Randomized Quicksort

(CLRS textbook: chapter 7)

Randomized Quicksort Algorithm (High-Level Description)

Randomized Quicksort Algorithm (High-Level Description)

Input: Set S of n distinct keys

Randomized Quicksort Algorithm (High-Level Description)

Input: Set S of n distinct keys

Output: The keys of S in increasing order

Randomized Quicksort Algorithm (High-Level Description)

Input: Set S of n distinct keys

Output: The keys of S in increasing order

RQS(S) (* recursive, divide-and-conquer algorithm *)

Randomized Quicksort Algorithm (High-Level Description)

Input: Set S of n distinct keys

Output: The keys of S in increasing order

RQS(S) (* recursive, divide-and-conquer algorithm *)

- If S is empty then return.

Randomized Quicksort Algorithm (High-Level Description)

Input: Set S of n distinct keys

Output: The keys of S in increasing order

RQS(S) (* recursive, divide-and-conquer algorithm *)

- If S is empty then return.
- If $|S| = 1$ then output the key in S and return.

Randomized Quicksort Algorithm (High-Level Description)

Input: Set S of n distinct keys

Output: The keys of S in increasing order

RQS(S) (* recursive, divide-and-conquer algorithm *)

- If S is empty then return.
- If $|S| = 1$ then output the key in S and return.
- Select a key p (called a "pivot") uniformly at random from S.

Randomized Quicksort Algorithm (High-Level Description)

Input: Set S of n distinct keys

Output: The keys of S in increasing order

RQS(S) (* recursive, divide-and-conquer algorithm *)

- If S is empty then return.
- If $|S| = 1$ then output the key in S and return.
- Select a key p (called a "pivot") uniformly at random from S. (each key of S is equally likely to be selected as pivot.)

Randomized Quicksort Algorithm (High-Level Description)

Input: Set S of n distinct keys

Output: The keys of S in increasing order

RQS(S) (* recursive, divide-and-conquer algorithm *)

- If S is empty then return.
- If $|S| = 1$ then output the key in S and return.
- Select a key p (called a "pivot") uniformly at random from S. (each key of S is equally likely to be selected as pivot.)
- By comparing p to every other key in S, split the set S into two subsets:

Randomized Quicksort Algorithm (High-Level Description)

Input: Set S of n distinct keys

Output: The keys of S in increasing order

RQS(S) (* recursive, divide-and-conquer algorithm *)

- If S is empty then return.
- If $|S| = 1$ then output the key in S and return.
- Select a key p (called a "pivot") uniformly at random from S. (each key of S is equally likely to be selected as pivot.)
- By comparing p to every other key in S, split the set S into two subsets:
  - $S_< = \{s \in S \mid s < p\}$

Randomized Quicksort Algorithm (High-Level Description)

Input: Set S of n distinct keys
Output: The keys of S in increasing order

RQS(S) (* recursive, divide-and-conquer algorithm *)

- If S is empty then return.
- If $|S| = 1$ then output the key in S and return.
- Select a key p (called a "pivot") uniformly at random from S. (each key of S is equally likely to be selected as pivot.)
- By comparing p to every other key in S, split the set S into two subsets:
  - $S_< = \{s \in S \mid s < p\}$
  - $S_> = \{s \in S \mid s > p\}$

Randomized Quicksort Algorithm (High-Level Description)

Input: Set S of n distinct keys

Output: The keys of S in increasing order

RQS(S) (* recursive, divide-and-conquer algorithm *)

- If S is empty then return.
- If $|S| = 1$ then output the key in S and return.
- Select a key p (called a "pivot") uniformly at random from S.
  (each key of S is equally likely to be selected as pivot.)
- By comparing p to every other key in S, split the set S into two subsets:
  - $S_< = \{s \in S \mid s < p\}$
  - $S_> = \{s \in S \mid s > p\}$

  (doing this for $|S| = n$ takes $n - 1$ key comparisons)

Randomized Quicksort Algorithm (High-Level Description)

Input: Set S of n distinct keys

Output: The keys of S in increasing order

RQS(S) (* recursive, divide-and-conquer algorithm *)

