

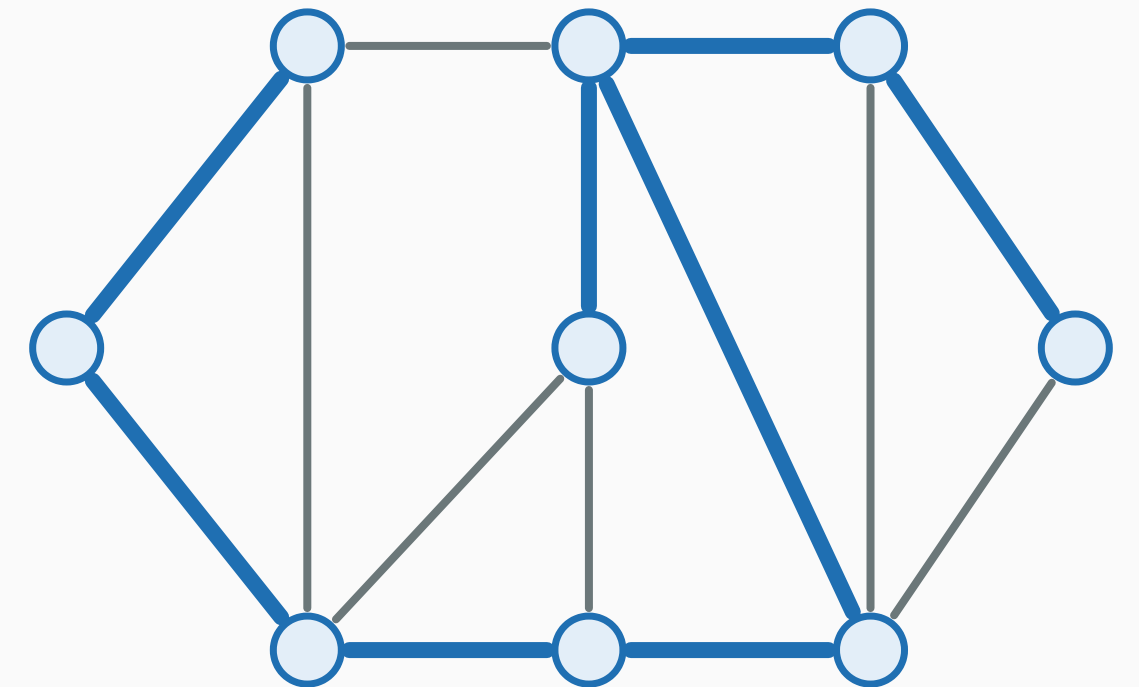
ALGORITHMS AND DATA STRUCTURES · LECTURE 7

Greedy algorithms and minimum spanning trees

Heng Guo

School of Informatics, University of Edinburgh

Monday 12 October 2026 · 16:10–17:00



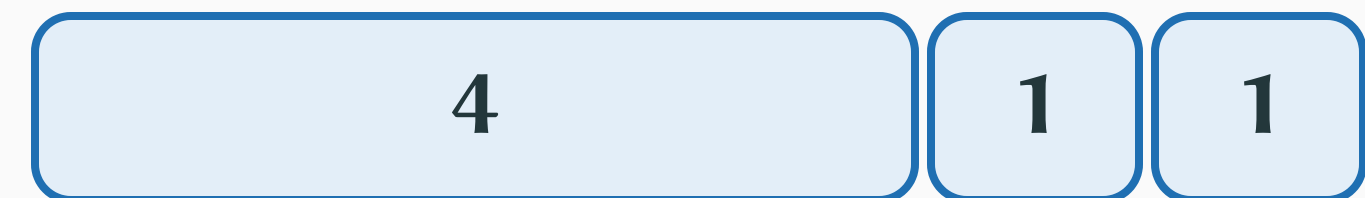
Last time: the last lecture on divide and conquer (average-case complexity).

GREEDY. Build the solution in small steps. At each step take the choice that looks best *now*, and never undo it.

? *Greedy can be wrong.*
When is the locally optimal solution globally optimal?

Today: minimum spanning trees.
Three greedy algorithms, and why they work.

Pay 6 with coins of value 1, 3, 4:



greedy (largest coin first): 3 coins



best: 2 coins

Graph $G = (V, E)$: nodes V , edges E ; $n = |V|$, $m = |E|$.

An edge is $\{v, w\}$ (undirected) or (v, w) (directed).

Neighbours of v : the nodes joined to v by an edge.

Degree $\deg v$: the number of neighbours of v .

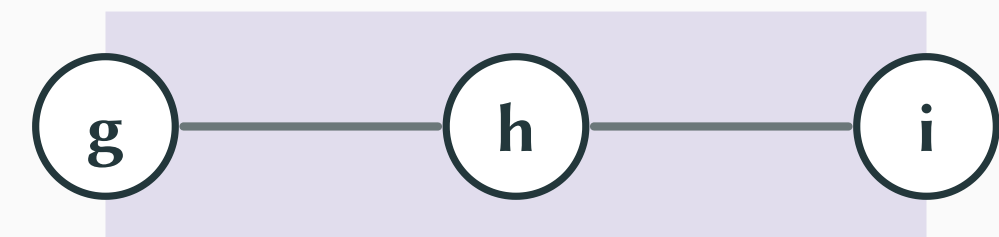
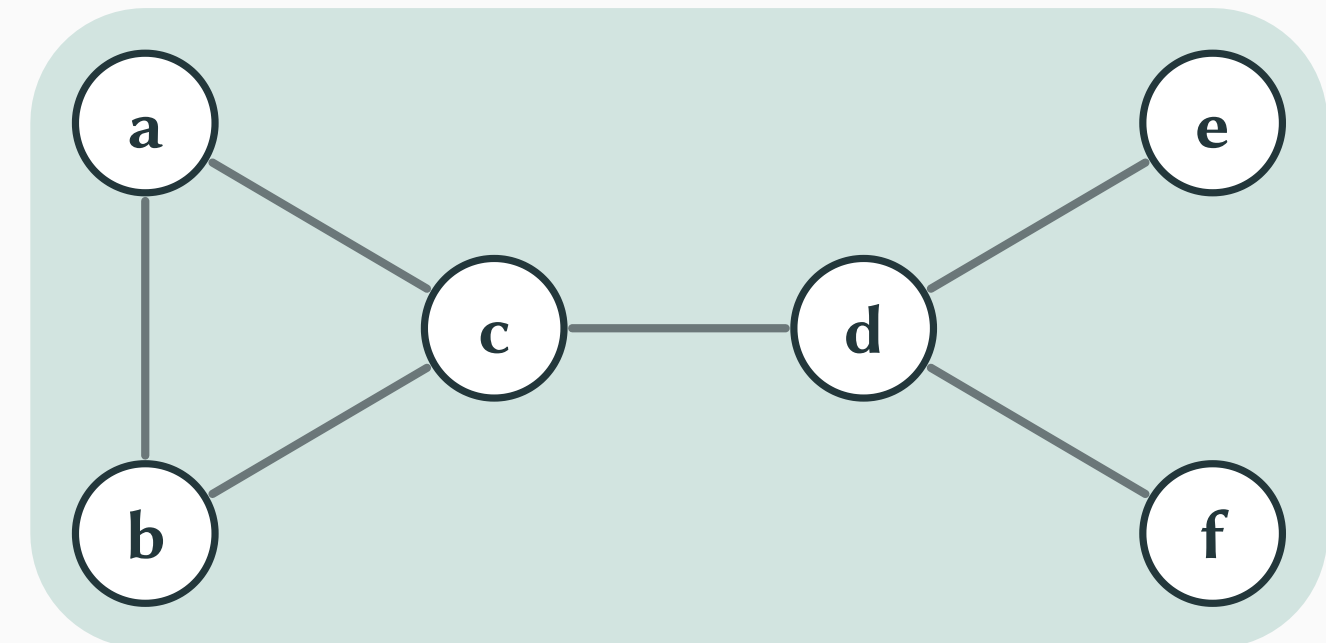
Path: distinct nodes $v_0, \dots, v_k$, each joined to the next.

Length of the path: k .

Cycle: a path $v_0, \dots, v_k$ ($k \geq 2$) plus the edge $\{v_k, v_0\}$.

Connected: a path between any two nodes.

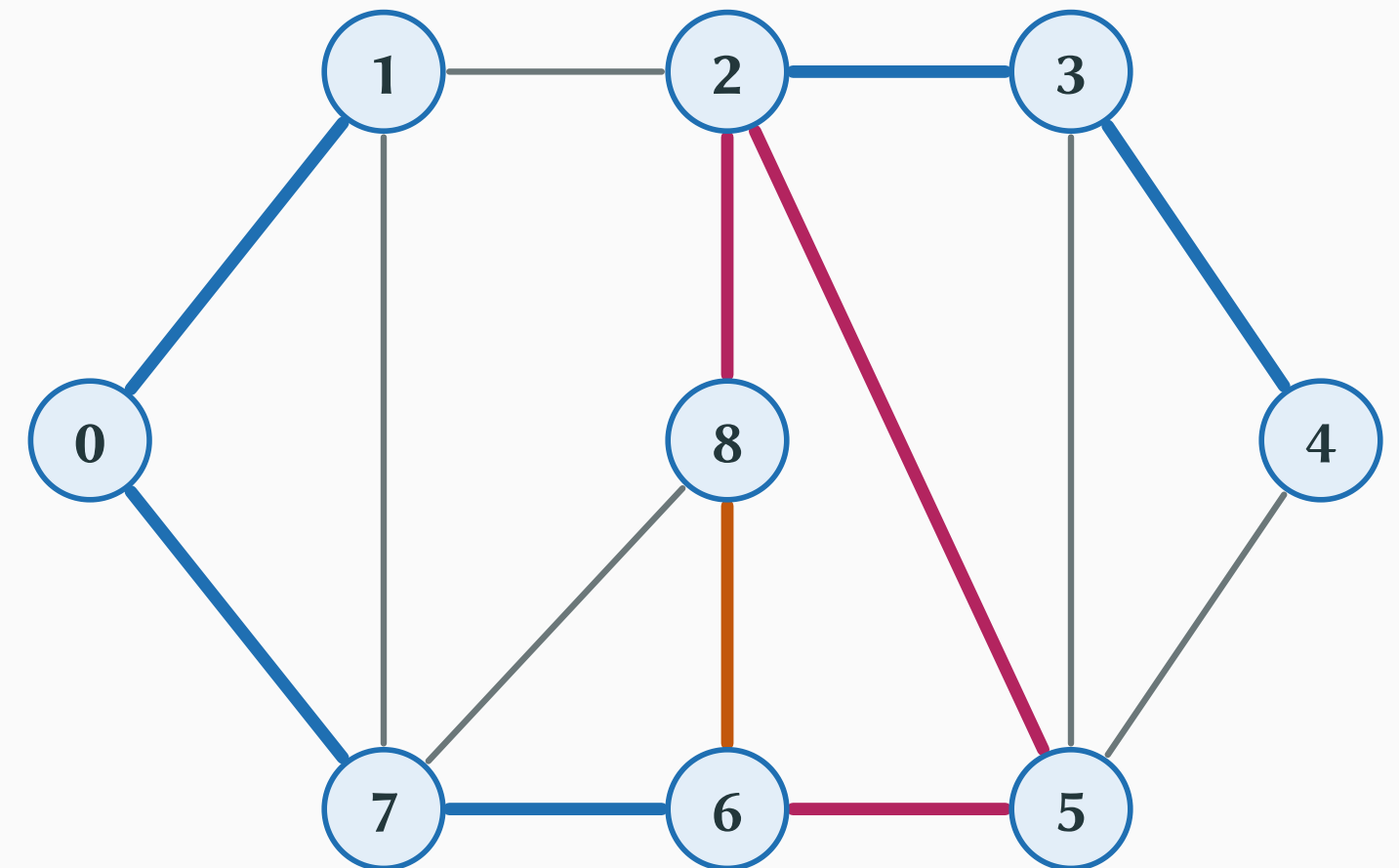
Component: a maximal connected part.

