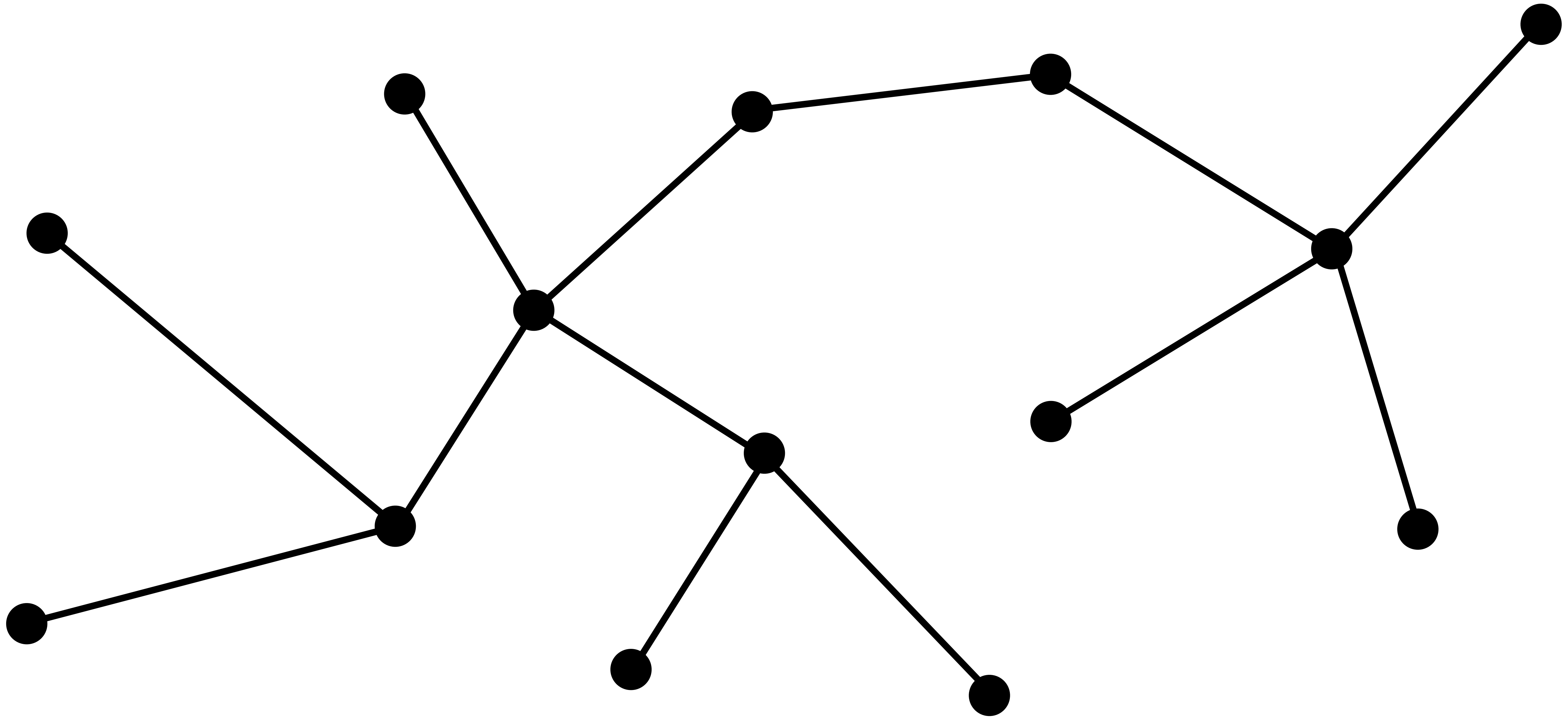
Let T be an MST that does not contain $\{u, v\}$.

Cut Connection of MST

Proof: Let $\{u, v\}$ be a least weight edge in the cut-set of C with weight x.

Let T be an MST that does not contain $\{u, v\}$.

T

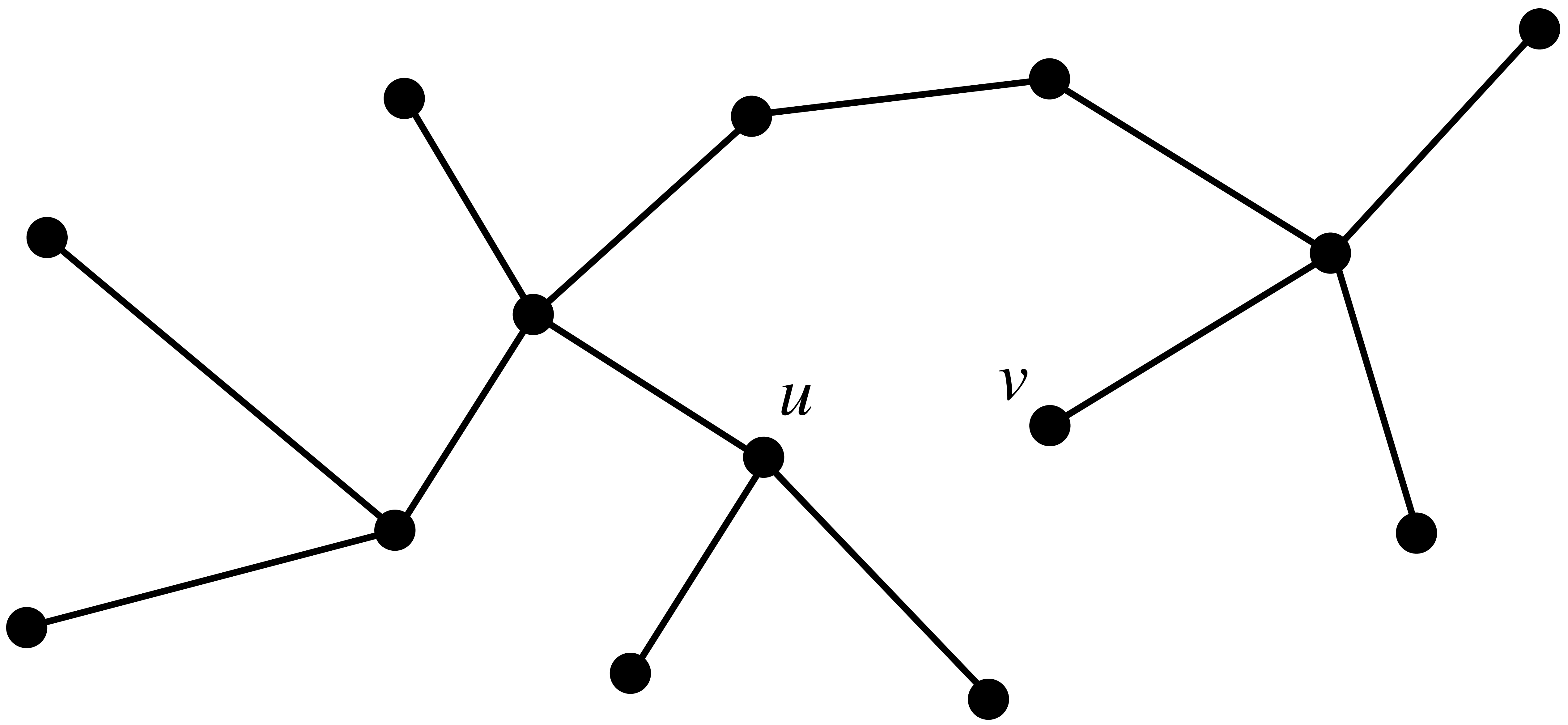


Cut Connection of MST

Proof: Let $\{u, v\}$ be a least weight edge in the cut-set of C with weight x.

Let T be an MST that does not contain $\{u, v\}$.

