Network Simplex Method (Primal and Dual)

Network (N, A)

Spanning Tree $T = (N, \tilde{A})$, connected, acyclic

(Very easy to find a spanning tree: [V] p. 253 Ex 14.14)

Network Simplex Method (Primal and Dual)

Network $(N, \mathcal{A})$

Spanning Tree $T = (N, \tilde{\mathcal{A}})$, connected, acyclic

(Very easy to find a spanning tree: [V] p. 253 Ex 14.14)

(1) Primal flow always exists:

$$B\tilde{X}_B = -\tilde{b}, \quad x_{ij} = 0, (i,j) \in \mathcal{A} \setminus T$$

Start from a leaf node and go inwards.

(Might be infeasible, some $x_{ij} < 0$)

Network Simplex Method (Primal and Dual)

Network $(N, \mathcal{A})$

Spanning Tree $T = (N, \tilde{\mathcal{A}})$, connected, acyclic

(Very easy to find a spanning tree: [V] p. 253 Ex 14.14)

(2) Dual flow always exists:

$(i,j) \in \mathcal{A}$: $\boxed{y_j - y_i + z_{ij} = c_{ij}}$

$(i,j) \in T$: $y_j - y_i = c_{ij}$, $z_{ij} = 0$,

$(i,j) \in \mathcal{A} \setminus T$: $z_{ij} = c_{ij} - (y_j - y_i)$

(Might be infeasible: $z_{ij} < 0$)

Network Simplex Method (Primal and Dual)

Network (N, A)

Spanning Tree $T = (N, \tilde{A})$, connected, acyclic

(Very easy to find a spanning tree: [V] p. 253 Ex 14.14)

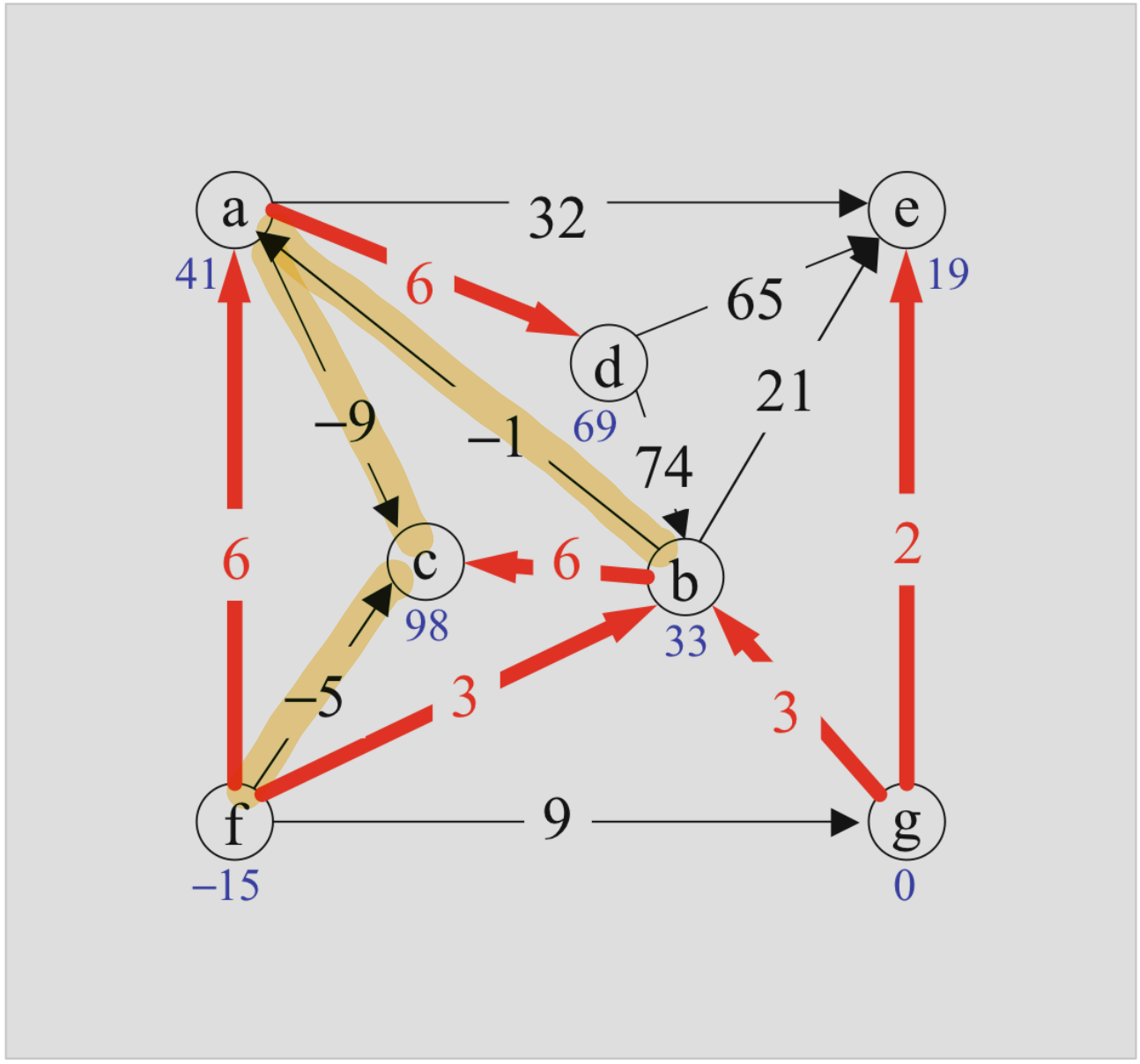
(3) T + one arc $\Rightarrow$ there is a cycle

T − one arc $\Rightarrow$ tree becomes disconnected

(Bertsimas-Tsitsiklis: Intro. Lin. Opt. p. 279

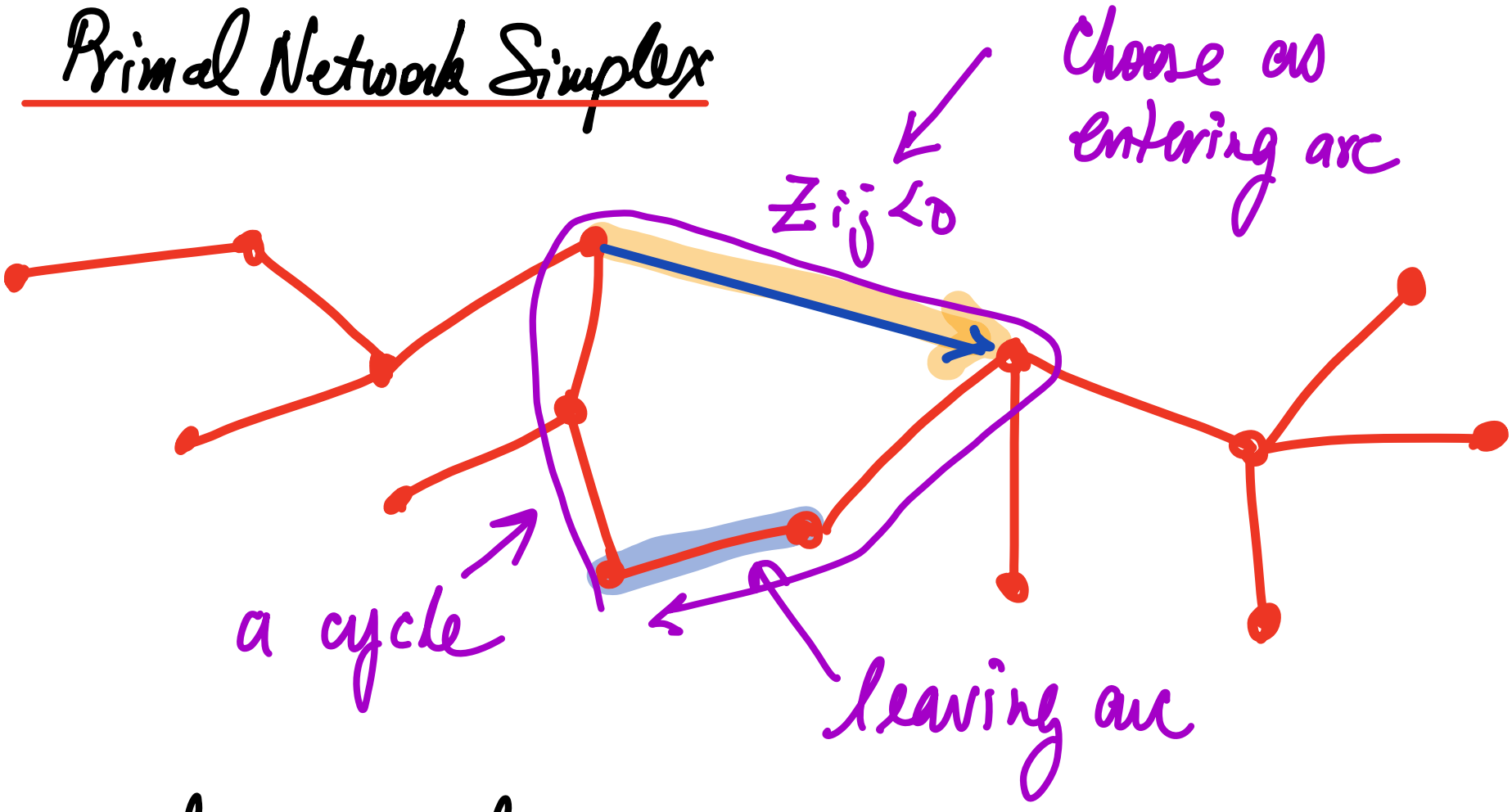
> The network simplex algorithm is widely used in practice, and is included in many commercial optimization codes, due to its simplicity and efficiency. In particular, it tends to run an order of magnitude faster than a general purpose simplex code applied to a network flow problem.

Primal Network Simplex



Not optimal as some $z_{ij} < 0$

Primal Network Simplex



Modification of (spanning) tree and variables:

(1) choose entering and leaving arcs

(2) update primal and dual flows.

Primal Network Simplex

Modify primal flow and spanning tree

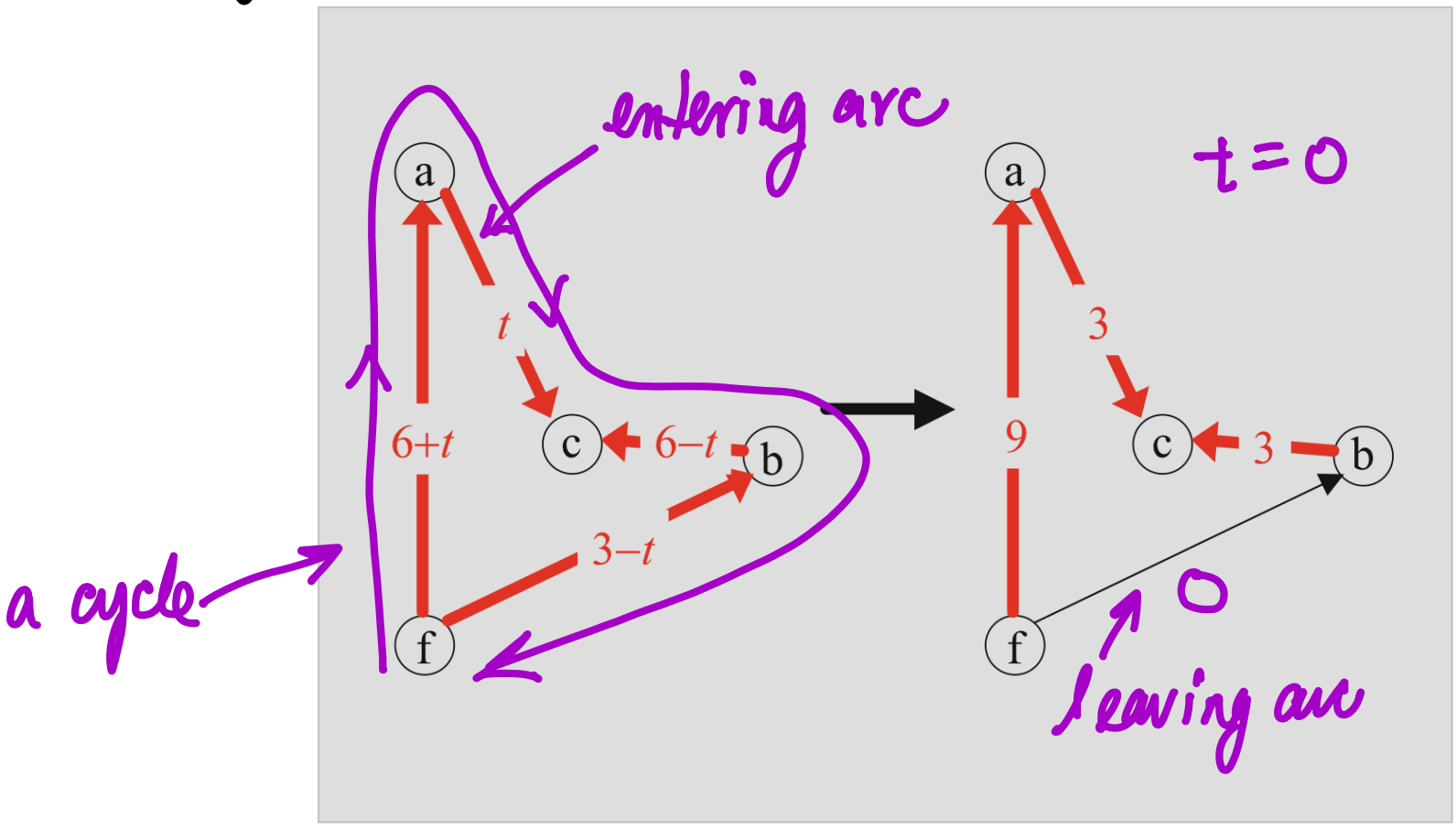


FIGURE 14.7. The cycle produced by including the entering arc with the spanning tree. As the flow t on the entering arc increases, eventually the flow on arc (f,b) becomes zero (when $t = 3$). Hence, arc (f,b) is the leaving arc.

Primal Network Simplex

Modify primal flow and spanning tree

With a little thought, one realizes that the selection rule for the leaving arc in a primal pivot is as follows:

Leaving arc selection rule:

- the leaving arc must be oriented along the cycle in the reverse direction from the entering arc, and
- among all such arcs, it must have the smallest flow.