- If S is empty then return.
- If $|S| = 1$ then output the key in S and return.
- Select a key p (called a "pivot") uniformly at random from S.
  (each key of S is equally likely to be selected as pivot.)
- By comparing p to every other key in S, split the set S into two subsets:
  - $S_< = \{s \in S \mid s < p\}$
  - $S_> = \{s \in S \mid s > p\}$

  (doing this for $|S| = n$ takes $n - 1$ key comparisons)
- $RQS(S_<)$;

Randomized Quicksort Algorithm (High-Level Description)

Input: Set S of n distinct keys

Output: The keys of S in increasing order

RQS(S) (* recursive, divide-and-conquer algorithm *)

- If S is empty then return.
- If $|S| = 1$ then output the key in S and return.
- Select a key p (called a "pivot") uniformly at random from S.
  (each key of S is equally likely to be selected as pivot.)
- By comparing p to every other key in S, split the set S into two subsets:
  - $S_< = \{s \in S \mid s < p\}$
  - $S_> = \{s \in S \mid s > p\}$

  (doing this for $|S| = n$ takes $n - 1$ key comparisons)
- $RQS(S_<)$; output p ;

Randomized Quicksort Algorithm (High-Level Description)

Input: Set S of n distinct keys
Output: The keys of S in increasing order

RQS(S) (* recursive, divide-and-conquer algorithm *)

- If S is empty then return.
- If $|S| = 1$ then output the key in S and return.
- Select a key p (called a "pivot") uniformly at random from S. (each key of S is equally likely to be selected as pivot.)
- By comparing p to every other key in S, split the set S into two subsets:
  - $S_< = \{s \in S \mid s < p\}$
  - $S_> = \{s \in S \mid s > p\}$

  (doing this for $|S| = n$ takes $n - 1$ key comparisons)
- $RQS(S_<)$; output p ; $RQS(S_>)$

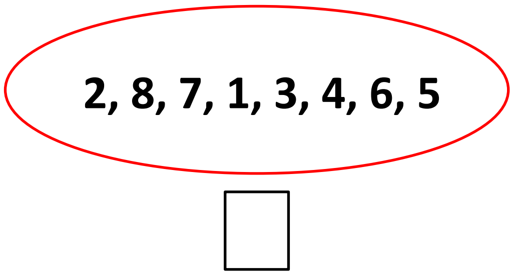
Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}