T

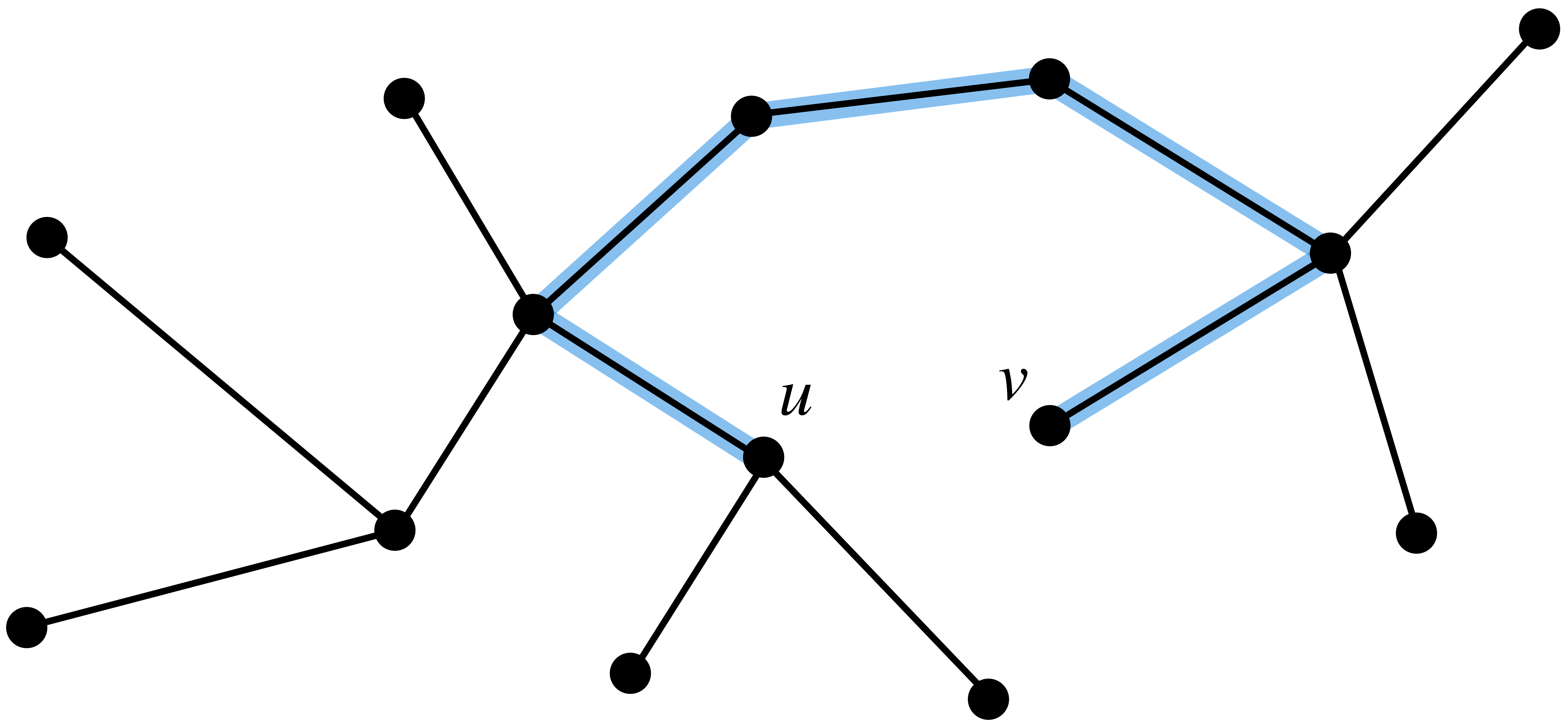


Cut Connection of MST

Proof: Let $\{u, v\}$ be a least weight edge in the cut-set of C with weight x.

Let T be an MST that does not contain $\{u, v\}$.

T

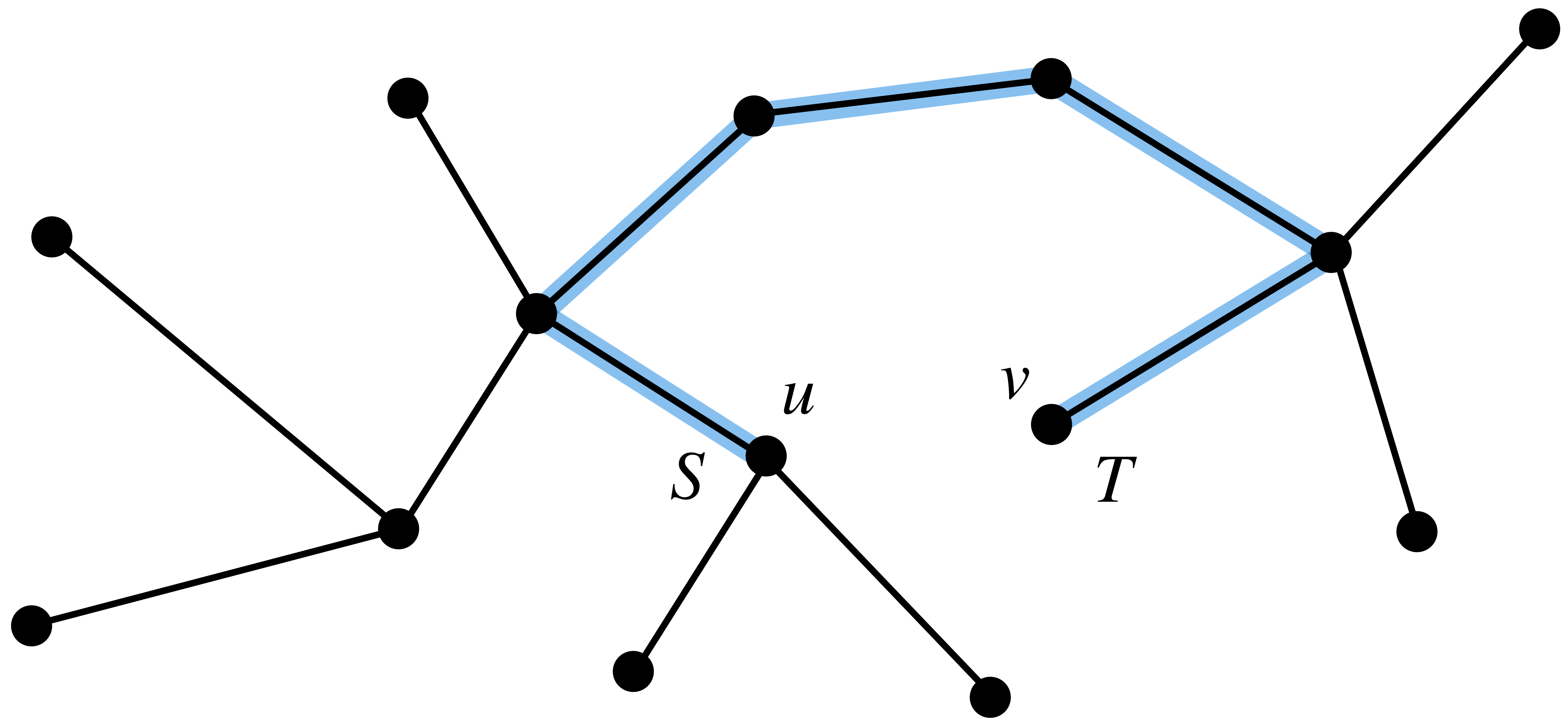


Cut Connection of MST

Proof: Let $\{u, v\}$ be a least weight edge in the cut-set of C with weight x.

Let T be an MST that does not contain $\{u, v\}$.