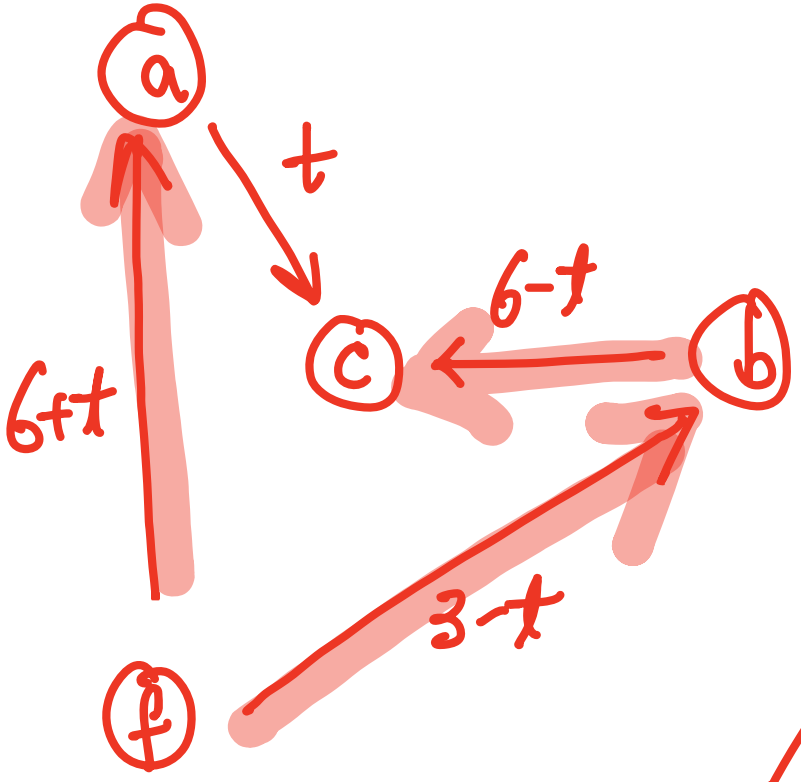
Also, the flows on the cycle get updated as follows:

Primal flows update:

- Flows oriented in the same direction as the leaving arc are decreased by the amount of flow that was on the leaving arc, whereas flows in the opposite direction are increased by this amount.

Primal Network Simplex

Proof of the fact that the cost is reduced



New Cost

$$= (6+t)\, C_{fa} + t\, C_{ac} + (6-t)\, C_{bc}$$

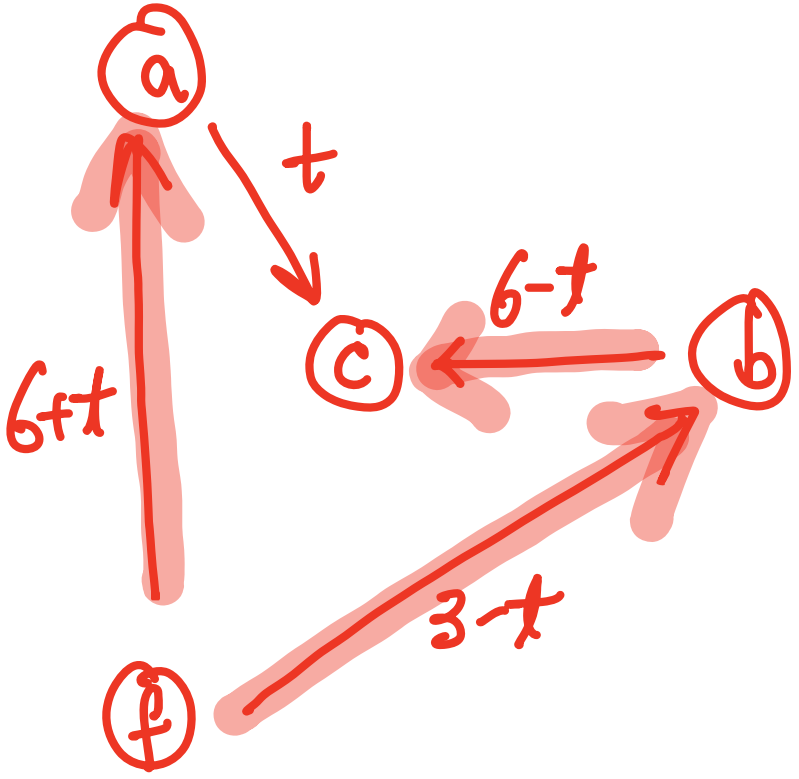
$$+ (3-t)\, C_{fb}$$

$$= \underbrace{6\, C_{fa} + 6\, C_{bc} + 3\, C_{fb}}_{\text{old cost}}$$

$$+ \underbrace{t\,(C_{fa} + C_{ac} - C_{bc} - C_{fb})}_{\text{modified cost}}$$

Primal Network Simplex

Proof of the fact that the cost is reduced



$$t\,(c_{fa} + c_{ac} - c_{bc} - c_{fb})$$

modified cost

$$= t\big((y_a - y_f) + c_{ac} - (y_c - y_b) - (y_b - y_f)\big)$$

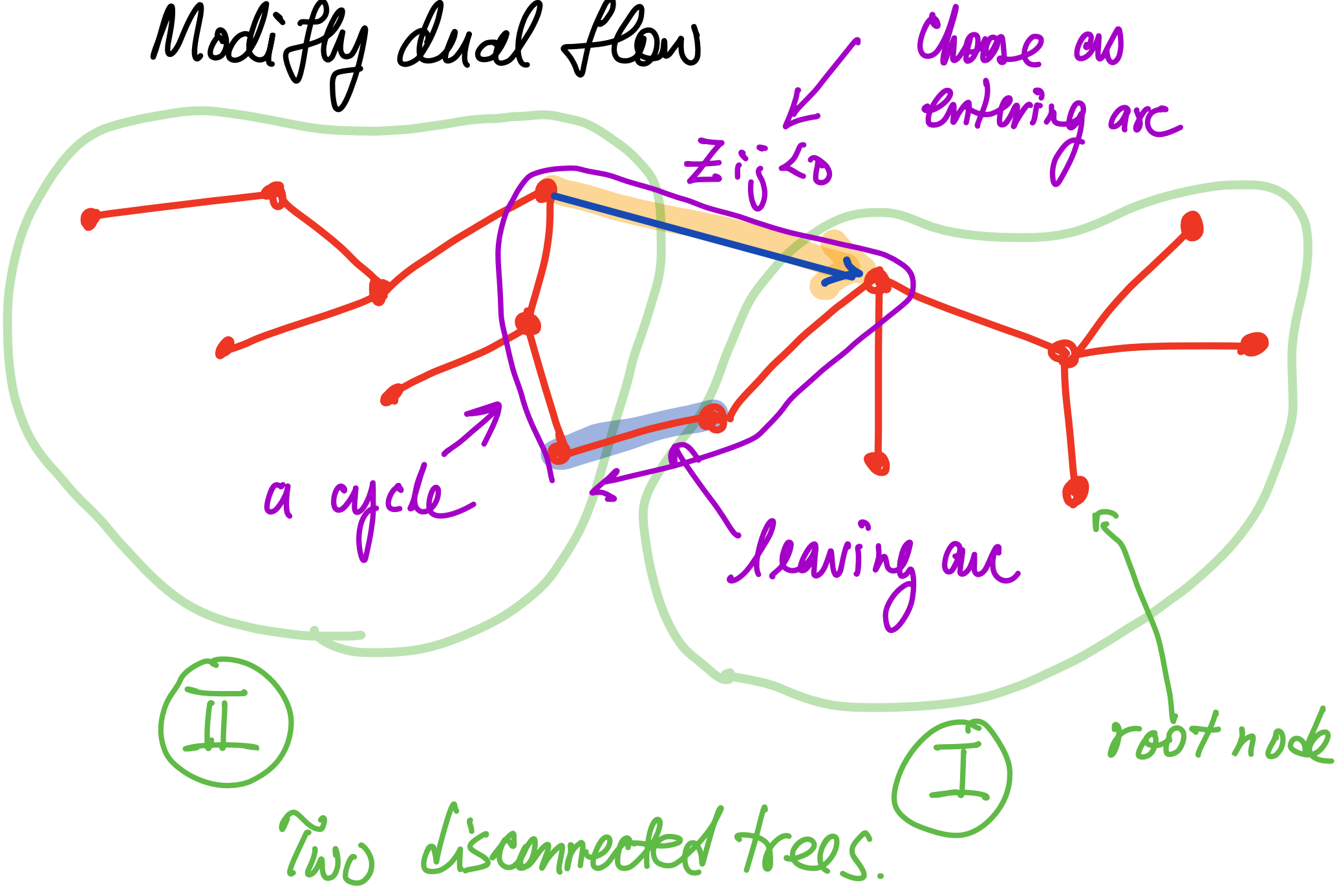
$$= t\,(y_a - y_c + c_{ac})$$

$$= t\,z_{ac} < 0$$

$y_c - y_a + z_{ac} = c_{ac}$

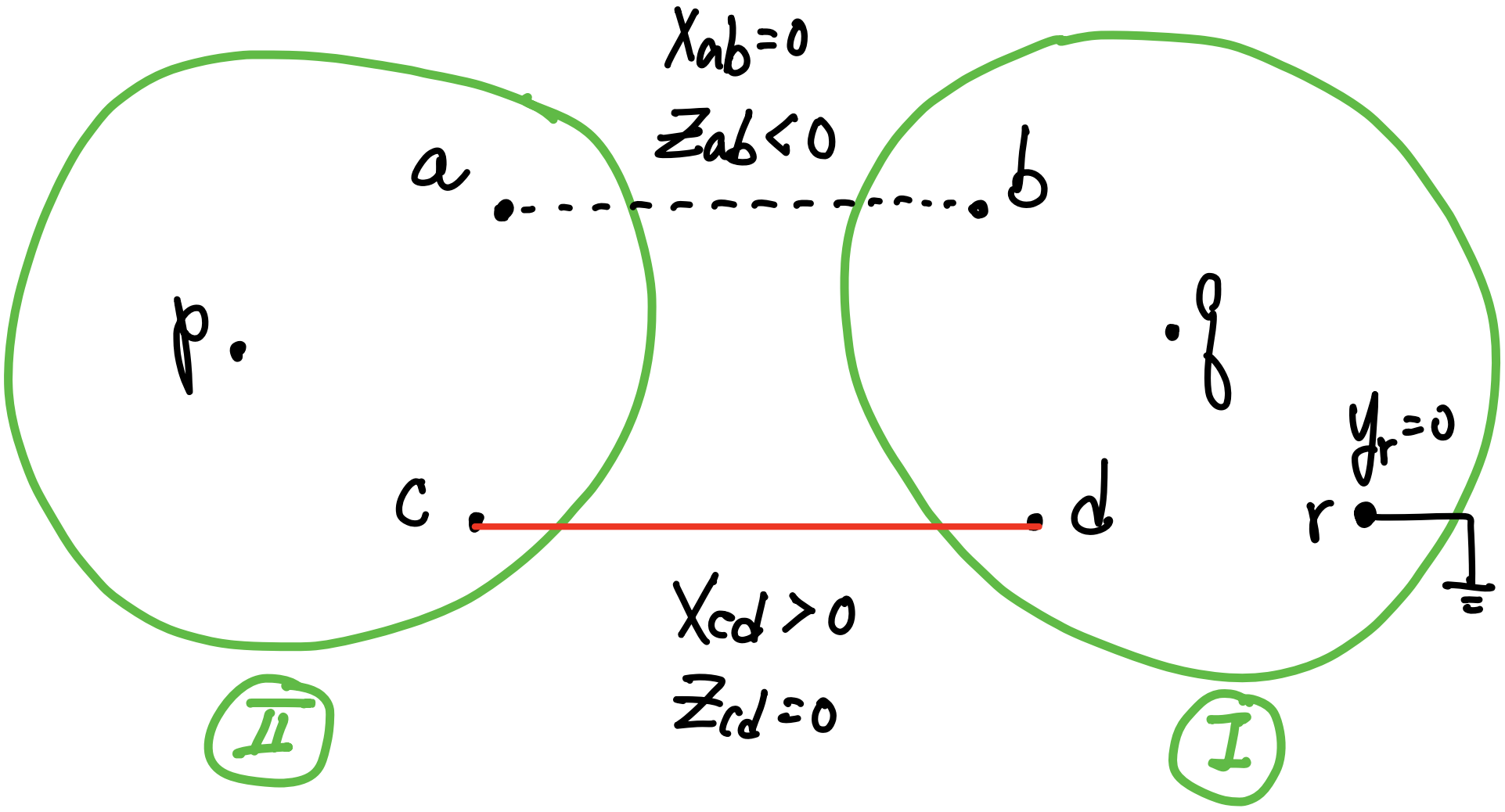
Primal Network Simplex

Modify dual flow



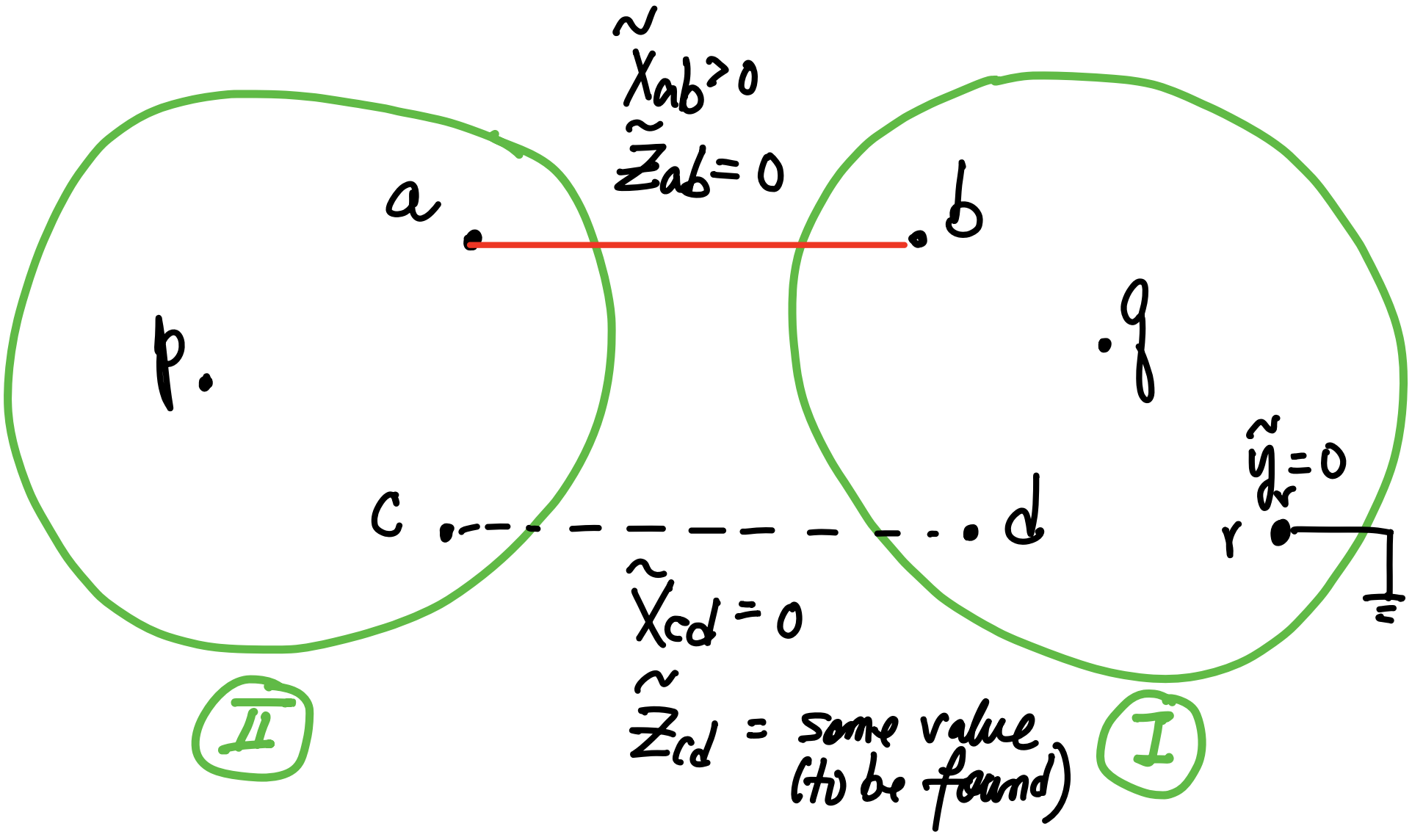
Primal Network Simplex

Before



Primal Network Simplex

After



Primal Network Simplex

Updating dual variables

In (I) (tree containing the root node):

nothing is changed:

$$\tilde{y}_q = y_q \qquad q \in (I)$$

$$\tilde{z}_{ij} = z_{ij} \qquad (i,j) \in (I)$$

Primal Network Simplex

Updating dual variables

Crossing (I) and (II): if $a \in$ (II) $\longrightarrow b \in$ (I)

$$\tilde{y}_b - \tilde{y}_a = c_{ab} \Rightarrow \tilde{y}_a = \tilde{y}_b - c_{ab}$$

$$(y_b - y_a + z_{ab} = c_{ab}) \quad = y_b - c_{ab}$$

$$= y_a - z_{ab}$$

$$\Rightarrow \tilde{y}_p = y_p - z_{ab}, \quad p \in \text{(II)}$$

Primal Network Simplex

Updating dual variables

Crossing (I) and (II): if $a \in$ (II) $\longrightarrow b \in$ (I)