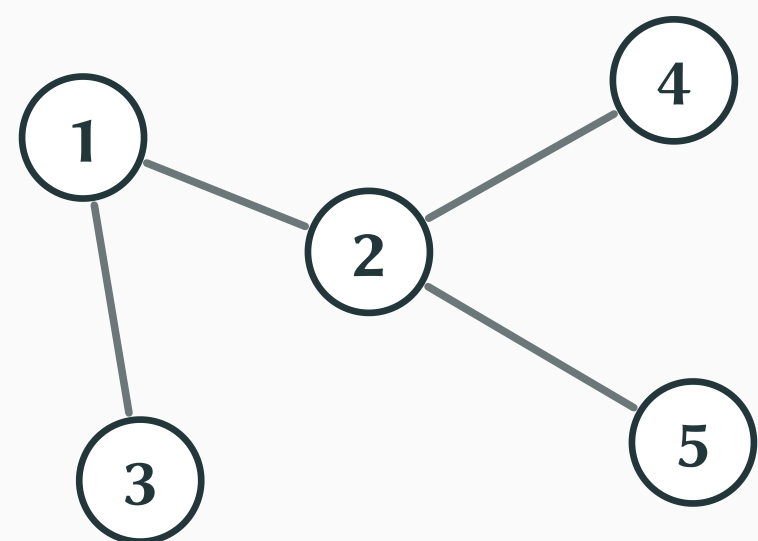


DEFINITION. A **tree** is connected and acyclic. A **forest** is acyclic: its components are trees. A **spanning tree** of $G = (V, E)$ is a set of edges $T \subseteq E$ such that (V, T) is a tree.

FACTS. A tree with n nodes has

- exactly one path between any two nodes,
- exactly $n - 1$ edges,
- exactly **two components** if an edge is removed,
- exactly **one cycle** if an **edge** is added.

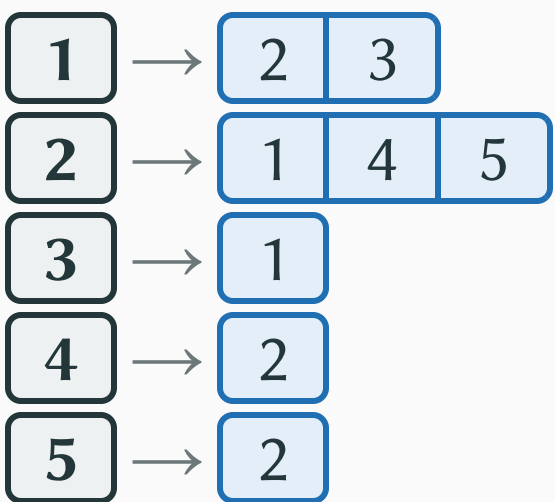




Adjacency matrix A

	1	2	3	4	5
1	0	1	1	0	0
2	1	0	0	1	1
3	1	0	0	0	0
4	0	1	0	0	0
5	0	1	0	0	0

Adjacency list



	adjacency matrix	adjacency list
memory	$O(n^2)$	$O(n + m)$
is $\{u, v\}$ an edge?	$O(1)$	$O(\min(\deg u, \deg v))$
all neighbours of u	$O(n)$	$O(\deg u)$

Recall from IADS: breadth-first and depth-first search
take $O(n + m)$ time with adjacency lists. KT 3.2–3.3, CLRS 20.2–20.3.

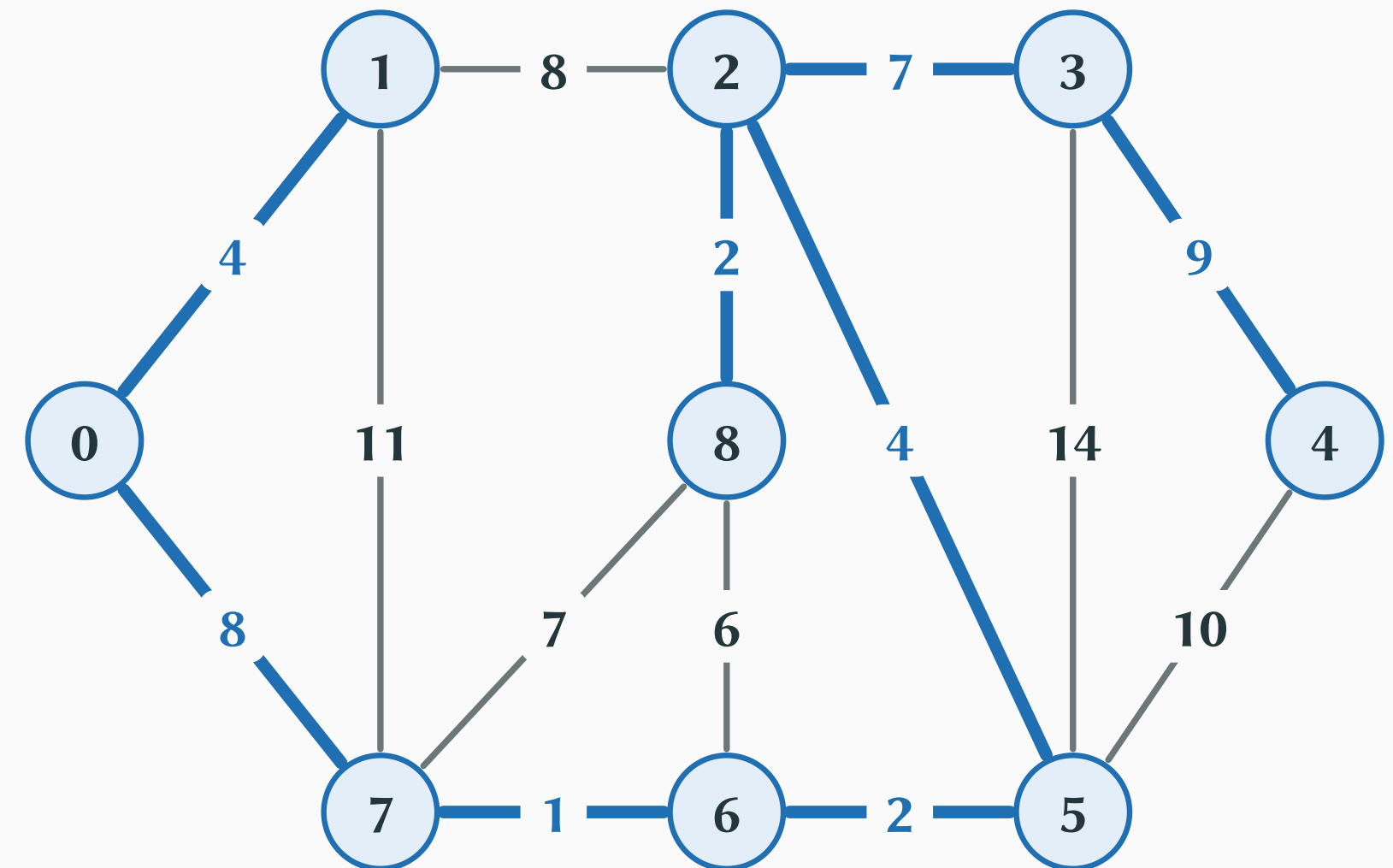
THE MINIMUM SPANNING TREE PROBLEM

TWO GREEDY ALGORITHMS

Connect locations by a network, as cheaply as possible: every direct link has a cost.

Input: a connected graph $G = (V, E)$ and a cost $c_e > 0$ for every edge e .
Output: $T \subseteq E$ with (V, T) connected and of minimum cost $c(T) = \sum_{e \in T} c_e$.

Trying all spanning trees is hopeless: the complete graph K_n on n nodes has n^{n-2} of them, $2.6 \cdot 10^{23}$ for $n = 20$.



cheapest: $1 + 2 + 2 + 4 + 4 + 7 + 8 + 9 = 37$

CLAIM. If T is a solution, then (V, T) has no cycle: it is a tree.

Suppose (V, T) has a cycle C .

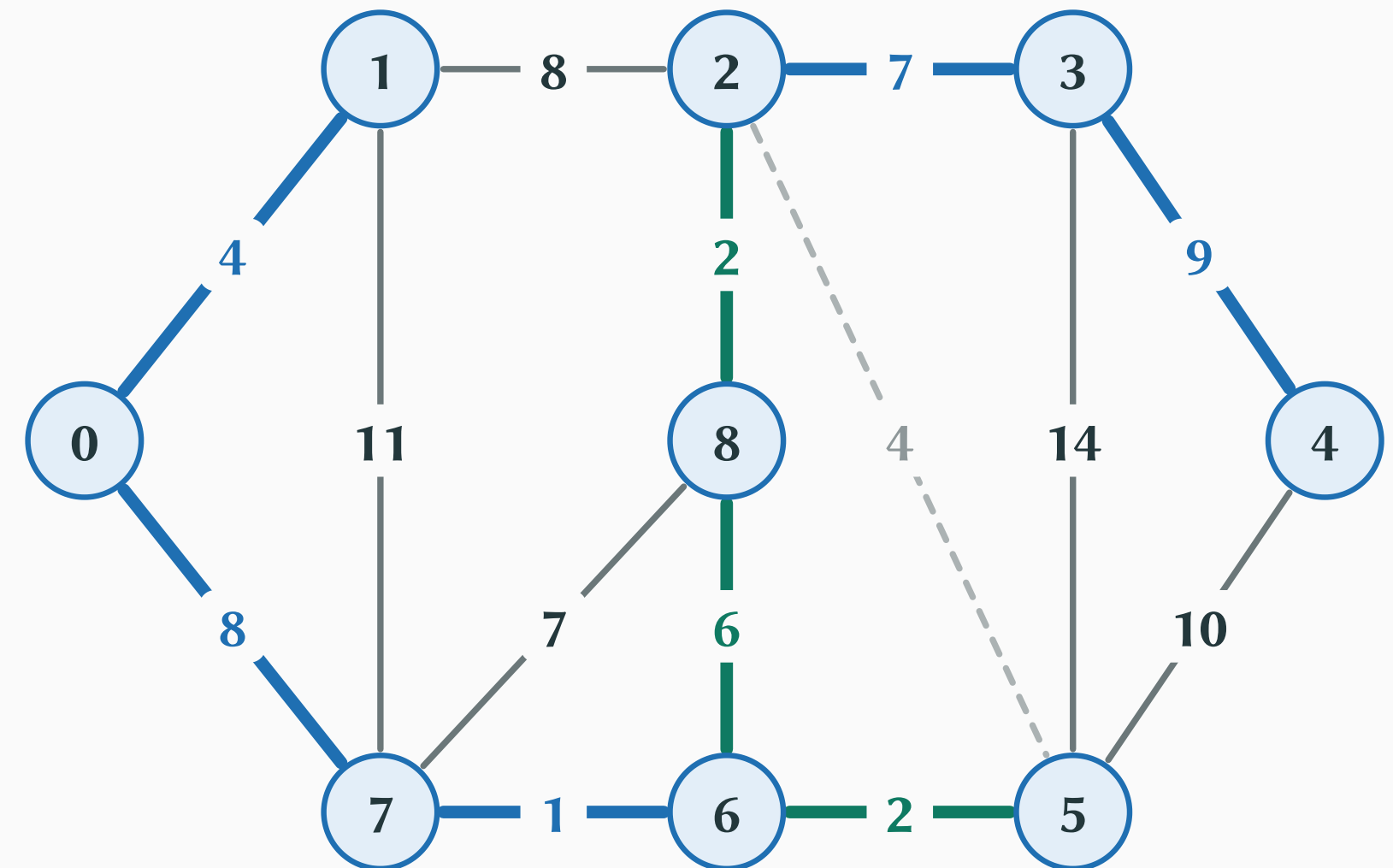
Let e be an edge of C .

$(V, T \setminus \{e\})$ is still connected: a path through e can go around the rest of C .