2, 8, 7, 1, 3, 4, 6, 5

Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}

2, 8, 7, 1, 3, 4, 6, 5

Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}

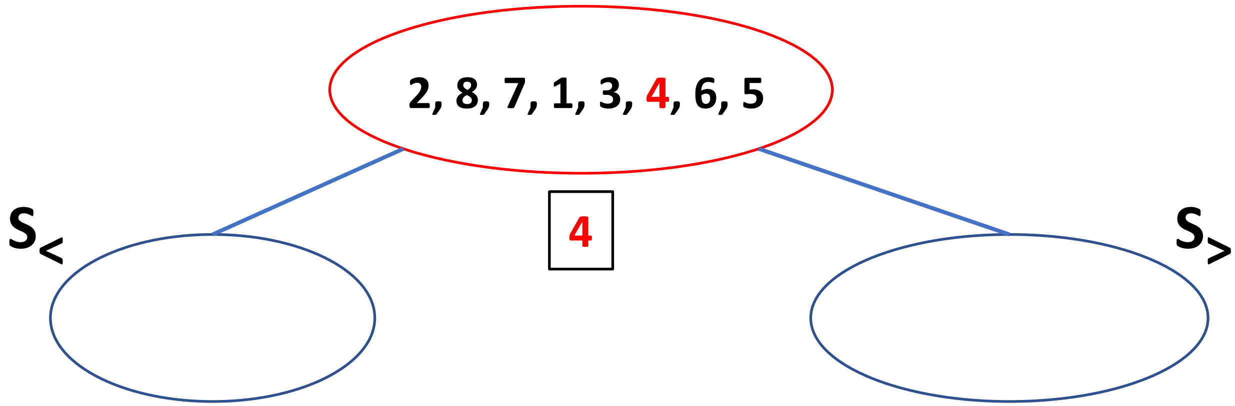


Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}

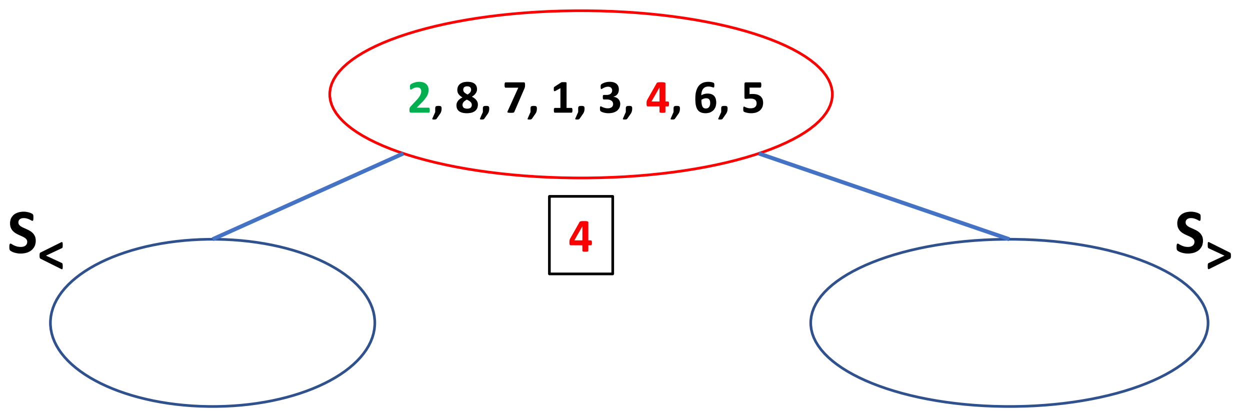
2, 8, 7, 1, 3, 4, 6, 5

$\boxed{\mathbf{4}}$

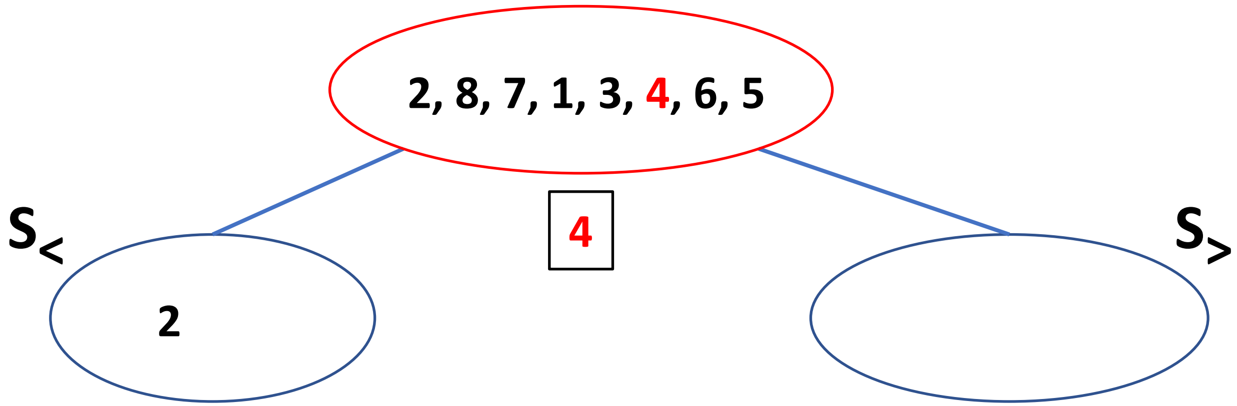
Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