T

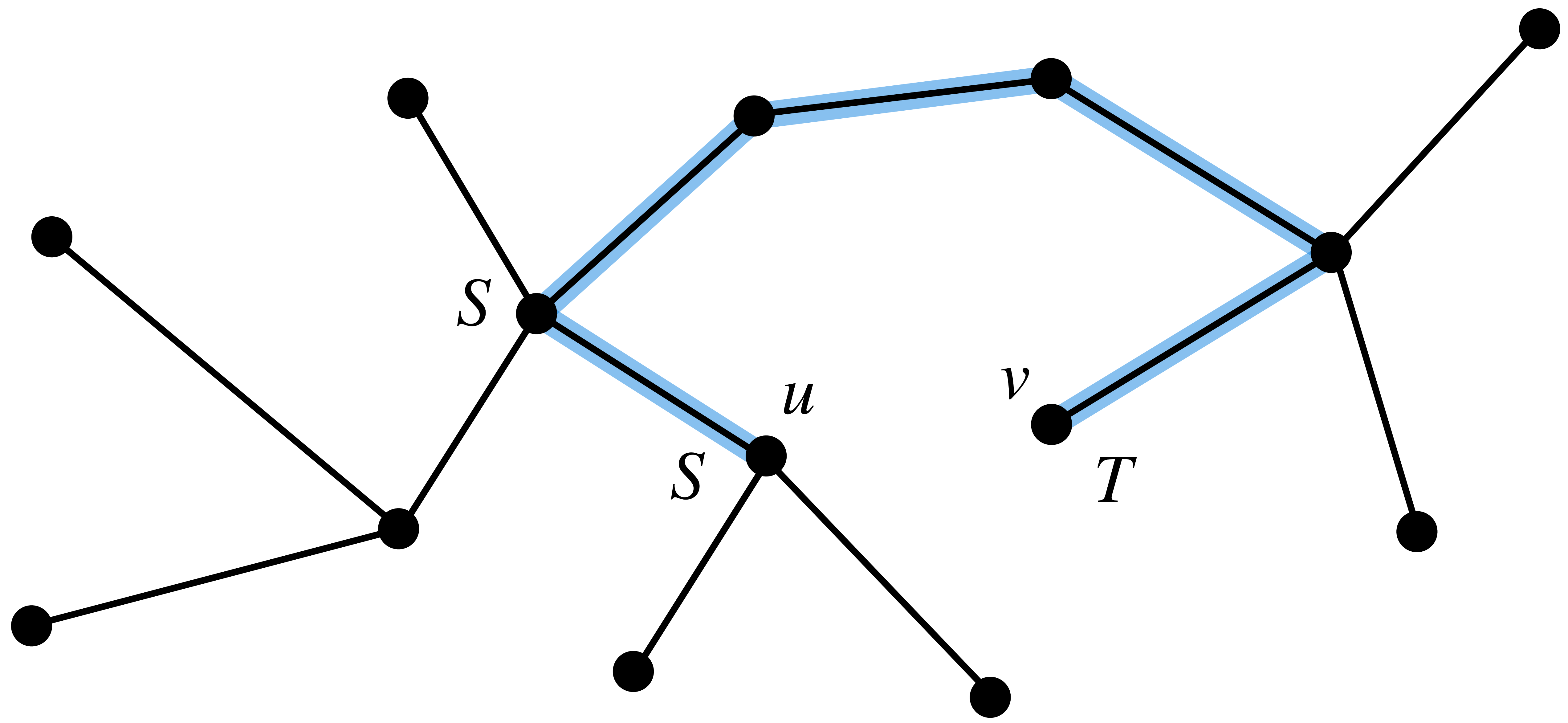


Cut Connection of MST

Proof: Let $\{u, v\}$ be a least weight edge in the cut-set of C with weight x.

Let T be an MST that does not contain $\{u, v\}$.

T

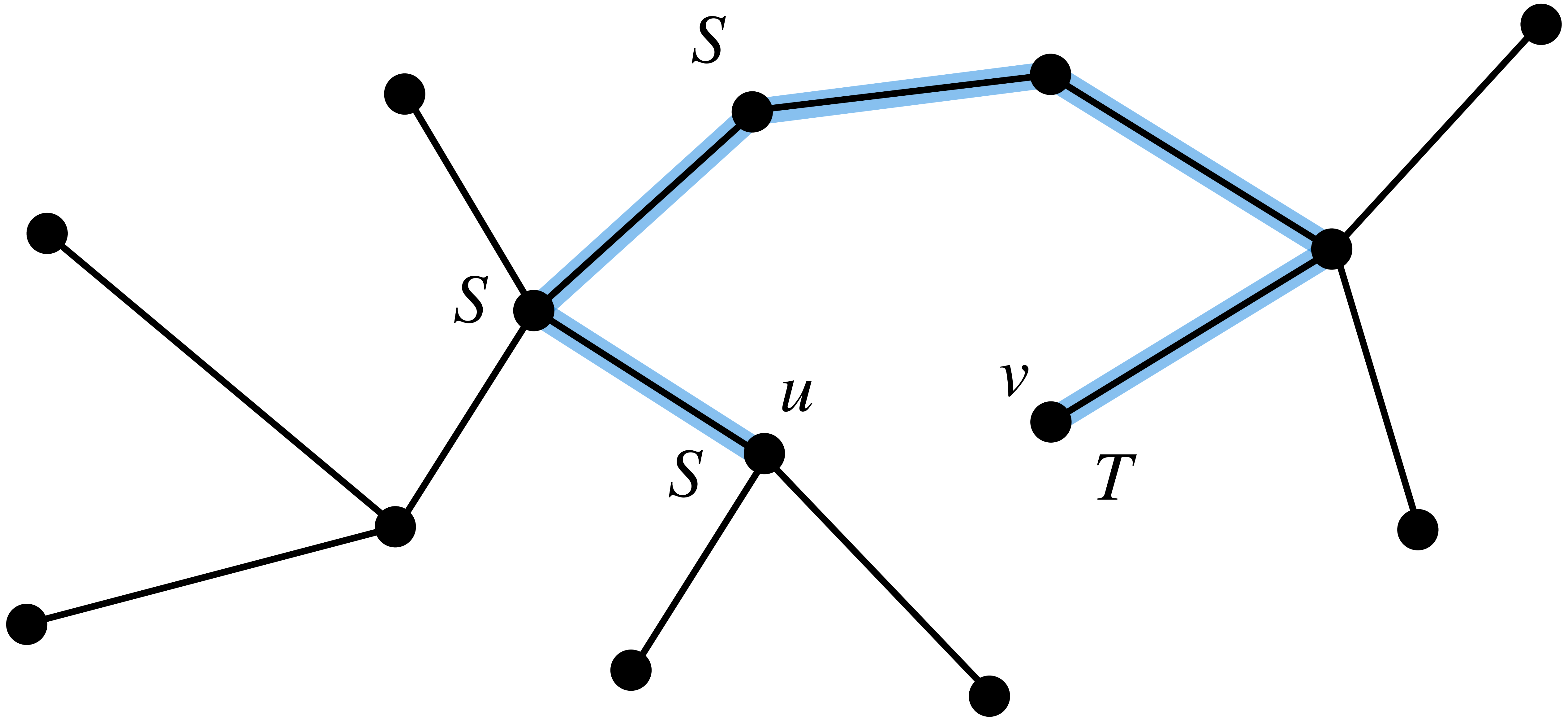


Cut Connection of MST

Proof: Let $\{u, v\}$ be a least weight edge in the cut-set of C with weight x.

Let T be an MST that does not contain $\{u, v\}$.

