Disjoint Set - Union/Find

Disjoint Set – Union/Find

- n distinct elements named $1, 2, \ldots, n$
- Initially, each element is in its own set

$$S_1 = \{1\}, \quad S_2 = \{2\}, \ldots, \quad S_n = \{n\}$$

- Each set has a representative element
- S_x : Set represented by element x

Operations:

Union(S_x, S_y): Create set $S = S_x \cup S_y$ and return the representative of S

Find(z): Given (a ptr to) z, find set S that contains z and return the representative of S

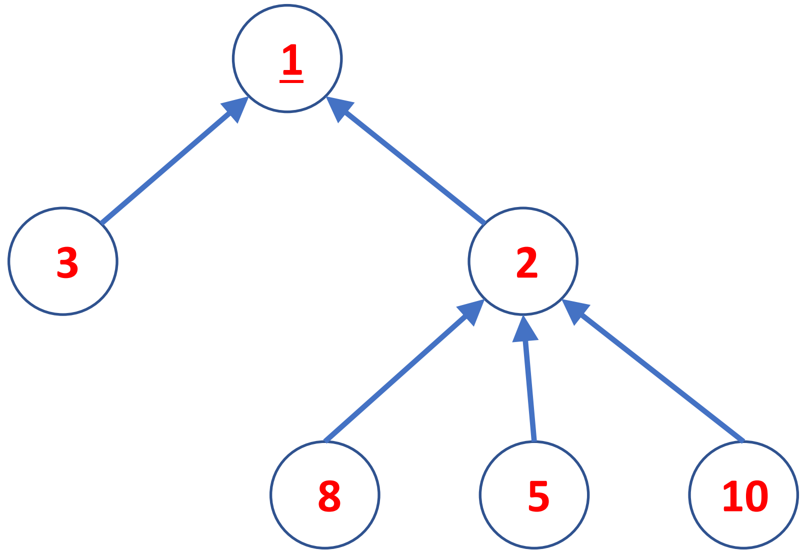
σ : Sequence of $n - 1$ **Unions** mixed with $m \geq n$ **Finds**

Goal: a data structure that minimizes the total cost of executing such sequences

Forest structure for Union-Find

- Each set is represented by a tree
- The root contains the set representative

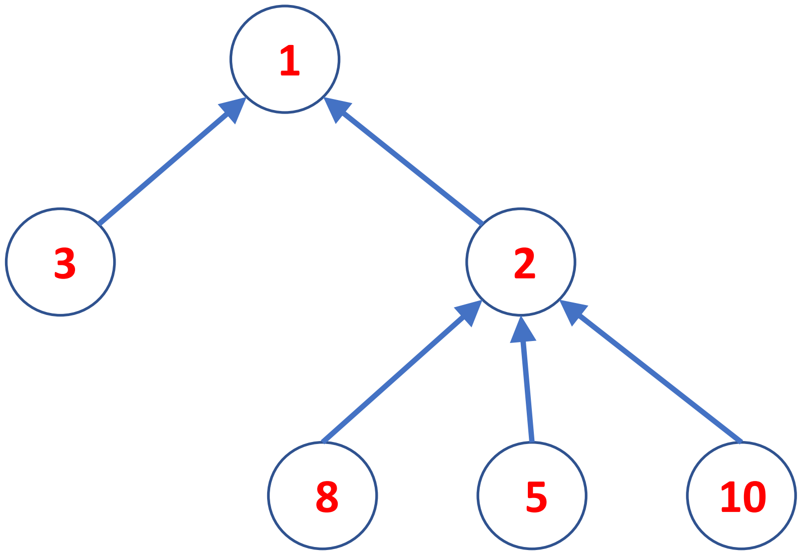
$S_1 = \{\underline{1}, 3, 2, 8, 5, 10\}$