It is cheaper by $c_e > 0$:

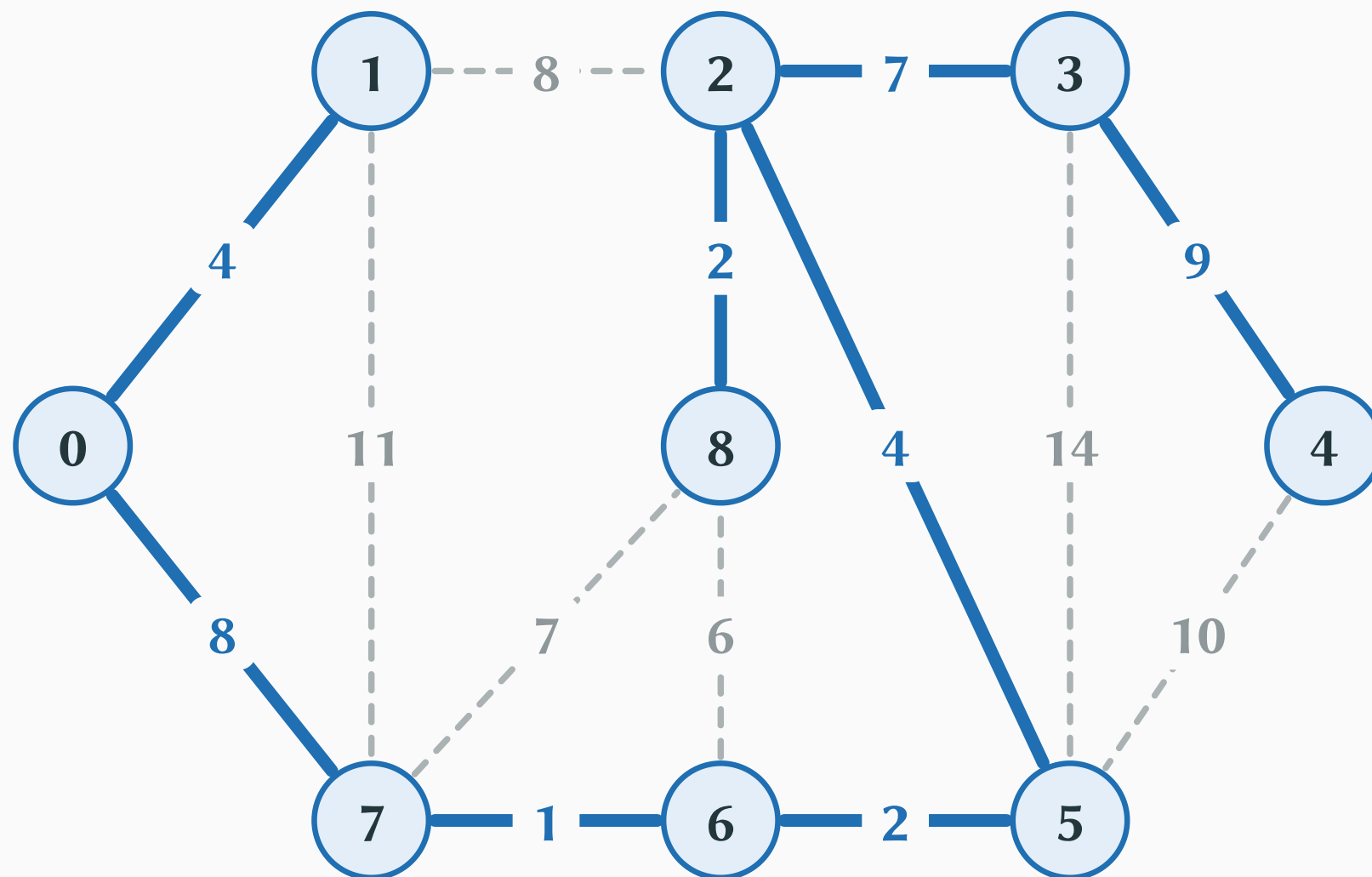
T was not minimum. Contradiction.

A solution is a **minimum spanning tree (MST)**.



cost: $43 - 4 = 39$

GREEDY #1: KRUSKAL'S ALGORITHM

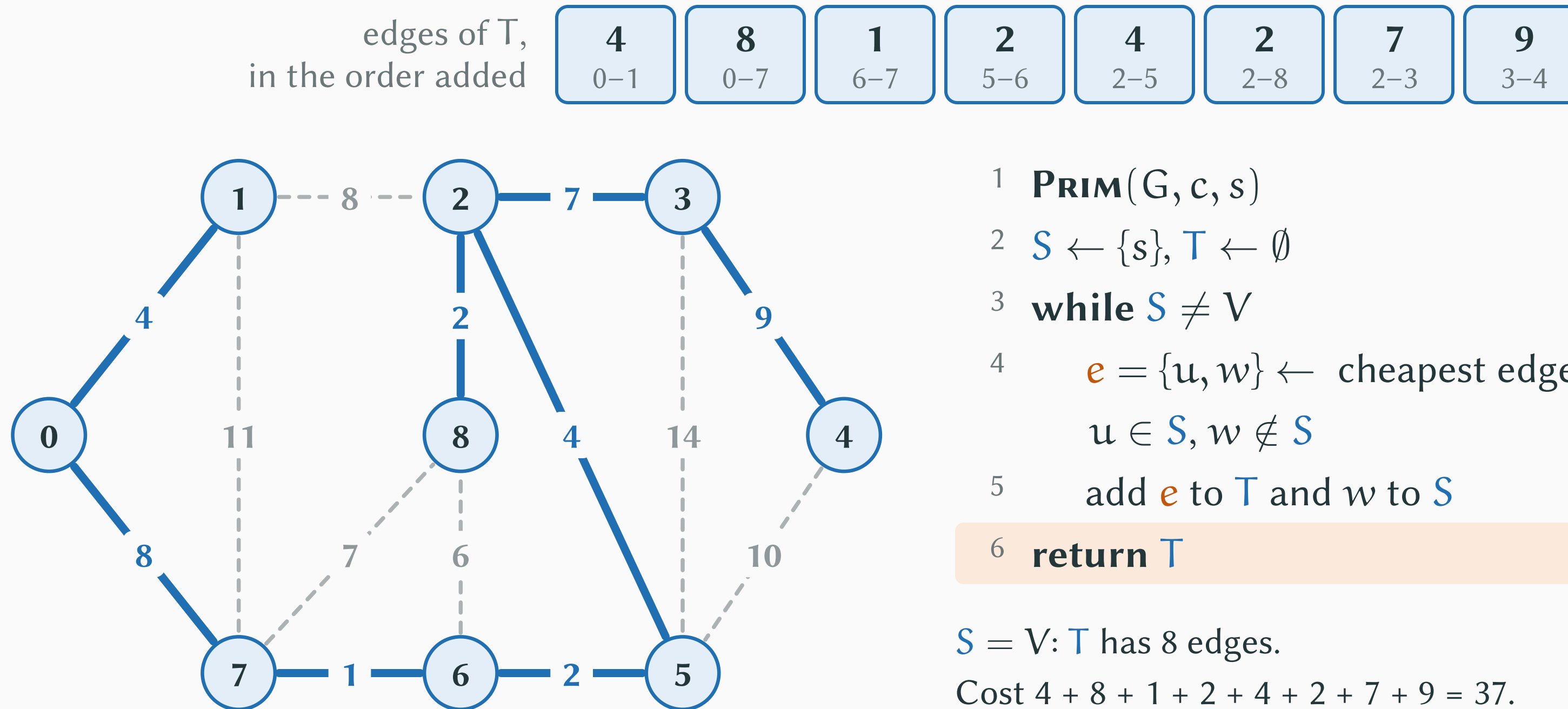


```
1 KRUSKAL(G, c)
2  $T \leftarrow \emptyset$ 
3 sort the edges by cost
4 for each edge  $e = \{u, v\}$  in this order
5     if  $u, v$  in different components
6         add  $e$  to  $T$ 
7 return  $T$ 
```

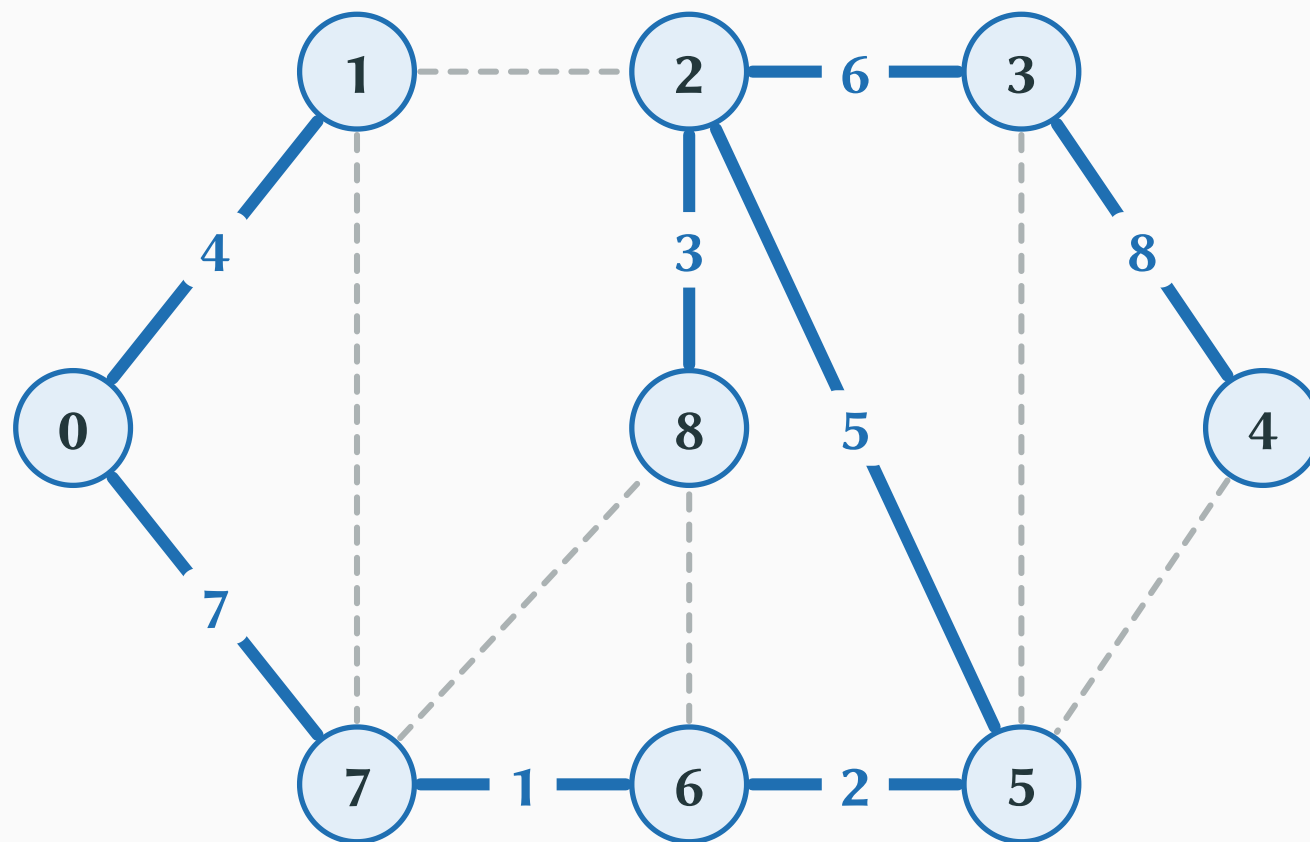
T has $n - 1 = 8$ edges: done.

Cost $1 + 2 + 2 + 4 + 4 + 7 + 8 + 9 = 37$.