$(i,j) \in$ (II)

$$\tilde{y}_j - \tilde{y}_i + \tilde{z}_{ij} = c_{ij}$$

$$(y_j - z_{ab}) - (y_i - z_{ab}) + \tilde{z}_{ij} = c_{ij}$$

$$\tilde{z}_{ij} = c_{ij} - (y_j - y_i) = z_{ij}$$

Primal Network Simplex

Updating dual variables

Crossing (I) and (II): if $a \in$ (II) $\longrightarrow b \in$ (I)

$i \in$ (II), $j \in$ (I)

$$\tilde{y}_j - \tilde{y}_i + \tilde{z}_{ij} = c_{ij}$$

$$y_j - (y_i - z_{ab}) + \tilde{z}_{ij} = c_{ij}$$

$$\tilde{z}_{ij} = c_{ij} - (y_j - y_i) - z_{ab}$$

$$\tilde{z}_{ij} = z_{ij} - z_{ab}$$

Primal Network Simplex

Updating dual variables

Crossing Ⅰ and Ⅱ: if $a \in$ Ⅱ $\longrightarrow b \in$ Ⅰ

$i \in$ Ⅰ, $j \in$ Ⅱ

$$\tilde{y}_j - \tilde{y}_i + \tilde{z}_{ij} = c_{ij}$$

$$(y_j - z_{ab}) - y_i + \tilde{z}_{ij} = c_{ij}$$

$$\tilde{z}_{ij} = c_{ij} - (y_j - y_i) + z_{ab}$$

$$\tilde{z}_{ij} = z_{ij} + z_{ab}$$

Primal Network Simplex

Updating dual variables

Crossing (I) and (II): if $a \in$ (II) $\longleftarrow$ $b \in$ (I)

$$\tilde{y}_a - \tilde{y}_b = c_{ba} \Rightarrow \tilde{y}_a = \tilde{y}_b + c_{ba}$$

$$= y_b + c_{ba}$$

$$(y_a - y_b + z_{ba} = c_{ba}) \quad = y_a + z_{ba}$$

$$\Rightarrow \tilde{y}_p = y_p + z_{ba}, \; p \in \text{(II)}$$

Primal Network Simplex

Updating dual variables

Crossing Ⅰ and Ⅱ: if $a \in$ Ⅱ $\longleftarrow$ $b \in$ Ⅰ

$(i,j) \in$ Ⅱ

$$\tilde{y}_j - \tilde{y}_i + \tilde{z}_{ij} = c_{ij}$$

$$(y_j + \tilde{z}_{ba}) - (y_i + \tilde{z}_{ba}) + \tilde{z}_{ij} = c_{ij}$$

$$\tilde{z}_{ij} = c_{ij} - (y_j - y_i) = z_{ij}$$

Primal Network Simplex

Updating dual variables

Crossing (I) and (II): if $a \in$ (II) $\longleftarrow$ $b \in$ (I)

$i \in$ (II), $j \in$ (I)

$$\tilde{y}_j - \tilde{y}_i + \tilde{z}_{ij} = c_{ij}$$

$$y_j - (y_i + z_{ba}) + \tilde{z}_{ij} = c_{ij}$$

$$\tilde{z}_{ij} = c_{ij} - (y_j - y_i) + z_{ab}$$

$$\tilde{z}_{ij} = z_{ij} + z_{ab}$$

Primal Network Simplex

Updating dual variables

Crossing (I) and (II): if $a \in$ (II) $\longleftarrow$ $b \in$ (I)

$i \in$ (I), $j \in$ (II)

$$\tilde{y}_j - \tilde{y}_i + \tilde{z}_{ij} = c_{ij}$$

$$(y_j + z_{ba}) - y_i + \tilde{z}_{ij} = c_{ij}$$

$$\tilde{z}_{ij} = c_{ij} - (y_j - y_i) - z_{ab}$$

$$\tilde{z}_{ij} = z_{ij} - z_{ab}$$

Primal Network Simplex

Updating dual variables

Dual variables update:

- If the entering arc crosses from the root-containing tree to the non-root-containing tree, then increase all dual variables on the non-root-containing tree by the dual slack of the entering arc.
- Otherwise, decrease these dual variables by this amount.

Dual slacks update:

- The dual slacks corresponding to those arcs that bridge in the same direction as the entering arc get decremented by the old dual slack on the entering arc, whereas those that correspond to arcs bridging in the opposite direction get incremented by this amount.

Primal Network Simplex

Modify dual flow

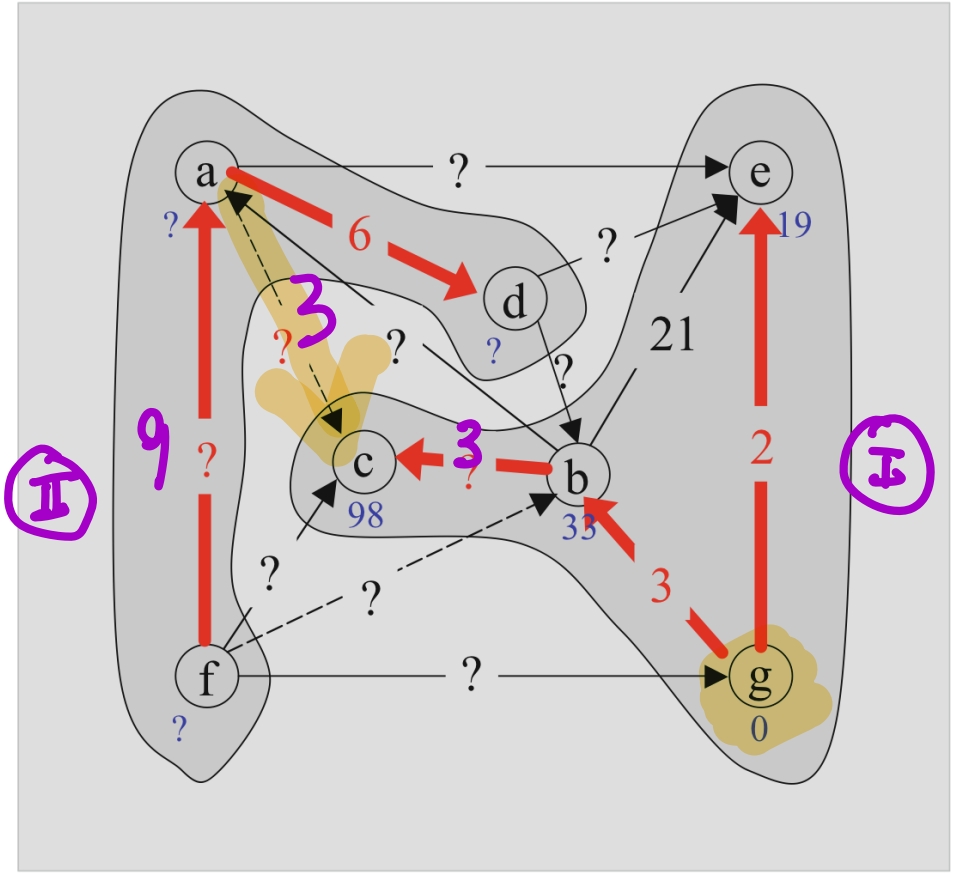
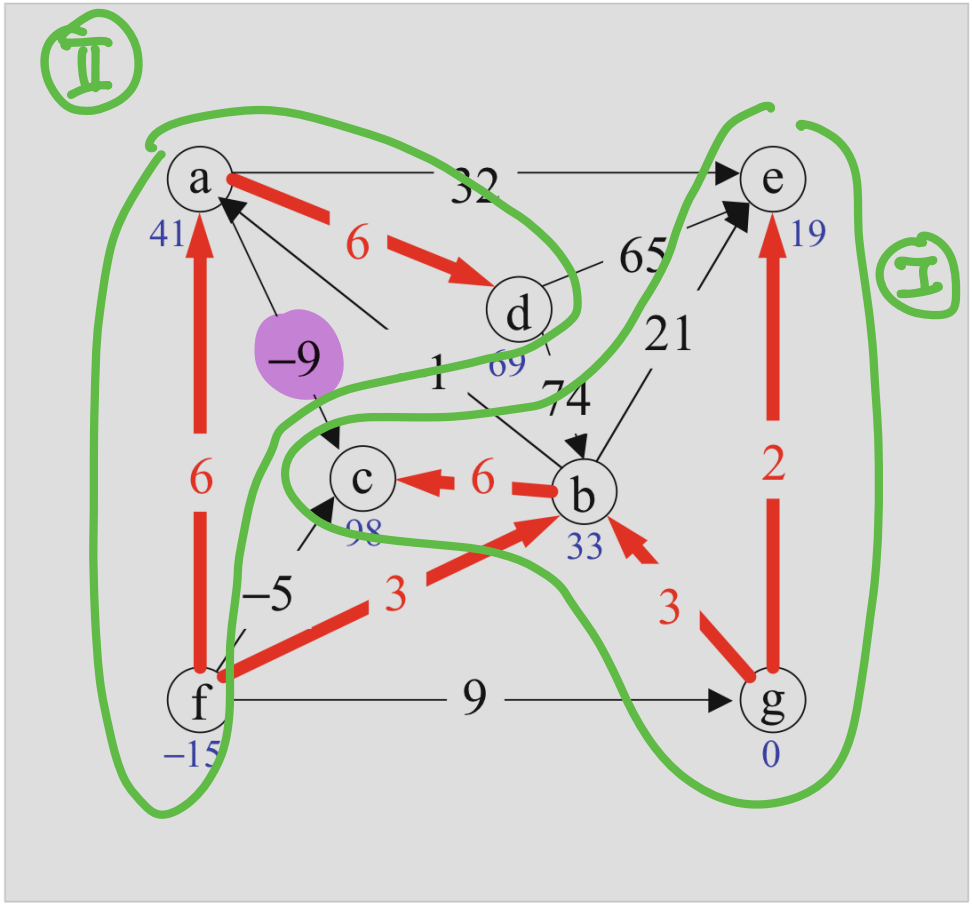
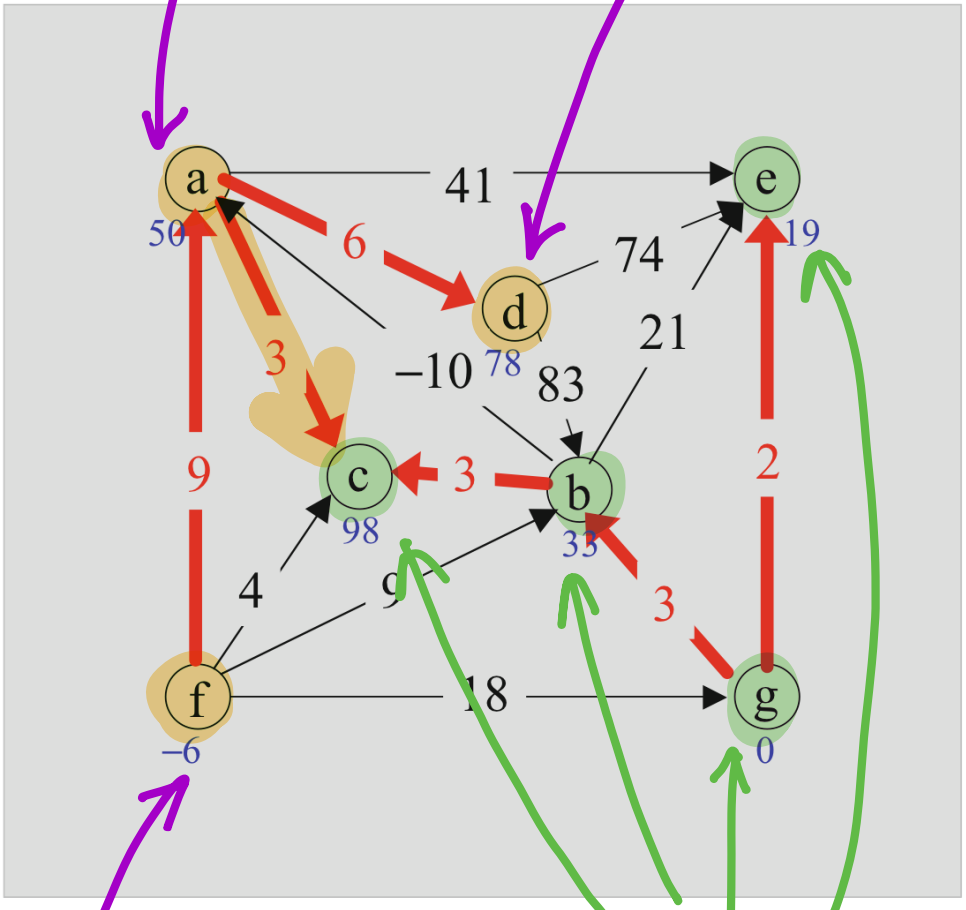


FIGURE 14.8. The two disjoint trees. Primal and dual values that remained unchanged are shown, whereas those that need to be updated are shown as question marks.