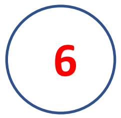
Example with $n = 10$

$S = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$

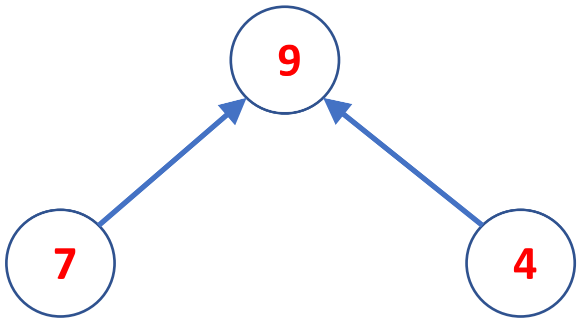
$S_1 = \{1, 3, 2, 8, 5, 10\}$



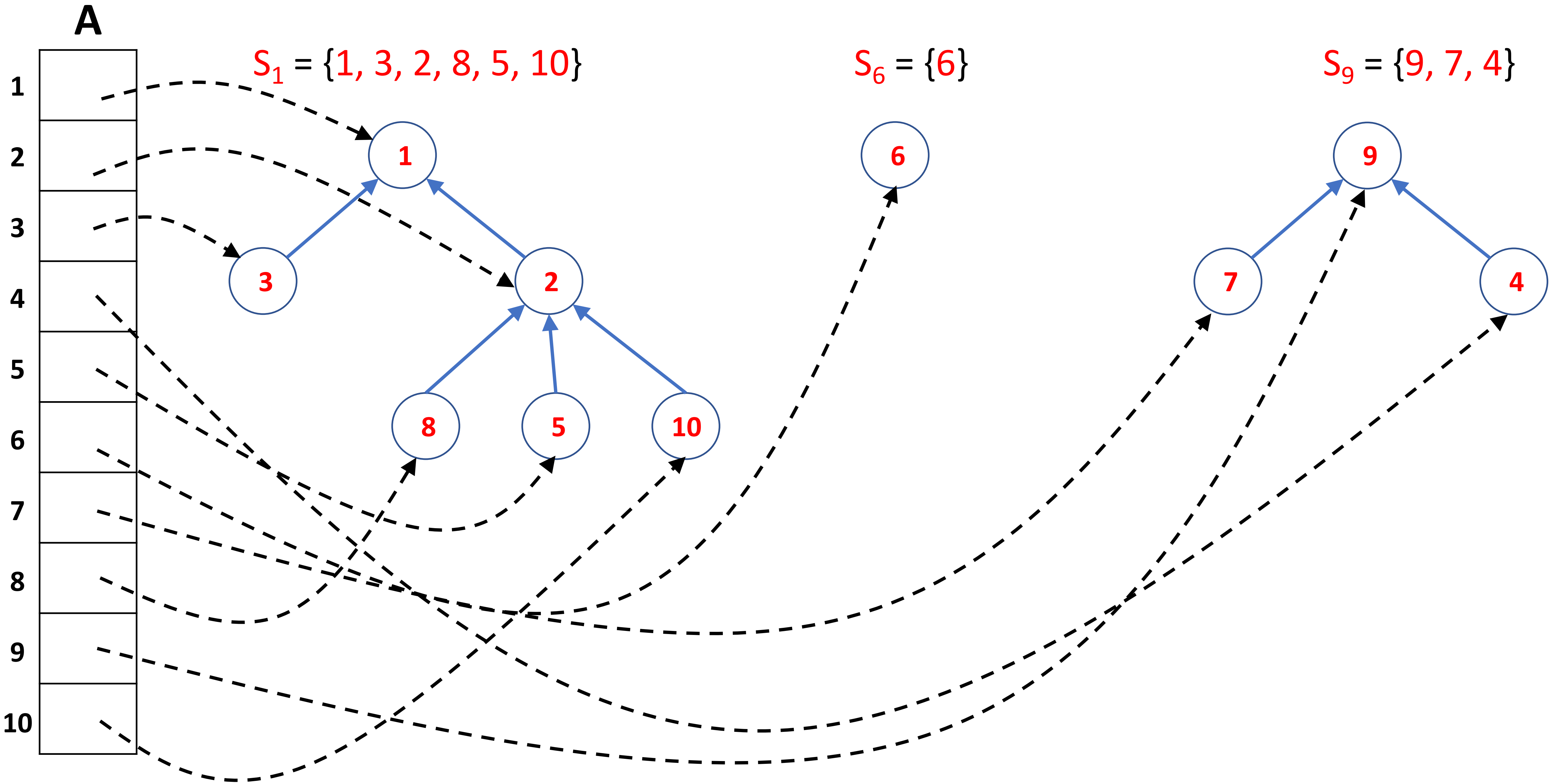
$S_6 = \{6\}$



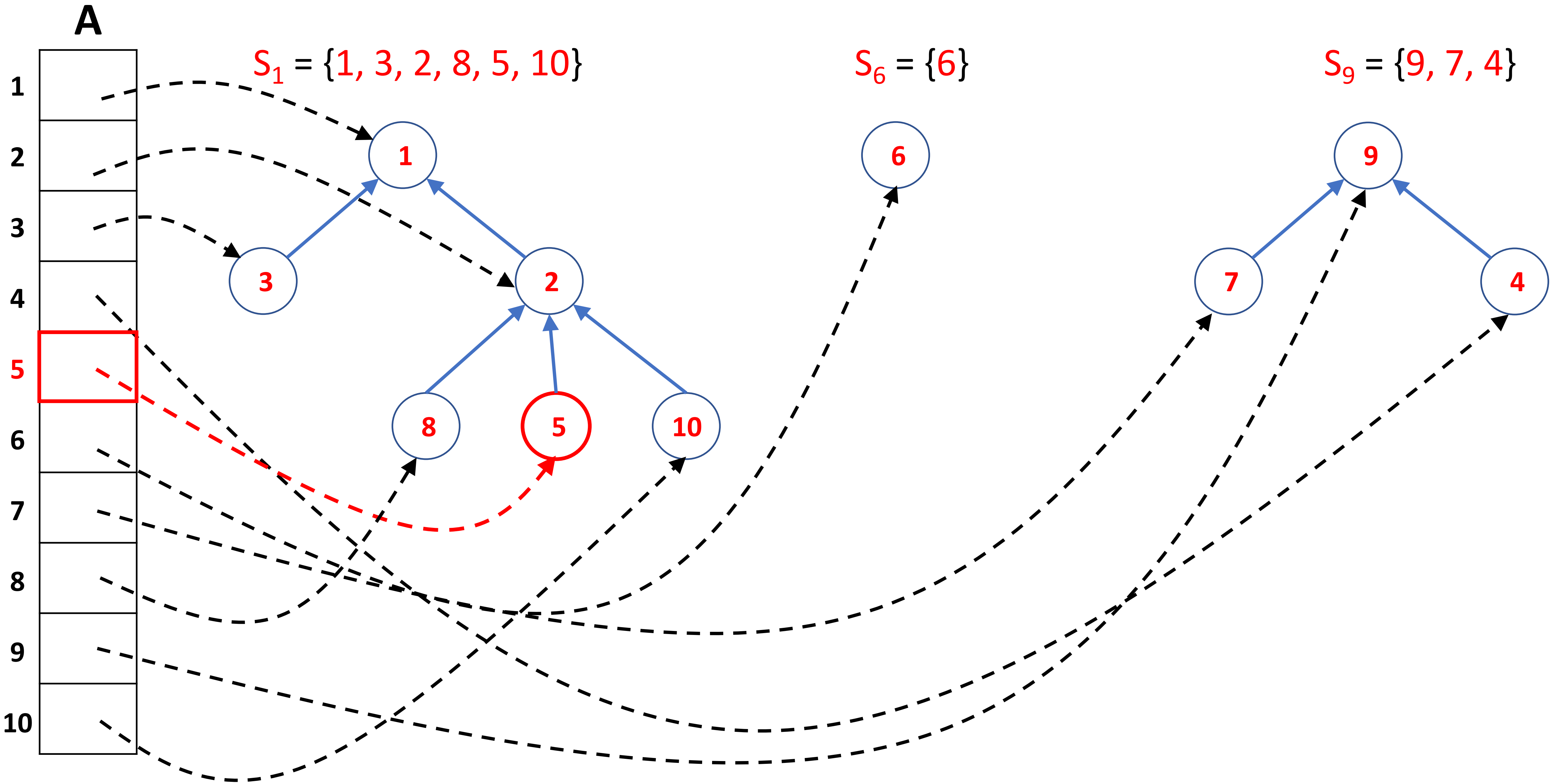
$S_9 = \{9, 7, 4\}$



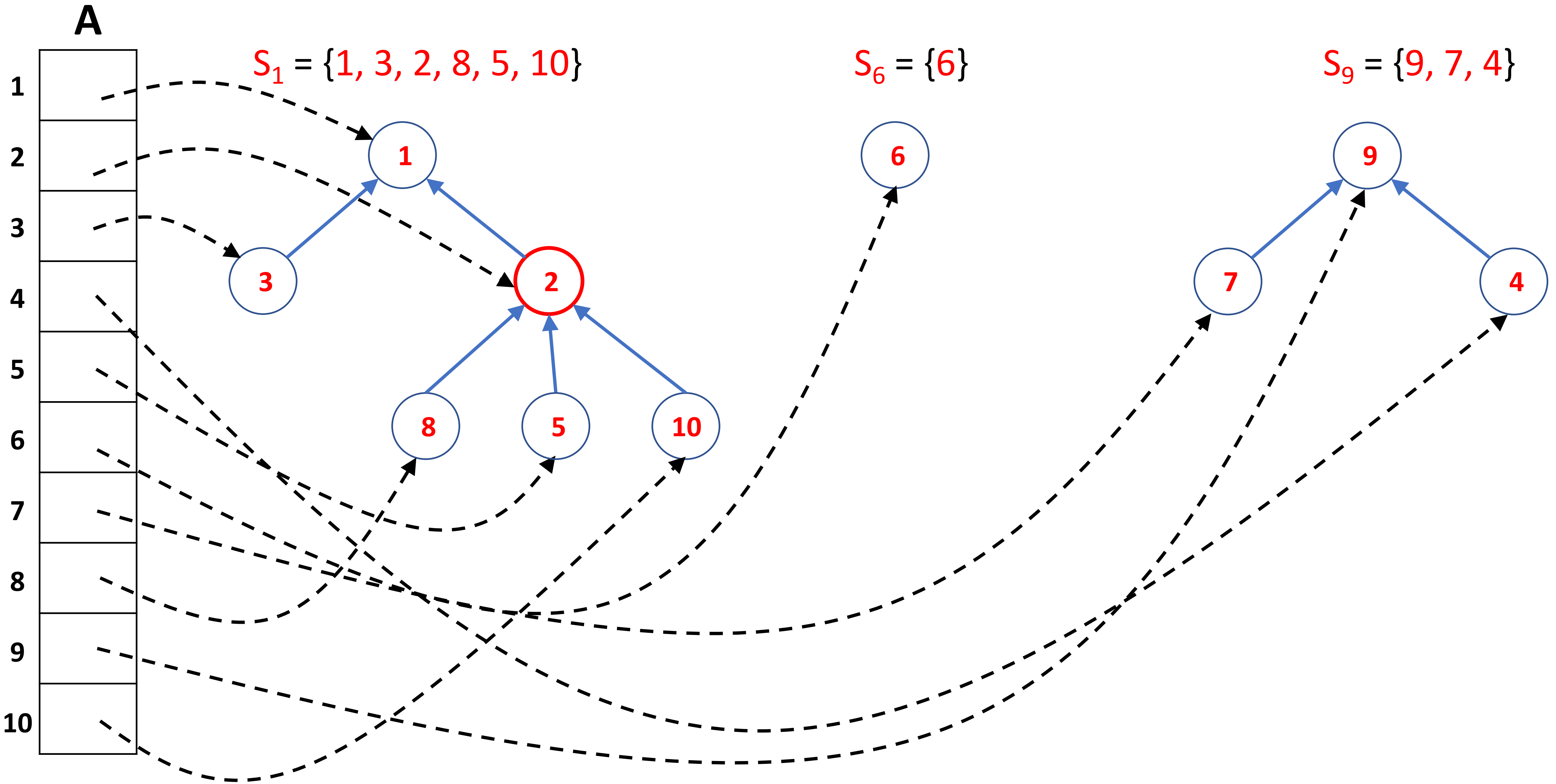
Example with $n = 10$



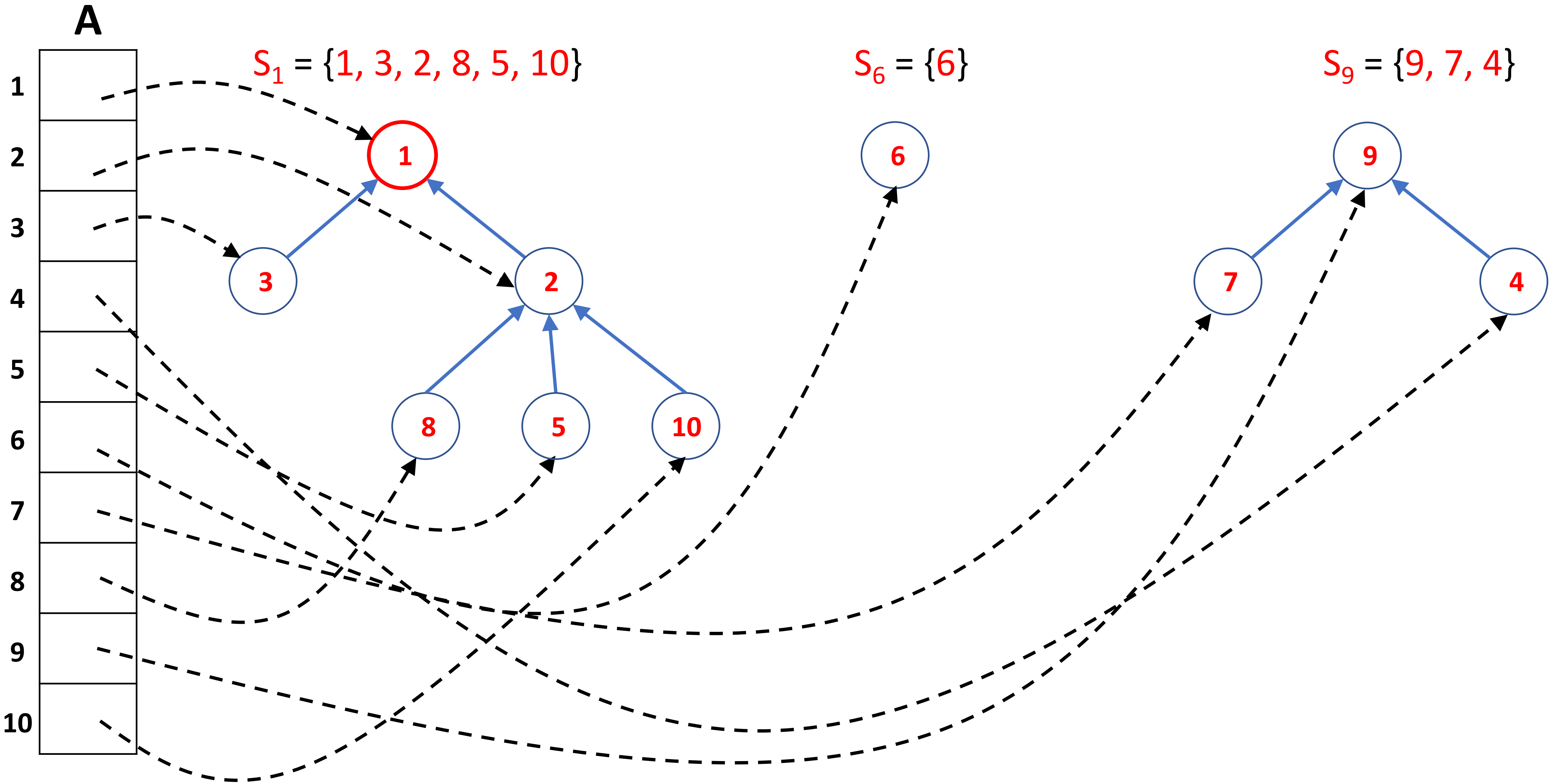
Find(5)



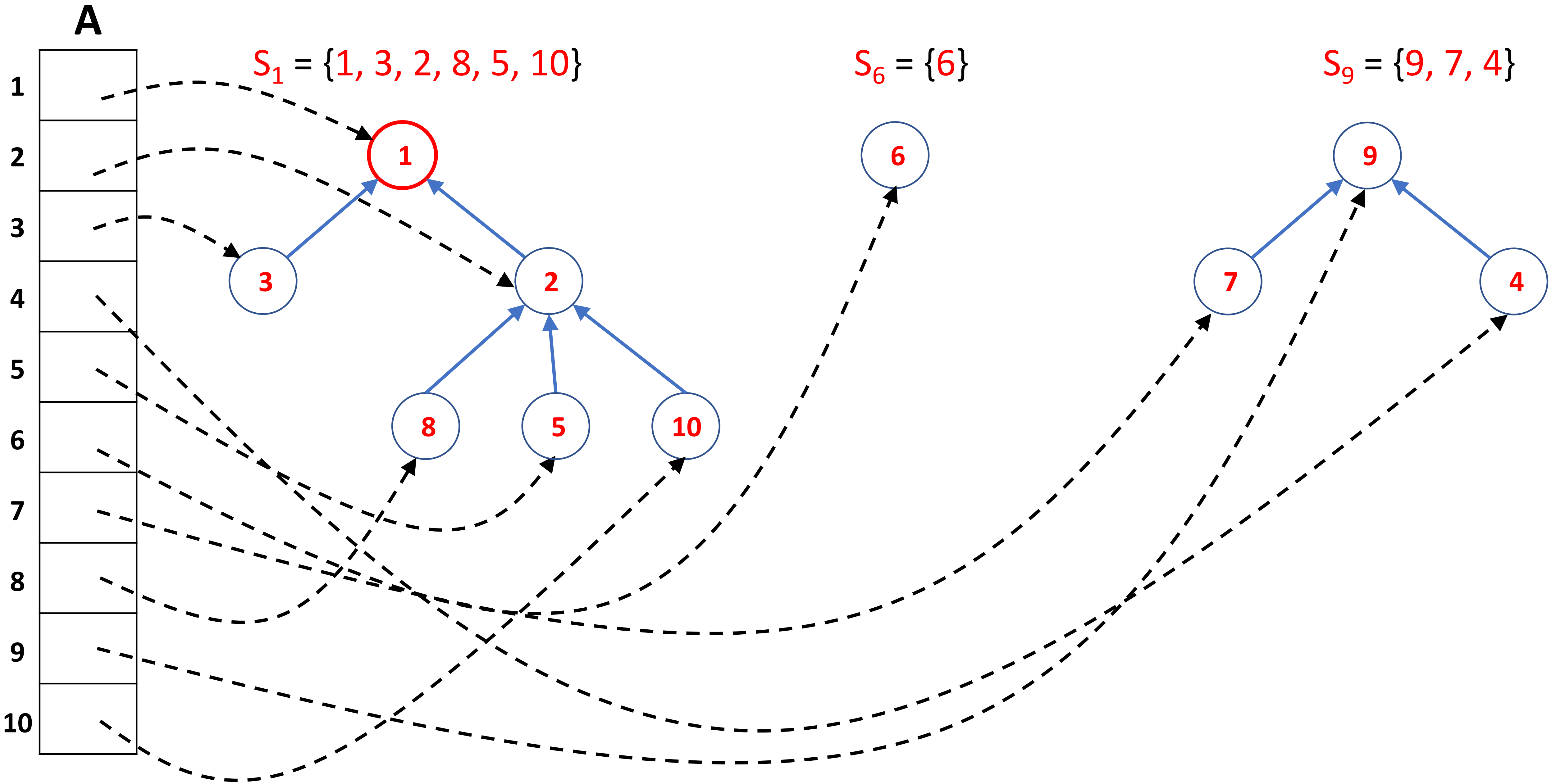
Find(5)



Find(5)



Find(5) : Return (ptr to) 1



Operations

- **Find**(x): Follow path from x up to root, return ptr to the root

    Cost is O(1 + length of the **Find** path)

- **Union**(S_x, S_y): Make root of S_x the child of root of S_y

    Cost is O(1)

Disjoint Forest: Time Complexity

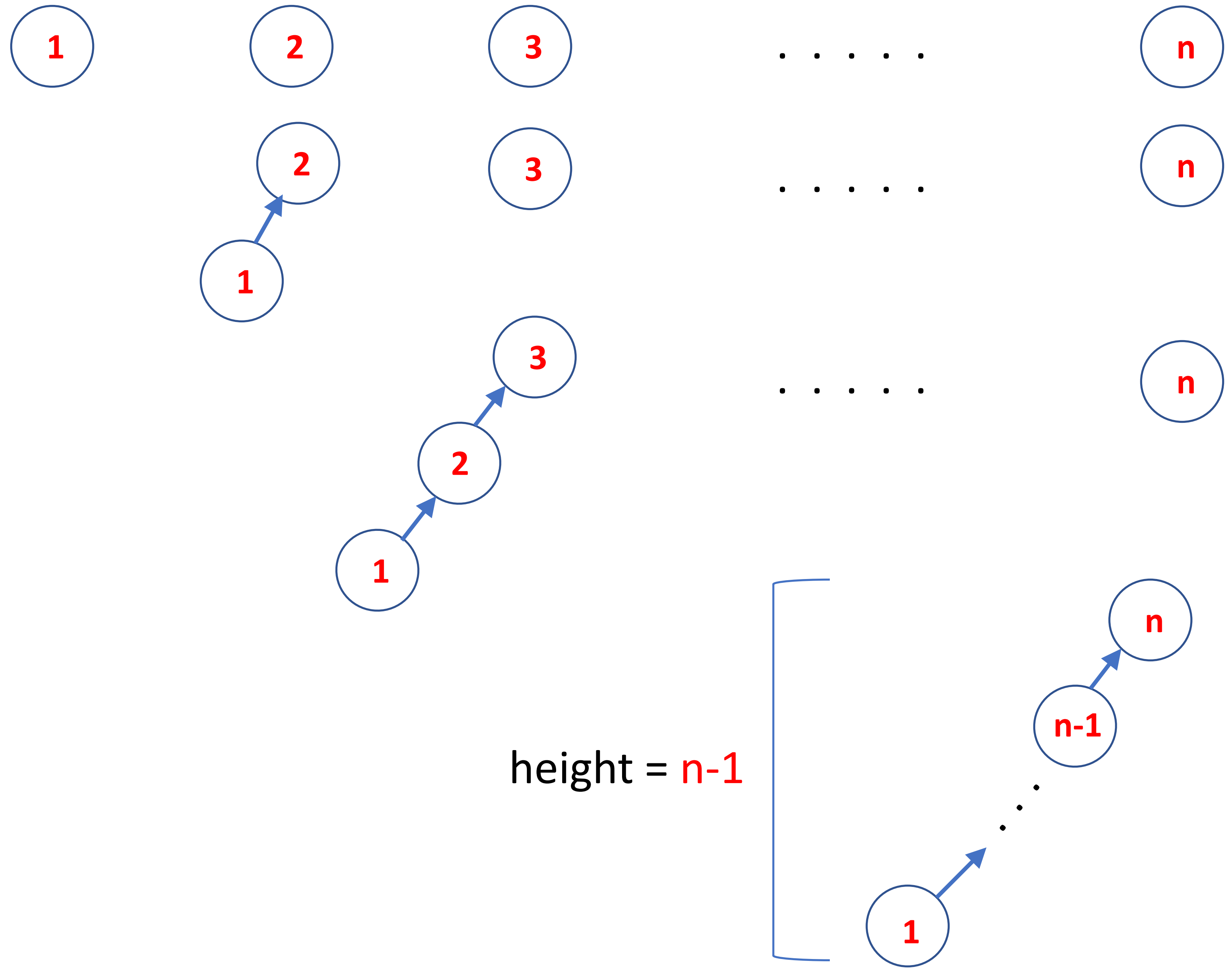
Initially :

Union(1, 2) :

Union(2, 3) :

.
.
.
.
.

Union(n-1, n) :

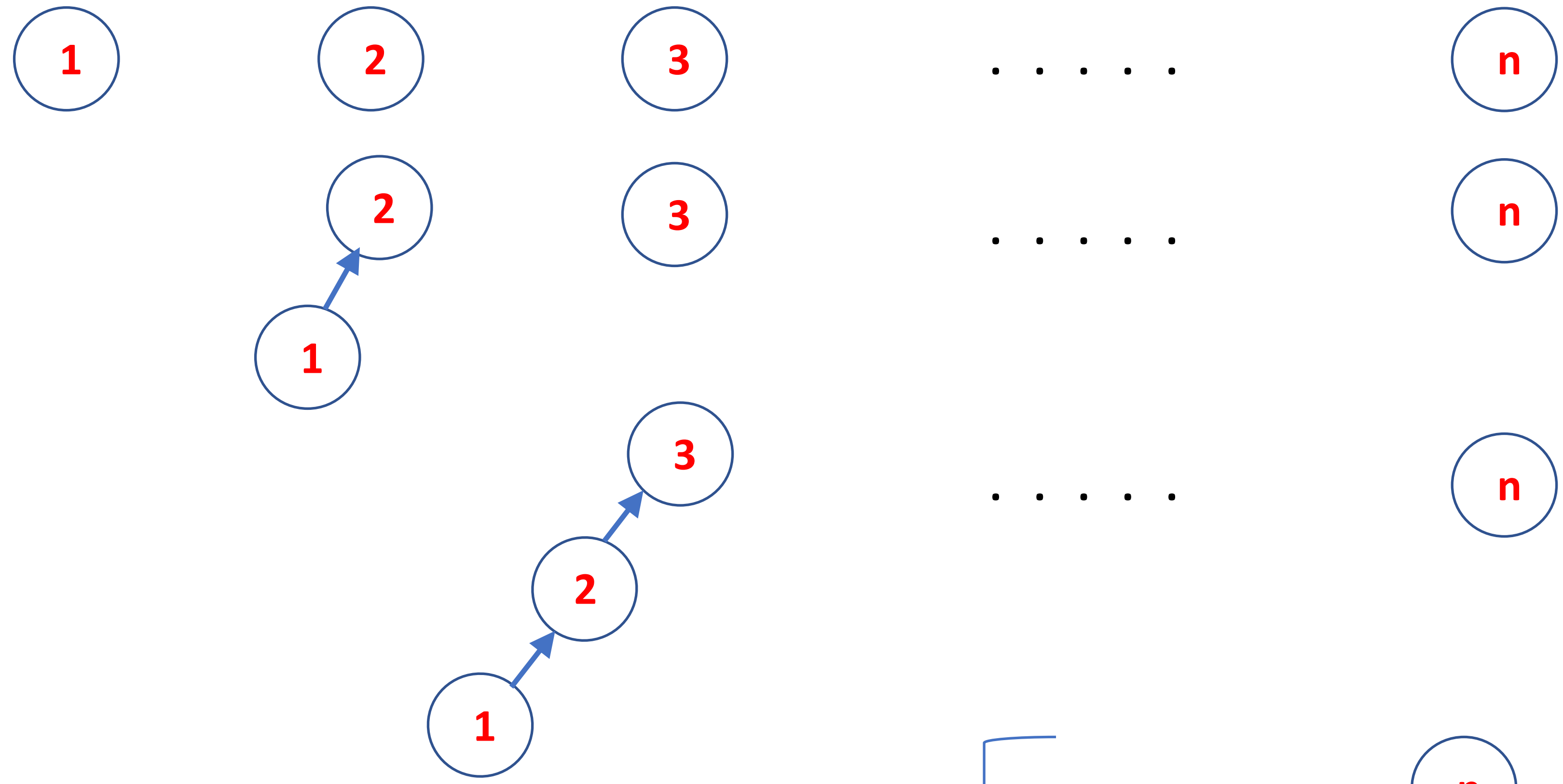


Disjoint Forest: Time Complexity

Initially :

Union(1, 2) :

Union(2, 3) :



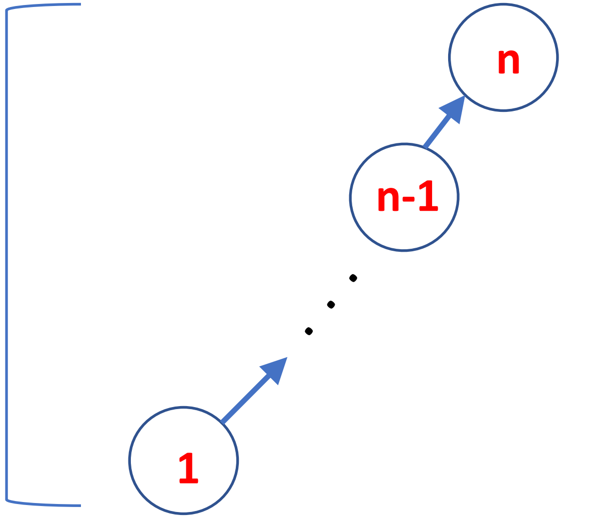
.
.
.
.
.