GREEDY #2: PRIM'S ALGORITHM

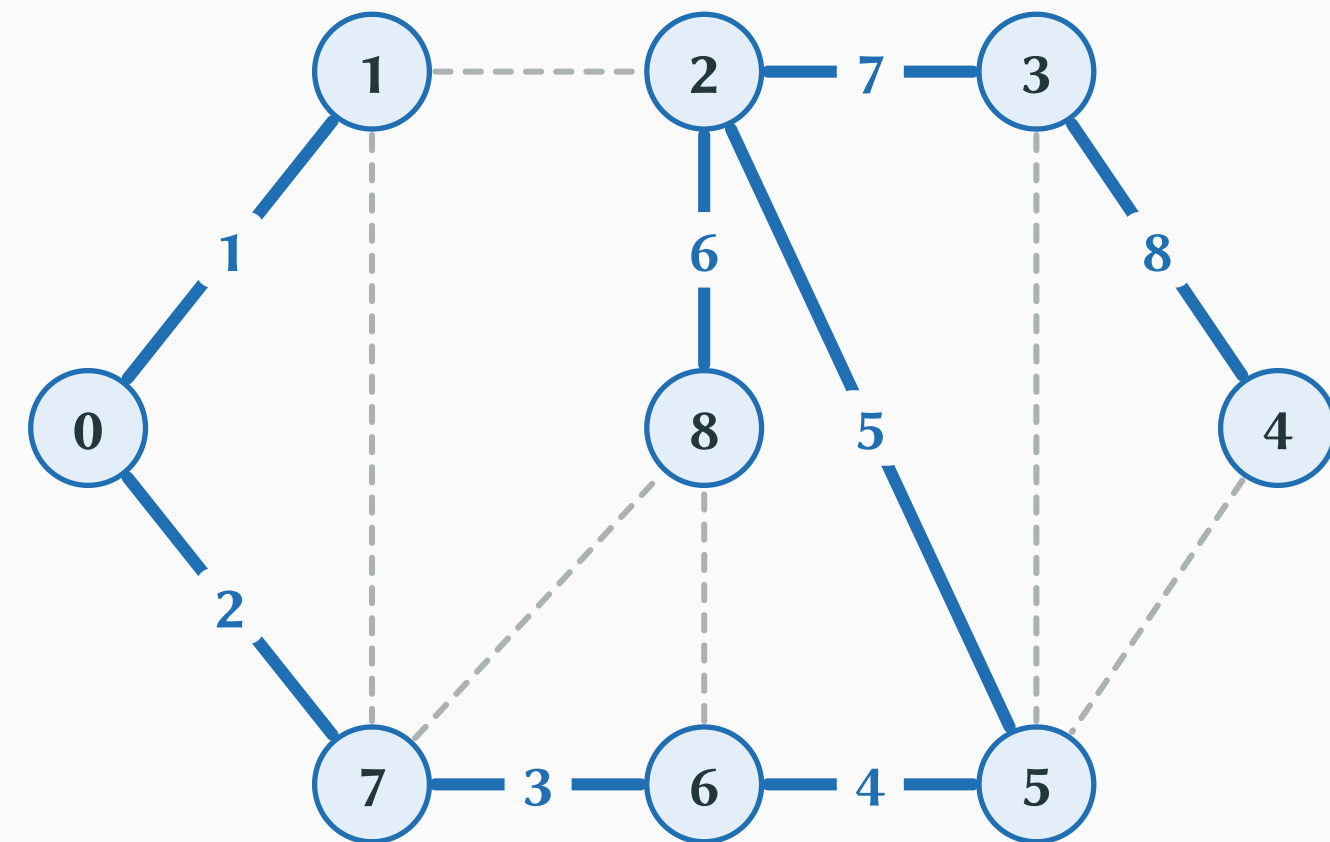


Kruskal's algorithm: a forest that merges



numbers on the edges: order of adding
costs in that order: 1 2 2 4 4 7 8 9

Prim's algorithm: one tree that grows



numbers on the edges: order of adding
costs in that order: 4 8 1 2 4 2 7 9

The same tree, of cost 37.

? *Is it a minimum spanning tree?*

Step 3 of Prim's algorithm: S , the nodes of its tree.

magenta: the **cut**, the edges between S and $V \setminus S$

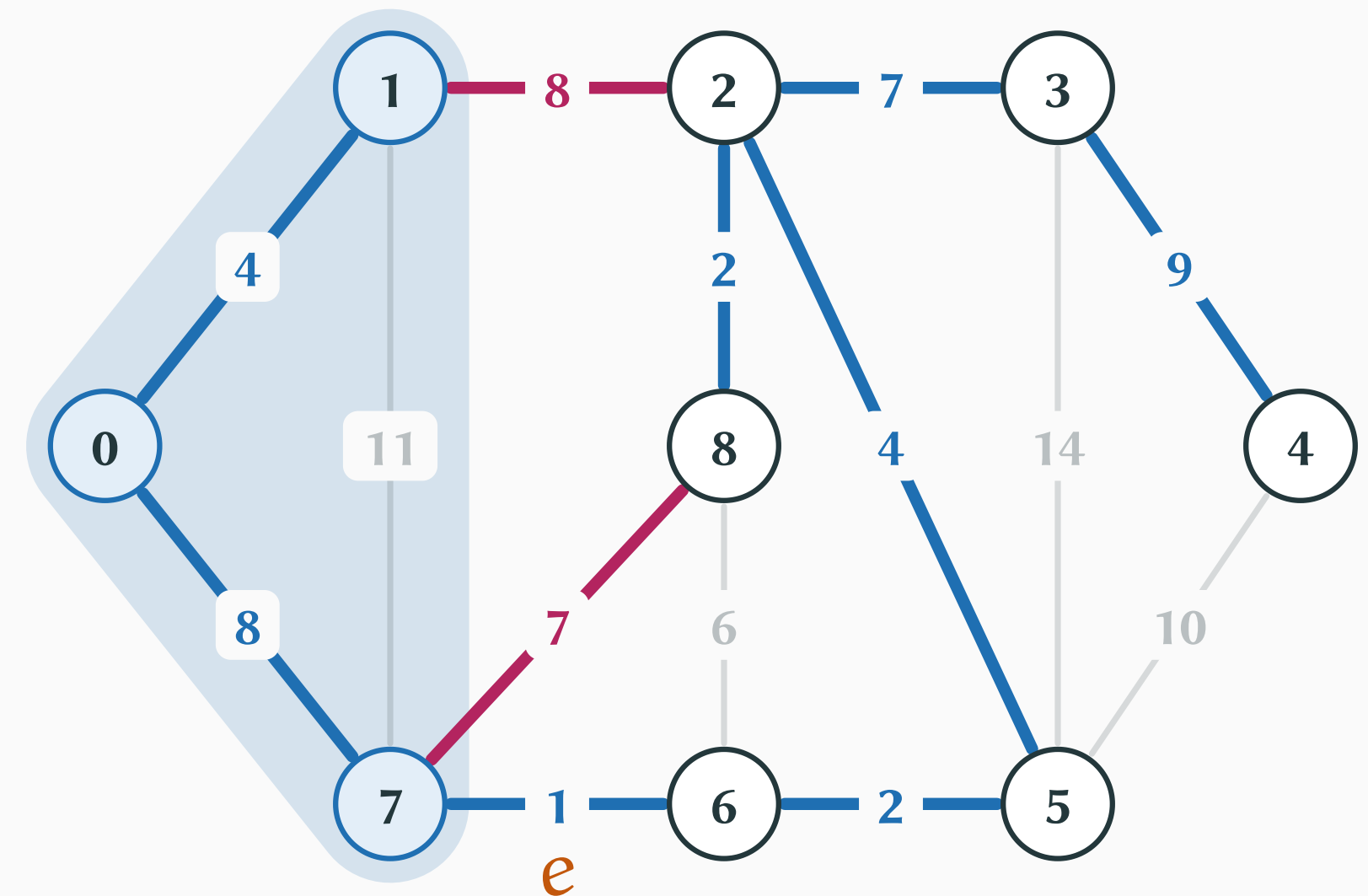
orange: e , the cheapest edge of the cut

blue: an MST. It contains e .

$S = \{0, 1, 7\}$, cut: 6-7 (1), 7-8 (7), 1-2 (8)

$e = 6-7$ (1) $\in$ MST

CUT PROPERTY. Let all costs be distinct, $S \subseteq V$ with $S \neq \emptyset, V$, and e the cheapest edge between S and $V \setminus S$. Then **every** MST contains e .



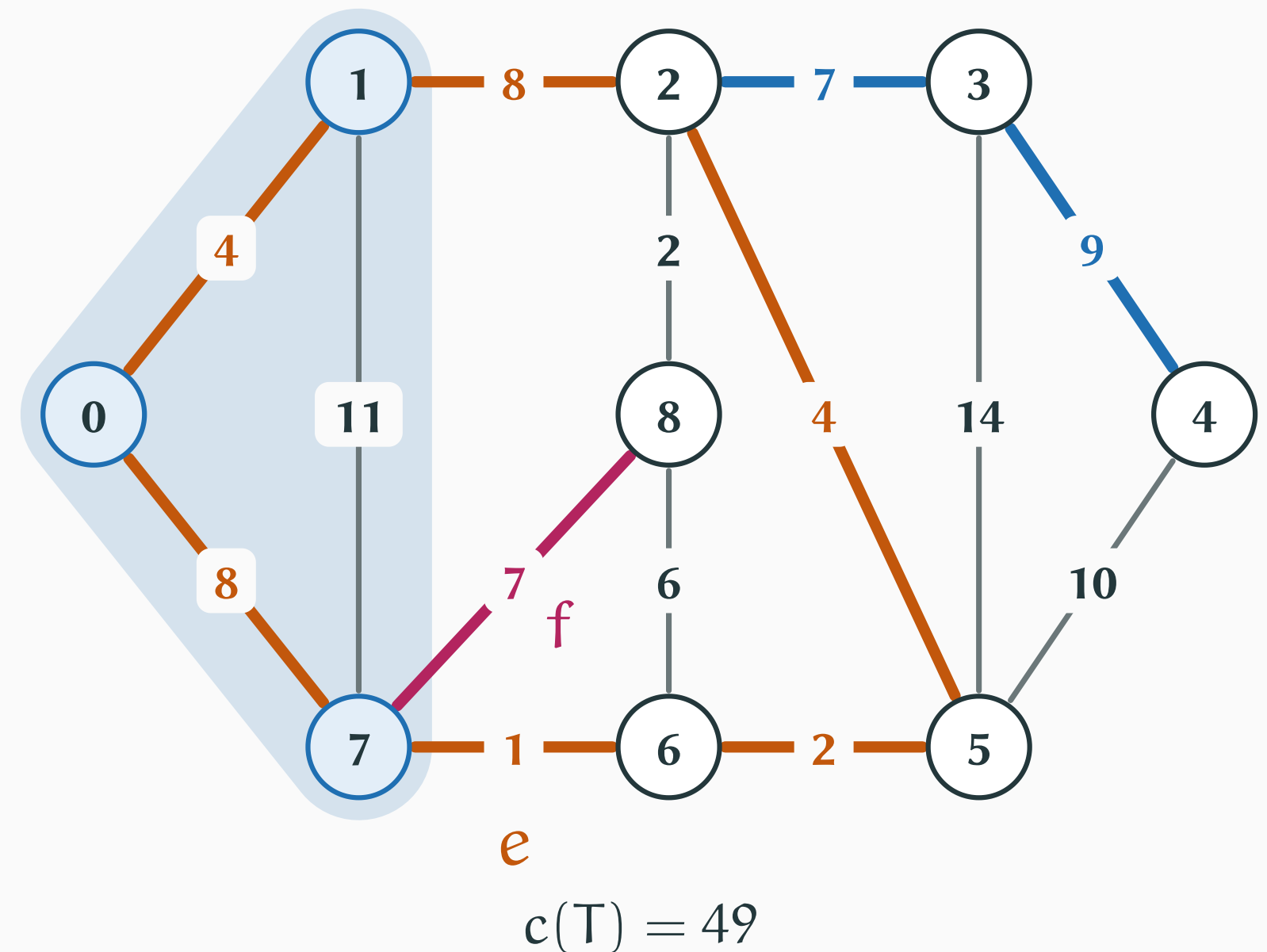
Let T be a spanning tree with $e \notin T$.
We want a **cheaper** spanning tree.