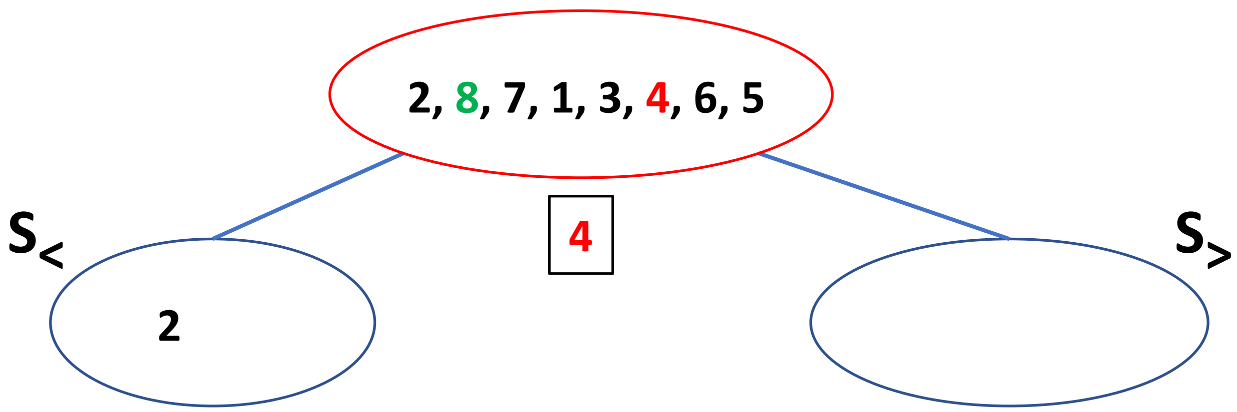
Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



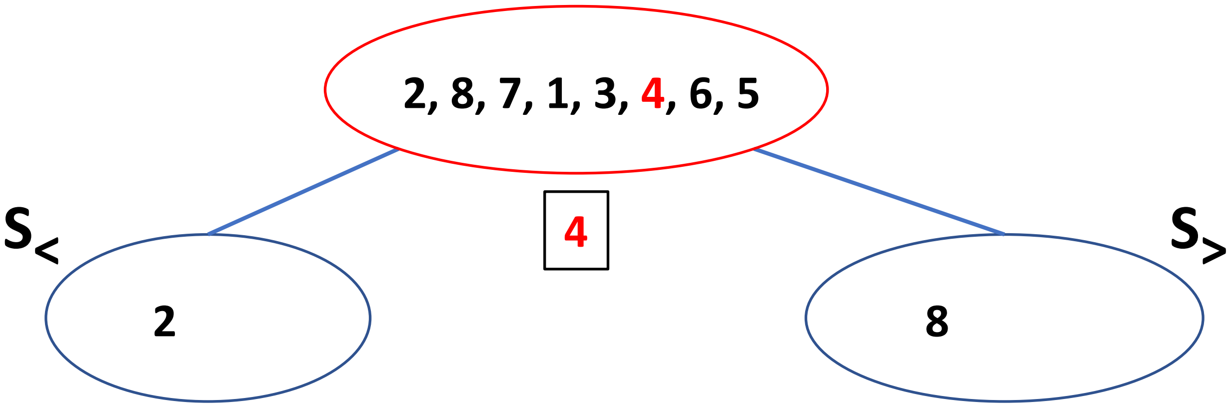
Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



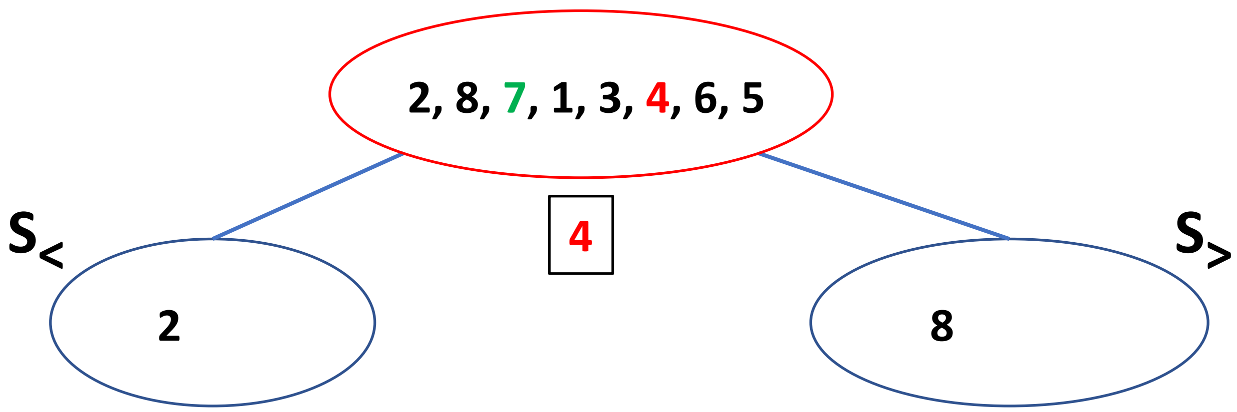
Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