Primal Network Simplex

Modify dual flow





50 = 41 − (−9)

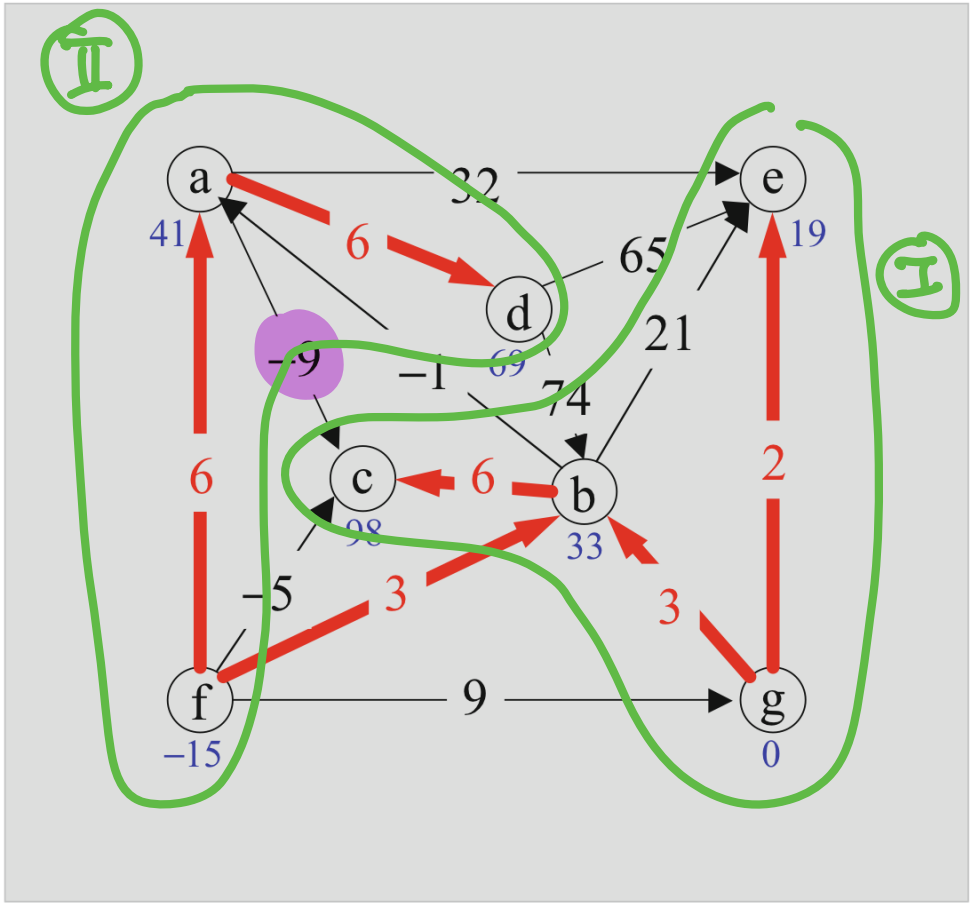
78 = 69 − (−9)

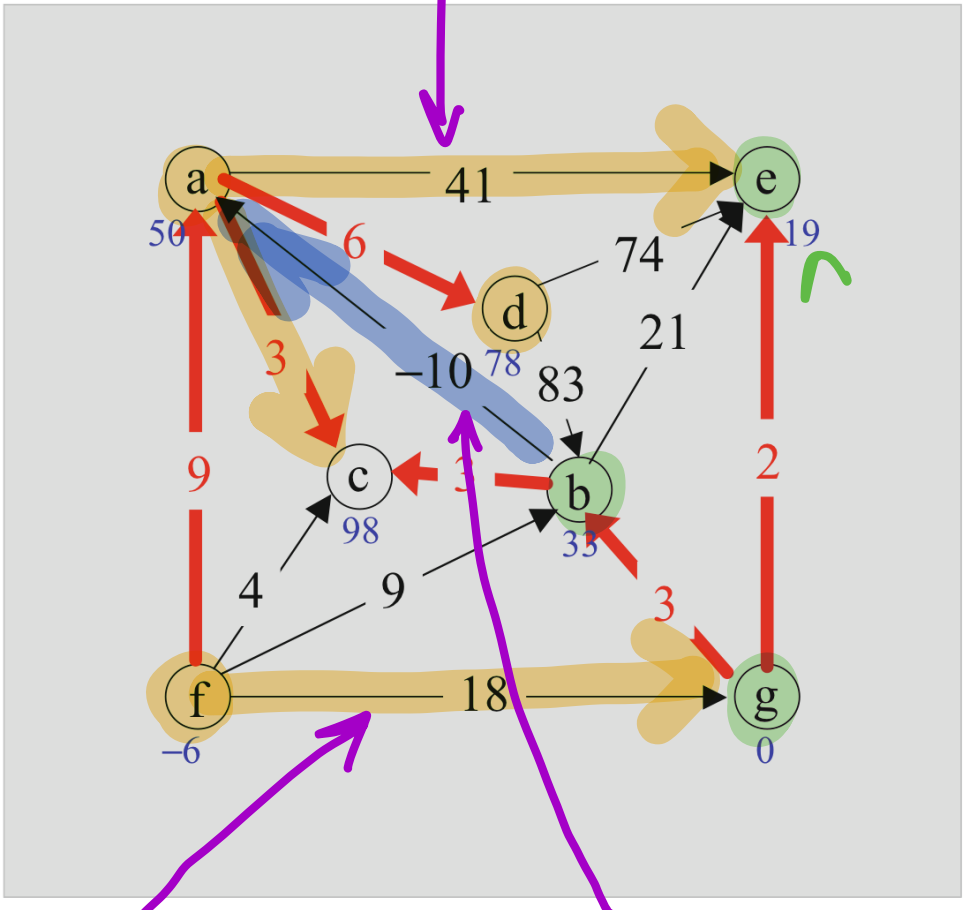
−6 = −15 − (−9)