T

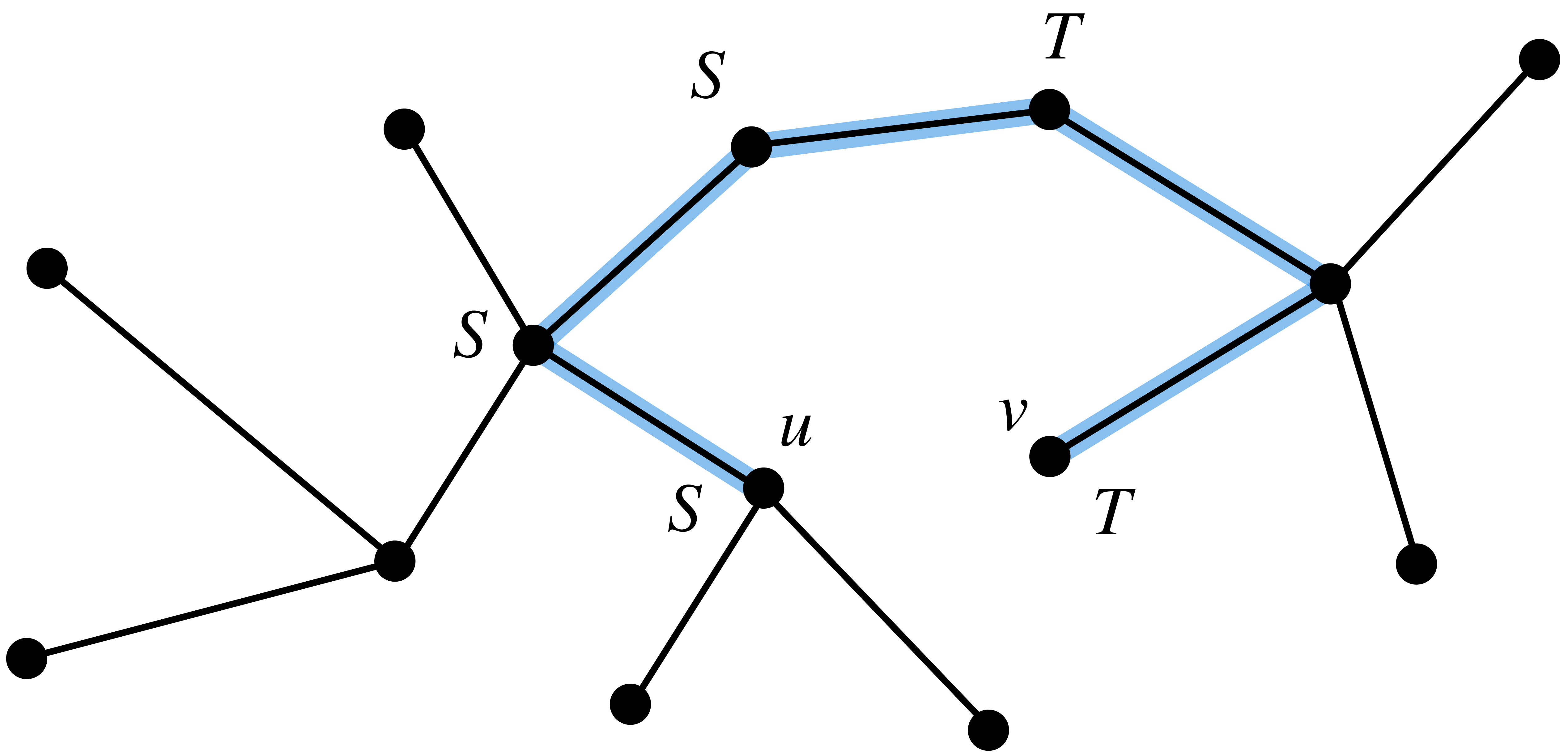


Cut Connection of MST

Proof: Let $\{u, v\}$ be a least weight edge in the cut-set of C with weight x.

Let T be an MST that does not contain $\{u, v\}$.

T

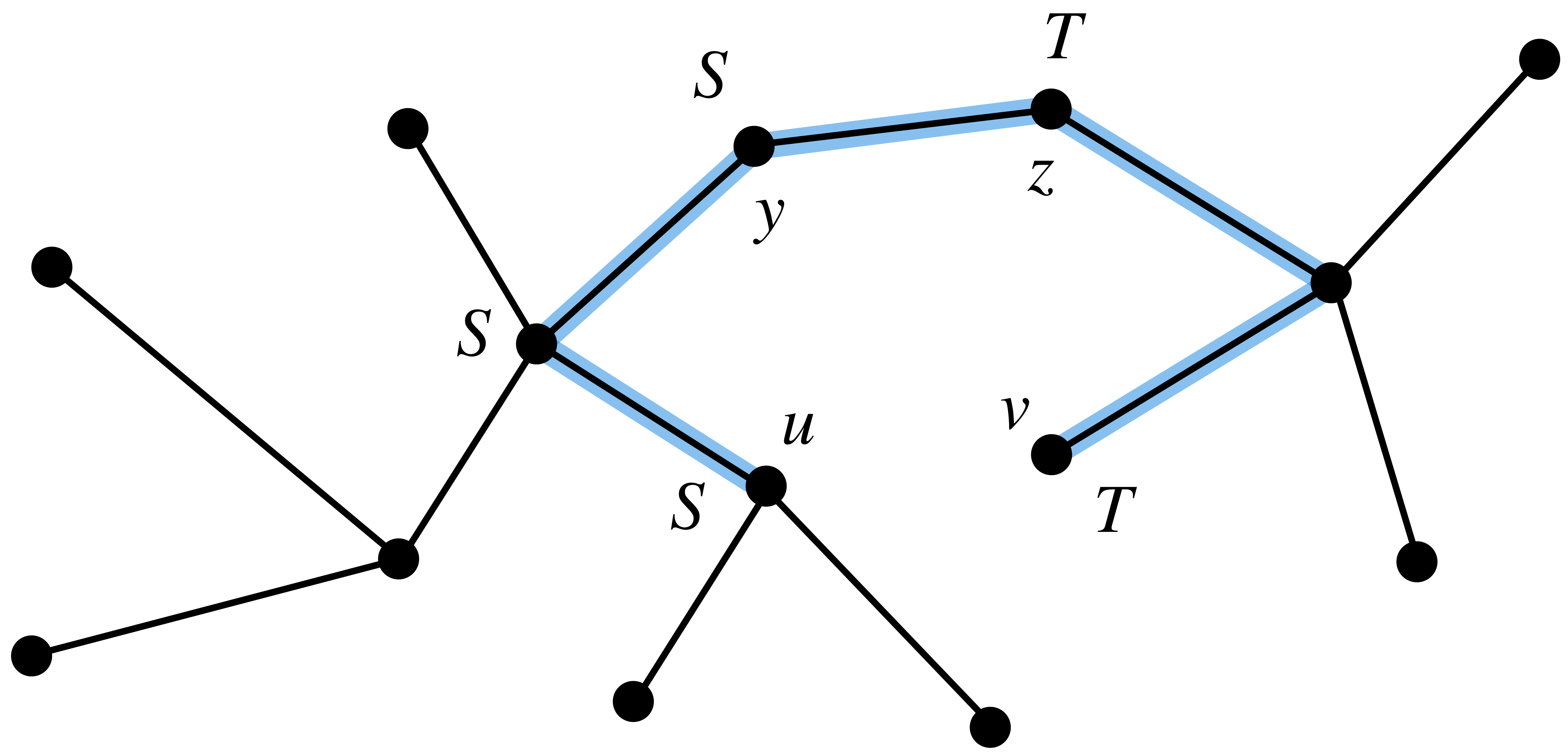


Cut Connection of MST

Proof: Let $\{u, v\}$ be a least weight edge in the cut-set of C with weight x.

Let T be an MST that does not contain $\{u, v\}$.

T

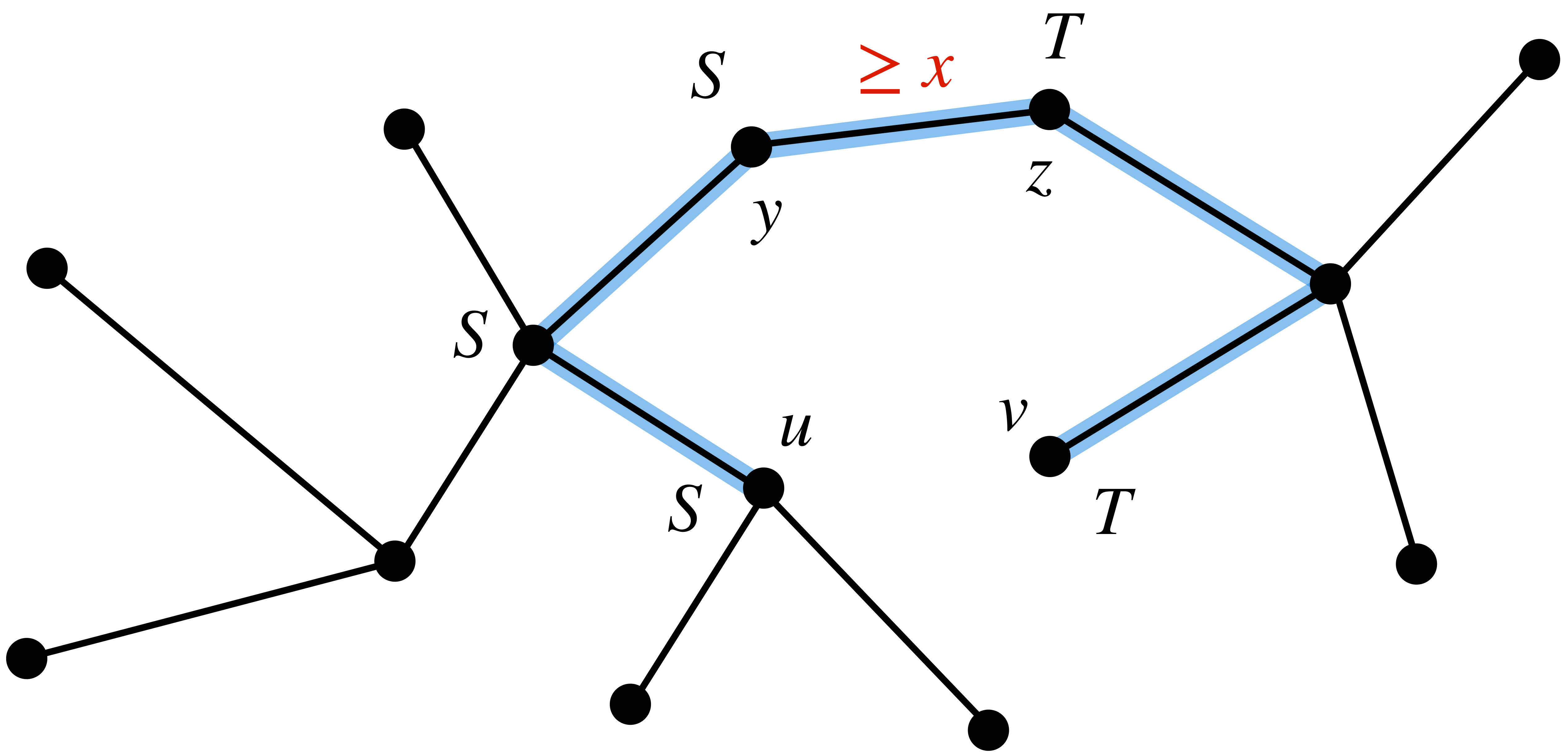


Cut Connection of MST

Proof: Let $\{u, v\}$ be a least weight edge in the cut-set of C with weight x.