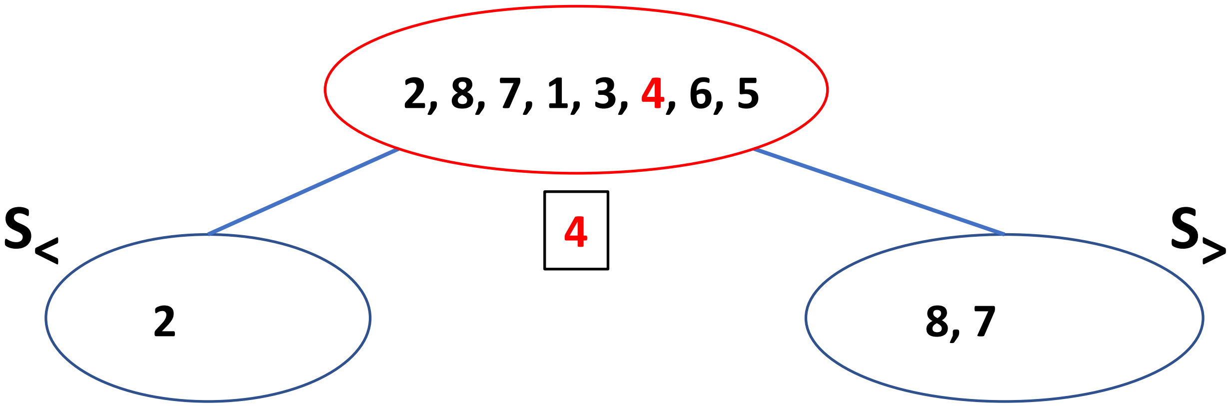
Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



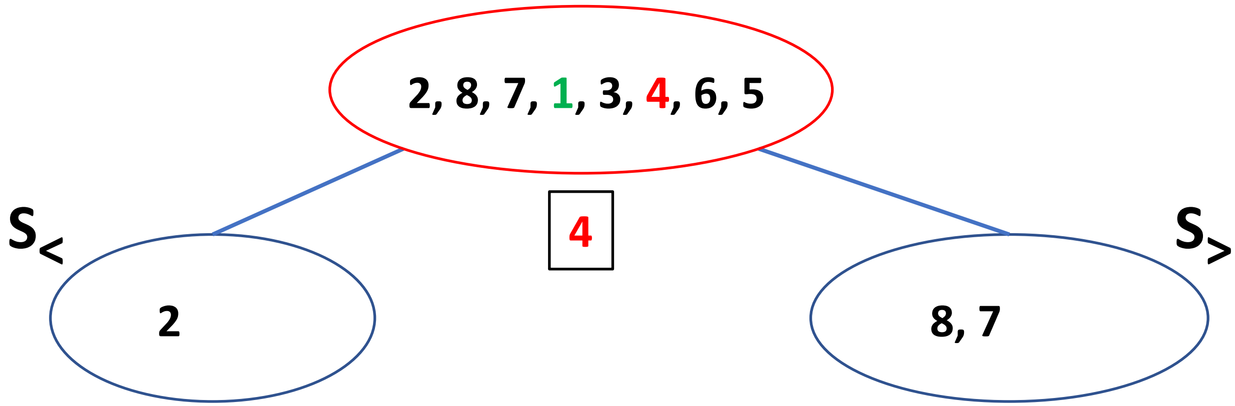
Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