Union(n-1, n) :

height = n-1

Cost of **Find**(1) : Ω(n) !

⇒ worst−case cost of executing σ is Ω(m.n)



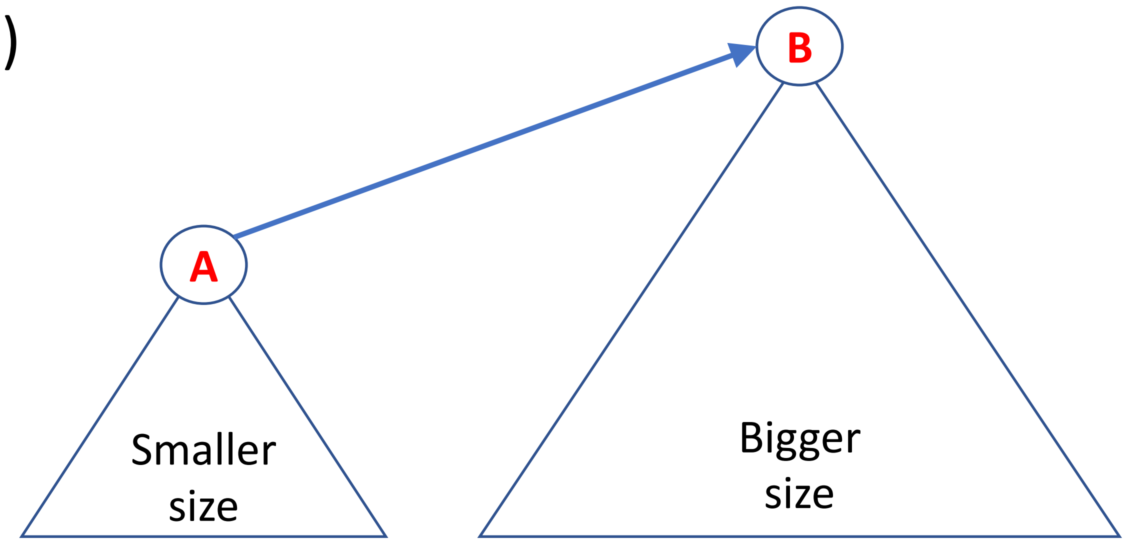
Disjoint Forest: Time Complexity

To reduce cost of executing σ, reduce the length of **Find** paths

$\boxed{\Rightarrow \text{ reduce height of the trees formed during the execution of } \sigma}$

Heuristic 1: **Weighted Union (WU)** by Size

Union(A, B)

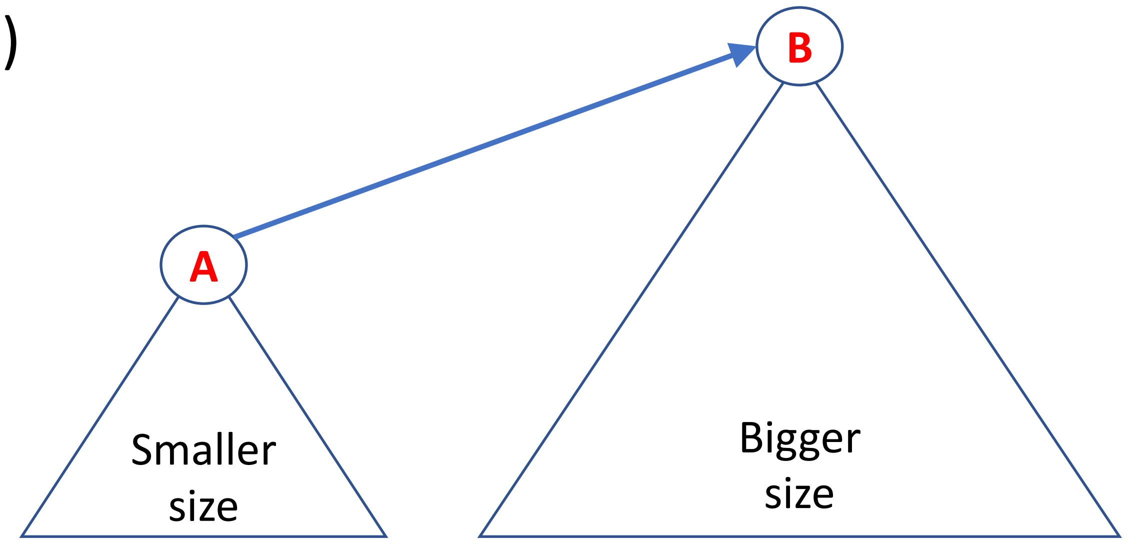


size: # of nodes in the tree

WU rule (by size): Smaller size tree becomes the child of the bigger size tree

Heuristic 1: **Weighted Union (WU)** by Size

Union(A, B)



size: # of nodes in the tree

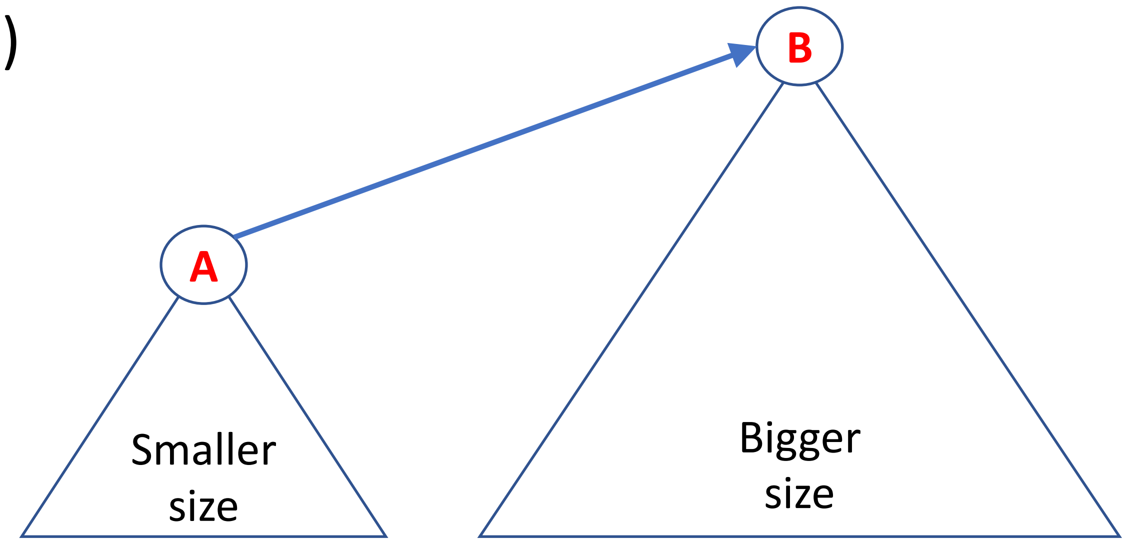
WU rule (by size): Smaller size tree becomes the child of the bigger size tree

With **WU**:

- Any tree T created during the execution of σ has height at most $\log_2 n$

Heuristic 1: **Weighted Union** (**WU**) by Size

Union(A, B)



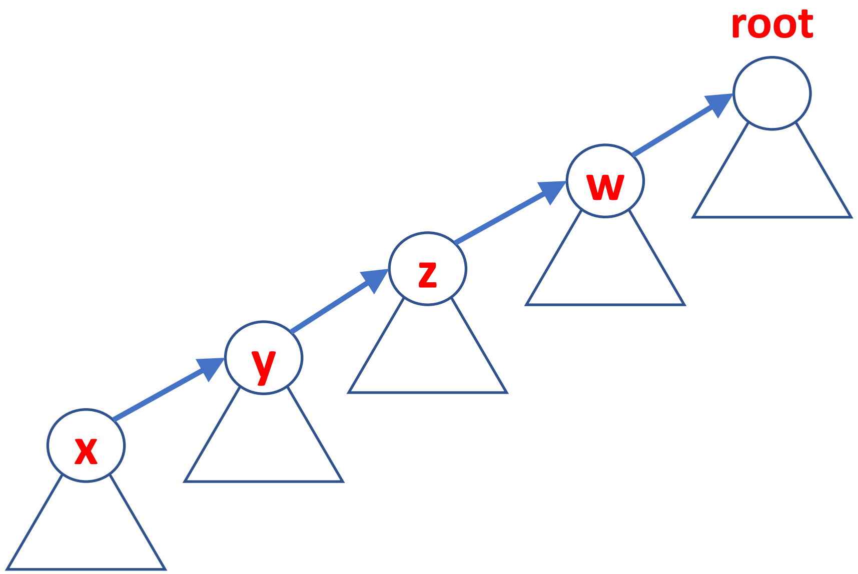
size: # of nodes in the tree

WU rule (by size): Smaller size tree becomes the child of the bigger size tree

With **WU**:

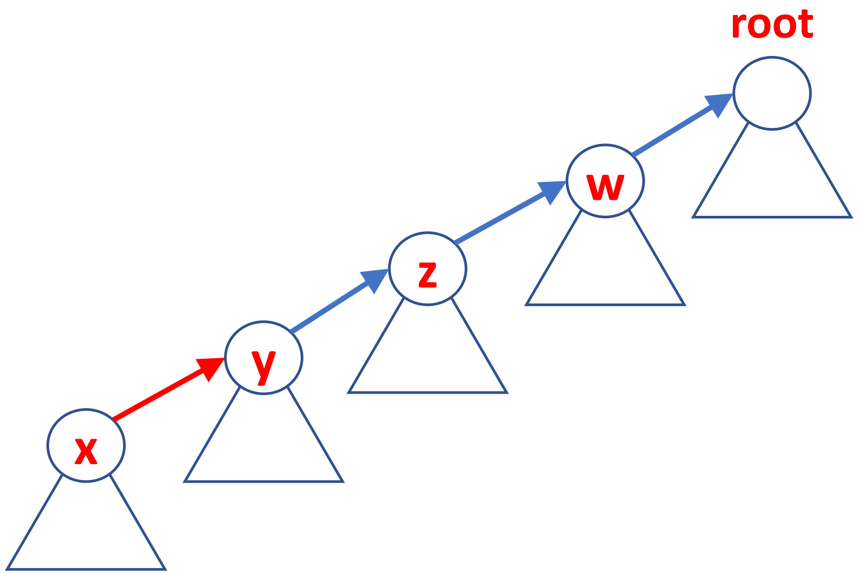
- Any tree T created during the execution of σ has height at most $\log_2 n$
- The worst-case cost of executing σ is O(m log n)

Heuristic 2: **P**ath **C**ompression (**PC**)



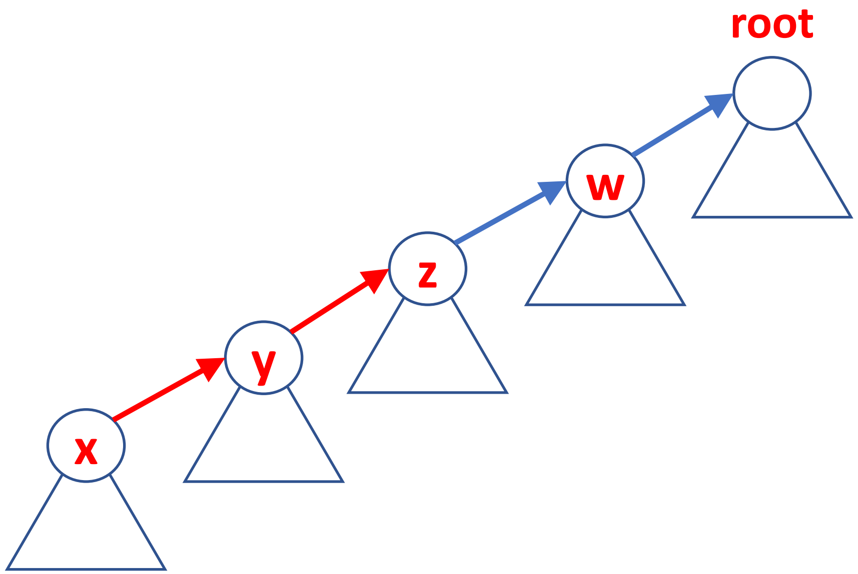
Find(x)

Heuristic 2: Path Compression (PC)



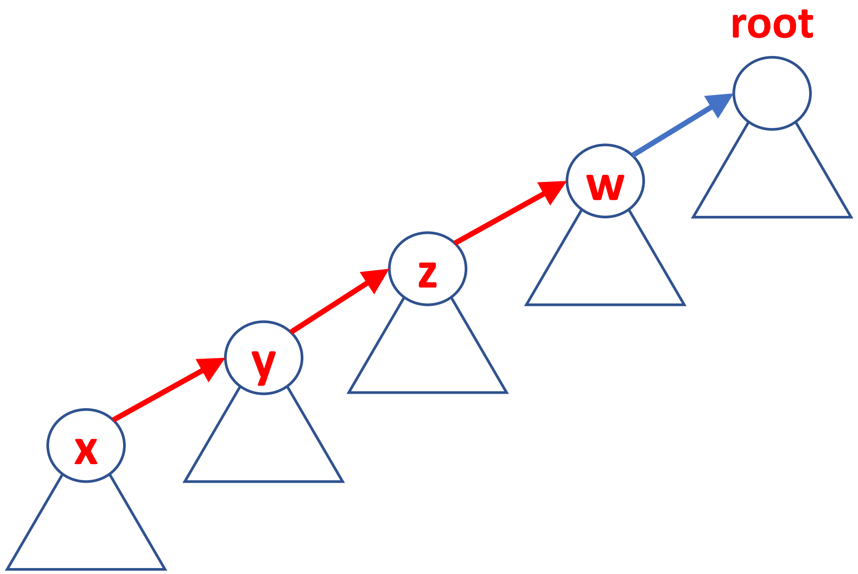
Find(x)

Heuristic 2: **P**ath **C**ompression (**PC**)



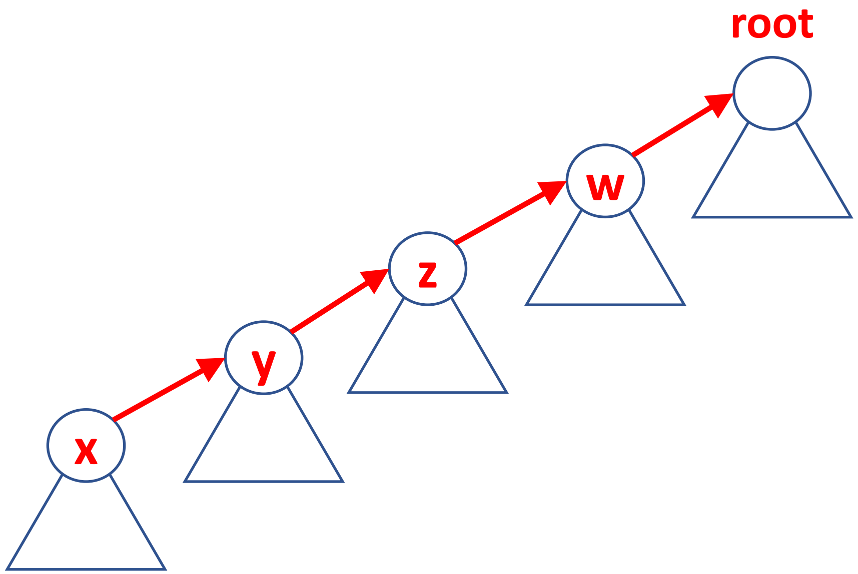
Find(x)

Heuristic 2: Path Compression (PC)



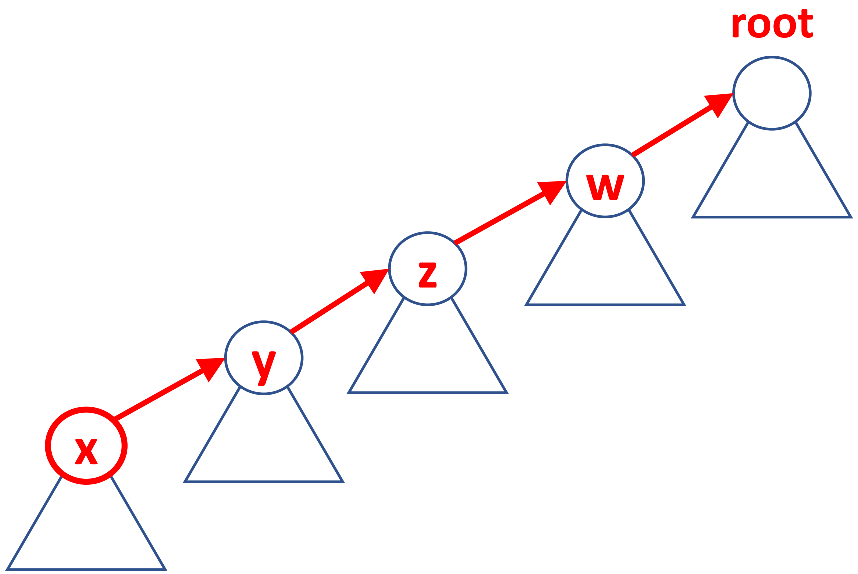
Find(x)