Let T be an MST that does not contain $\{u, v\}$.

T

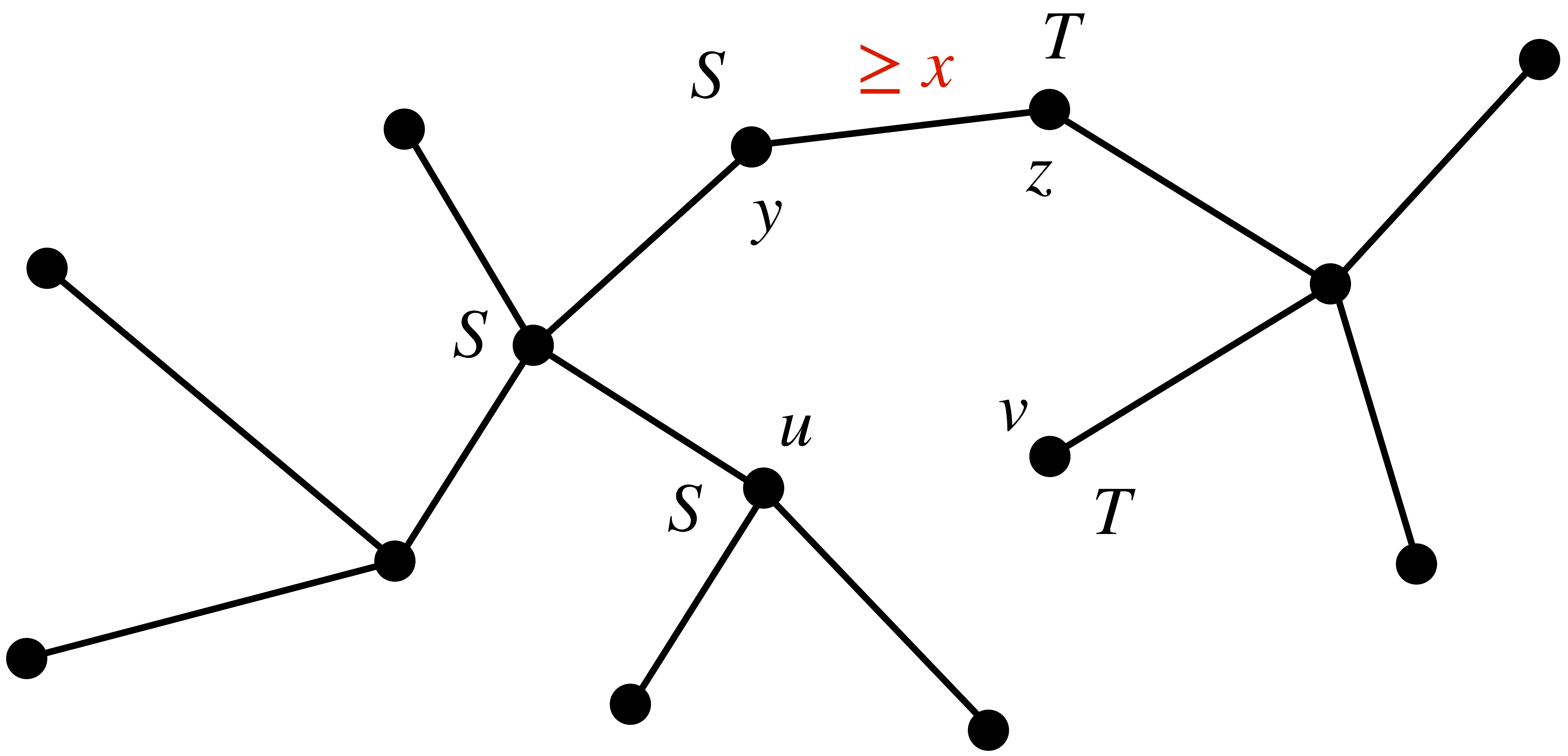


Cut Connection of MST

Proof: Let $\{u, v\}$ be a least weight edge in the cut-set of C with weight x.

Let T be an MST that does not contain $\{u, v\}$.

T

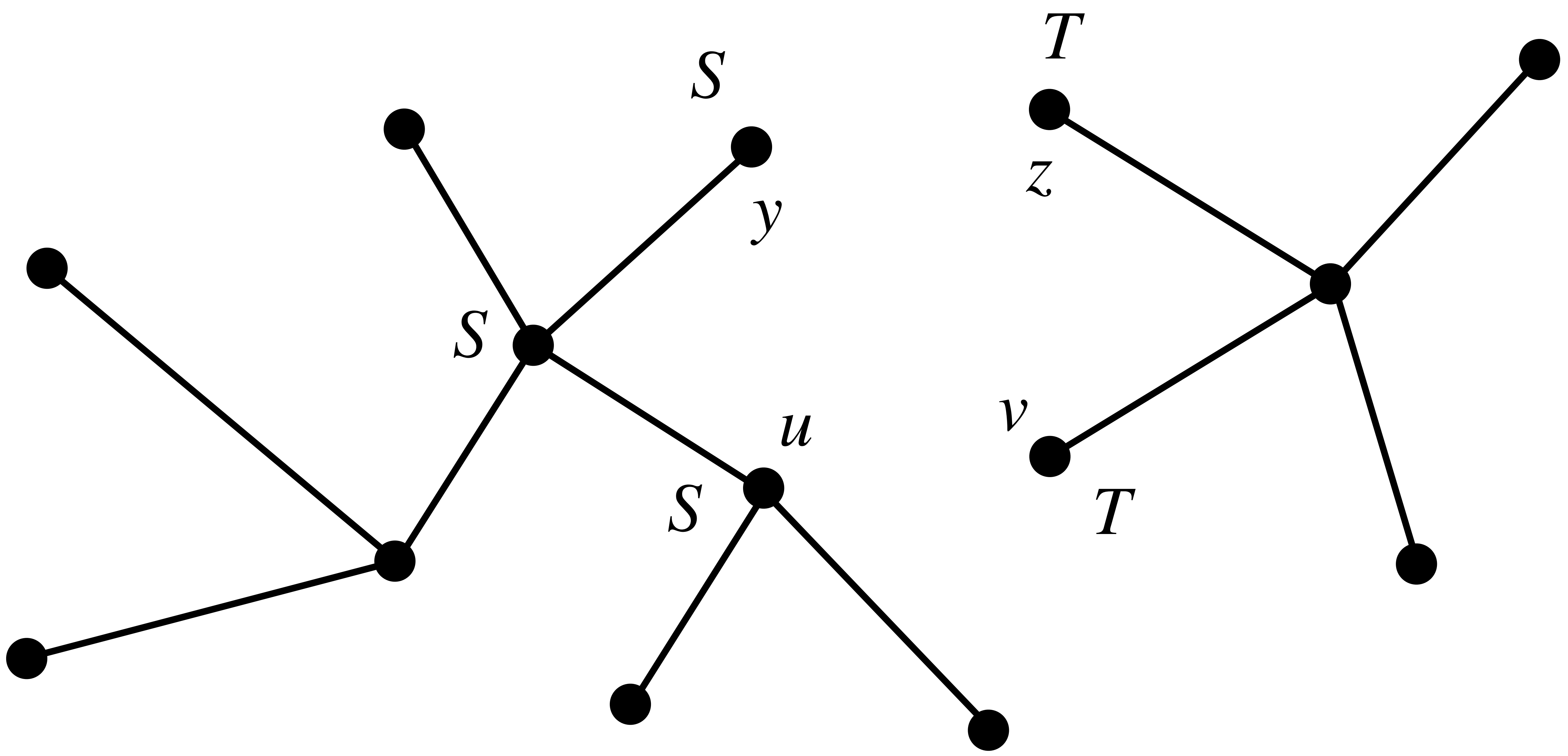


Cut Connection of MST

Proof: Let $\{u, v\}$ be a least weight edge in the cut-set of C with weight x.