Exchange: add e and remove
an edge f of T that crosses the cut.
As $c_e < c_f$, $T - f + e$ is cheaper.

? *Is $T - f + e$ a spanning tree?*

No: node 8 is cut off, and e closes a **cycle**.

f must lie on the **cycle** that e closes in T .



Let T be a spanning tree with $e = \{v, w\} \notin T$.

T contains a path P from $v \in S$ to $w \notin S$.

P leaves S at an edge $e' = \{v', w'\}$.

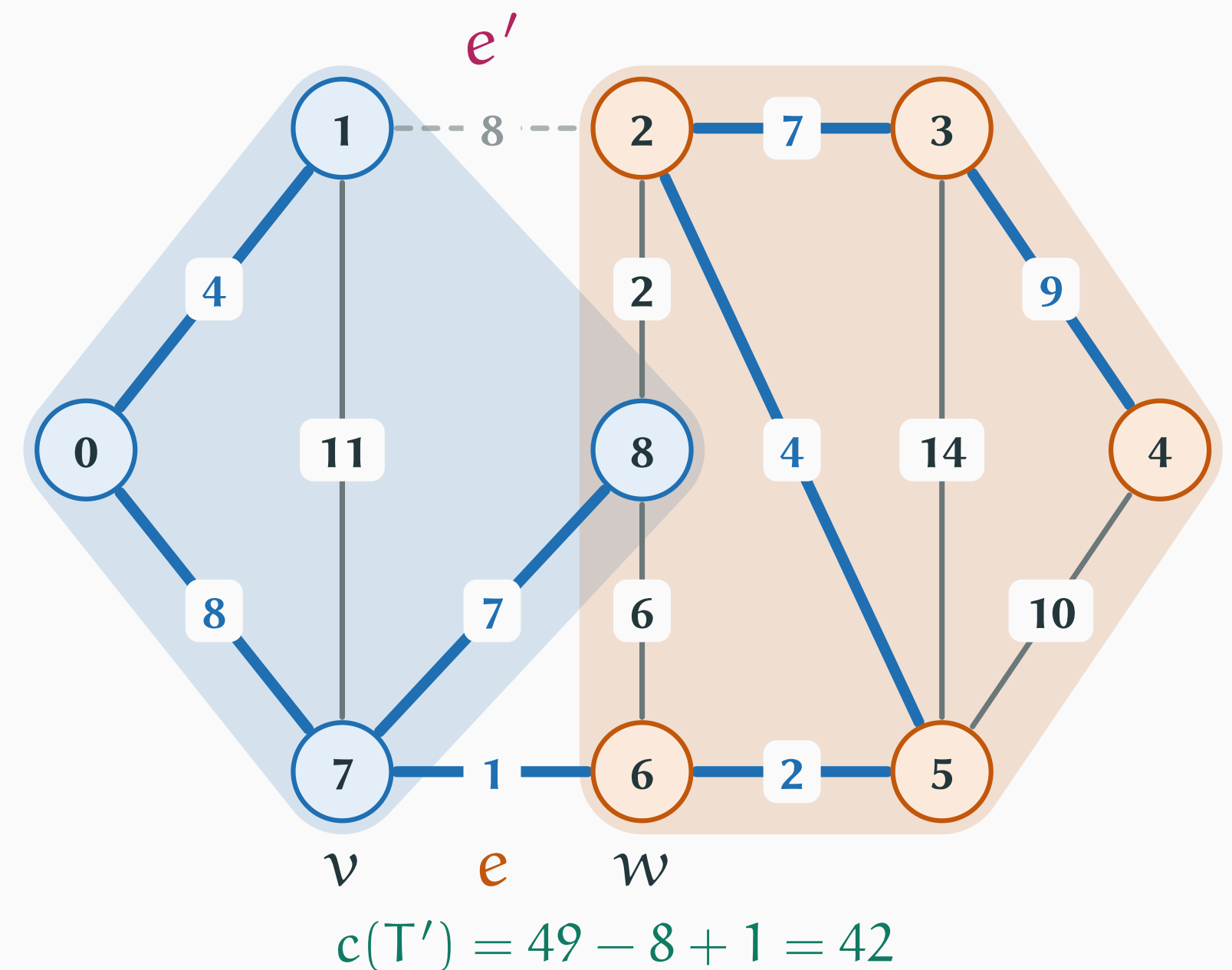
Since e' is a cut edge and $e' \neq e: c_e < c_{e'}$.

$T' = T - e' + e$ is a spanning tree:

$T - e'$ has two components, with v in one and w in the other, and e joins them.

$$c(T') = c(T) - c_{e'} + c_e < c(T).$$

So T is not minimum: **every** MST contains e . $\square$



Kruskal's algorithm adds an edge $e = \{u, w\}$.

S : the component of u just before.

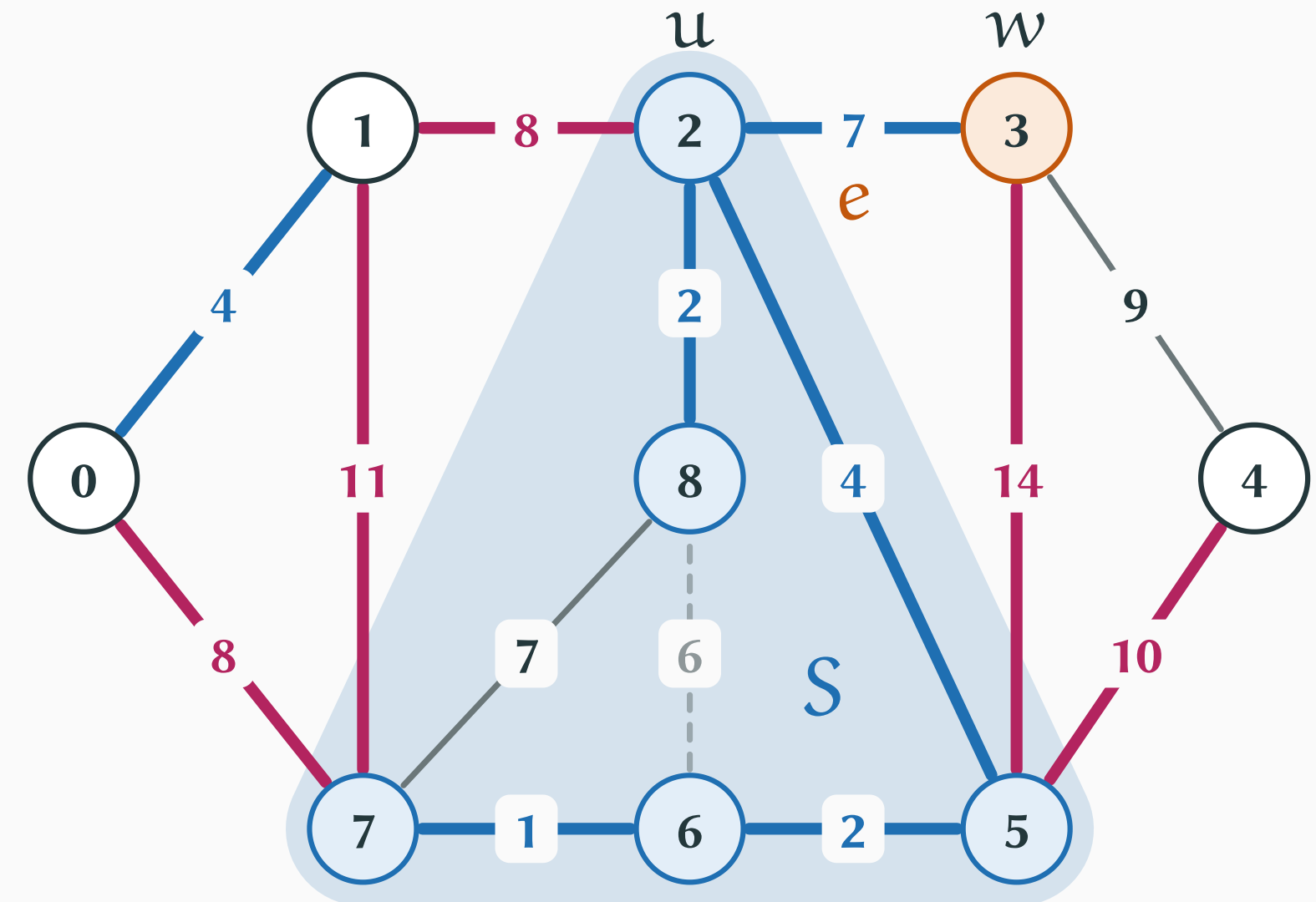
$u \in S, w \notin S$ (else e closes a cycle):

e is a **cut edge**.

? *Is e the cheapest cut edge?*

Yes. A cut edge examined earlier would have joined two components (it still does), so Kruskal's algorithm would have added it, putting both its ends in S .

By the cut property (costs distinct),
every edge that Kruskal's algorithm adds
is in **every** MST.



? *Is the output T a spanning tree?*

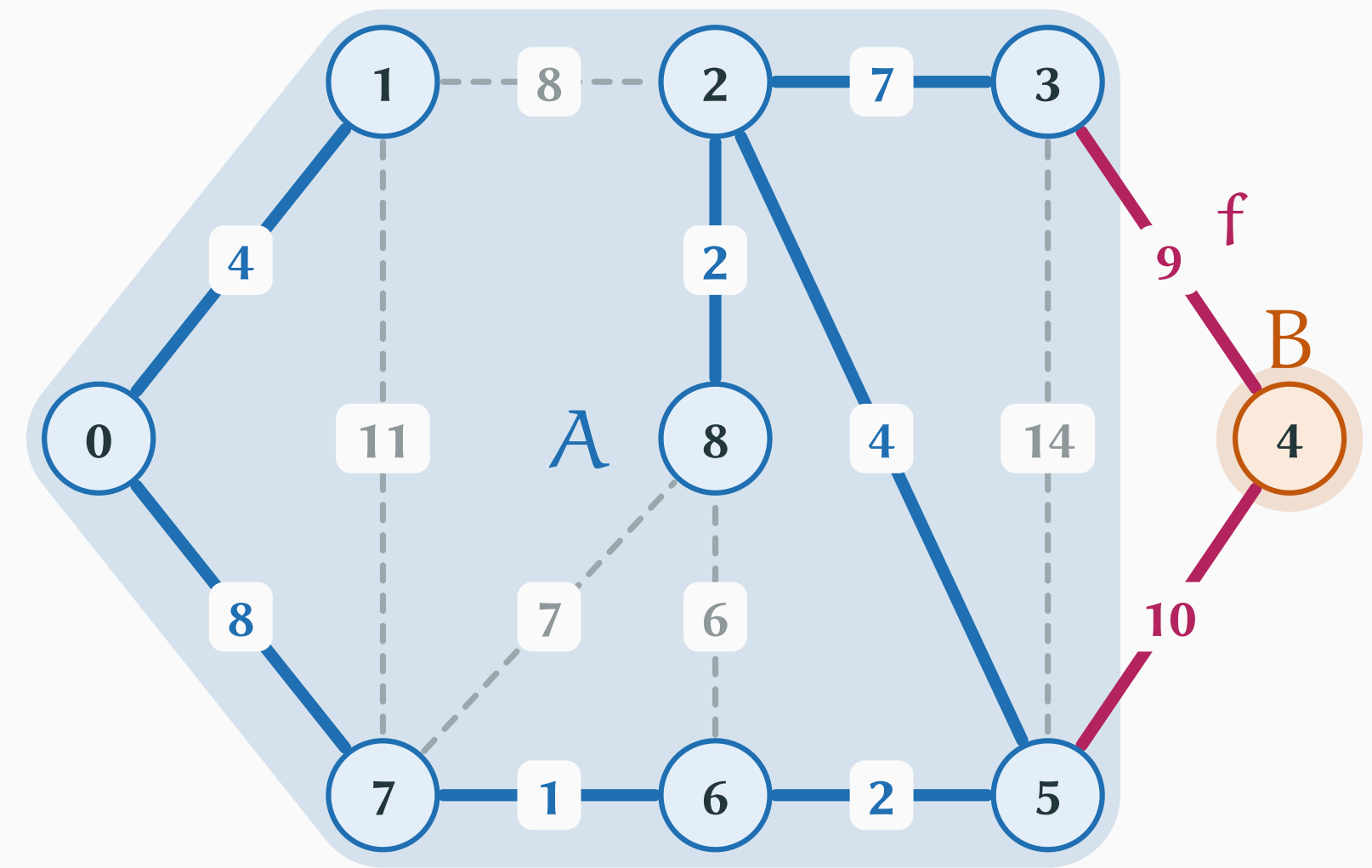
No cycle: an edge is added only if its ends are in different components.