Heuristic 2: **P**ath **C**ompression (**PC**)



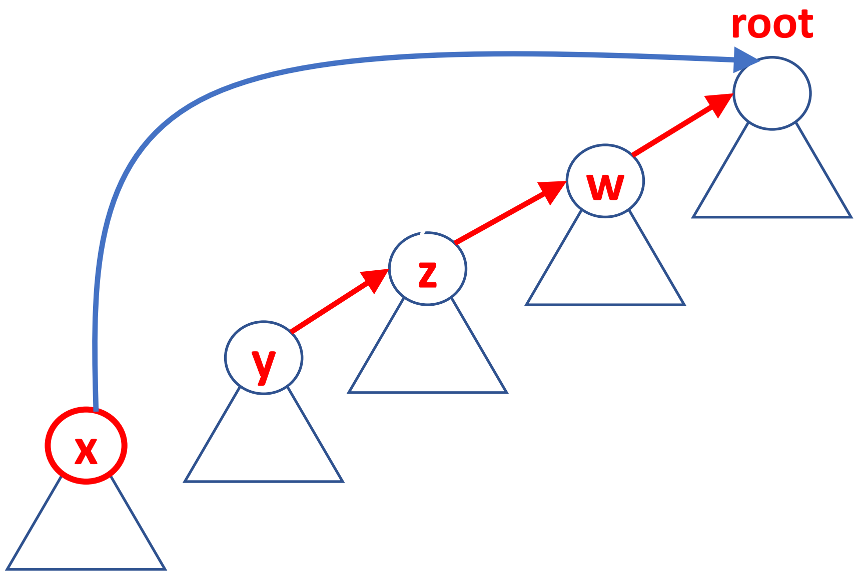
Find(x)

Heuristic 2: **P**ath **C**ompression (**PC**)



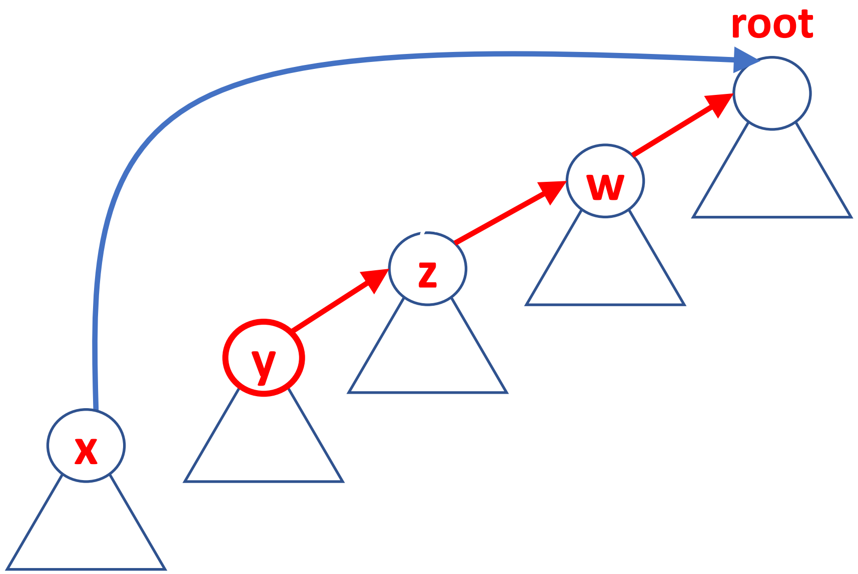
Find(x)

Heuristic 2: Path Compression (PC)



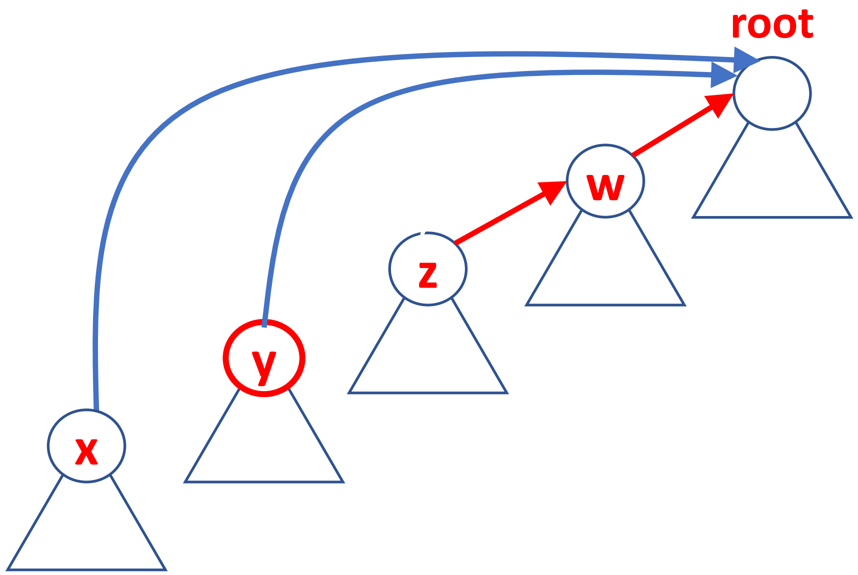
Find(x)

Heuristic 2: Path Compression (PC)



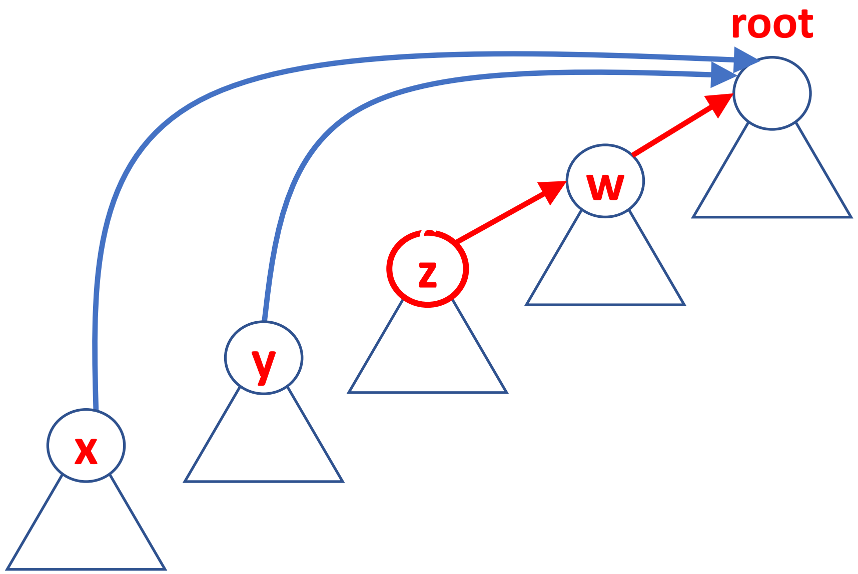
Find(x)

Heuristic 2: **P**ath **C**ompression (**PC**)



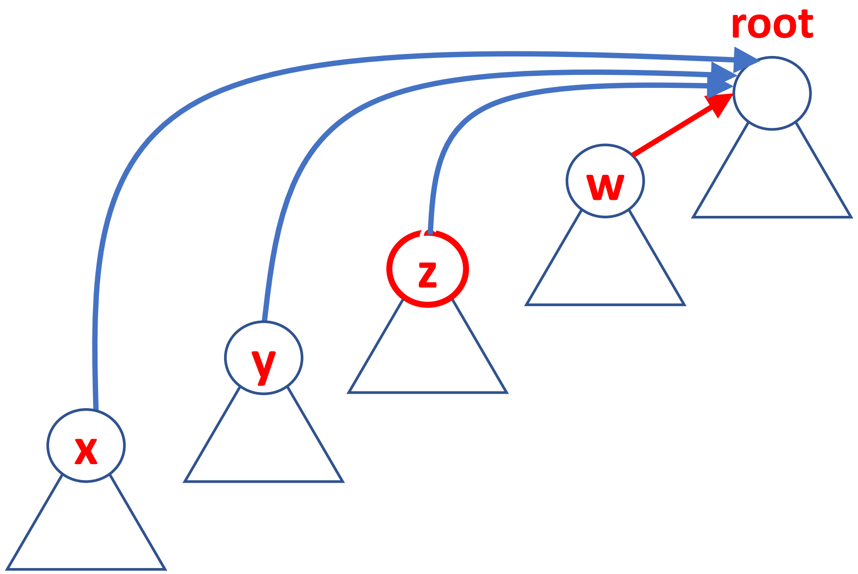
Find(x)

Heuristic 2: **P**ath **C**ompression (**PC**)



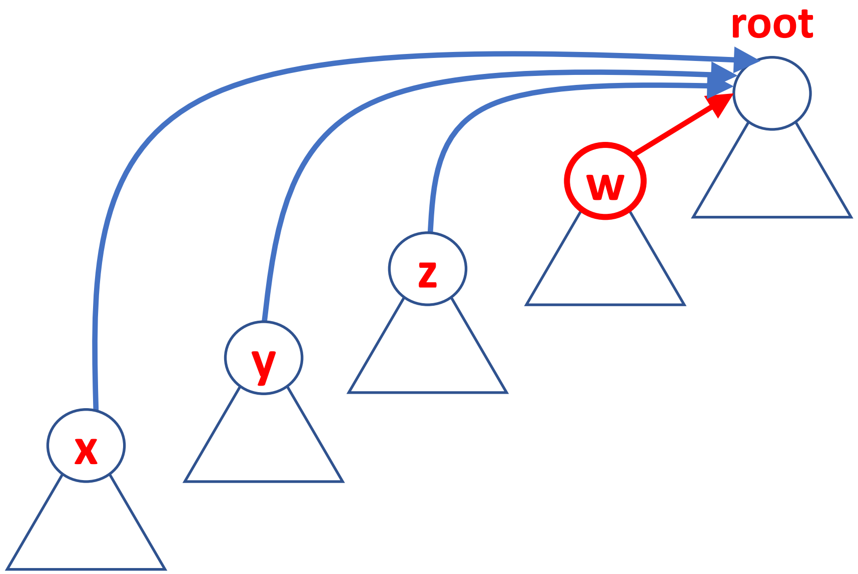
Find(x)

Heuristic 2: **P**ath **C**ompression (**PC**)



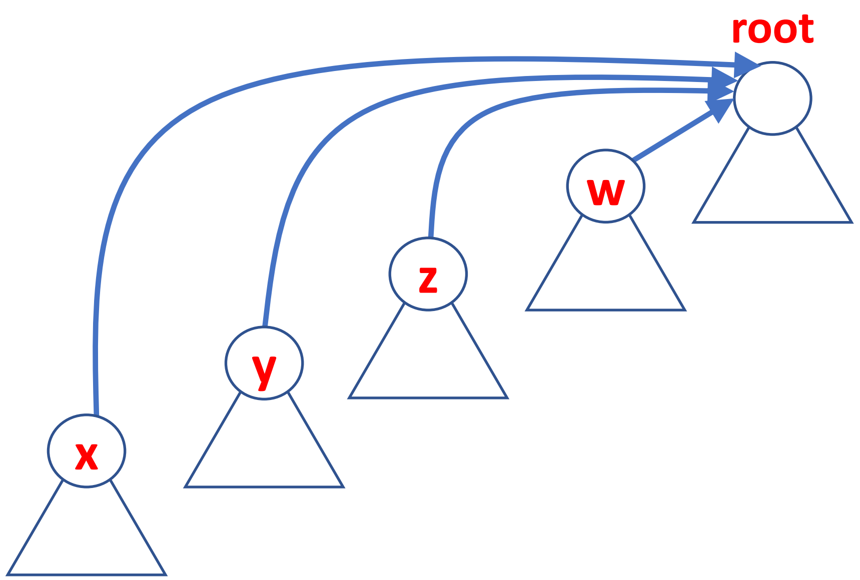
Find(x)

Heuristic 2: **P**ath **C**ompression (**PC**)



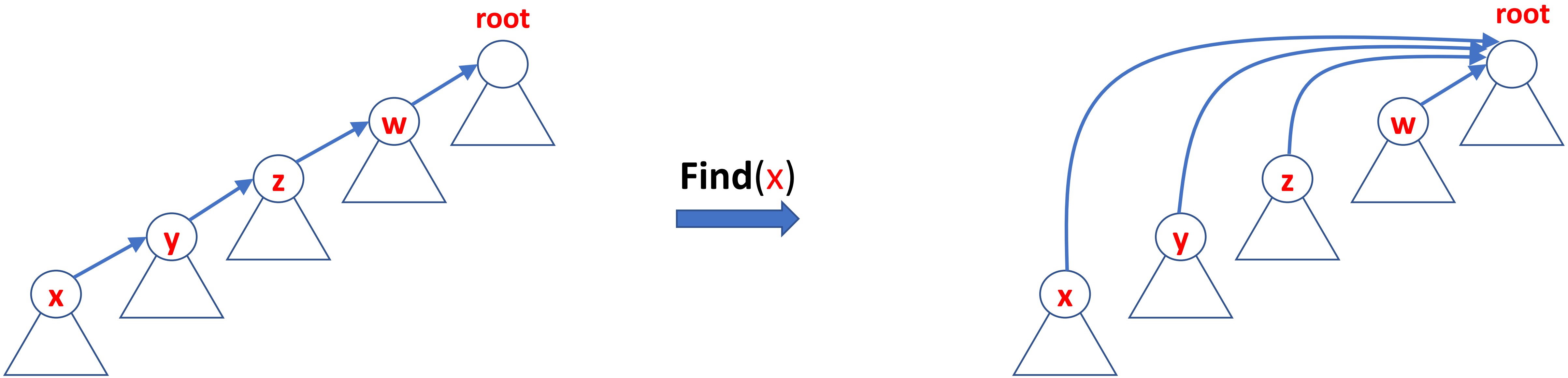
Find(x)

Heuristic 2: **P**ath **C**ompression (**PC**)



Find(x)