Let T be an MST that does not contain $\{u, v\}$.

T

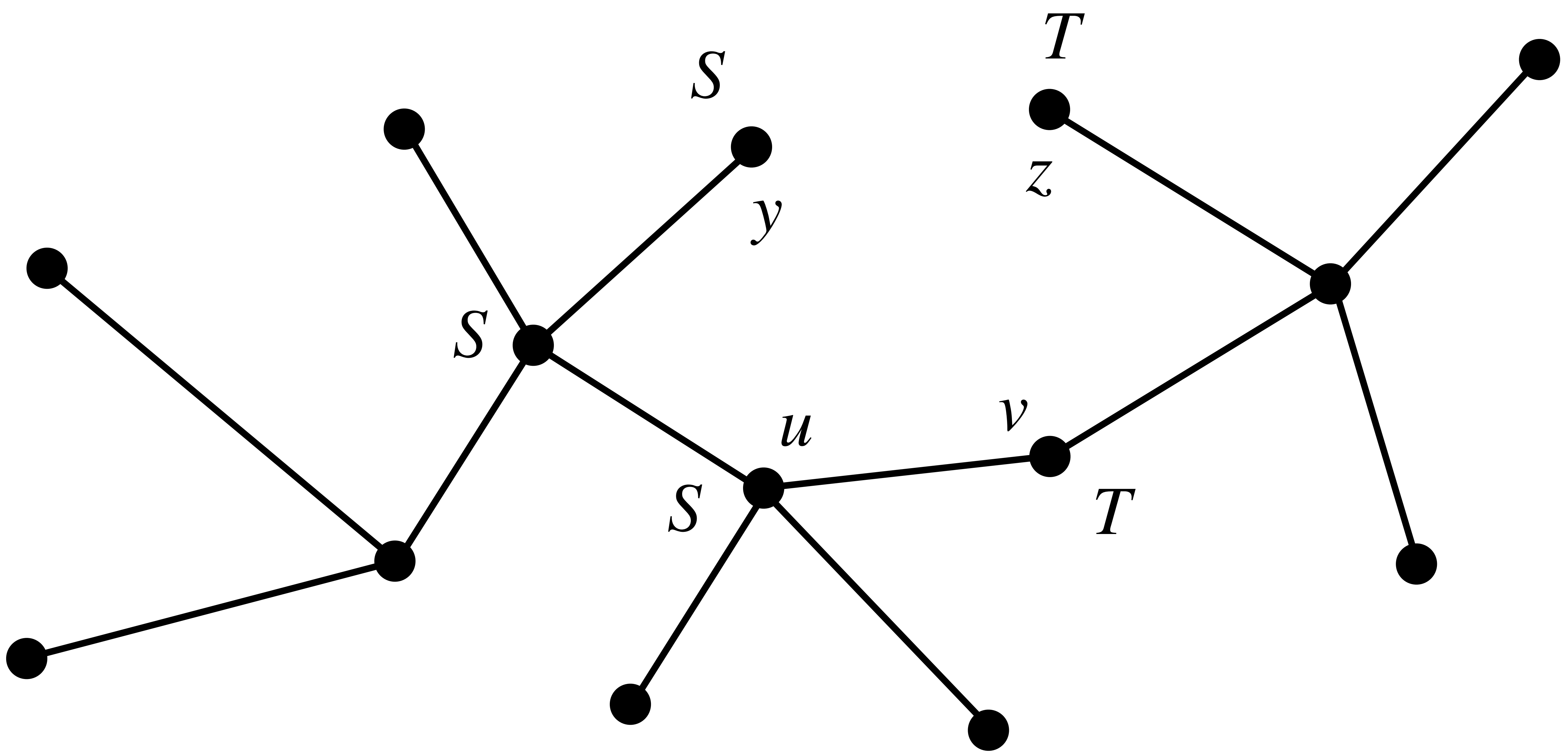


Cut Connection of MST

Proof: Let $\{u, v\}$ be a least weight edge in the cut-set of C with weight x.

Let T be an MST that does not contain $\{u, v\}$.

T

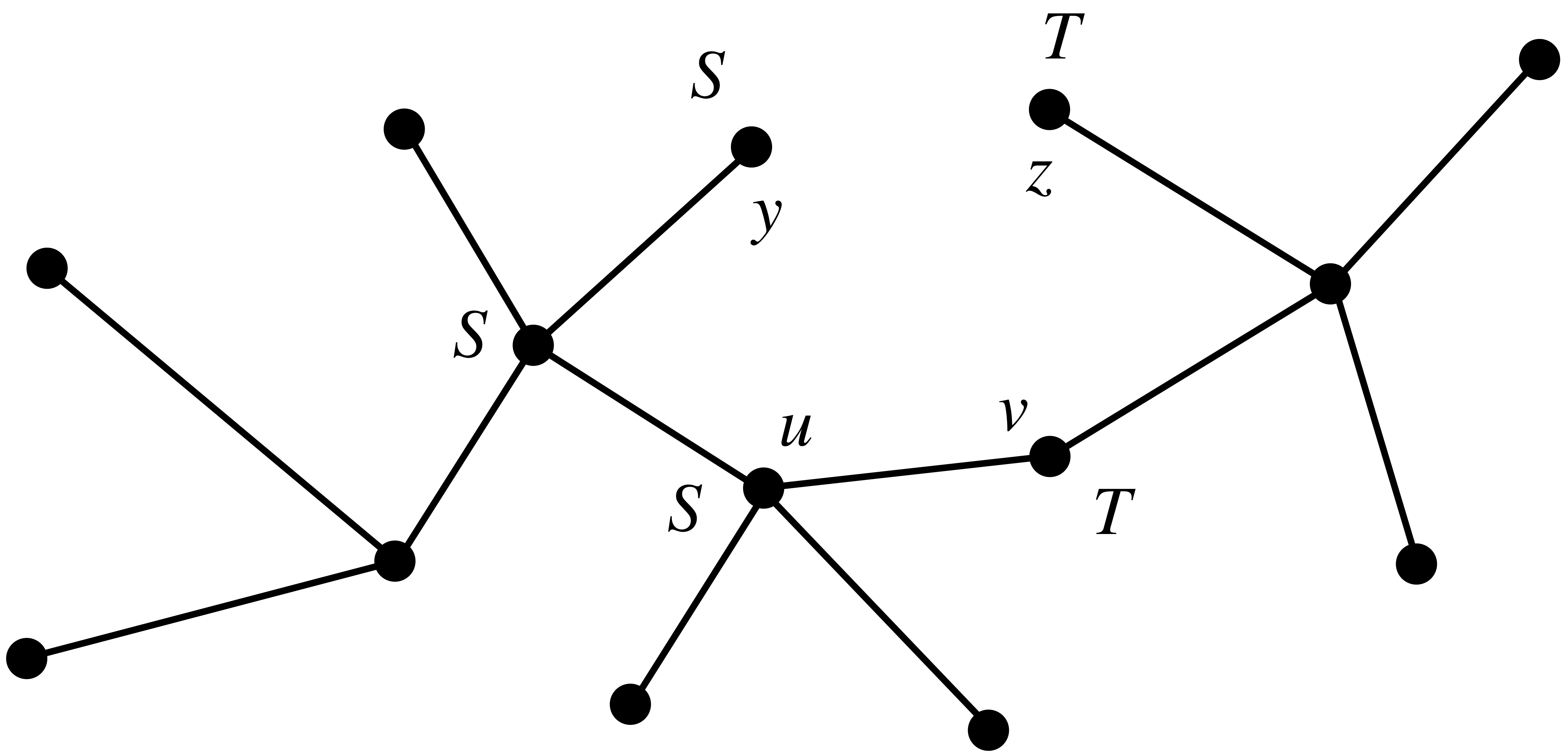


Cut Connection of MST

Proof: Let $\{u, v\}$ be a least weight edge in the cut-set of C with weight x.