Connected? Suppose not: let A be a component of (V, T) and $B = V \setminus A$.

G is connected, so some edge f of G joins A and B .

When f was examined, its ends were in different components (they still are), so Kruskal's algorithm added it. Contradiction.

So T is a minimum spanning tree.



if the output were this forest ...

PRIM'S ALGORITHM IS OPTIMAL AND FEASIBLE

WHY GREEDY WORKS

S : the nodes of the tree T built so far.

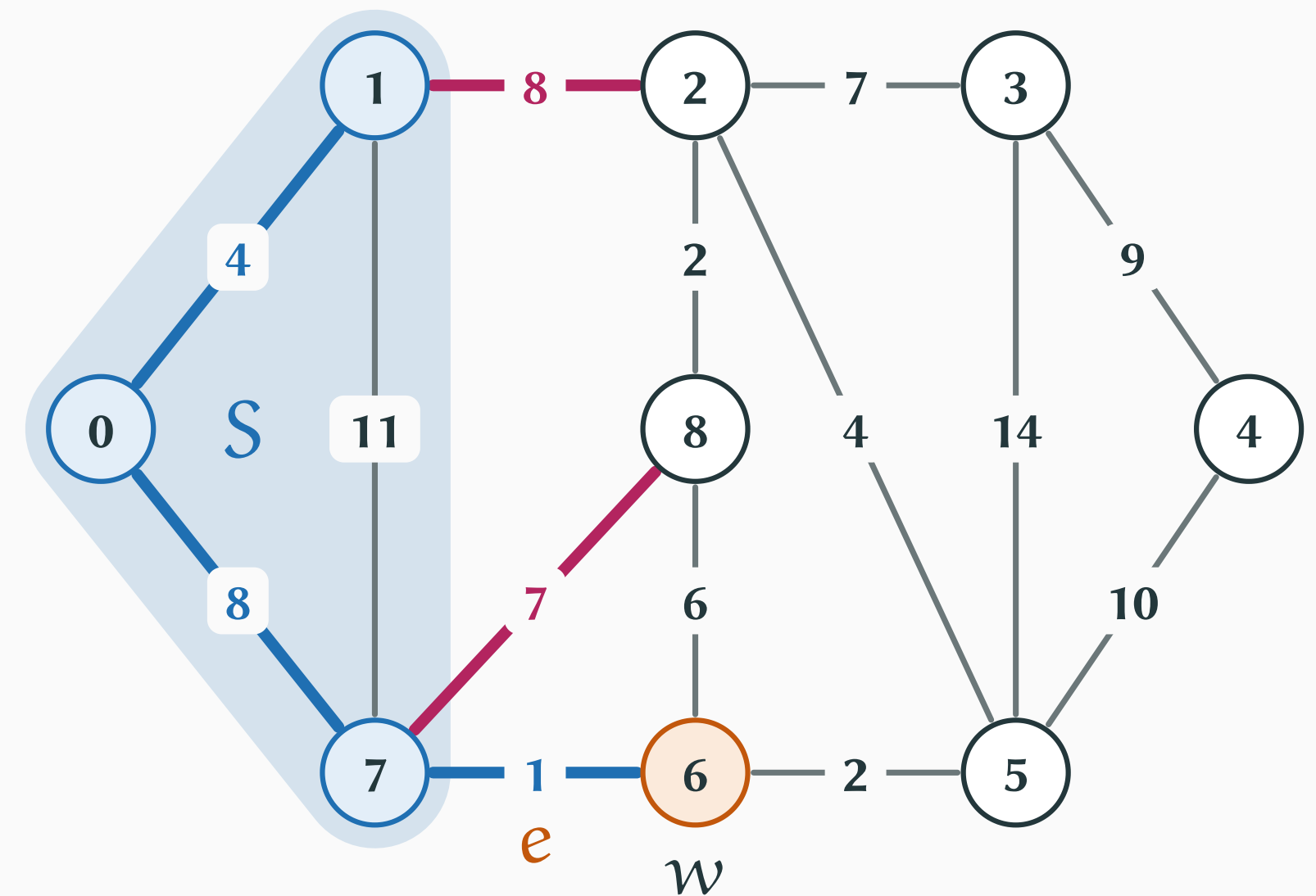
T is a tree on S .

Prim's algorithm adds the cheapest edge e between S and $V \setminus S$.

By the cut property (costs distinct),
 e is in every MST.

Feasible: e adds one new node w ,
so T is again a tree.

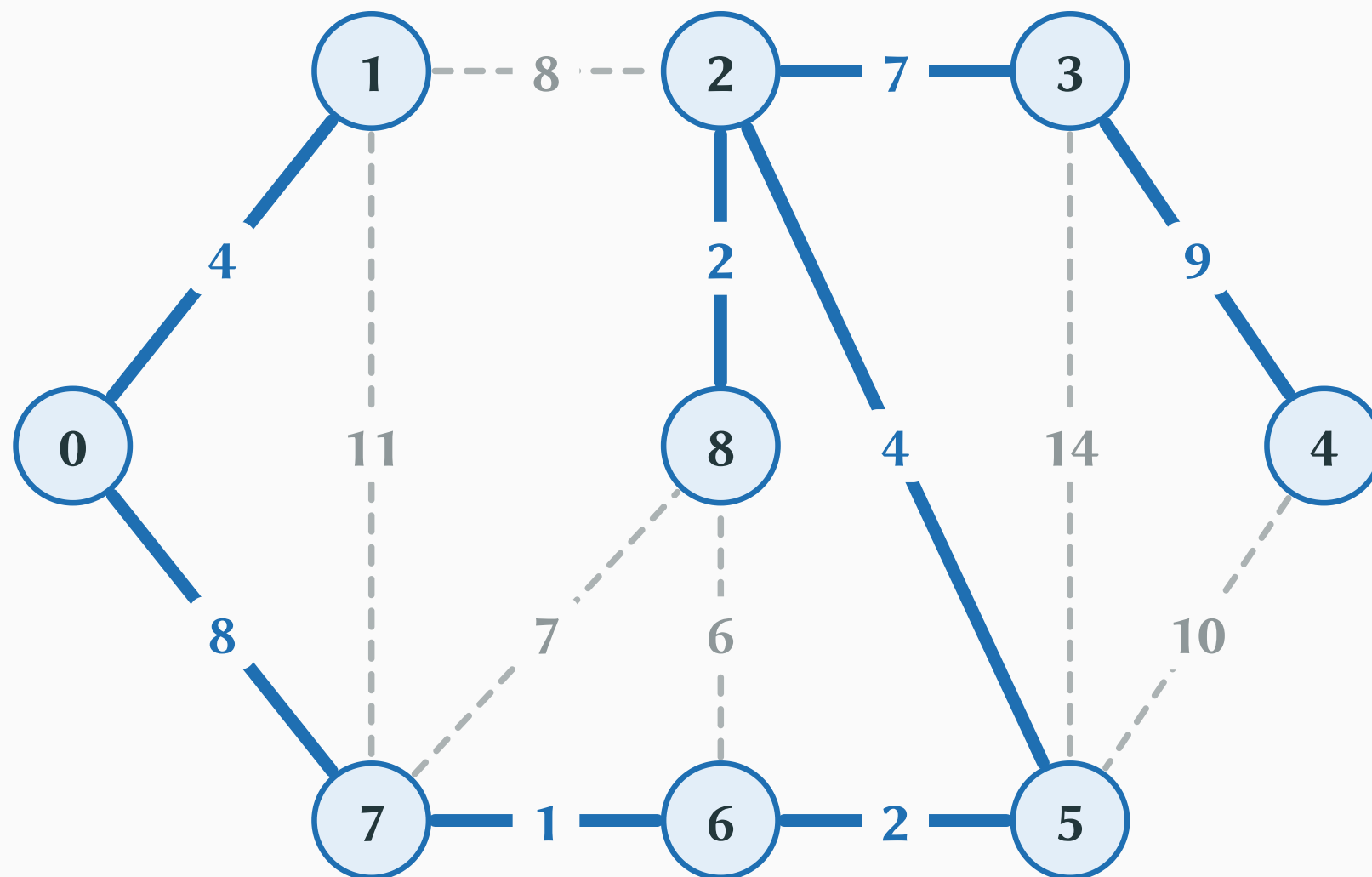
While $S \neq V$, some edge leaves S (G is connected):
Prim's algorithm goes on until $S = V$.



So Prim's algorithm outputs a minimum spanning tree.

GREEDY #3: REVERSE-DELETE

A THIRD GREEDY



```
1 REVERSE-DELETE(G, c)
2  $T \leftarrow E$ 
3 sort the edges, most expensive first
4 for each edge  $e$  in this order
5     if  $(V, T \setminus \{e\})$  is connected
6         remove  $e$  from  $T$ 
7 return  $T$ 
```

T has $n - 1 = 8$ edges: every edge is needed.
Cost 37.

CYCLE PROPERTY. Let all costs be distinct, C a cycle of G , and e its most expensive edge. Then e is in **no** MST.

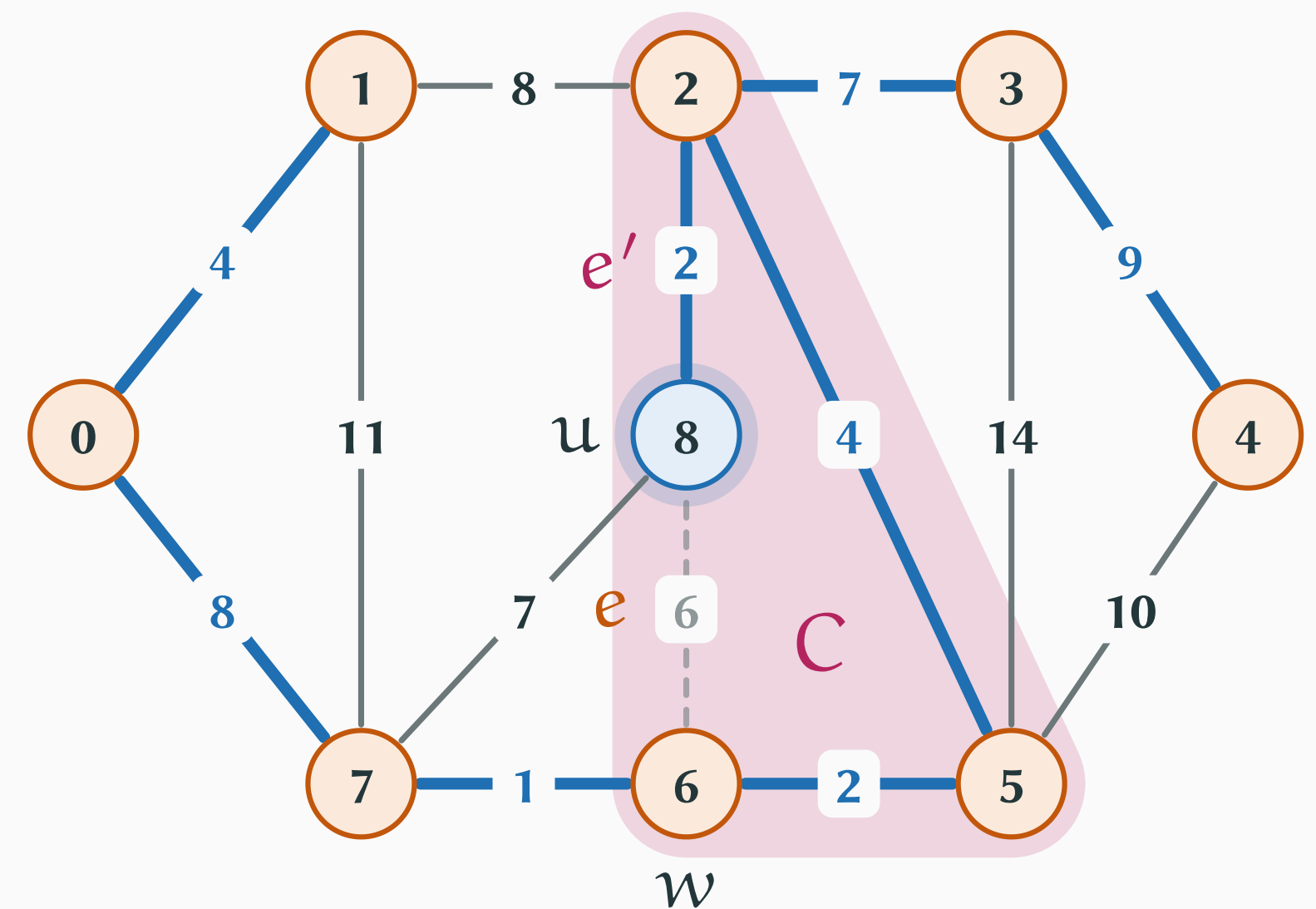
Proof. Let T be a spanning tree with $e = \{u, w\} \in T$.