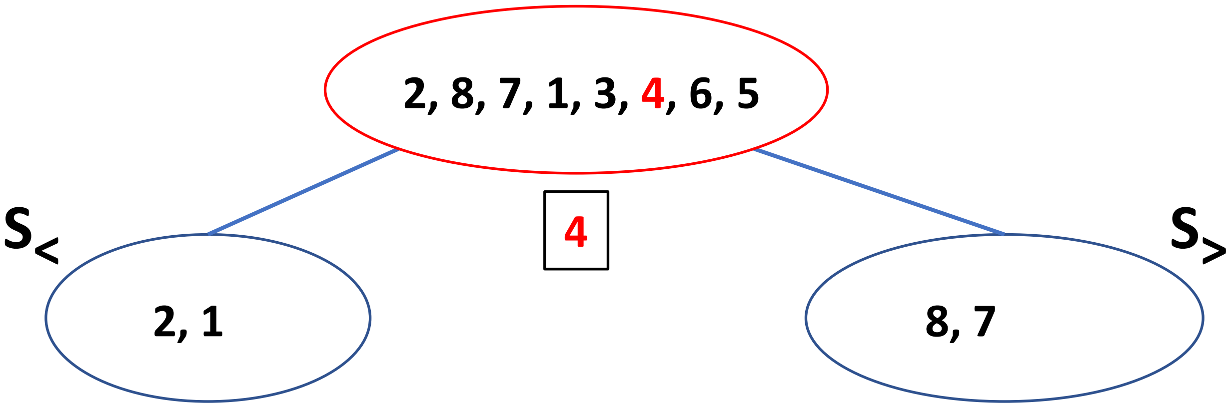
Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



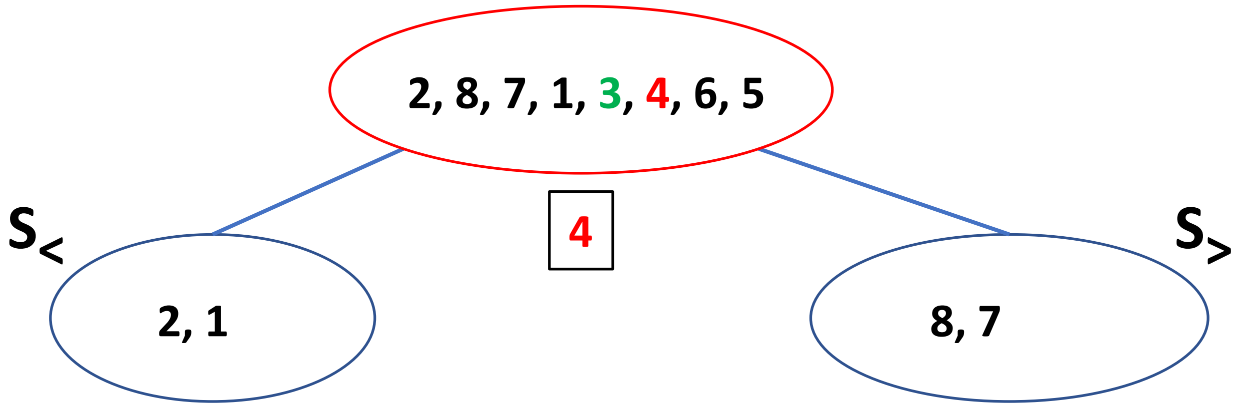
Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



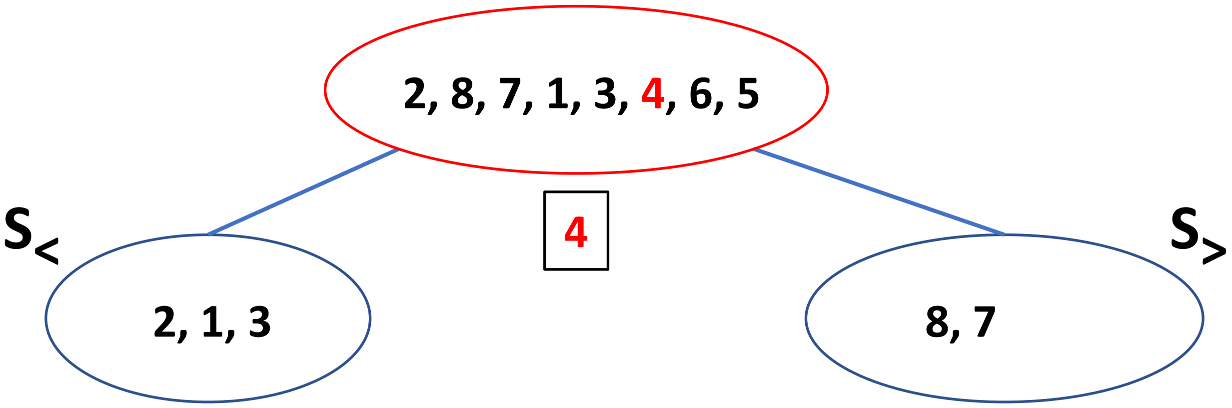
Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