no change

Primal Network Simplex

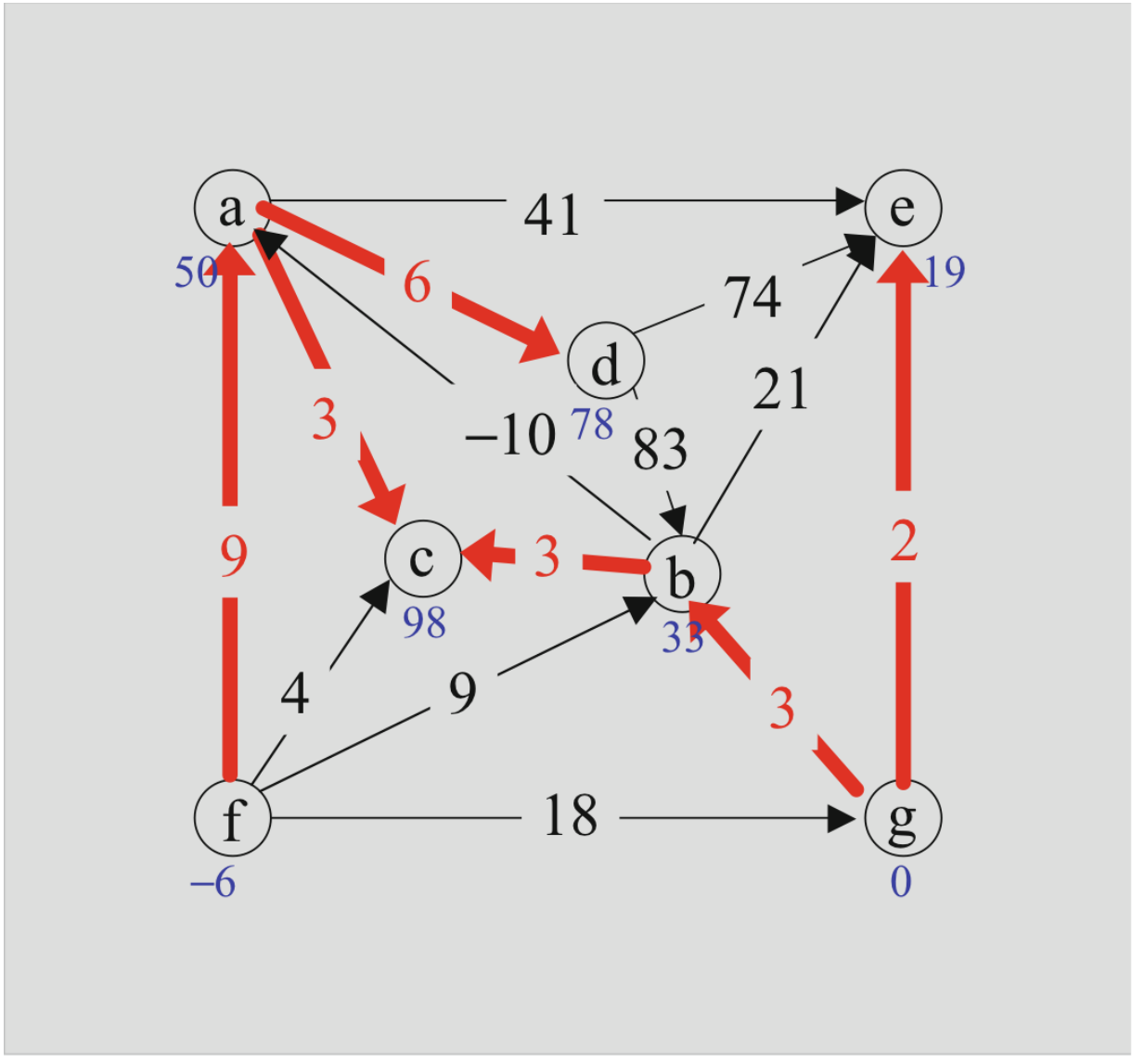
Modify dual flow





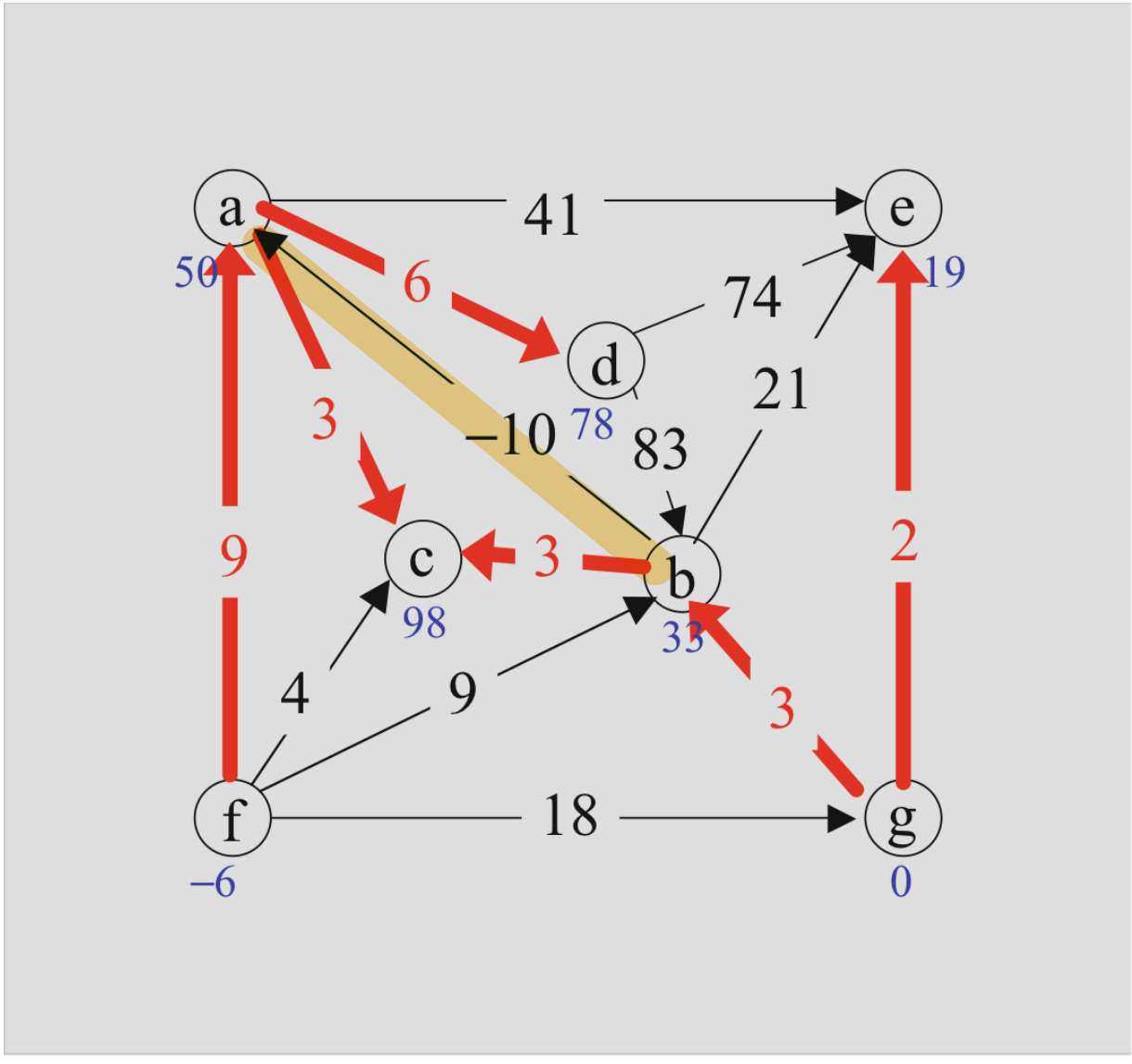
Primal Network Simplex

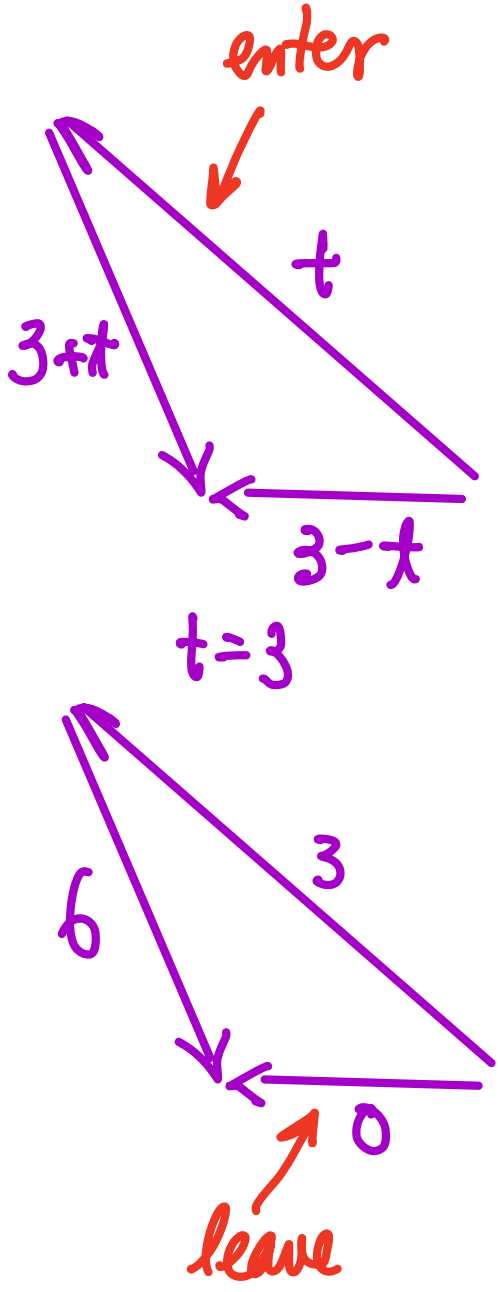
End of 1st step



Primal Network Simplex

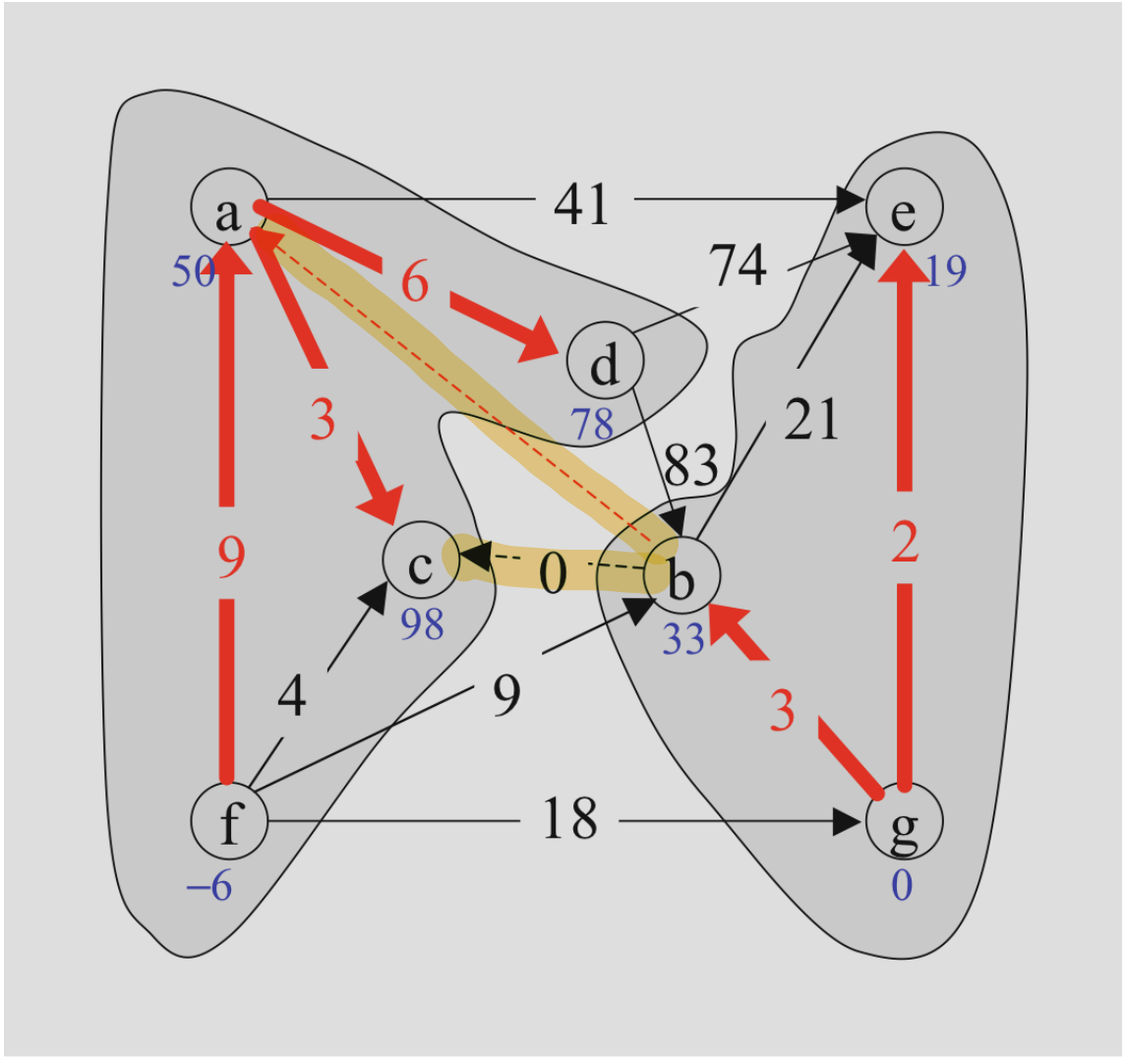
End of 1st step

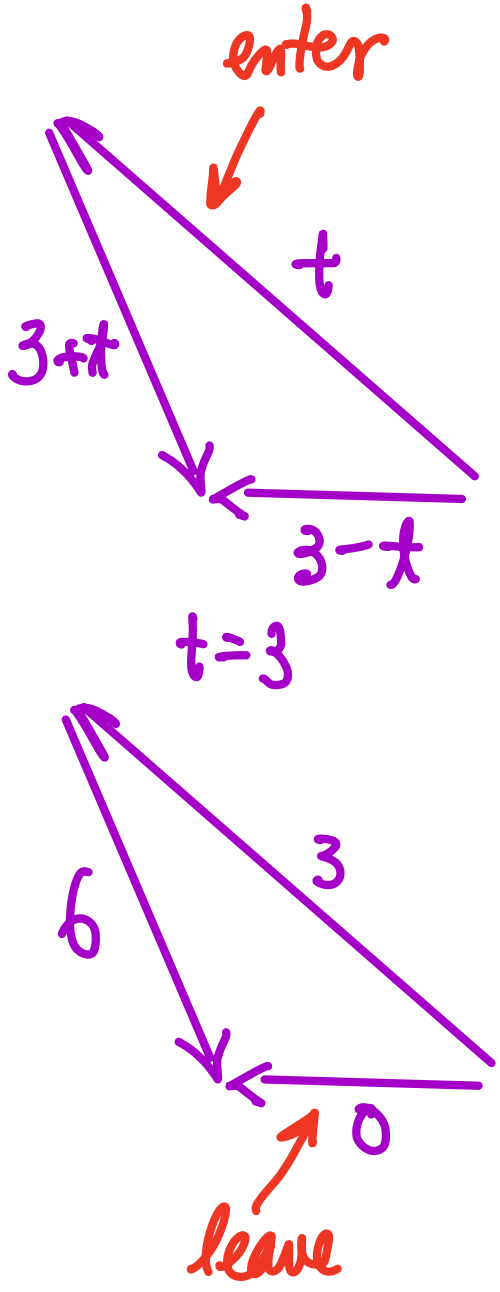




Primal Network Simplex

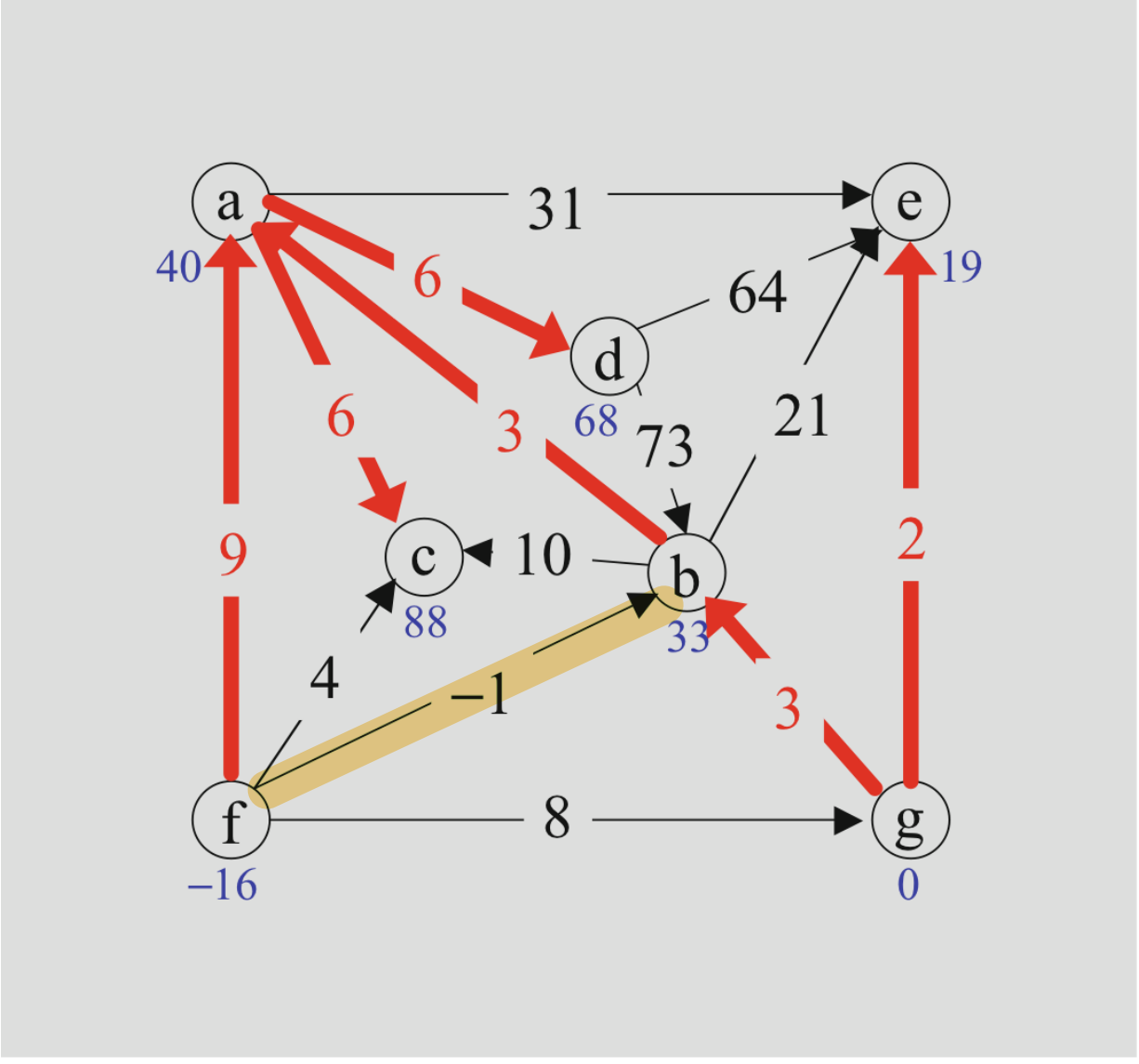
2nd Step





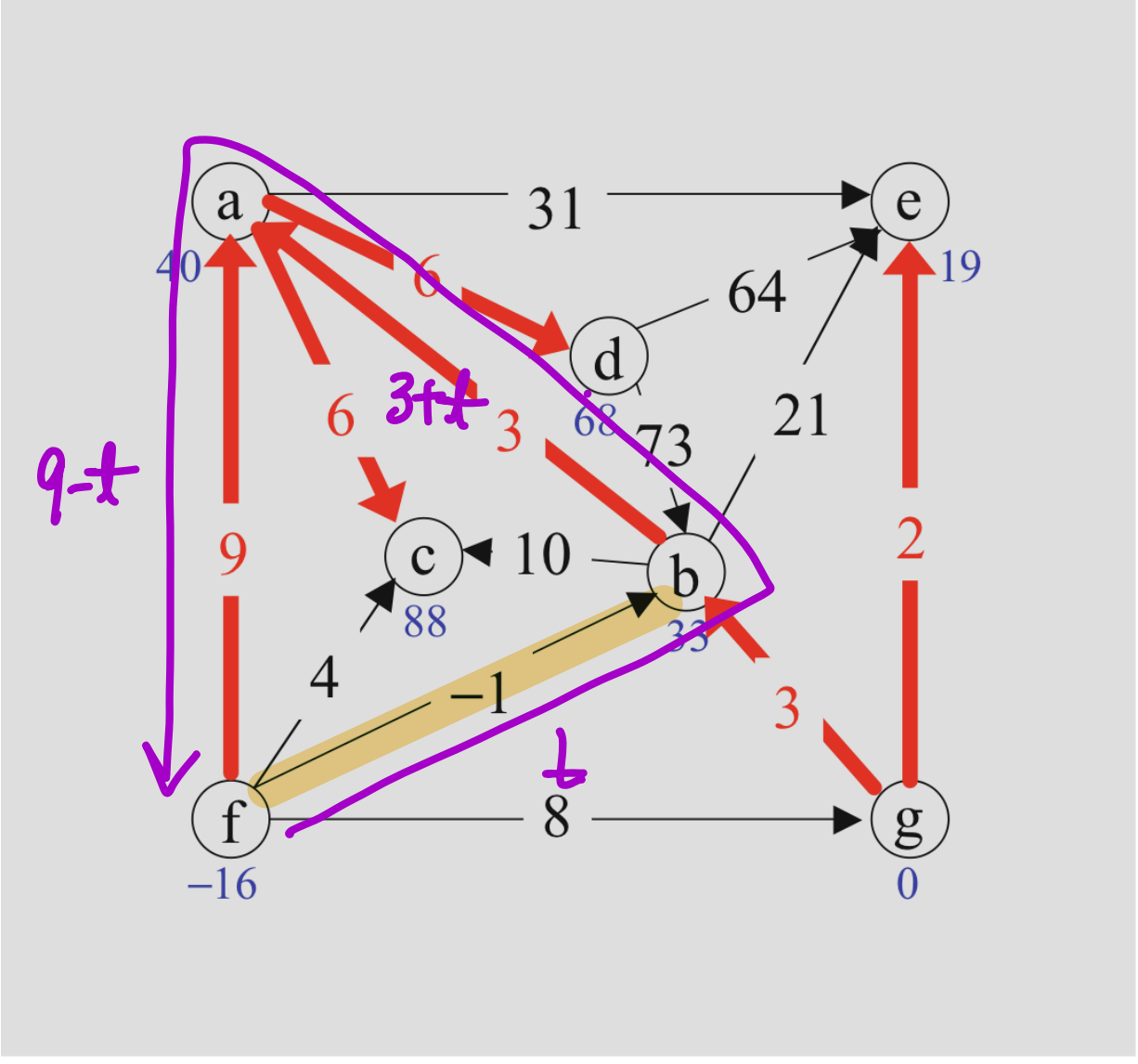
Primal Network Simplex

End of 2nd step



Primal Network Simplex

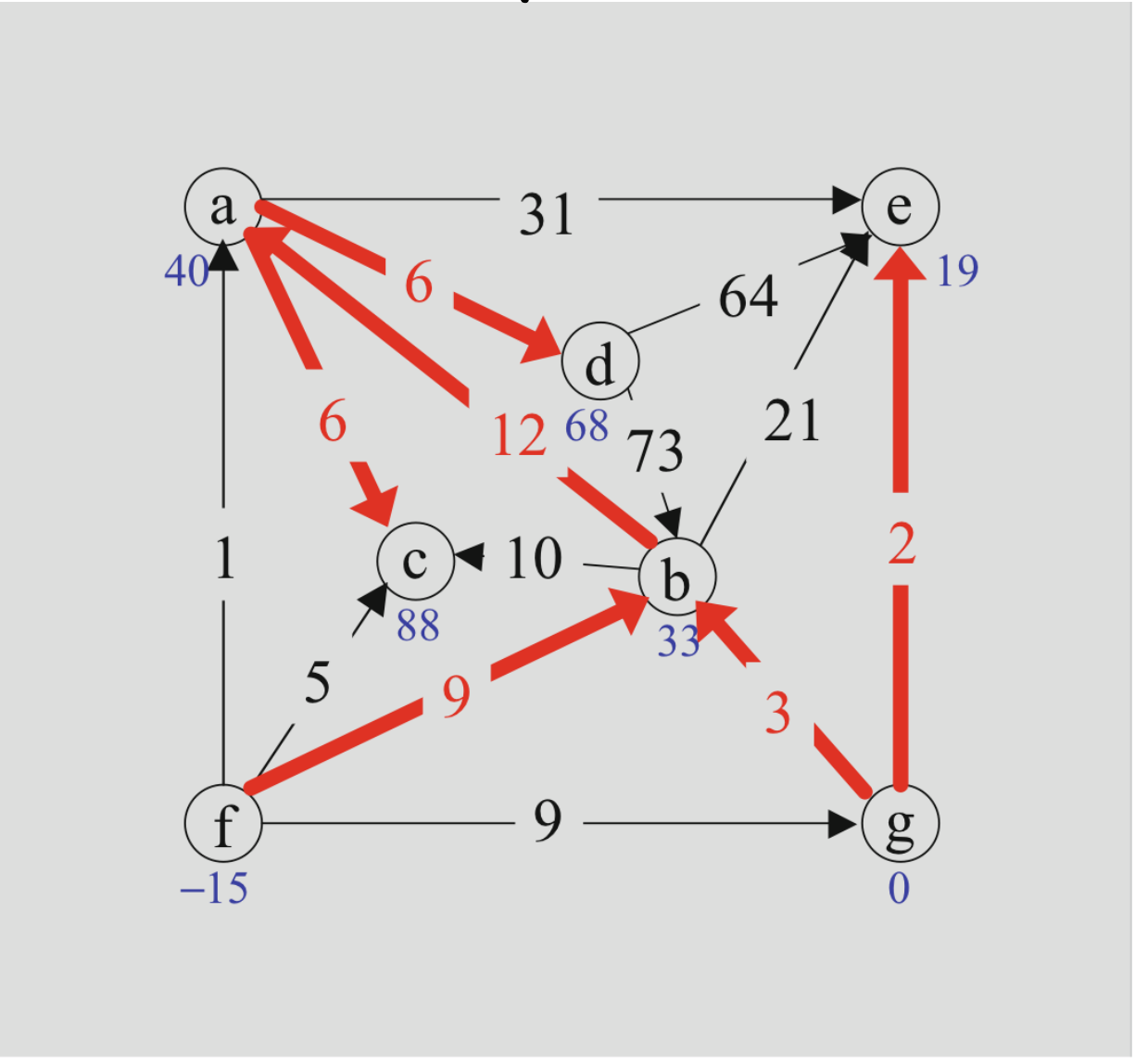
End of 2nd step



t=9

Primal Network Simplex

3rd and last step : optimal



Dual Network Simplex

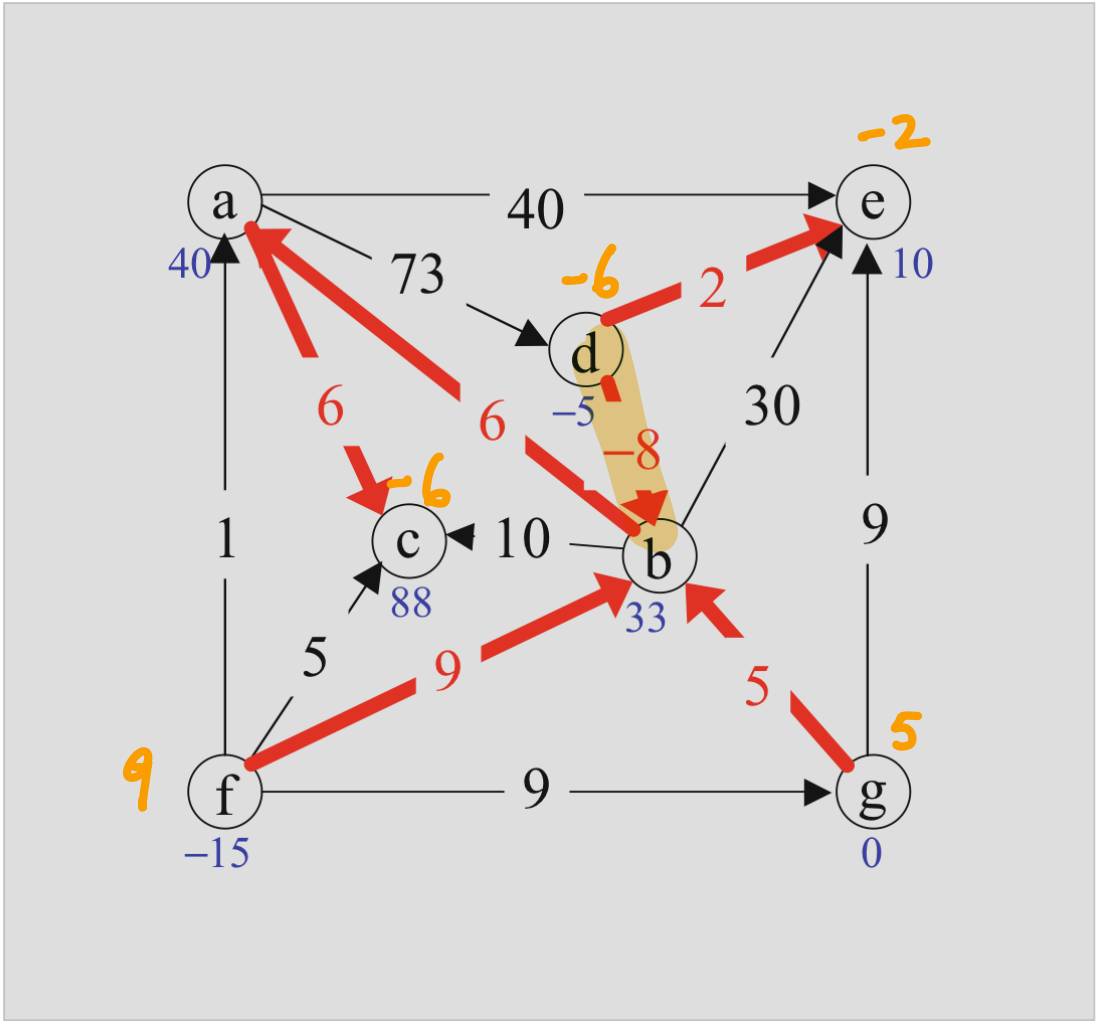


FIGURE 14.13. A tree solution that is dual feasible but not primal feasible.

Dual Network Simplex

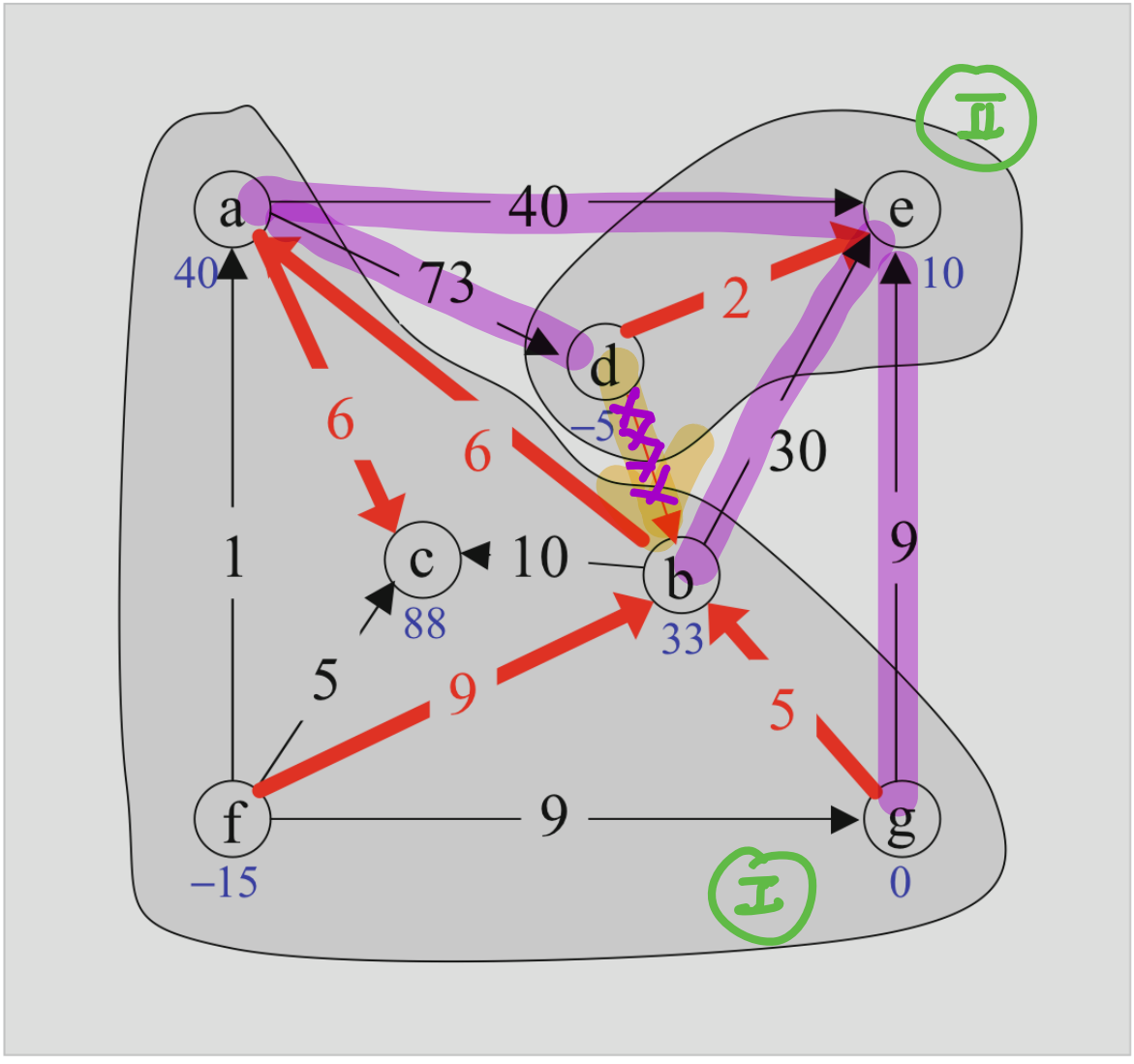


FIGURE 14.14. The two subtrees for the first pivot of the dual simplex method.

Dual Network Simplex

To this end, let us consider the general situation. As mentioned above, the spanning tree with the leaving arc removed consists of two disjoint trees. The entering arc must reconnect these two trees.

First, consider a reconnecting arc that connects in the same direction as the leaving arc. If chosen as the entering arc, this arc will become basic in which case its dual slack will drop from a positive value to zero. The dual slack for all other nontree arcs pointing in the same direction will drop by the same amount. The leaving arc will become a nontree arc and therefore its dual slack will also drop by this amount. But, it started out at zero. After the pivot, it will be negative. In other words, dual feasibility will be lost. For this reason, we restrict our attention to reconnecting arcs that point in the opposite direction.

Entering arc selection rule:

- the entering arc must bridge the two subtrees in the opposite direction from the leaving arc, and
- among all such arcs, it must have the smallest dual slack.