$T - e$ has two components: $S \ni u$ and $V \setminus S \ni w$.

The path $C - e$ from u to w leaves S at an edge e' , another edge of C : so $c_{e'} < c_e$.

$T - e + e'$ is a spanning tree (e' joins the two components), cheaper by $c_e - c_{e'}$.

So T is not minimum: no MST contains e . $\square$



$$c(T - e + e') = 41 - 8 + 2 = 35$$

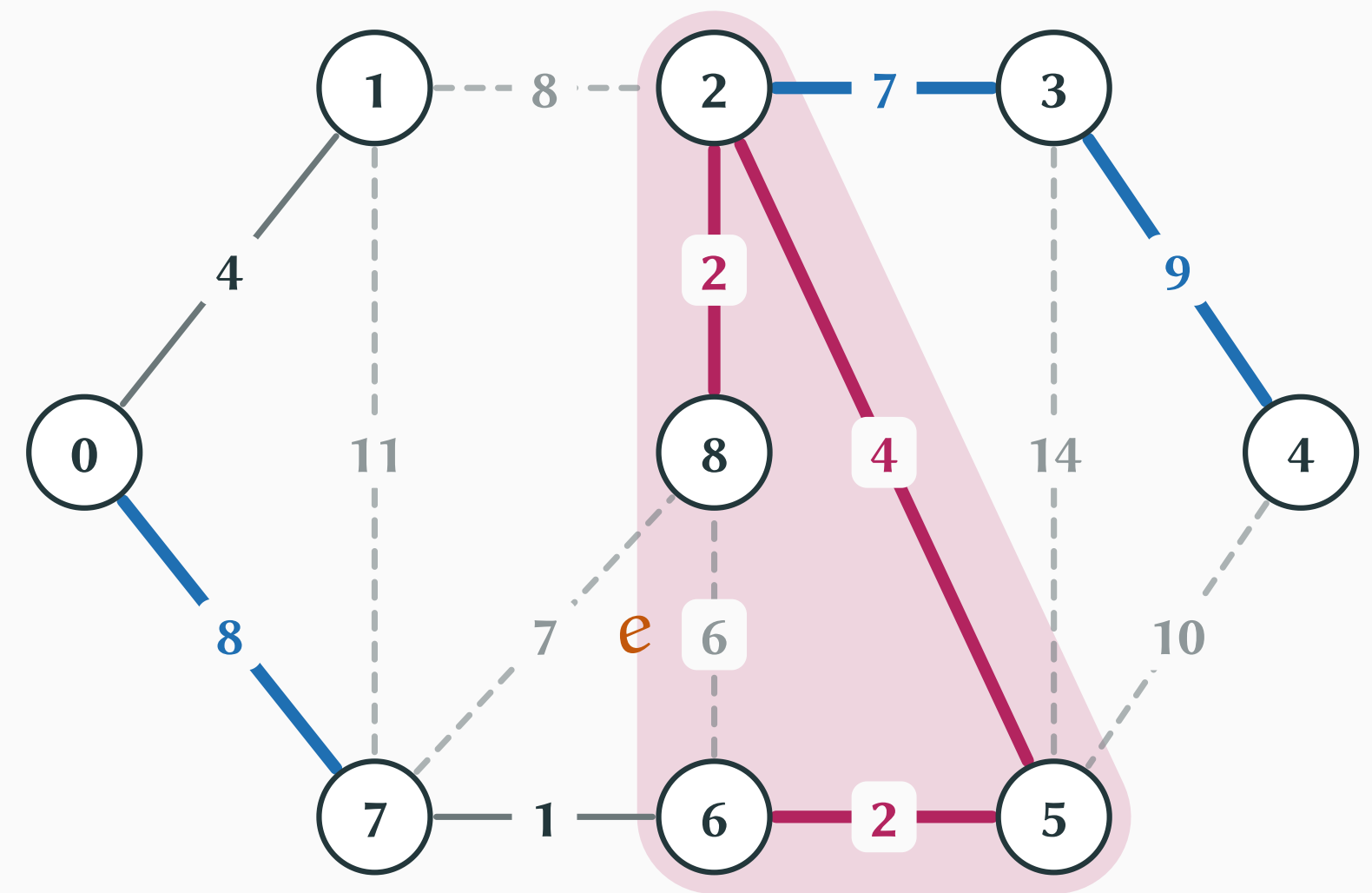
Reverse-delete removes an edge e .

It is removed only if the graph stays connected:
 e lies on a cycle C of the current graph.

? *Is e the most expensive edge of C ?*

Yes. A costlier edge $f \in C$ would have been examined earlier, with all of C present, and removed. But f is still there.

By the cycle property (costs distinct),
no removed edge belongs to any MST.



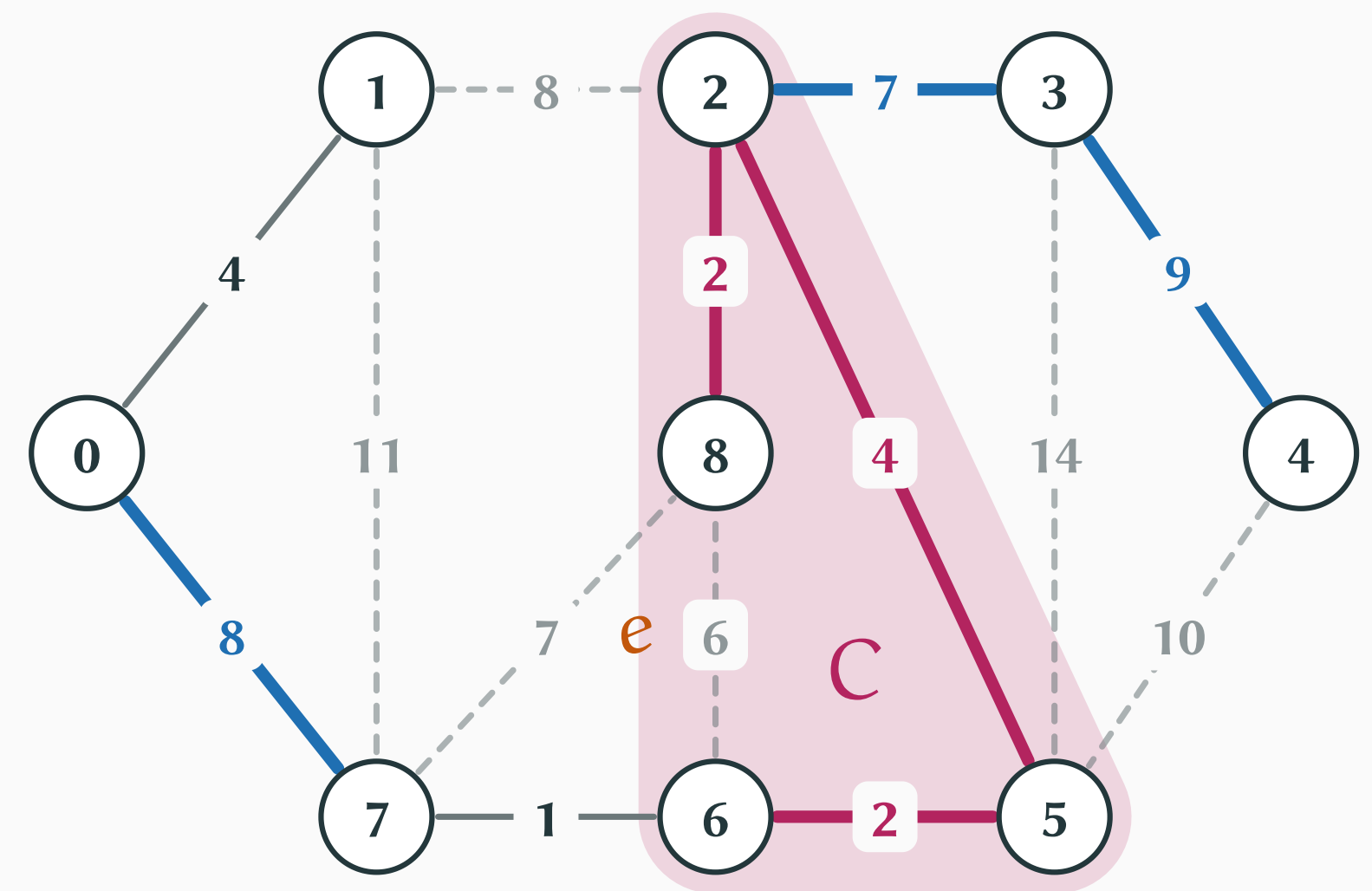
? *Is the output T a spanning tree?*

Connected: an edge is removed only if the graph stays connected.

No cycle: suppose T contains a cycle C . Let e be its most expensive edge.

When e was examined, C was in the graph, so e could be removed and was. Contradiction.

So T is a minimum spanning tree.



if the output contained a cycle ...

WHAT IF COSTS ARE EQUAL?