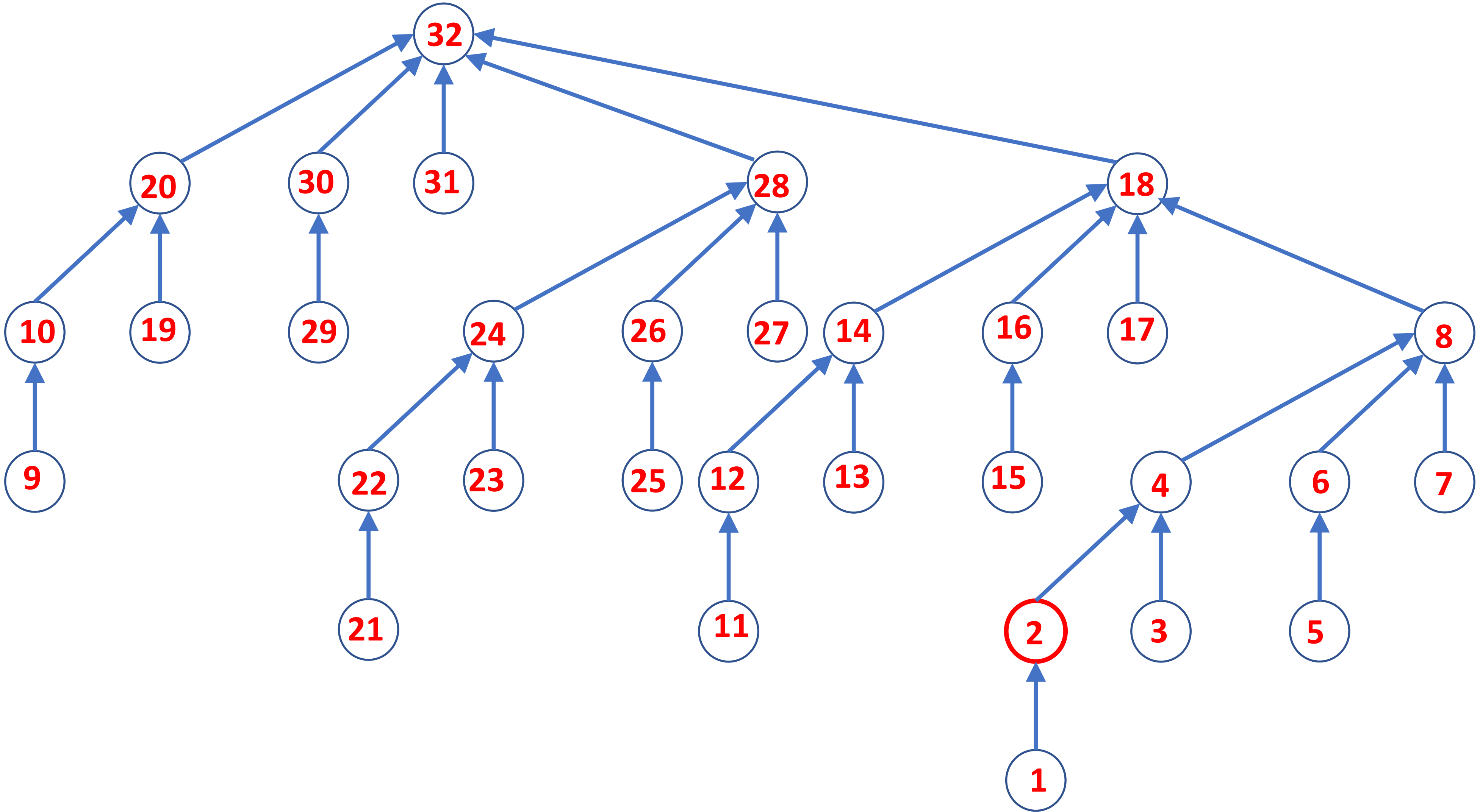
Heuristic 2: **Path Compression (PC)**



PC rule: In **Find**(x), make each vertex along the **Find** path a child of root

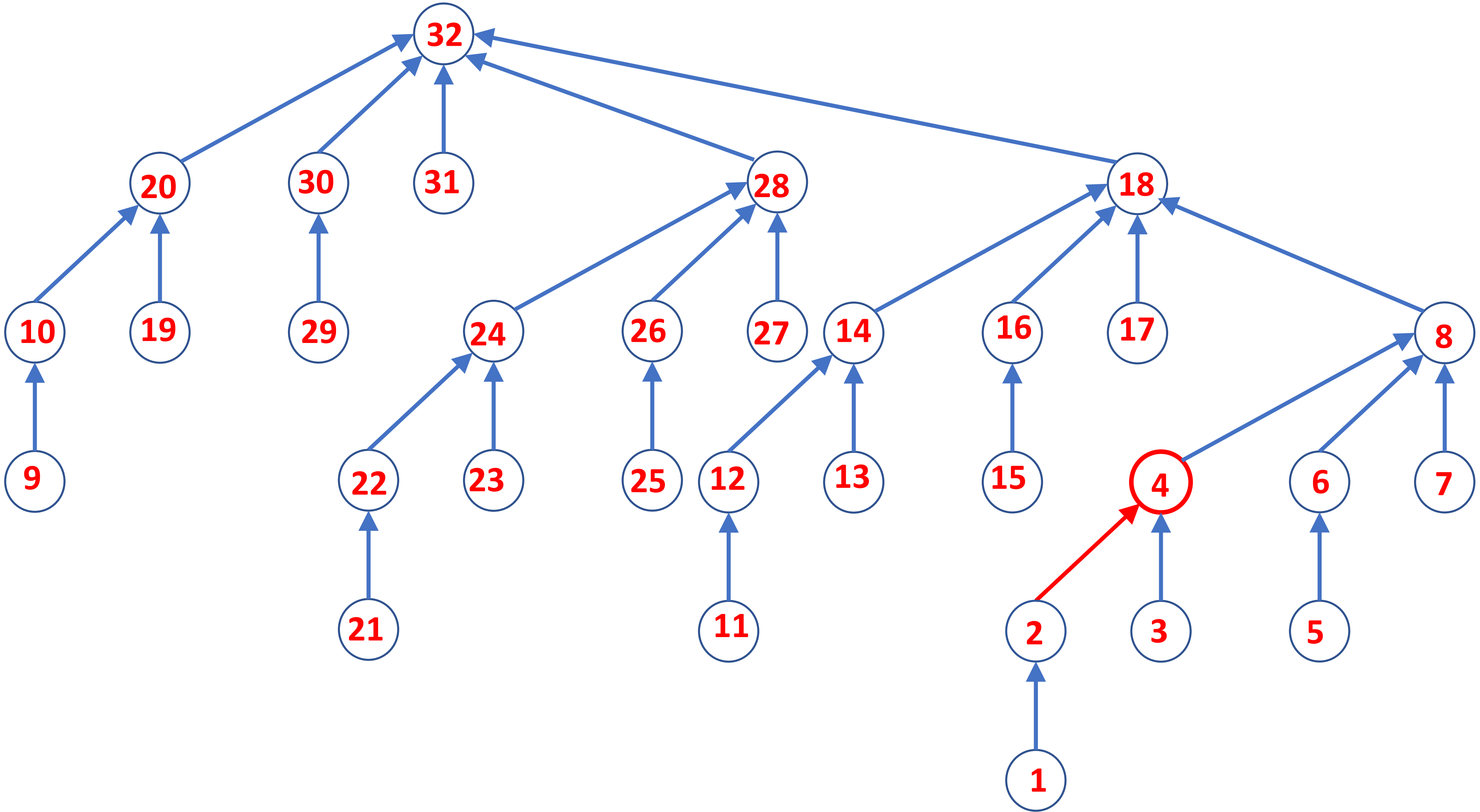
Example of Path Compression (PC)

Find(2)



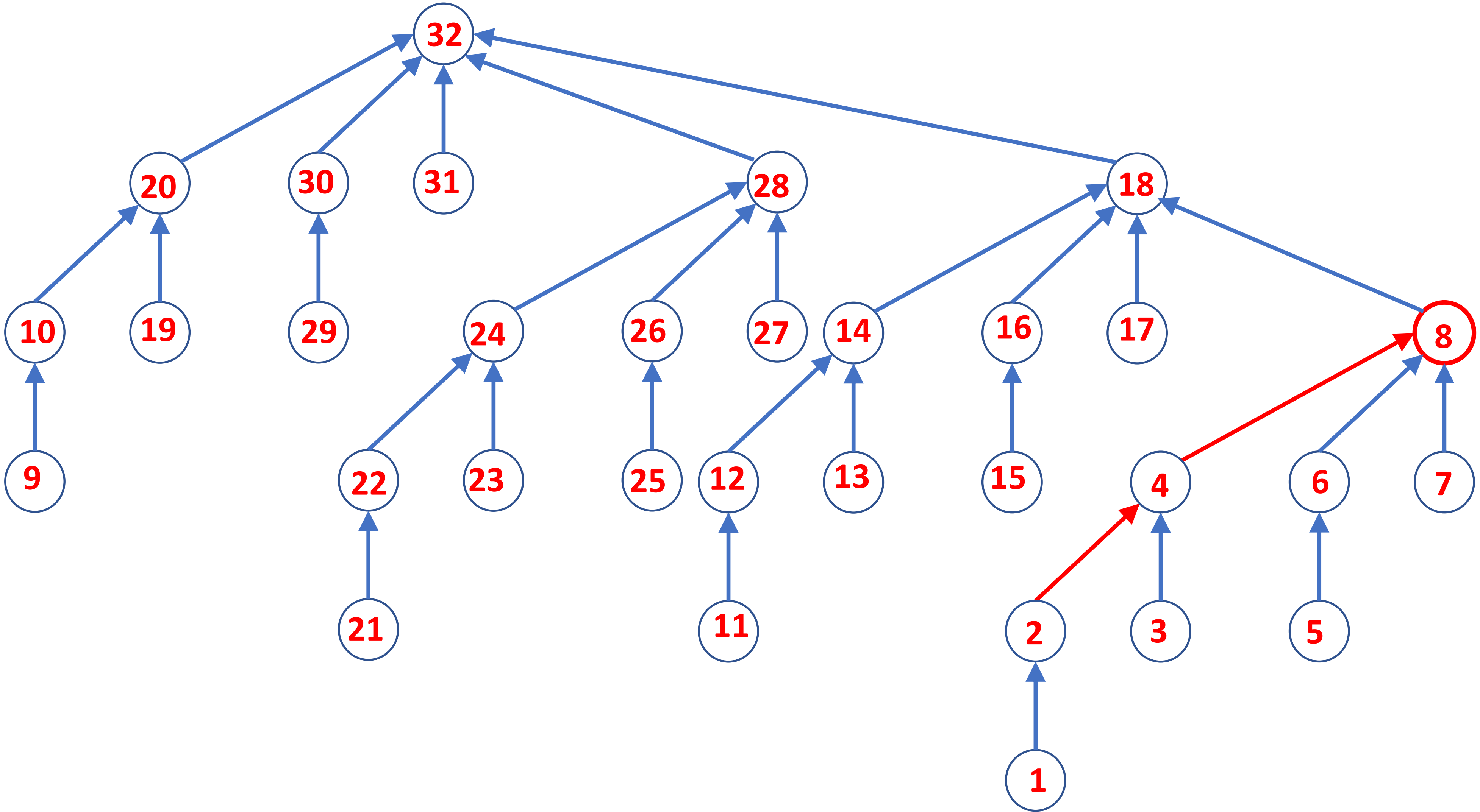
Example of Path Compression (PC)

Find(2)



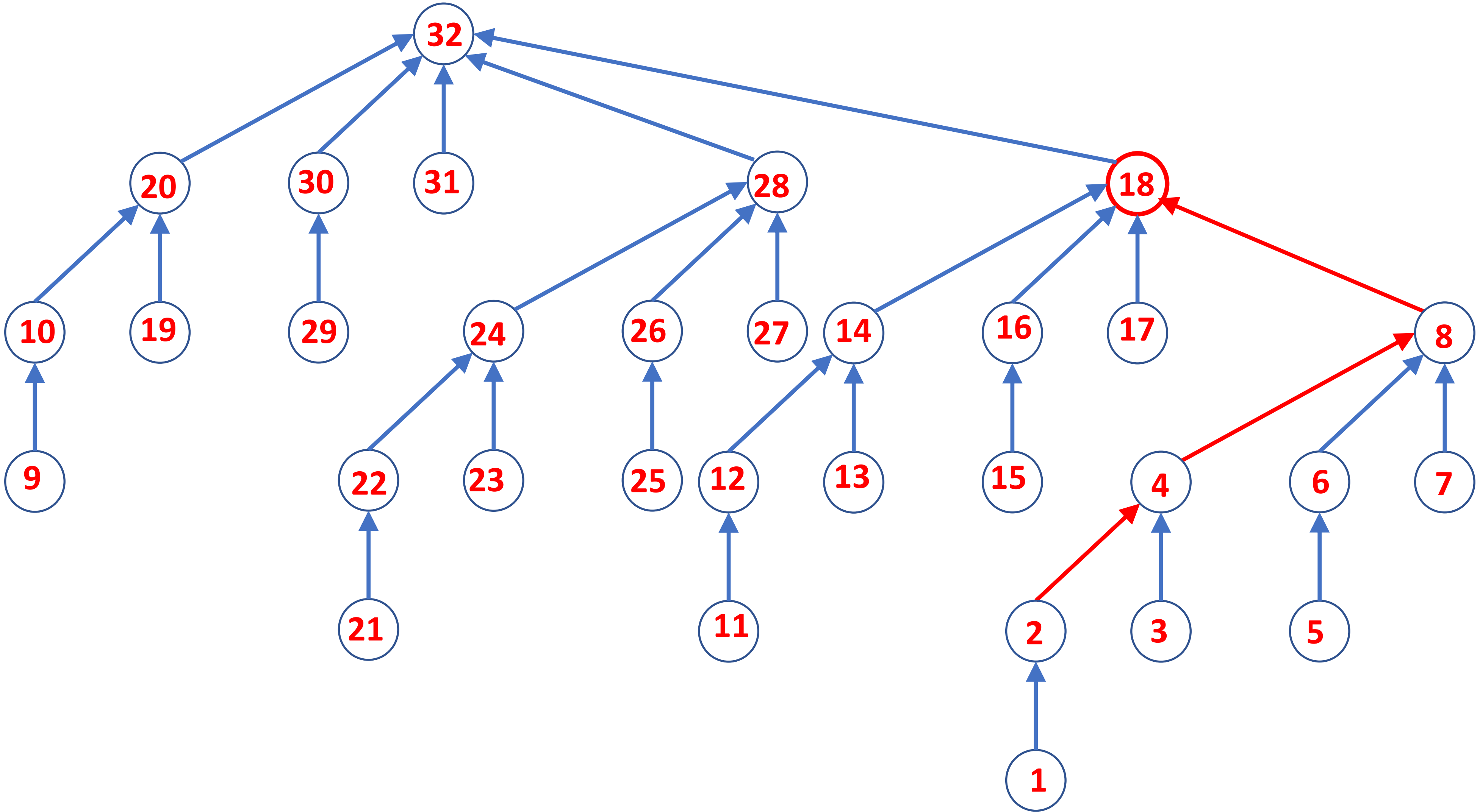
Example of Path Compression (PC)

Find(2)



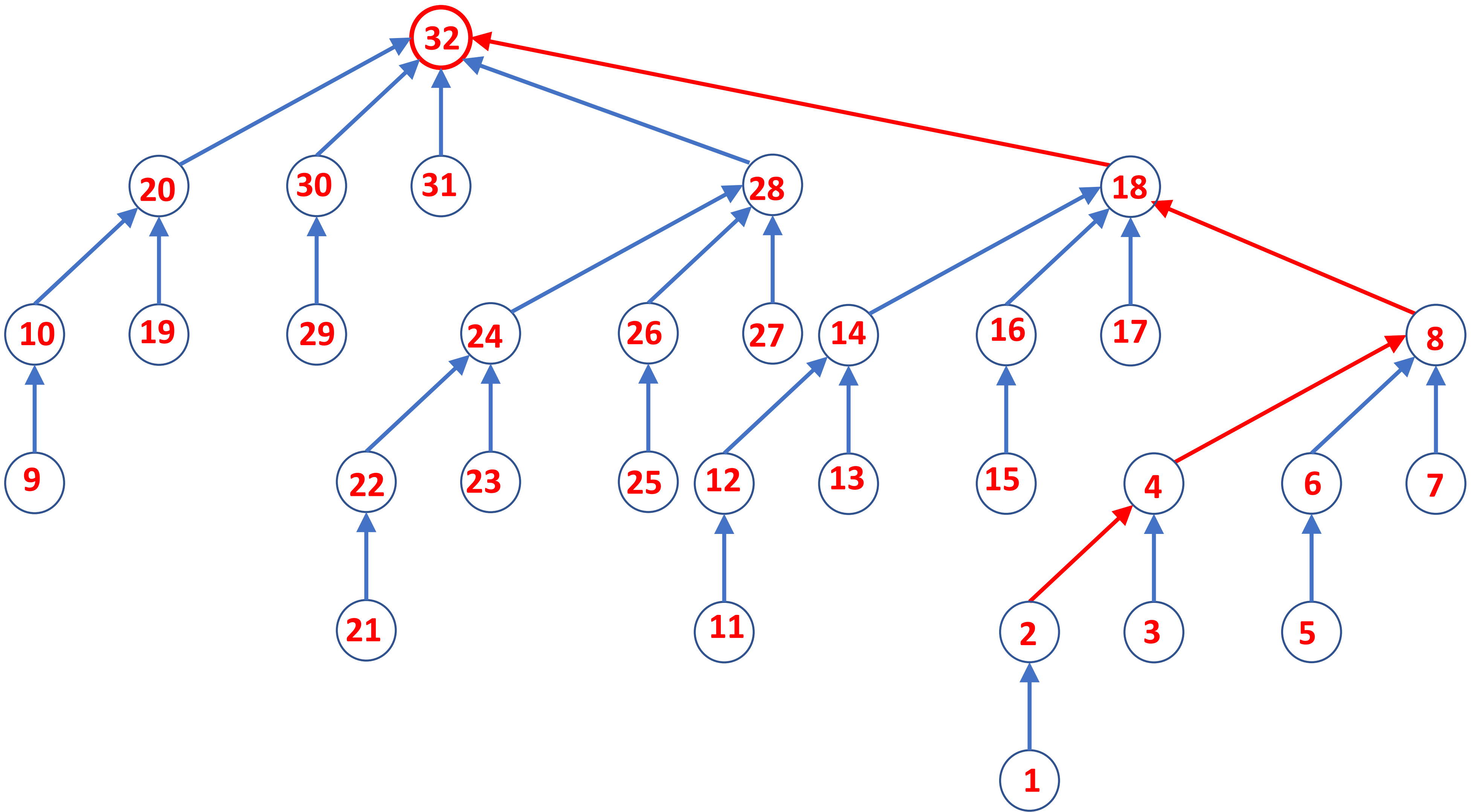
Example of Path Compression (PC)