Let T be an MST that does not contain $\{u, v\}$.

T'

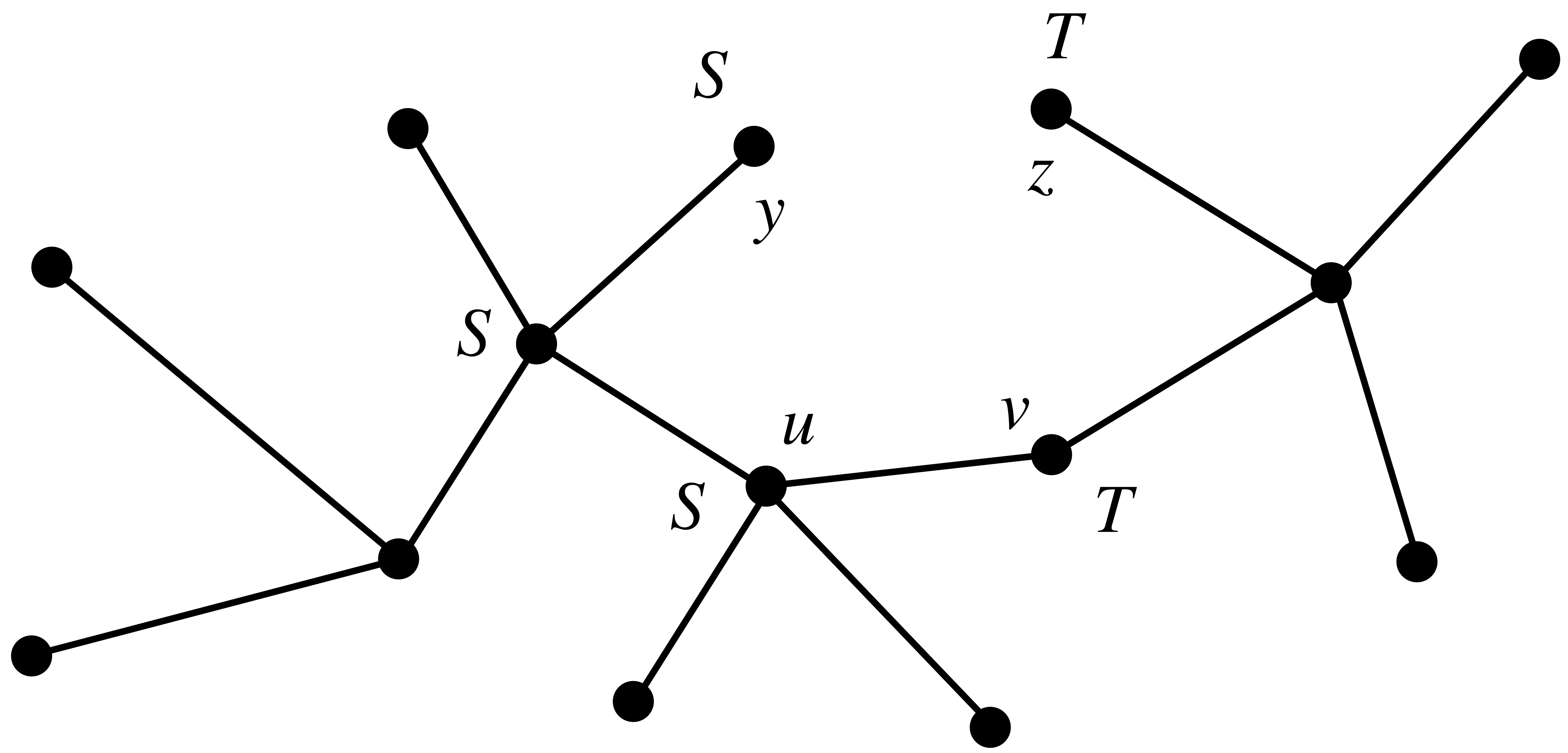


Cut Connection of MST

Proof: Let $\{u, v\}$ be a least weight edge in the cut-set of C with weight x.

Let T be an MST that does not contain $\{u, v\}$.

T'



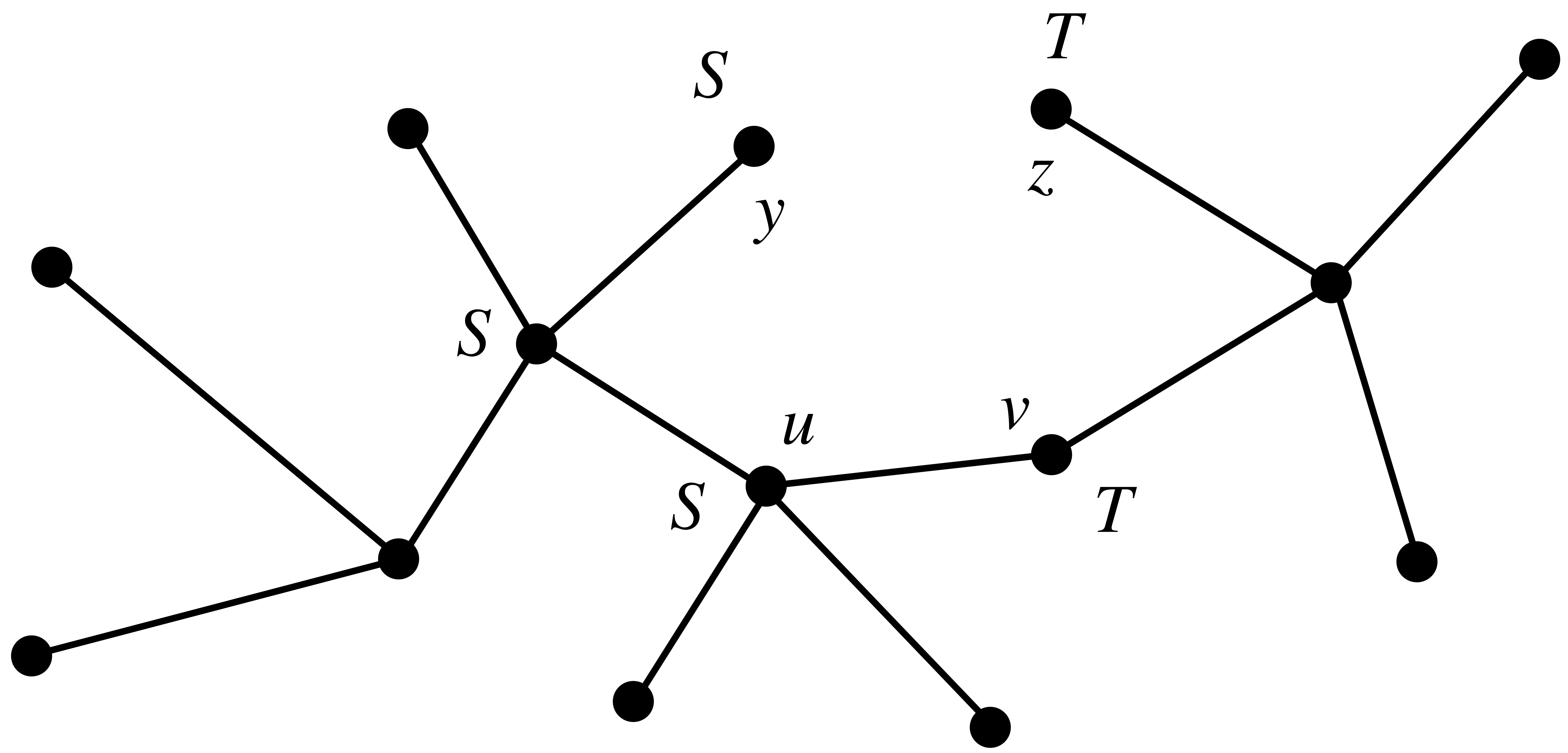
Then, $T' = T - \{y, z\} + \{u, v\}$ will a spanning tree with $w(T') \leq w(T)$.

Cut Connection of MST

Proof: Let $\{u, v\}$ be a least weight edge in the cut-set of C with weight x.

Let T be an MST that does not contain $\{u, v\}$.

T'



Then, $T' = T - \{y, z\} + \{u, v\}$ will a spanning tree with $w(T') \leq w(T)$.