Dual Network Simplex

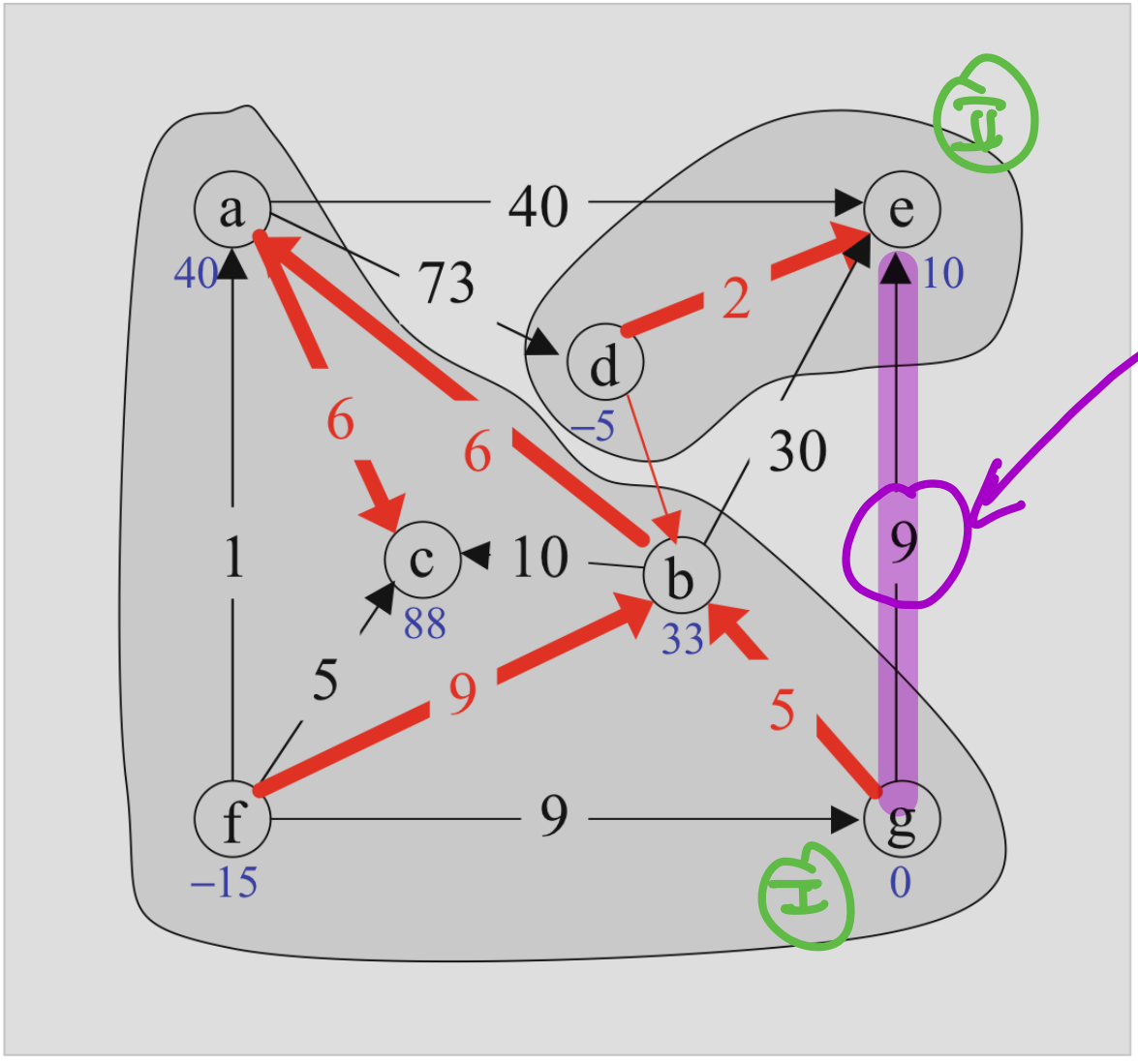
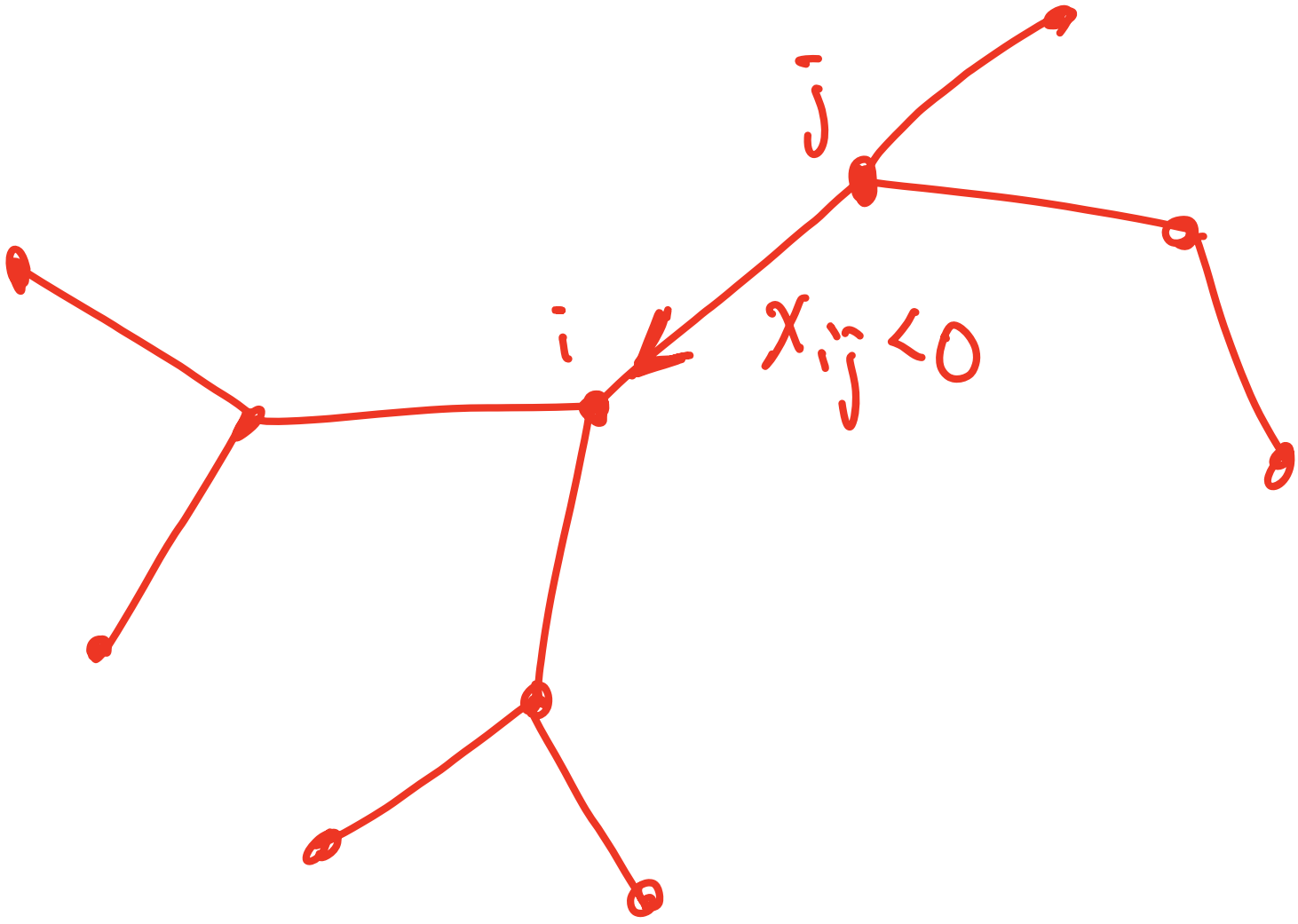
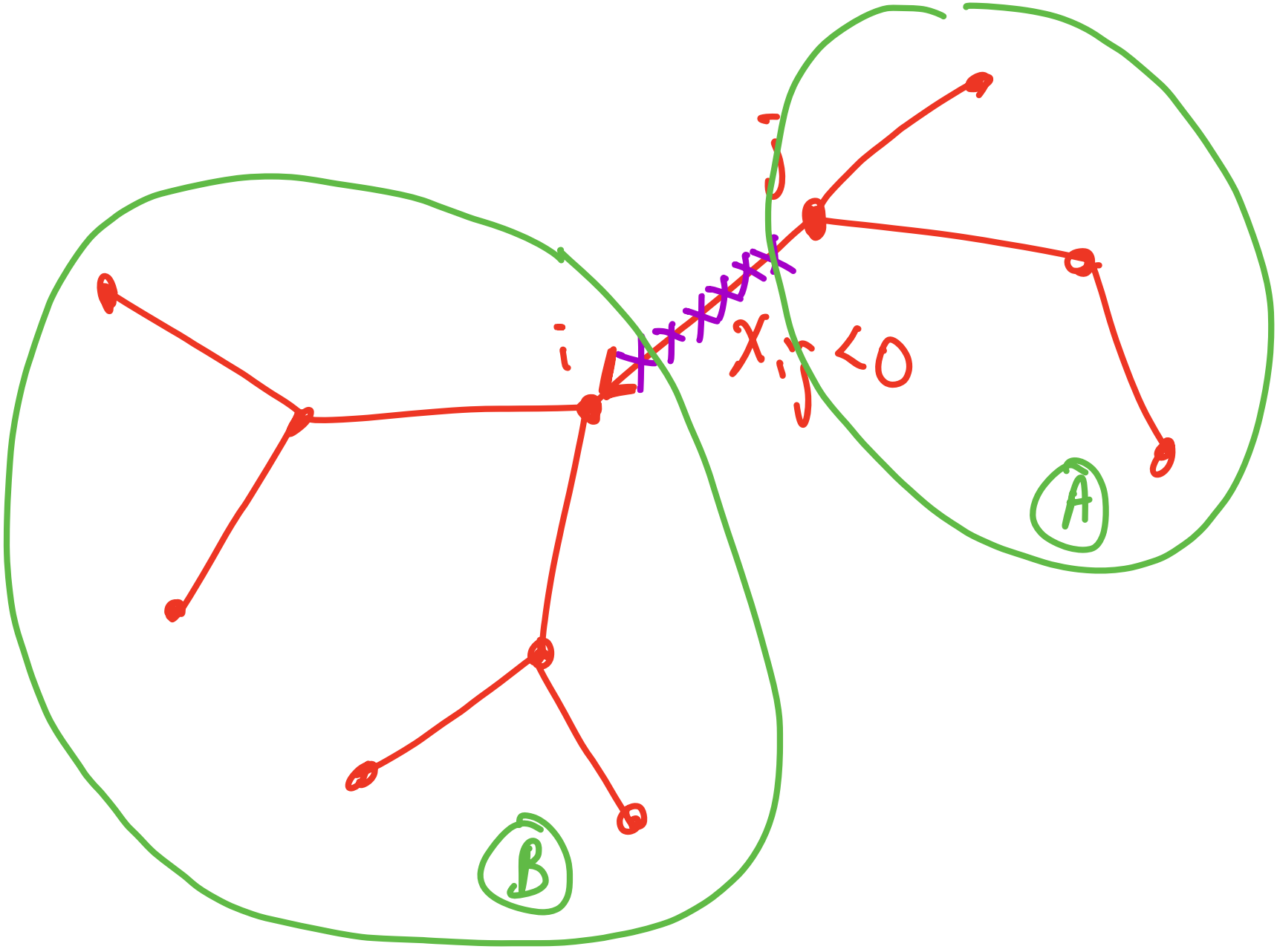


FIGURE 14.14. The two subtrees for the first pivot of the dual simplex method.

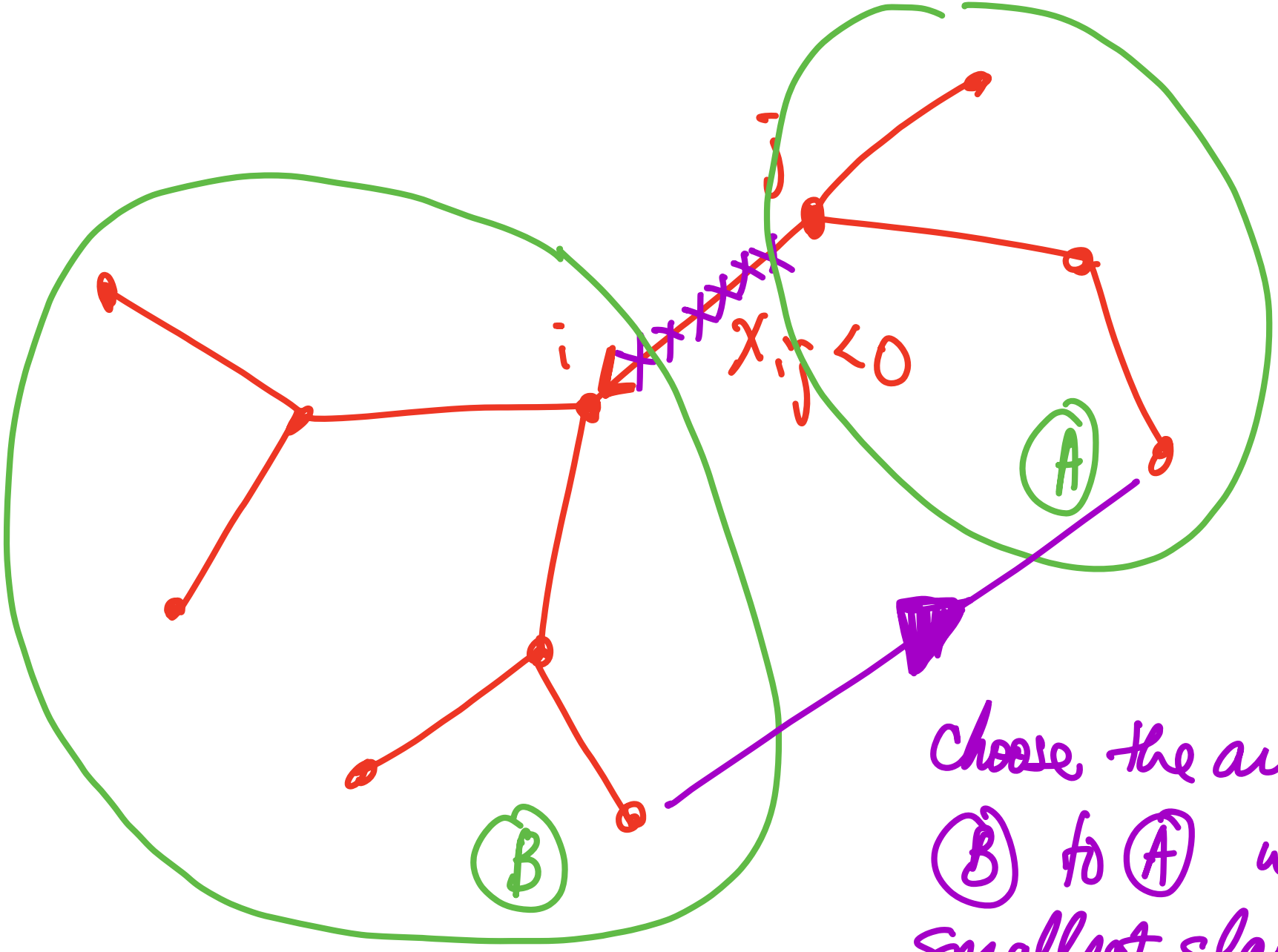
Dual Network Simplex



Dual Network Simplex



Dual Network Simplex



Choose the arc from (B) to (A) with smallest slack

Dual Network Simplex

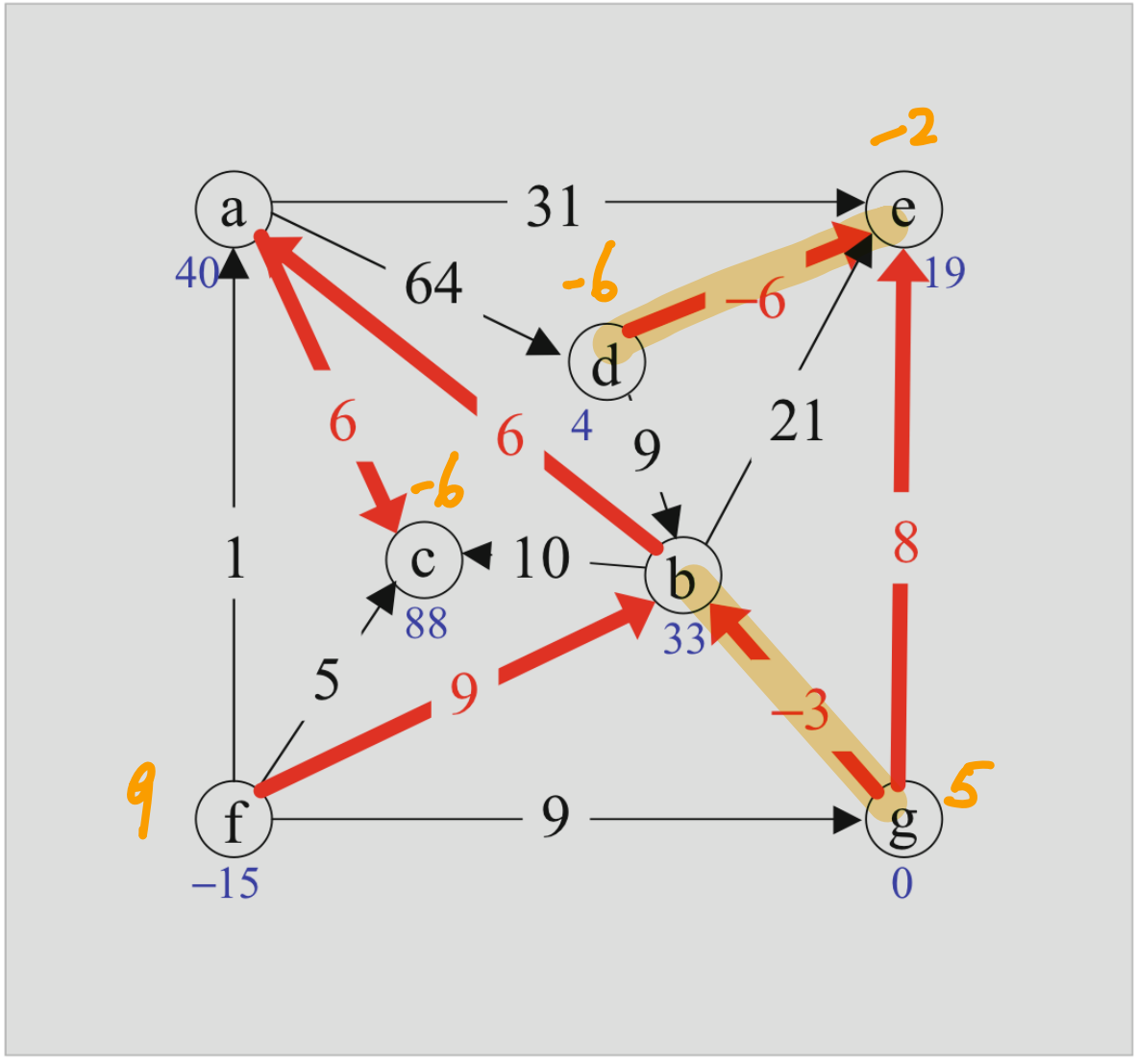


FIGURE 14.15. The tree solution after the first pivot.