Find(2)



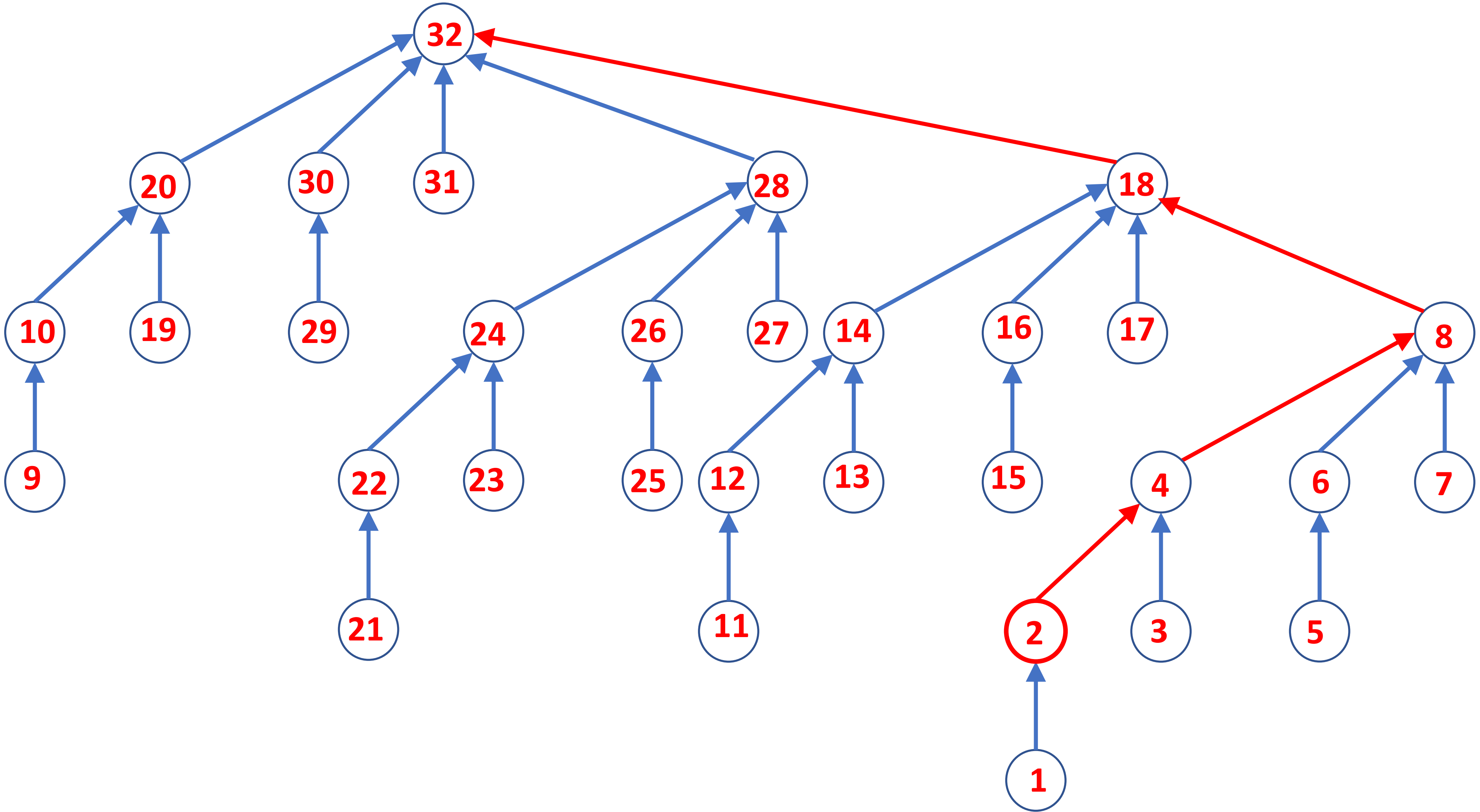
Example of Path Compression (PC)

Find(2)



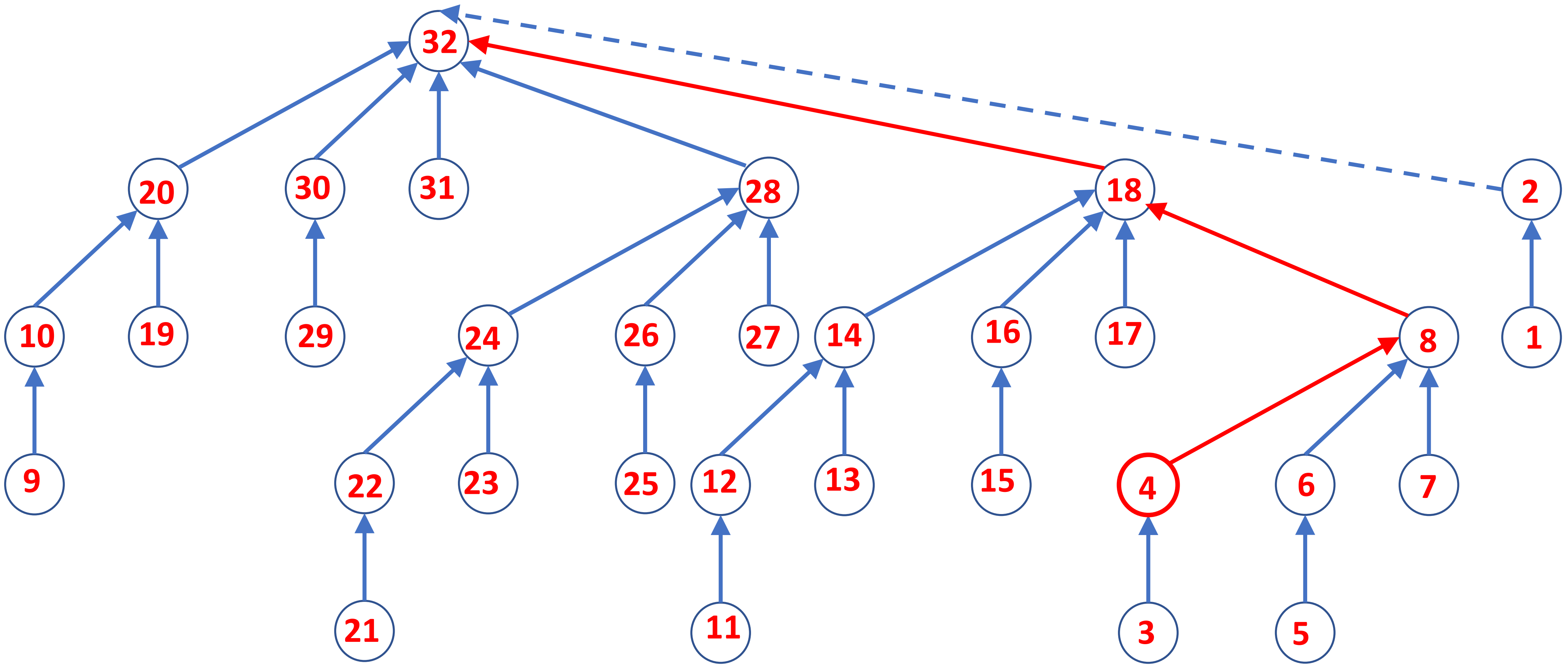
Example of Path Compression (PC)

Find(2)



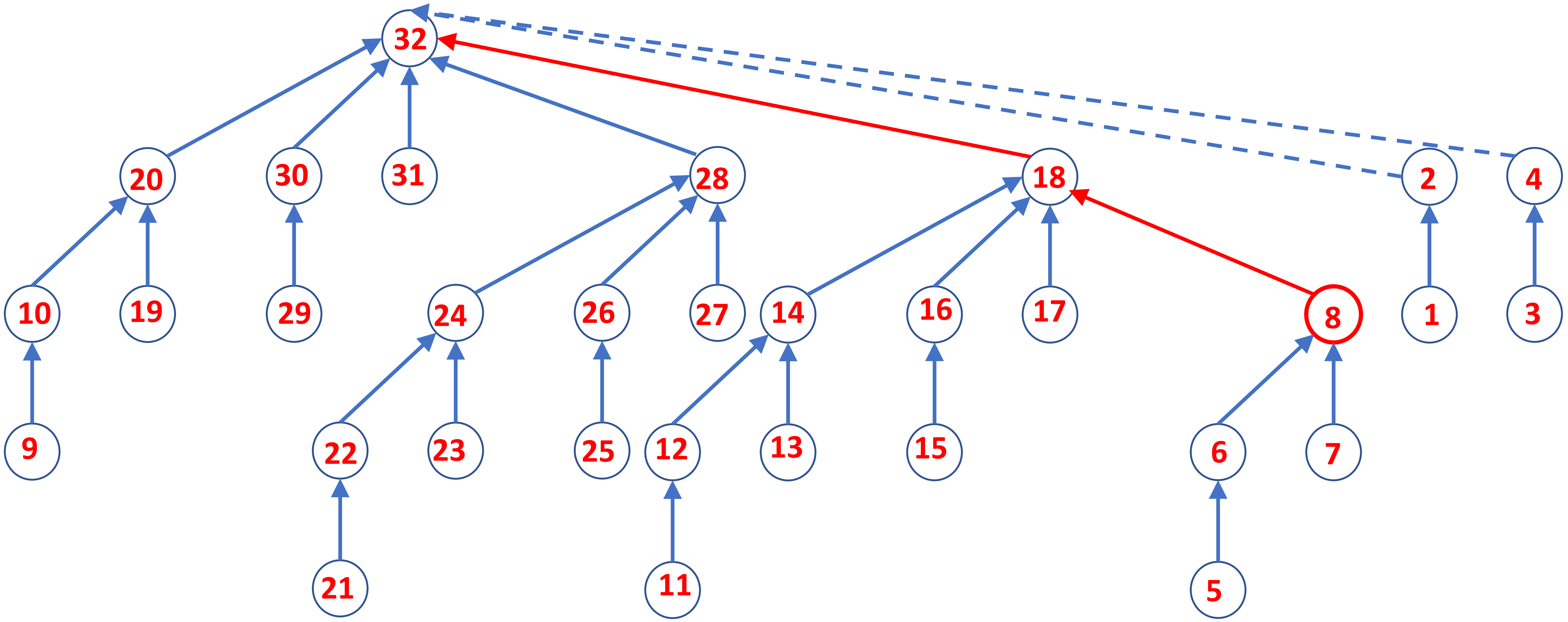
Example of Path Compression (PC)

Find(2)



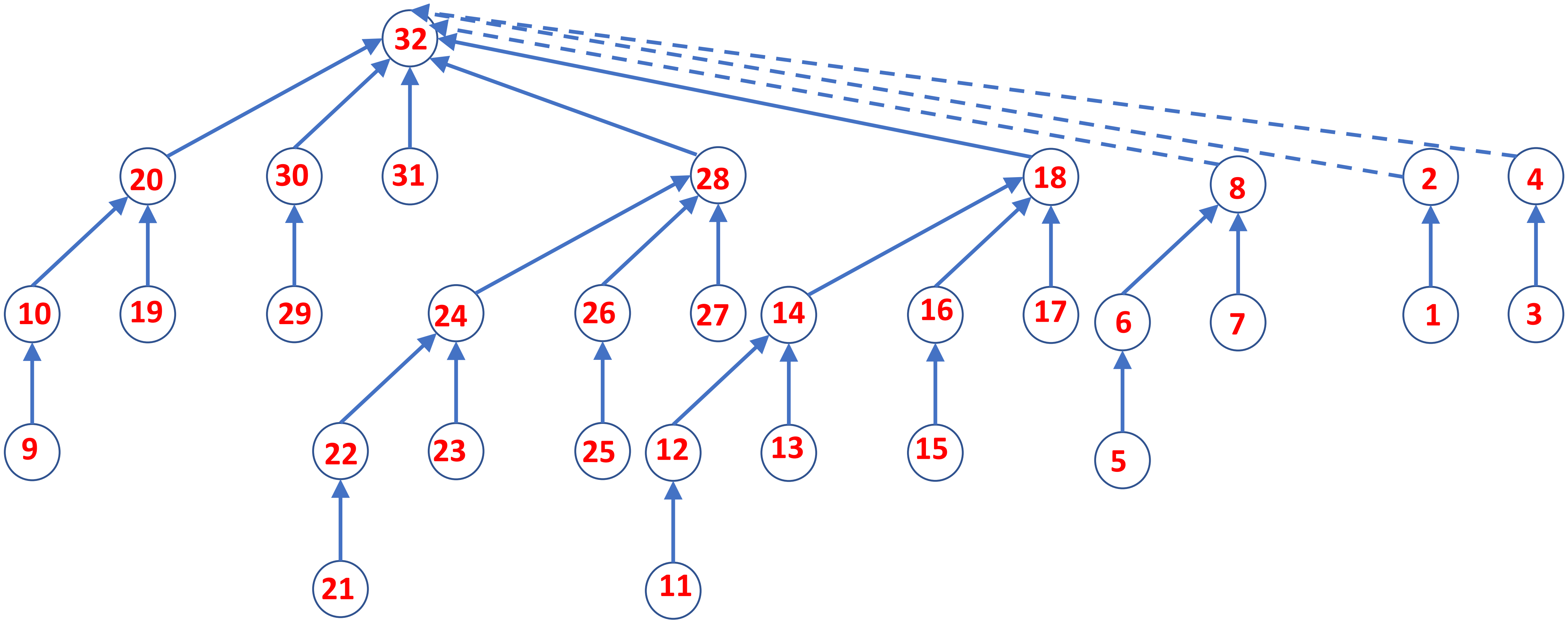
Example of Path Compression (PC)

Find(2)



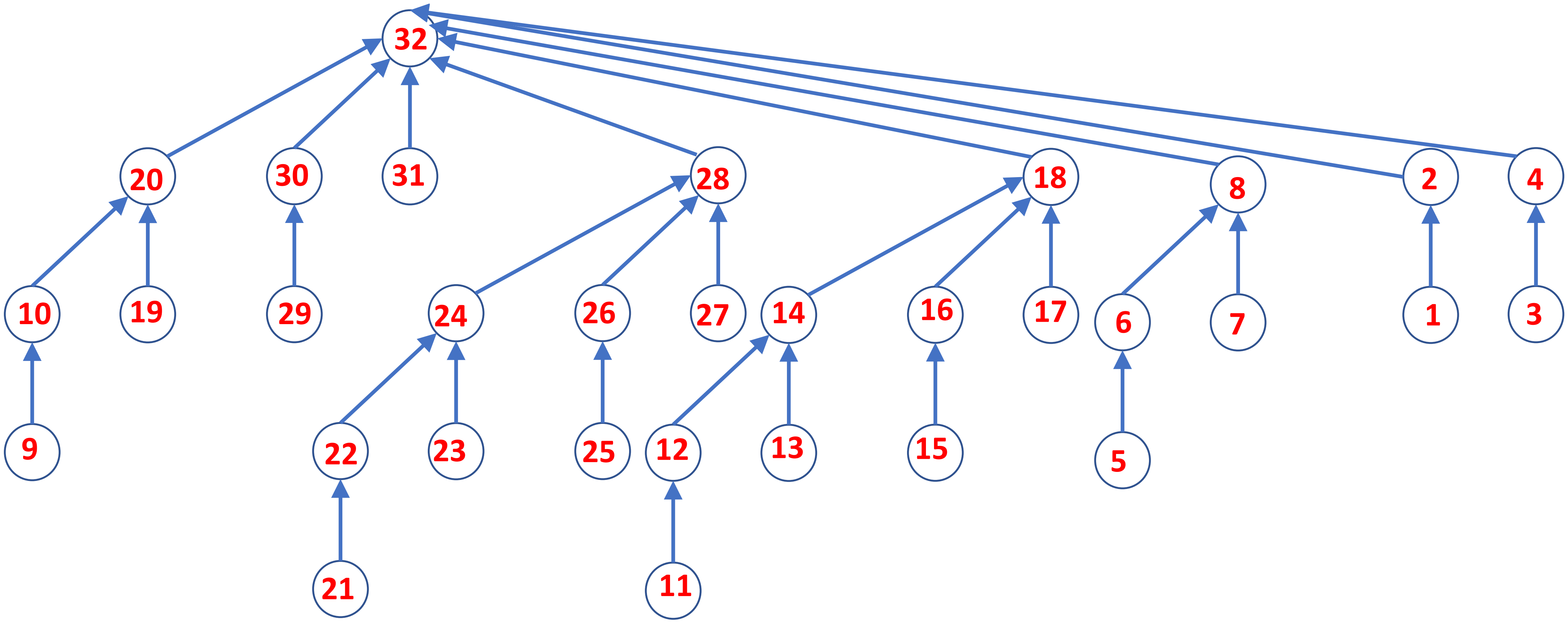
Example of Path Compression (PC)