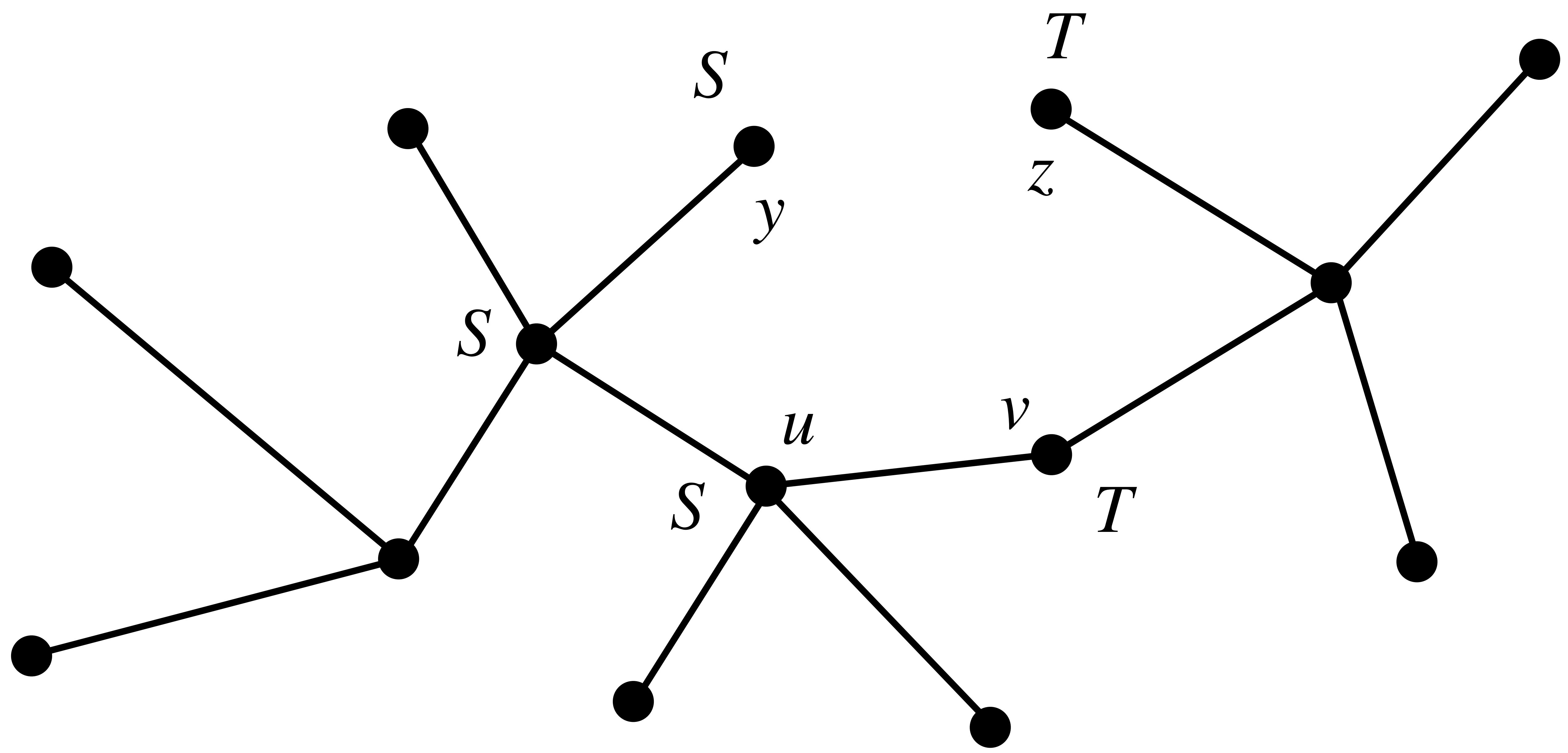
$w(T') < w(T)$ is not possible as T is an MST.

Cut Connection of MST

Proof: Let $\{u, v\}$ be a least weight edge in the cut-set of C with weight x.

Let T be an MST that does not contain $\{u, v\}$.

T'



Then, $T' = T - \{y, z\} + \{u, v\}$ will a spanning tree with $w(T') \leq w(T)$.

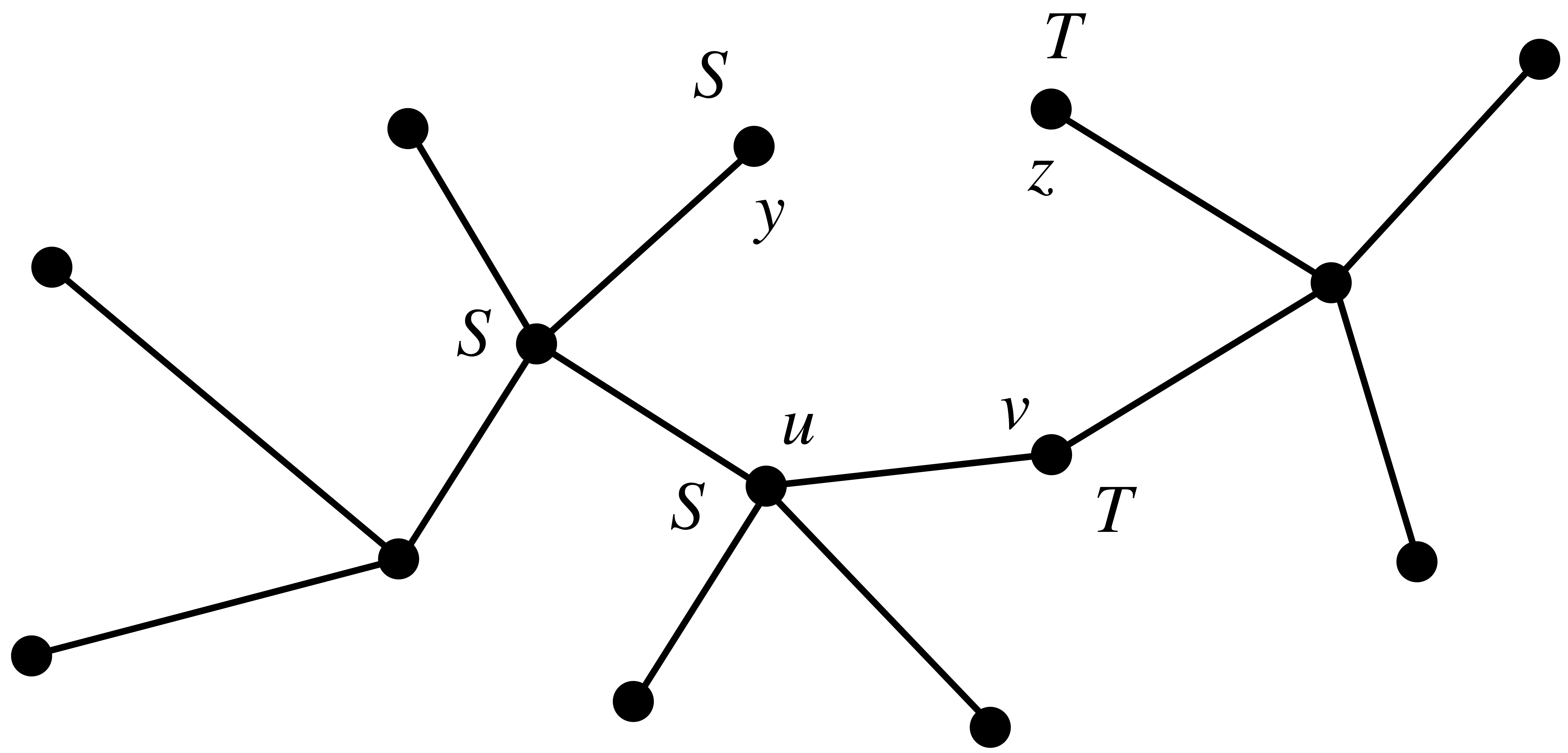
$w(T') < w(T)$ is not possible as T is an MST. Hence, T' is also an MST.

Cut Connection of MST

Proof: Let $\{u, v\}$ be a least weight edge in the cut-set of C with weight x.

Let T be an MST that does not contain $\{u, v\}$.

T'



Then, $T' = T - \{y, z\} + \{u, v\}$ will a spanning tree with $w(T') \leq w(T)$.

$w(T') < w(T)$ is not possible as T is an MST. Hence, T' is also an MST. ∎