EQUAL COSTS

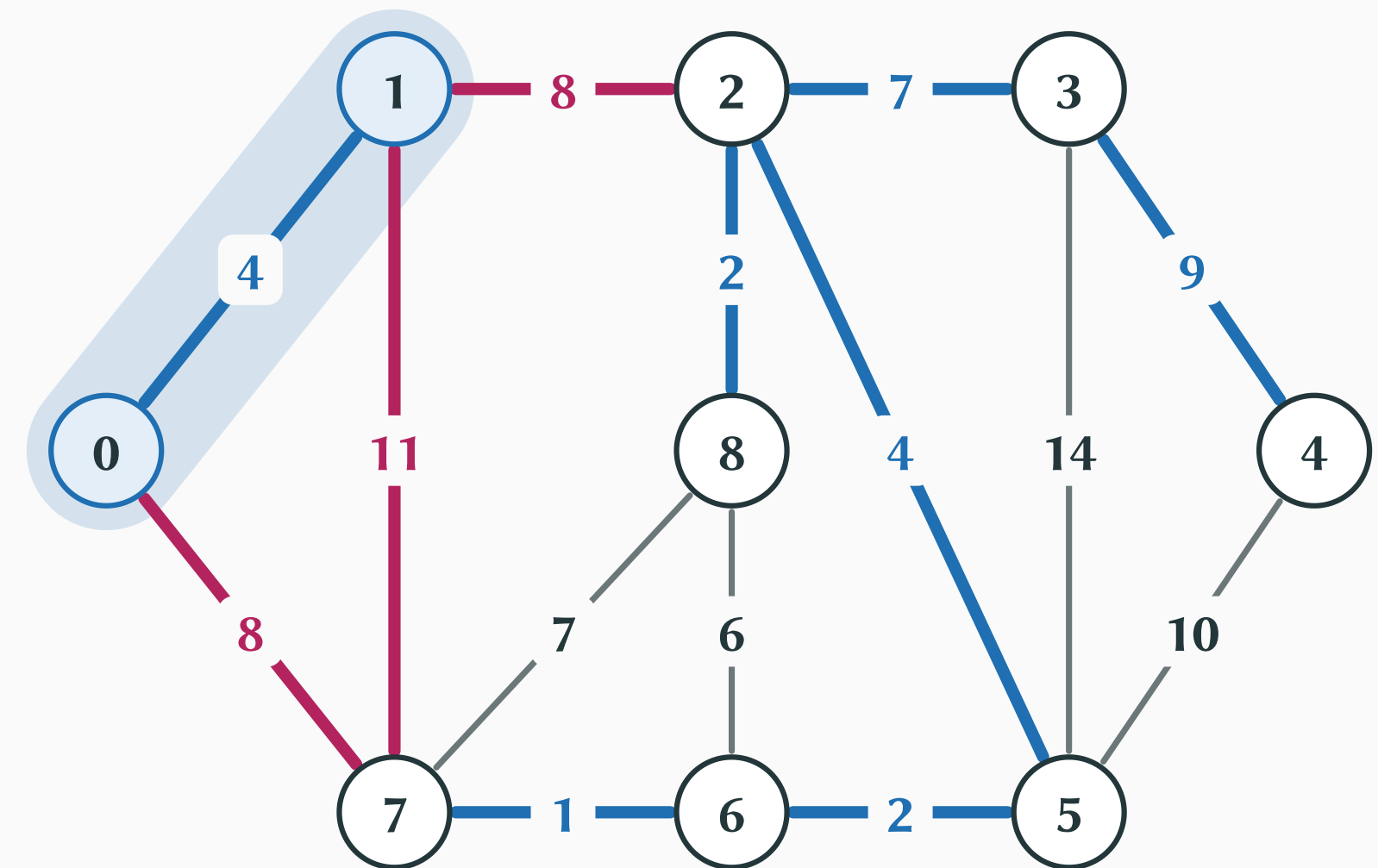
We assumed distinct costs. Our example has equal ones: $2, 2 \cdot 4, 4 \cdot 7, 7 \cdot 8, 8$.

The MST is not unique: T_1 and T_2 both cost 37.

The cut property fails: for $S = \{0, 1\}$ the cheapest cut edges $\{0, 7\}$ and $\{1, 2\}$ both cost 8, and neither is in every MST.

Do the algorithms still work?

Yes: tiny perturbations break the ties.



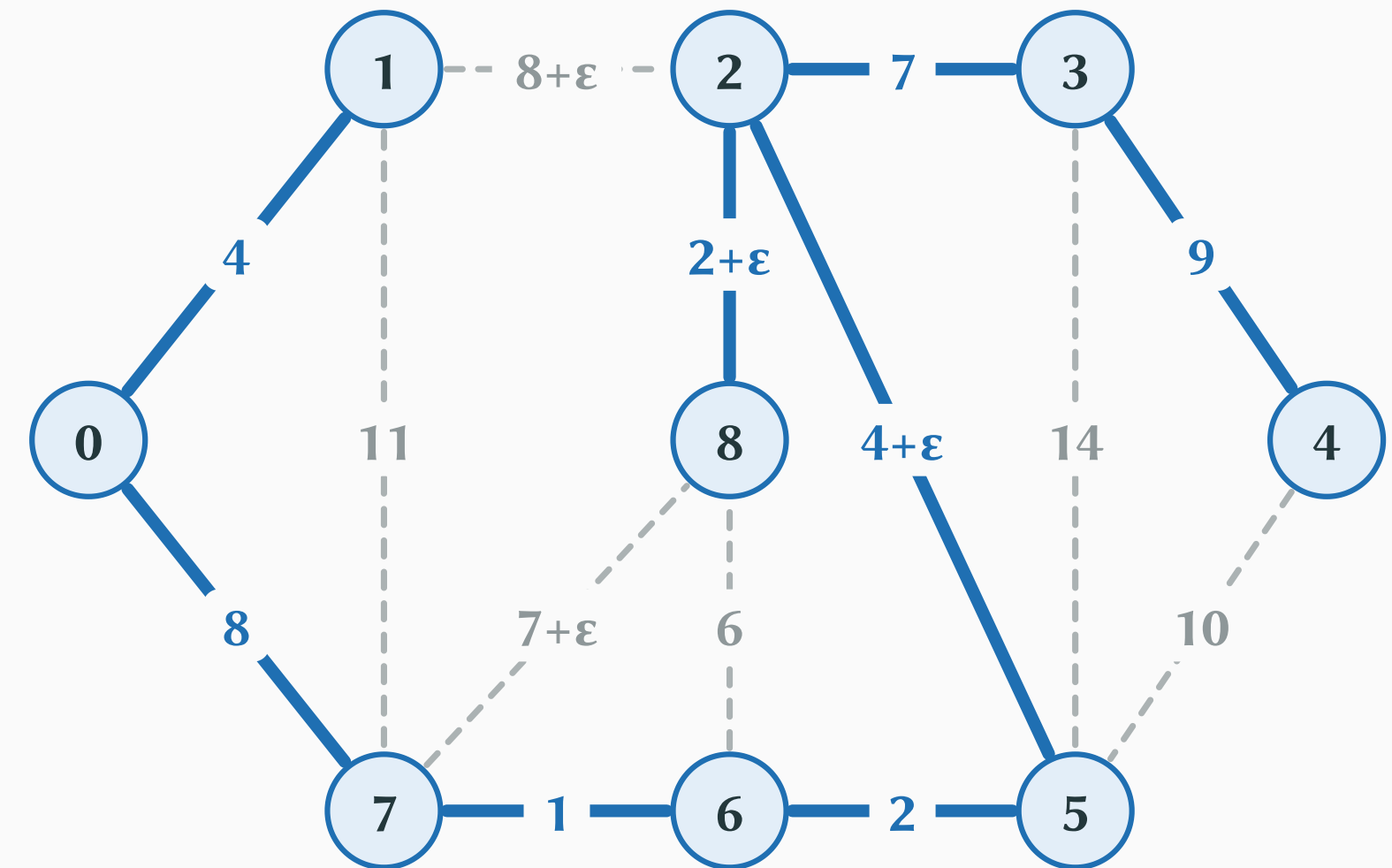
cut edges 0-7 (8), 1-2 (8), 1-7 (11)

Break the ties: $c'_e = c_e + \epsilon_e$, with tiny $\epsilon_e \geq 0$ that make all costs distinct.

Here: $+\epsilon$ for the later edge of each tied pair.

Run the algorithm on c' :
it returns an MST T of c' .

CLAIM. T is an MST for the original costs c .



the tree T of c' costs $37 + 2\epsilon$

Example: the costs of spanning trees are integers, so two of them are equal or at least $\Delta = 1$ apart.

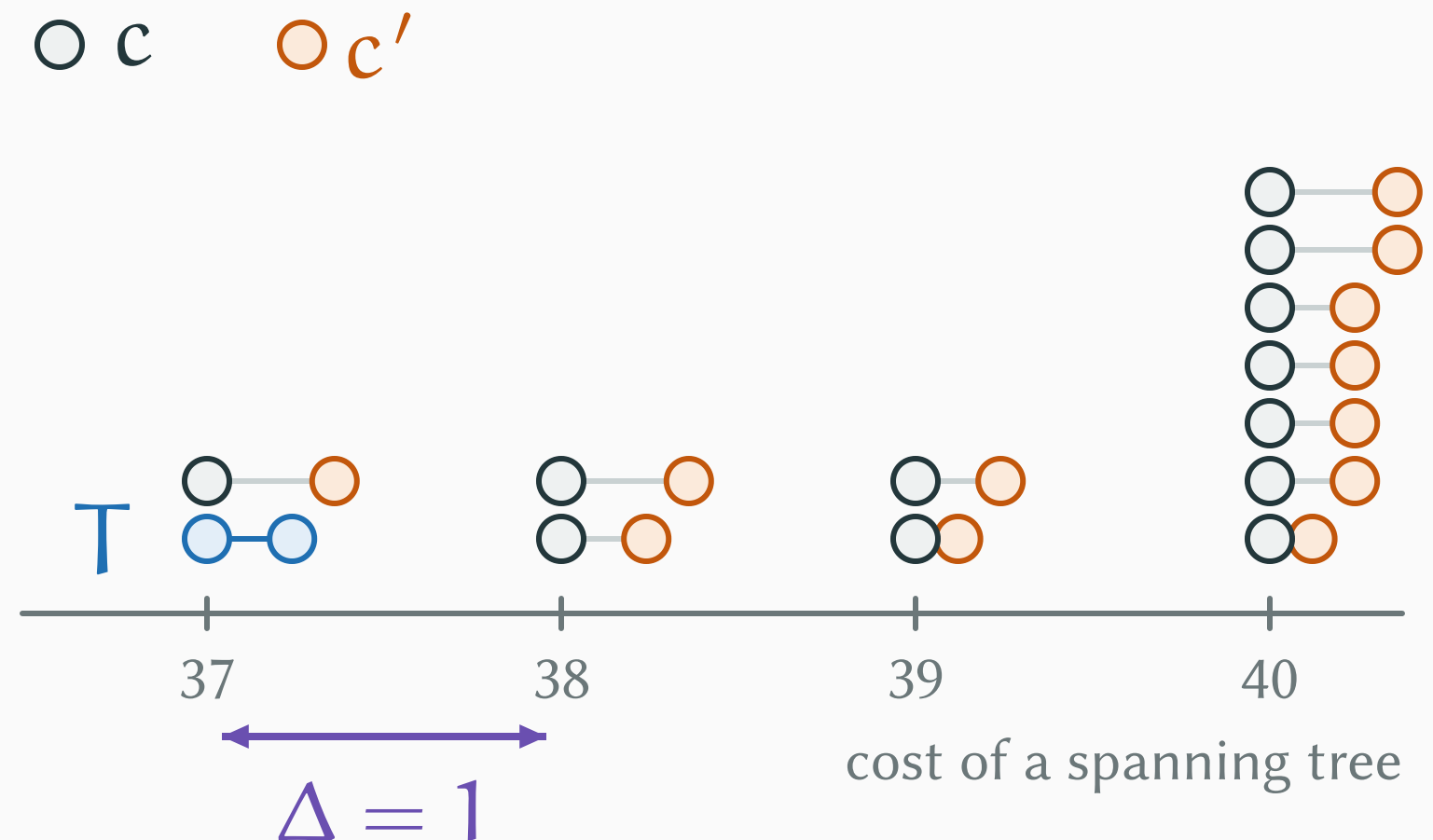
In general, let Δ be the smallest positive difference between the costs of two spanning trees, and choose $\varepsilon_e \geq 0$ with $\sum_e \varepsilon_e < \Delta$.

Let T^* be any spanning tree, T an MST for c' :

$$c(T) \leq c'(T) \leq c'(T^*) < c(T^*) + \Delta.$$

Costs of spanning trees differ by 0 or by at least Δ , so $c(T) \leq c(T^*)$.

T is an MST for c .



- 1 **Greedy:** the best-looking choice at each step, never undone.
It needs a proof.
- 2 **Three algorithms:**
Kruskal's algorithm: cheapest edge unless it closes a cycle
Prim's algorithm: cheapest edge leaving the tree
Reverse-delete: most expensive edge unless it disconnects
- 3 **Why they work:**
Cut property and cycle property, by exchange arguments
Equal costs: perturb

Reference: CLRS, Chapters 21.1, 21.2

Kruskal's algorithm,
Prim's algorithm and
reverse-delete all return
a **minimum spanning tree**.

? *How fast?*
Kruskal's algorithm asks
“same component?”,
Prim's algorithm asks
“cheapest edge leaving S?”
Next lecture: running times