Find(2)



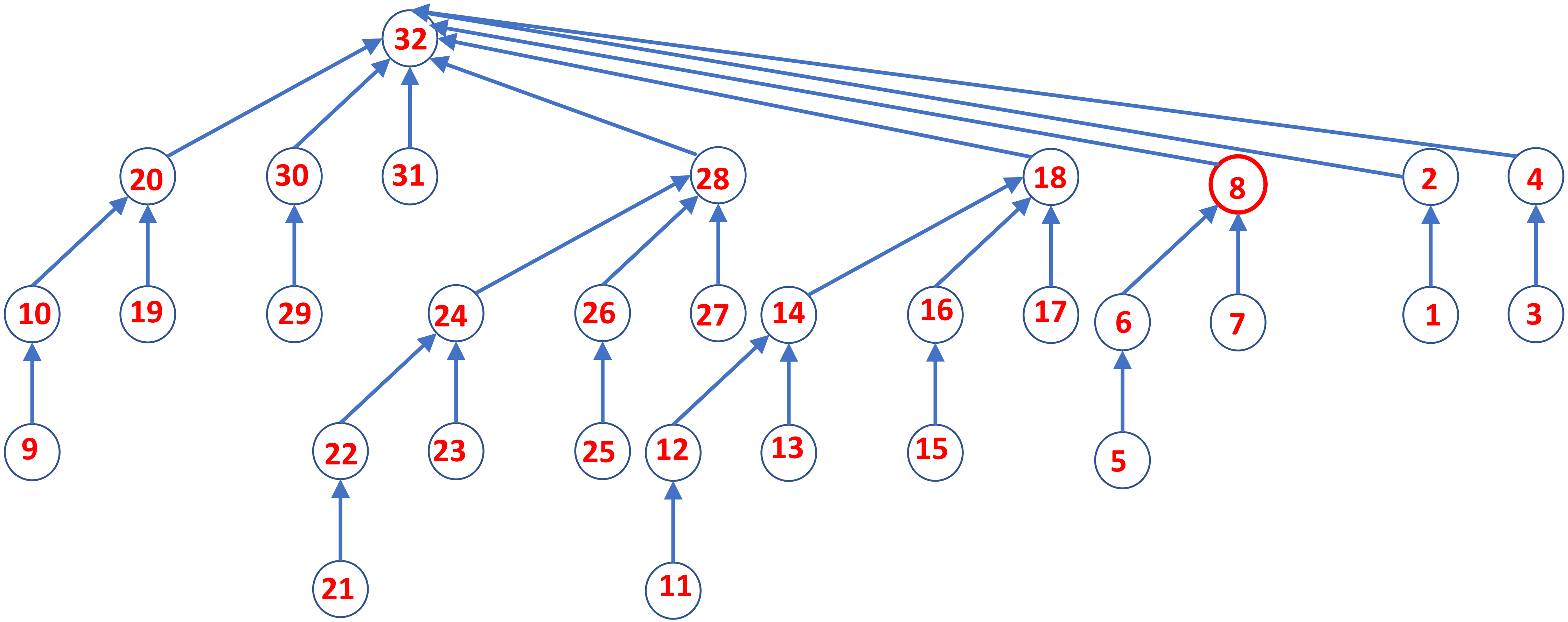
Example of Path Compression (PC)

Find(2)



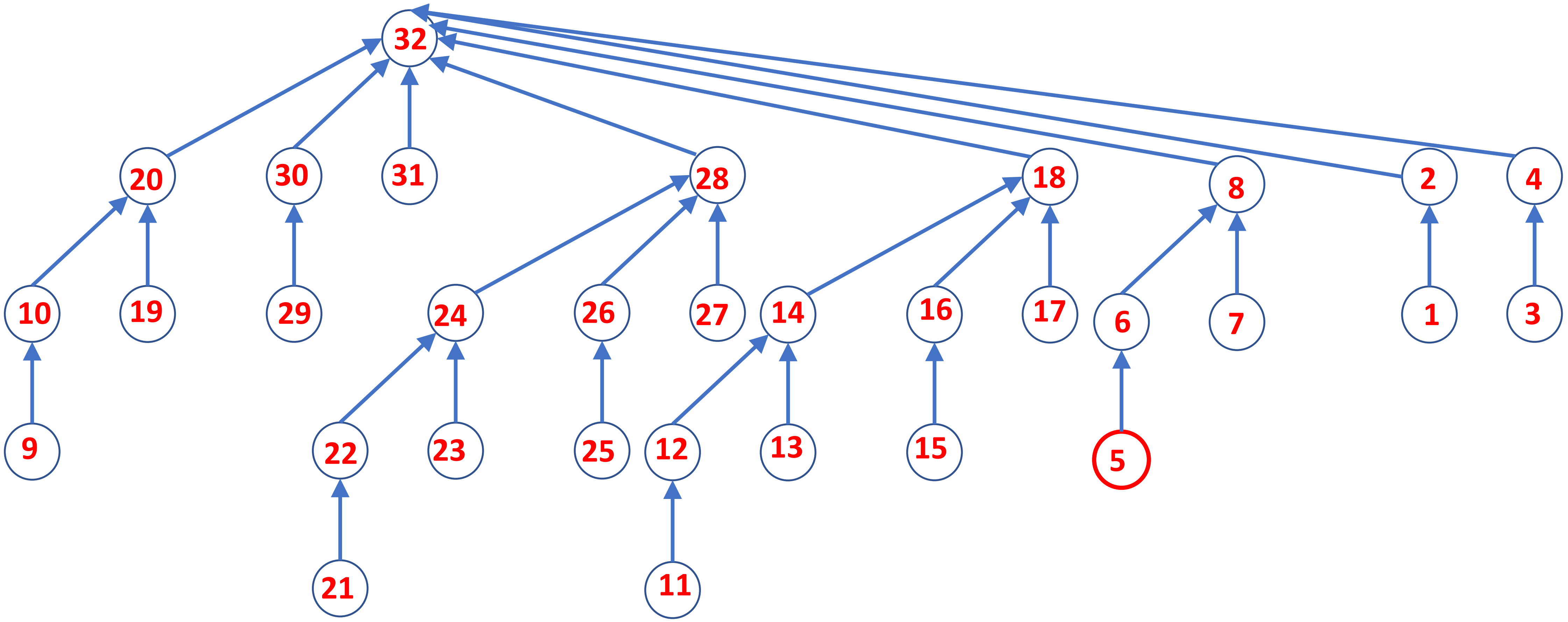
Example of Path Compression (PC)

Find(2)



Example of Path Compression (PC)

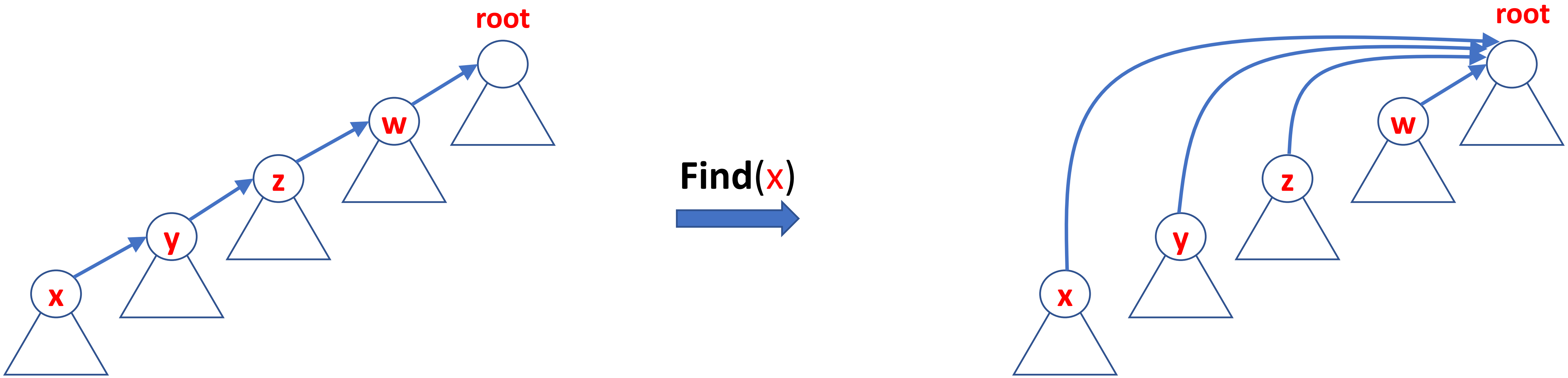
Find(2)



Heuristic 2: Path Compression (PC)

PC rule: In **Find**(x), make each vertex along the **Find** path a child of root

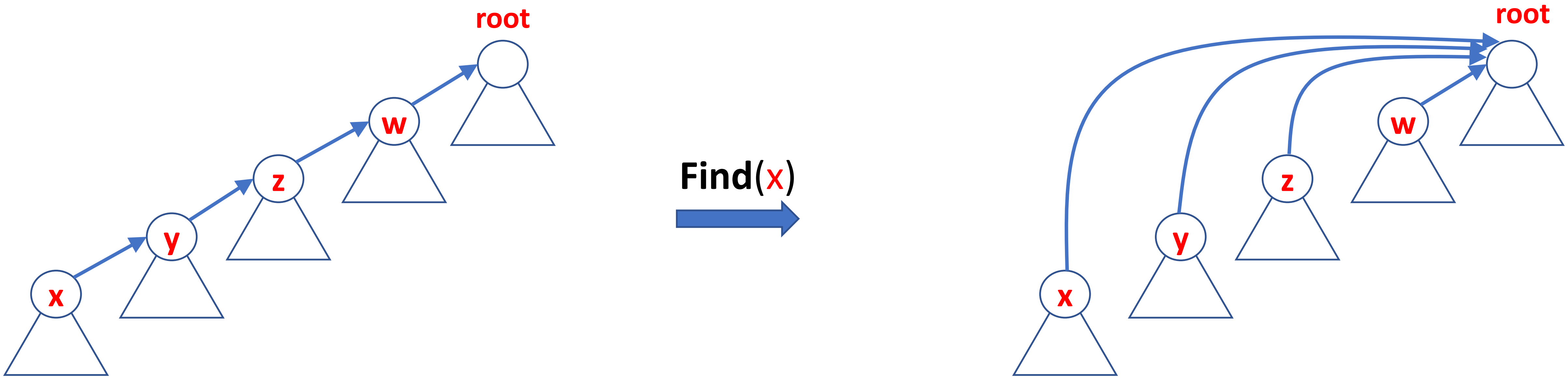
Heuristic 2: Path Compression (PC)



PC rule: In **Find**(x), make each vertex along the **Find** path a child of root

This increases the cost of **Find**(x), but makes several future finds cheaper

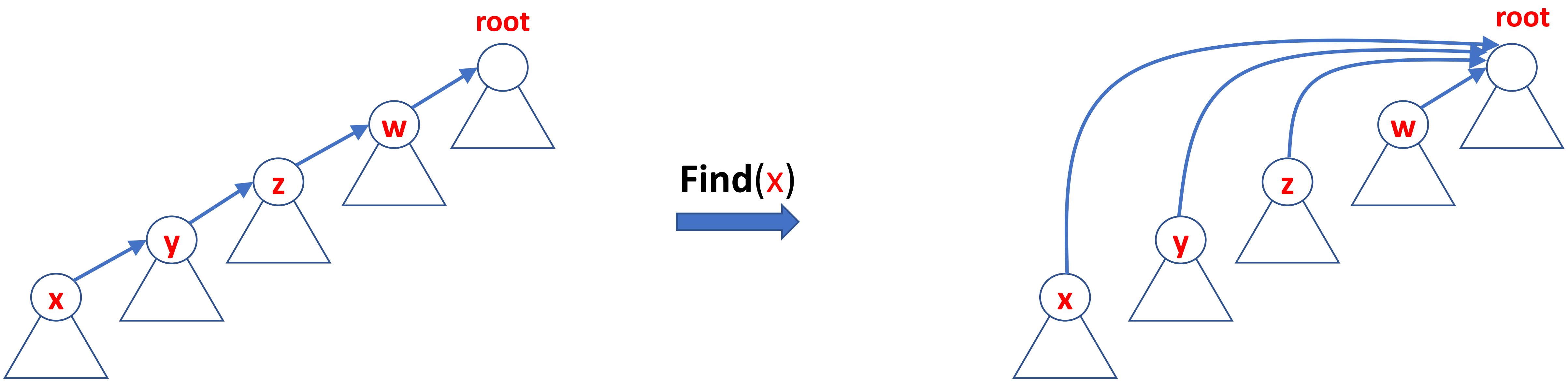
Heuristic 2: Path Compression (PC)



PC rule: In **Find**(x), make each vertex along the **Find** path a child of root

This increases the cost of **Find**(x), but makes several future finds cheaper:
Average cost of each **Find** decreases!

Heuristic 2: Path Compression (PC)



PC rule: In **Find**(x), make each vertex along the **Find** path a child of root

This increases the cost of **Find**(x), but makes several future finds cheaper:
Average cost of each **Find** decreases!

Amortization

With **WU** and **PC** , Time Complexity of σ?

σ: Sequence of n-1 **Unions** mixed with $m \geq n$ **Finds**

A quick detour…

Definition: 2^{*n}

A quick detour…

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, $n >= 0$

A quick detour…

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, $n >= 0$

$2^{*0} = 1$

A quick detour…

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, $n >= 0$

$2^{*0} = 1$

$2^{*1} =$

A quick detour…

Definition: $2{*}^{n}$

$2{*}^{0} = 1$

$2{*}^{n+1} = 2^{2{*}^{n}}$, $n >= 0$

$2{*}^{0} = 1$

$2{*}^{1} = 2^{2{*}^{0}}$

A quick detour…

Definition: $2*^{n}$

$2*^{0} = 1$

$2*^{n+1} = 2^{2*^{n}}$, $n >= 0$

$2*^{0} = 1$

$2*^{1} = 2^{2*^{0}} = 2^{1} = 2$

A quick detour…

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, $n >= 0$

$2^{*0} = 1$

$2^{*1} = 2^{2^{*0}} = 2^1 = 2$

$2^{*2} =$

A quick detour…

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, $n >= 0$

$2^{*0} = 1$

$2^{*1} = 2^{2^{*0}} = 2^1 = 2$

$2^{*2} = 2^{2^{*1}}$

A quick detour…

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, $n >= 0$

$2^{*0} = 1$

$2^{*1} = 2^{2^{*0}} = 2^1 = 2$

$2^{*2} = 2^{2^{*1}} = 2^2 = 4$

A quick detour…

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, $n >= 0$

$2^{*0} = 1$

$2^{*1} = 2^{2^{*0}} = 2^1 = 2$

$2^{*2} = 2^{2^{*1}} = 2^2 = 4$

$2^{*3} =$

A quick detour…

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, n >= 0

$2^{*0} = 1$

$2^{*1} = 2^{2^{*0}} = 2^1 = 2$

$2^{*2} = 2^{2^{*1}} = 2^2 = 4$

$2^{*3} = 2^{2^{*2}}$

A quick detour…

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, $n >= 0$

$2^{*0} = 1$

$2^{*1} = 2^{2^{*0}} = 2^1 = 2$

$2^{*2} = 2^{2^{*1}} = 2^2 = 4$

$2^{*3} = 2^{2^{*2}} = 2^4 = 16$

A quick detour…

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, n >= 0

$2^{*0} = 1$

$2^{*1} = 2^{2^{*0}} = 2^1 = 2$

$2^{*2} = 2^{2^{*1}} = 2^2 = 4$

$2^{*3} = 2^{2^{*2}} = 2^4 = 16$

$2^{*4} =$

A quick detour…

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, $n >= 0$

$2^{*0} = 1$

$2^{*1} = 2^{2^{*0}} = 2^1 = 2$

$2^{*2} = 2^{2^{*1}} = 2^2 = 4$

$2^{*3} = 2^{2^{*2}} = 2^4 = 16$

$2^{*4} = 2^{2^{*3}}$

A quick detour…

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, $n >= 0$

$2^{*0} = 1$

$2^{*1} = 2^{2^{*0}} = 2^1 = 2$

$2^{*2} = 2^{2^{*1}} = 2^2 = 4$

$2^{*3} = 2^{2^{*2}} = 2^4 = 16$

$2^{*4} = 2^{2^{*3}} = 2^{16} = 65536$

A quick detour…

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, $n >= 0$

$2^{*0} = 1$

$2^{*1} = 2^{2^{*0}} = 2^1 = 2$

$2^{*2} = 2^{2^{*1}} = 2^2 = 4$

$2^{*3} = 2^{2^{*2}} = 2^4 = 16$

$2^{*4} = 2^{2^{*3}} = 2^{16} = 65536$

$2^{*5} =$

A quick detour…

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, $n >= 0$

$2^{*0} = 1$

$2^{*1} = 2^{2^{*0}} = 2^1 = 2$

$2^{*2} = 2^{2^{*1}} = 2^2 = 4$

$2^{*3} = 2^{2^{*2}} = 2^4 = 16$

$2^{*4} = 2^{2^{*3}} = 2^{16} = 65536$

$2^{*5} = 2^{2^{*4}} =$

A quick detour...