Dual Network Simplex

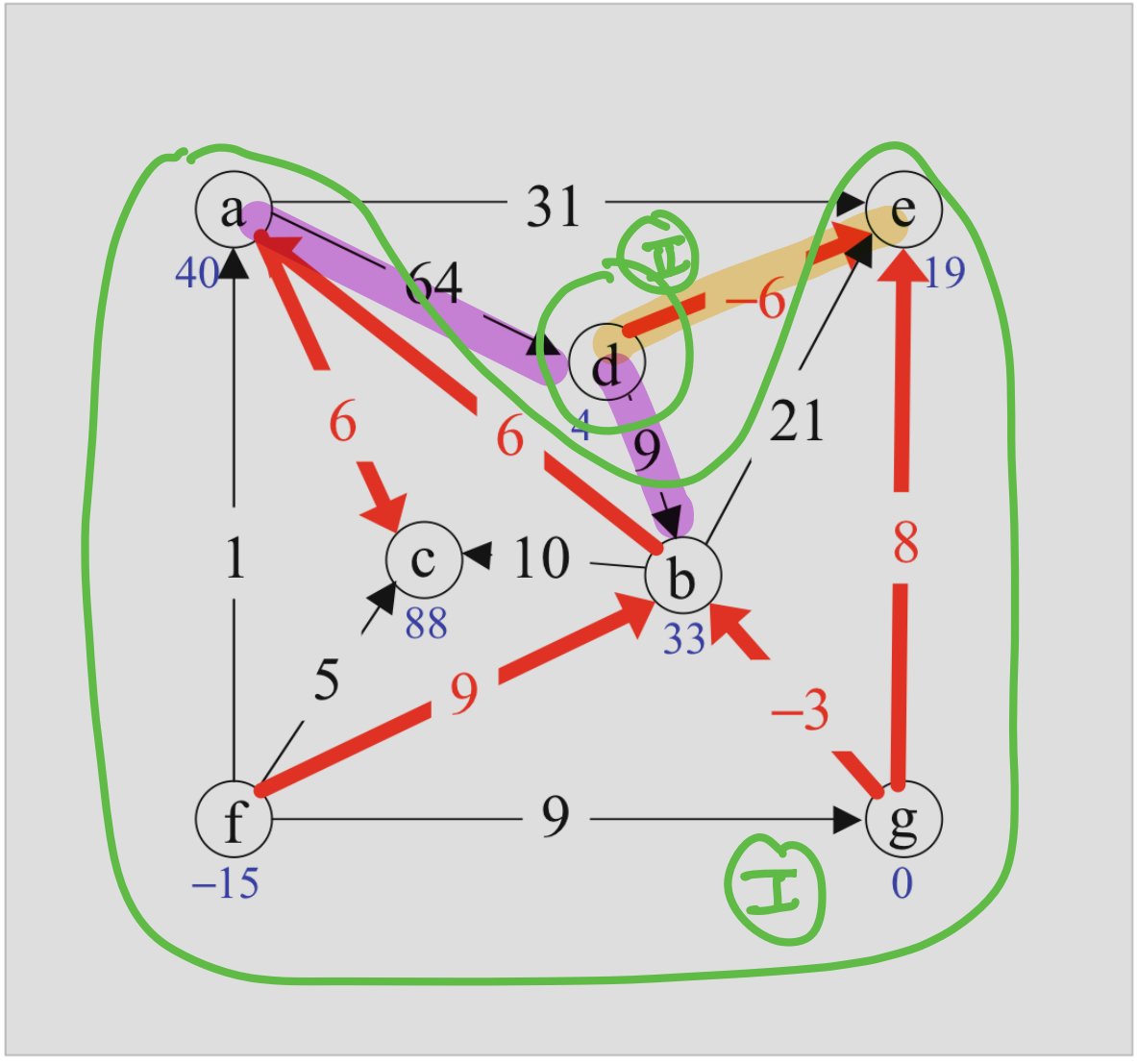


FIGURE 14.15. The tree solution after the first pivot.

Dual Network Simplex

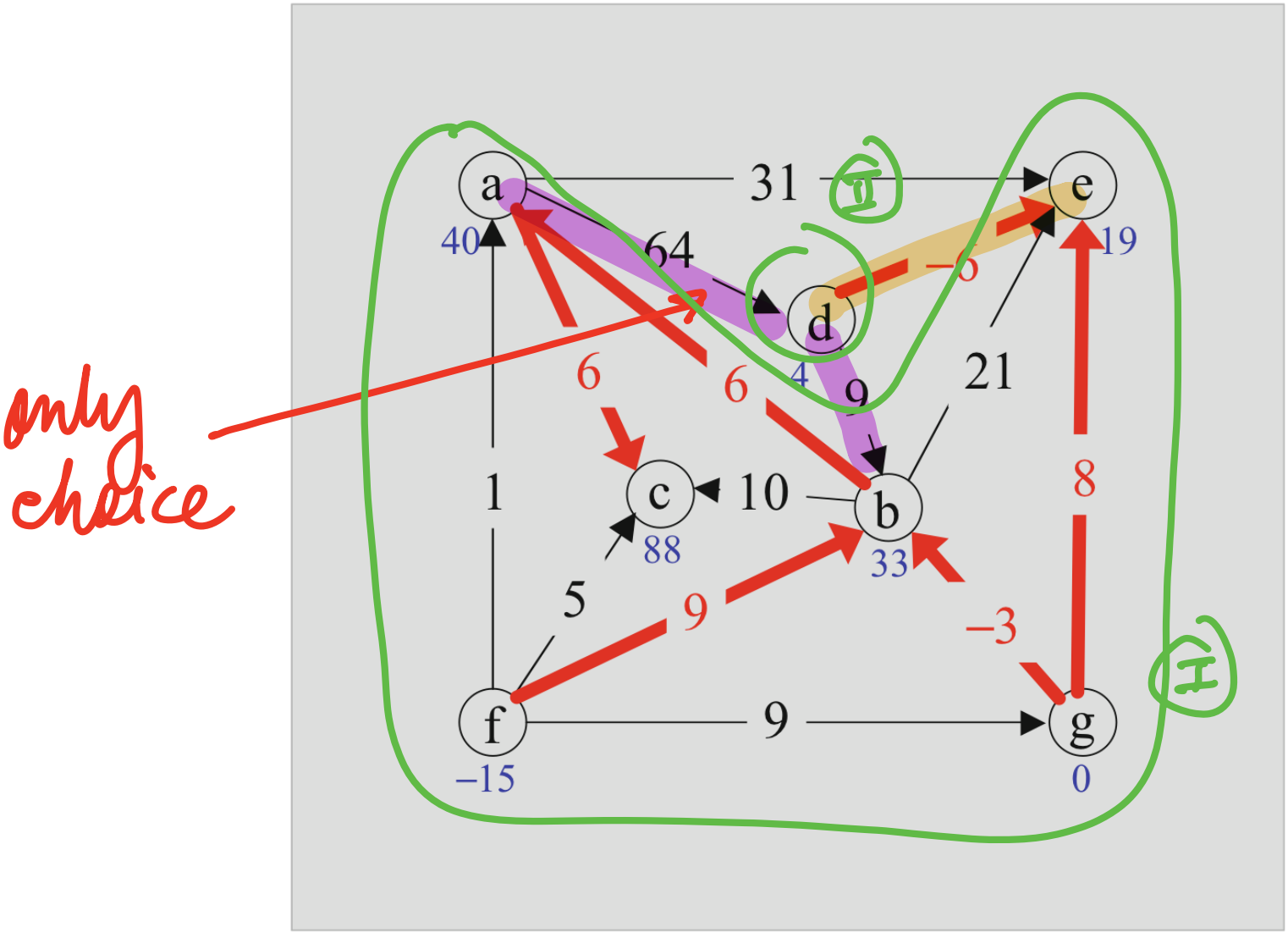


FIGURE 14.15. The tree solution after the first pivot.

Dual Network Simplex

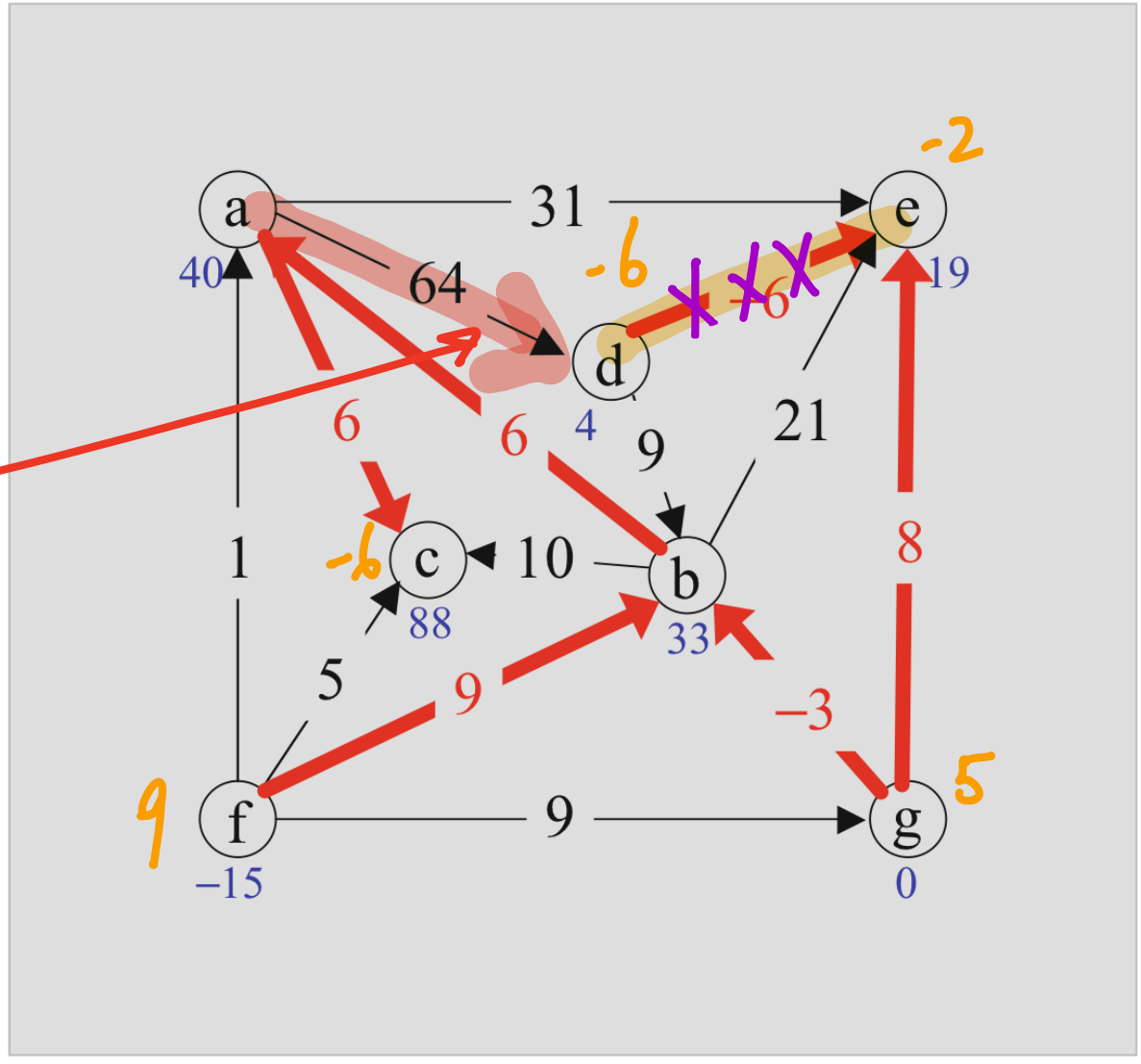


FIGURE 14.15. The tree solution after the first pivot.

Dual Network Simplex

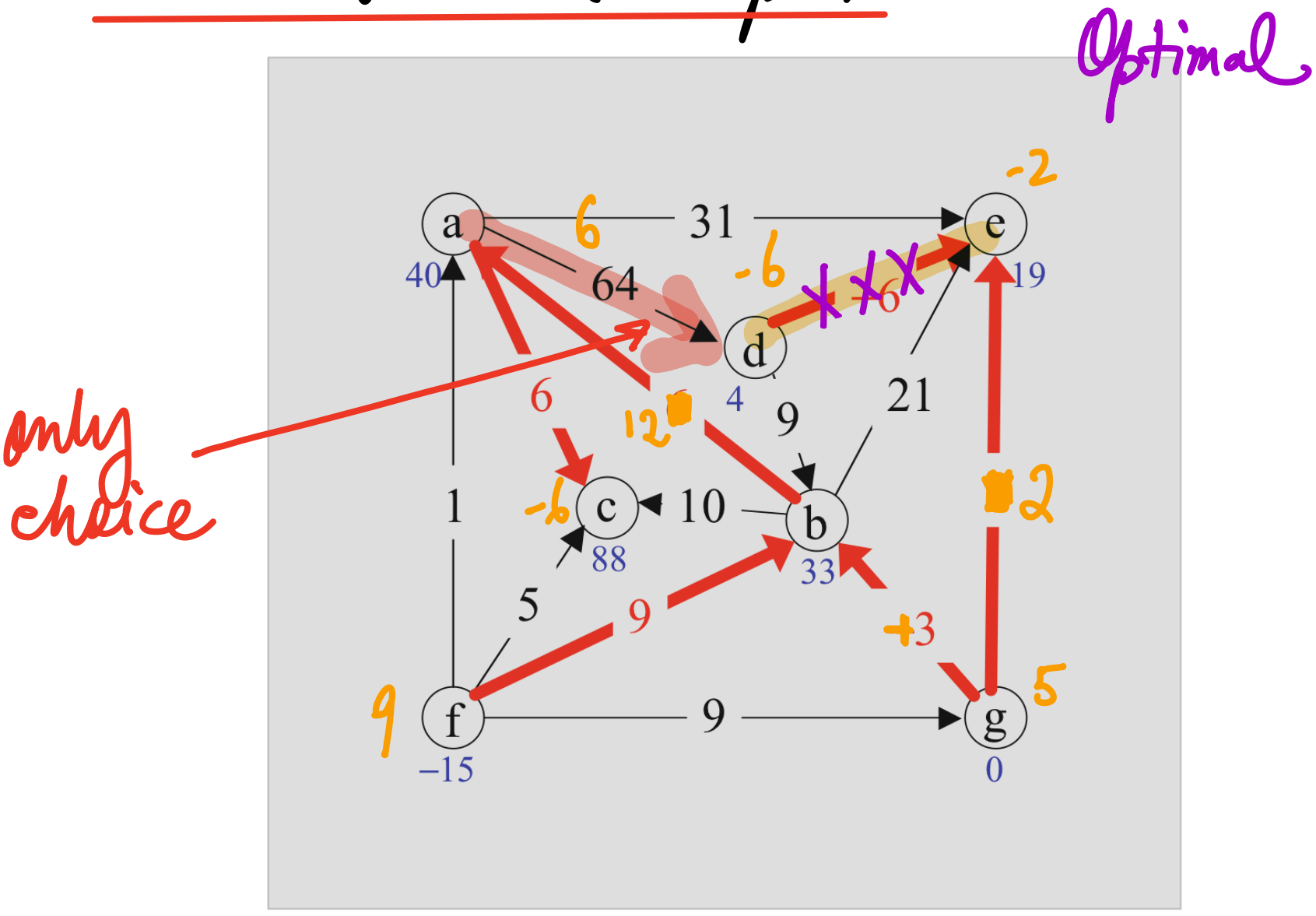


FIGURE 14.15. The tree solution after the first pivot.

5. Putting It All Together

As we saw in Chapter 5, for linear programming the primal and the dual simplex methods form the foundation on which one can build a few different variants of the simplex method. The same is true here in the context of network flows.

For example, one can build a two-phased procedure in which one first uses the dual network simplex method (with costs artificially and temporarily altered to ensure dual feasibility of an initial tree solution) to find a primal feasible solution and then uses the primal network simplex method to move from the feasible solution to an optimal one.

Alternatively, one can use the primal network simplex method (with supplies temporarily altered to ensure primal feasibility of an initial tree solution) to find a dual feasible solution and then use the dual network simplex method (with the original supplies) to move from the dual feasible solution to an optimal one.

Finally, as described for linear programming in Chapter 7, one can define a parametric self-dual method in which primal pivots and dual pivots are intermingled as needed so as to reduce a perturbation parameter μ from ∞ to zero.