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, n >= 0

$2^{*0} = 1$

$2^{*1} = 2^{2^{*0}} = 2^1 = 2$

$2^{*2} = 2^{2^{*1}} = 2^2 = 4$

$2^{*3} = 2^{2^{*2}} = 2^4 = 16$

$2^{*4} = 2^{2^{*3}} = 2^{16} = 65536$

$2^{*5} = 2^{2^{*4}} = 2^{65536}$

A quick detour…

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, $n >= 0$

$2^{*0} = 1$

$2^{*1} = 2^{2^{*0}} = 2^1 = 2$

$2^{*2} = 2^{2^{*1}} = 2^2 = 4$

$2^{*3} = 2^{2^{*2}} = 2^4 = 16$

$2^{*4} = 2^{2^{*3}} = 2^{16} = 65536$

$2^{*5} = 2^{2^{*4}} = 2^{65536} \approx 10^{19729}$

A quick detour…

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, $n >= 0$

$2^{*0} = 1$

$2^{*1} = 2^{2^{*0}} = 2^1 = 2$

$2^{*2} = 2^{2^{*1}} = 2^2 = 4$

$2^{*3} = 2^{2^{*2}} = 2^4 = 16$

$2^{*4} = 2^{2^{*3}} = 2^{16} = 65536$

$2^{*5} = 2^{2^{*4}} = 2^{65536} \approx 10^{19729}$

Estimated # of atoms in observable universe $\approx 10^{80}$

A quick detour...

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, $n >= 0$

$2^{*0} = 1$

$2^{*1} = 2^{2^{*0}} = 2^1 = 2$

$2^{*2} = 2^{2^{*1}} = 2^2 = 4$

$2^{*3} = 2^{2^{*2}} = 2^4 = 16$

$2^{*4} = 2^{2^{*3}} = 2^{16} = 65536$

$2^{*5} = 2^{2^{*4}} = 2^{65536} \approx 10^{19729}$

$2^{*6} =$

A quick detour…

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, n >= 0

$2^{*0} = 1$

$2^{*1} = 2^{2^{*0}} = 2^1 = 2$

$2^{*2} = 2^{2^{*1}} = 2^2 = 4$

$2^{*3} = 2^{2^{*2}} = 2^4 = 16$

$2^{*4} = 2^{2^{*3}} = 2^{16} = 65536$

$2^{*5} = 2^{2^{*4}} = 2^{65536} \approx 10^{19729}$

2^{*6} = REALLY BIG !

A quick detour...

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, $n >= 0$

$$2^{*n} = \left.2^{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}\right\}\text{n 2s}$$

$2^{*0} = 1$

$2^{*1} = 2^{2^{*0}} = 2^1 = 2$

$2^{*2} = 2^{2^{*1}} = 2^2 = 4$

$2^{*3} = 2^{2^{*2}} = 2^4 = 16$

$2^{*4} = 2^{2^{*3}} = 2^{16} = 65536$

$2^{*5} = 2^{2^{*4}} = 2^{65536} \approx 10^{19729}$

2^{*6} = REALLY BIG !

A quick detour…

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, $n >= 0$

$$2^{*n} = \left. 2^{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}} \right\} \text{n 2s}$$

This function grows very fast with n !!

$2^{*0} = 1$

$2^{*1} = 2^{2^{*0}} = 2^1 = 2$

$2^{*2} = 2^{2^{*1}} = 2^2 = 4$

$2^{*3} = 2^{2^{*2}} = 2^4 = 16$

$2^{*4} = 2^{2^{*3}} = 2^{16} = 65536$

$2^{*5} = 2^{2^{*4}} = 2^{65536} \approx 10^{19729}$

2^{*6} = REALLY BIG !

A quick detour…

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, $n >= 0$

$$2^{*n} = \left.2^{2^{2^{\cdot^{\cdot^{2}}}}}\right\} \text{n 2s}$$

This function grows very fast with n !!

$2^{*0} = 1$

$2^{*1} = 2^{2^{*0}} = 2^1 = 2$

$2^{*2} = 2^{2^{*1}} = 2^2 = 4$

$2^{*3} = 2^{2^{*2}} = 2^4 = 16$

$2^{*4} = 2^{2^{*3}} = 2^{16} = 65536$

$2^{*5} = 2^{2^{*4}} = 2^{65536} \approx 10^{19729}$

2^{*6} = REALLY BIG !

Definition: $\log^* n$

$\log^* n = \min\{k : 2^{*k} \geq n\}$

A quick detour…

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, $n >= 0$

$$2^{*n} = \left. 2^{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}} \right\} \text{n 2s}$$

This function grows very fast with n !!

$2^{*0} = 1$

$2^{*1} = 2^{2^{*0}} = 2^1 = 2$

$2^{*2} = 2^{2^{*1}} = 2^2 = 4$

$2^{*3} = 2^{2^{*2}} = 2^4 = 16$

$2^{*4} = 2^{2^{*3}} = 2^{16} = 65536$

$2^{*5} = 2^{2^{*4}} = 2^{65536} \approx 10^{19729}$

2^{*6} = REALLY BIG !

Definition: $\log^* n$

$\log^* n = \min\{k : 2^{*k} \geq n\}$

n	1
log* n	0

A quick detour…

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, n >= 0

$$2^{*n} = 2^{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}} \Big\} \text{ n 2s}$$

This function grows very fast with n !!

$2^{*0} = 1$

$2^{*1} = 2^{2^{*0}} = 2^1 = 2$

$2^{*2} = 2^{2^{*1}} = 2^2 = 4$

$2^{*3} = 2^{2^{*2}} = 2^4 = 16$

$2^{*4} = 2^{2^{*3}} = 2^{16} = 65536$

$2^{*5} = 2^{2^{*4}} = 2^{65536} \approx 10^{19729}$

2^{*6} = REALLY BIG !

Definition: log*n

$\log^* n = \min\{k : 2^{*k} \geq n\}$

n	1	2
log* n	0	1

A quick detour…

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, $n >= 0$

$$2^{*n} = \left. 2^{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}} \right\} \text{n 2s}$$

This function grows very fast with n !!

$2^{*0} = 1$

$2^{*1} = 2^{2^{*0}} = 2^1 = 2$

$2^{*2} = 2^{2^{*1}} = 2^2 = 4$

$2^{*3} = 2^{2^{*2}} = 2^4 = 16$

$2^{*4} = 2^{2^{*3}} = 2^{16} = 65536$

$2^{*5} = 2^{2^{*4}} = 2^{65536} \approx 10^{19729}$

2^{*6} = REALLY BIG !

Definition: $\log^* n$

$\log^* n = \min\{k : 2^{*k} \geq n\}$

n	1	2	3, 4
log* n	0	1	2

A quick detour…

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, $n >= 0$

$$2^{*n} = 2^{2^{2^{\cdot^{\cdot^{2}}}}} \quad \Big\} \text{ n 2s}$$

This function grows very fast with n !!

$2^{*0} = 1$

$2^{*1} = 2^{2^{*0}} = 2^1 = 2$

$2^{*2} = 2^{2^{*1}} = 2^2 = 4$

$2^{*3} = 2^{2^{*2}} = 2^4 = 16$

$2^{*4} = 2^{2^{*3}} = 2^{16} = 65536$

$2^{*5} = 2^{2^{*4}} = 2^{65536} \approx 10^{19729}$

2^{*6} = REALLY BIG !

Definition: $\log^* n$

$\log^* n = \min\{k : 2^{*k} \geq n\}$

n	1	2	3, 4	5, 6, 7,…., 16
log* n	0	1	2	3

A quick detour…

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, $n >= 0$

$$2^{*n} = \underbrace{2^{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}}_{n\ 2\text{s}}$$

This function grows very fast with n !!

$2^{*0} = 1$

$2^{*1} = 2^{2^{*0}} = 2^1 = 2$

$2^{*2} = 2^{2^{*1}} = 2^2 = 4$

$2^{*3} = 2^{2^{*2}} = 2^4 = 16$

$2^{*4} = 2^{2^{*3}} = 2^{16} = 65536$

$2^{*5} = 2^{2^{*4}} = 2^{65536} \approx 10^{19729}$

2^{*6} = REALLY BIG !

Definition: log*n

$\log^* n = \min\{k : 2^{*k} \geq n\}$

n	1	2	3, 4	5, 6, 7,…., 16	17, 18, 19, …., 65536
log* n	0	1	2	3	4

A quick detour…

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, $n >= 0$

$$2^{*n} = \left. 2^{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}} \right\} n \text{ 2s}$$

This function grows very fast with n !!

$2^{*0} = 1$

$2^{*1} = 2^{2^{*0}} = 2^1 = 2$

$2^{*2} = 2^{2^{*1}} = 2^2 = 4$

$2^{*3} = 2^{2^{*2}} = 2^4 = 16$

$2^{*4} = 2^{2^{*3}} = 2^{16} = 65536$

$2^{*5} = 2^{2^{*4}} = 2^{65536} \approx 10^{19729}$

2^{*6} = REALLY BIG !

Definition: log*n

$\log^* n = \min\{k : 2^{*k} \geq n\}$

n	1	2	3, 4	5, 6, 7,…., 16	17, 18, 19, …., 65536	65537,….., 10^{19729}
log* n	0	1	2	3	4	5

A quick detour…

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, n >= 0

$$2^{*n} = \left.2^{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}\right\} \text{n 2s}$$

This function grows very fast with n !!

$2^{*0} = 1$

$2^{*1} = 2^{2^{*0}} = 2^1 = 2$

$2^{*2} = 2^{2^{*1}} = 2^2 = 4$

$2^{*3} = 2^{2^{*2}} = 2^4 = 16$

$2^{*4} = 2^{2^{*3}} = 2^{16} = 65536$

$2^{*5} = 2^{2^{*4}} = 2^{65536} \approx 10^{19729}$

2^{*6} = REALLY BIG !

Definition: log*n

$\log^* n = \min\{k : 2^{*k} \geq n\}$

n	1	2	3, 4	5, 6, 7,…., 16	17, 18, 19, …., 65536	65537,….., 10^{19729}	…..
log* n	0	1	2	3	4	5	

A quick detour…

Definition: 2^{*n}

$2^{*0} = 1$

$2^{*n+1} = 2^{2^{*n}}$, $n >= 0$

$2^{*n} = 2^{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}$ $\Big\}$ n 2s

This function grows very fast with n !!

$2^{*0} = 1$

$2^{*1} = 2^{2^{*0}} = 2^1 = 2$

$2^{*2} = 2^{2^{*1}} = 2^2 = 4$

$2^{*3} = 2^{2^{*2}} = 2^4 = 16$

$2^{*4} = 2^{2^{*3}} = 2^{16} = 65536$

$2^{*5} = 2^{2^{*4}} = 2^{65536} \approx 10^{19729}$

2^{*6} = REALLY BIG !

Definition: $\log^* n$

$\log^* n = \min\{k : 2^{*k} \geq n\}$

n	1	2	3, 4	5, 6, 7,…., 16	17, 18, 19, …., 65536	65537,….., 10^{19729}	…..
log* n	0	1	2	3	4	5	

log* n grows very slowly with n !!

Time Complexity of σ, with **WU** and **PC**

σ: Sequence of n-1 **Unions** mixed with $m \geq n$ **Finds**

Theorem: With **WU** and **PC**, executing every such σ takes O(m log* n) time

Time Complexity of σ, with **WU** and **PC**

σ: Sequence of n-1 **Unions** mixed with $m \geq n$ **Finds**

Theorem: With **WU** and **PC**, executing every such σ takes $O(m \log^* n)$ time

That is, executing ***every*** such σ takes ***at most*** $m \log^* n$ time,
for large m, n, and within a constant factor

Time Complexity of σ, with **WU** and **PC**

σ: Sequence of n-1 **Unions** mixed with $m \geq n$ **Finds**

Theorem: With **WU** and **PC**, executing every such σ takes O(m log* n) time

That is, executing ***every*** such σ takes ***at most*** m log* n time,
for large m, n, and within a constant factor

However: Is there ***some*** σ that takes ***at least*** m log* n time ?

Time Complexity of σ, with **WU** and **PC**

σ: Sequence of $n-1$ **Unions** mixed with $m \geq n$ **Finds**

Theorem: With **WU** and **PC**, executing every such σ takes $O(m \log^* n)$ time

That is, executing ***every*** such σ takes ***at most*** $m \log^* n$ time,
for large m, n, and within a constant factor

However: Is there ***some*** σ that takes ***at least*** $m \log^* n$ time ?

Is the following claim true?

Claim: With **WU** and **PC**, executing every such σ takes $O(m)$ time

Analysis of Disjoint Forest: A Timeline

Analysis of Disjoint Forest: A Timeline

1964

Forest Implementation introduced

Bernard A. Galler
Michael J. Fischer

Analysis of Disjoint Forest: A Timeline

1964

Forest Implementation introduced

Bernard A. Galler
Michael J. Fischer

1973

O(m log* n) upper-bound for Forest Implementation

John E. Hopcroft
Jeffrey D. Ullman

Analysis of Disjoint Forest: A Timeline

1964

Forest Implementation introduced

Bernard A. Galler
Michael J. Fischer

1973

$O(m \log^* n)$ upper-bound for Forest Implementation

John E. Hopcroft
Jeffrey D. Ullman

1975

$O(m\ \alpha(m,n))$ upper-bound for Forest Implementation

Robert E. Tarjan

Analysis of Disjoint Forest: A Timeline

1964

Forest Implementation introduced

Bernard A. Galler
Michael J. Fischer

1973

$O(m \log^* n)$ upper-bound for Forest Implementation

John E. Hopcroft
Jeffrey D. Ullman

1975

$O(m\, \alpha(m,n))$ upper-bound for Forest Implementation

Robert E. Tarjan

1979

$\Omega(m\, \alpha(m,n))$ lower-bound for *any* Disjoint-Set data structure that satisfies some “technical assumptions”

Robert E. Tarjan

Analysis of Disjoint Forest: A Timeline

1964	Forest Implementation introduced	Bernard A. Galler, Michael J. Fischer
1973	$O(m \log^* n)$ upper-bound for Forest Implementation	John E. Hopcroft, Jeffrey D. Ullman
1975	$O(m\, \alpha(m,n))$ upper-bound for Forest Implementation	Robert E. Tarjan
1979	$\Omega(m\, \alpha(m,n))$ lower-bound for *any* Disjoint-Set data structure that satisfies some “technical assumptions”	Robert E. Tarjan
1989	$\Omega(m\, \alpha(m,n))$ lower-bound for *any* Disjoint-Set data structure	Michael L. Fredman, Michael E. Saks