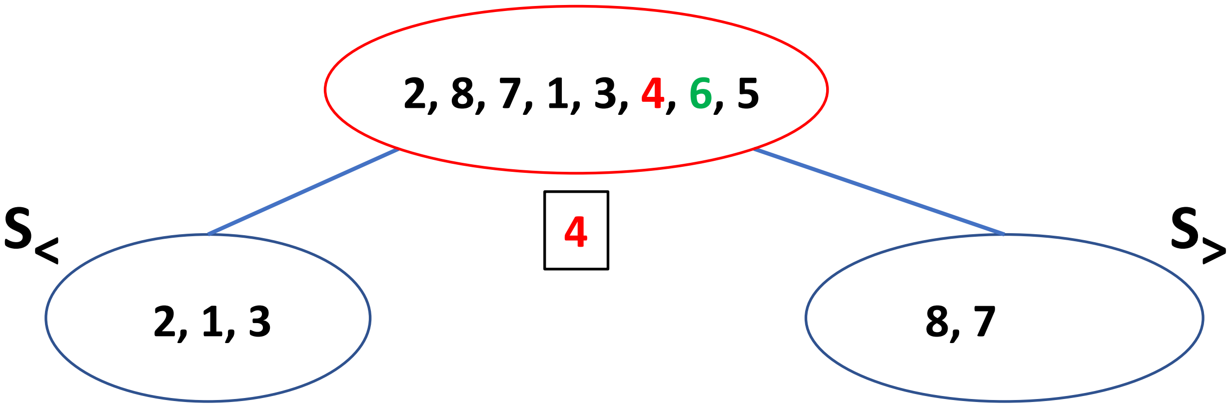
Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



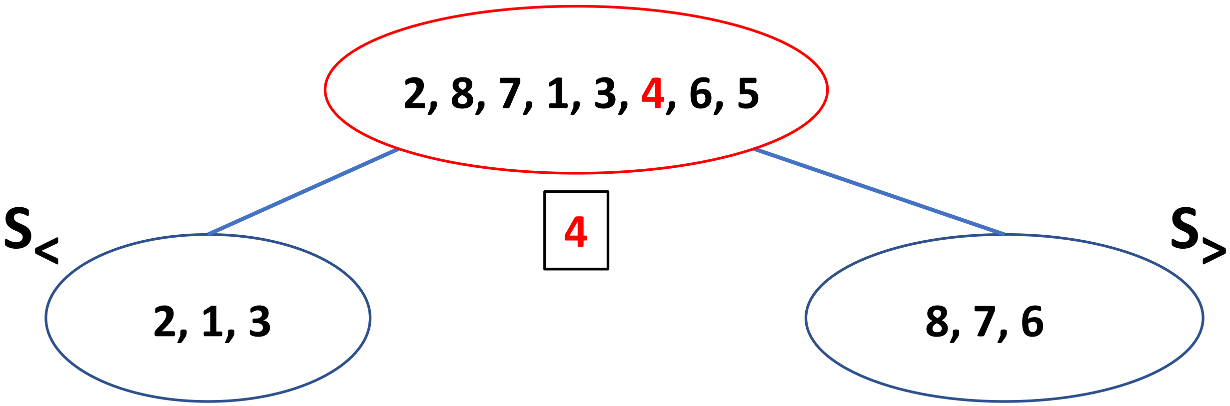
Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



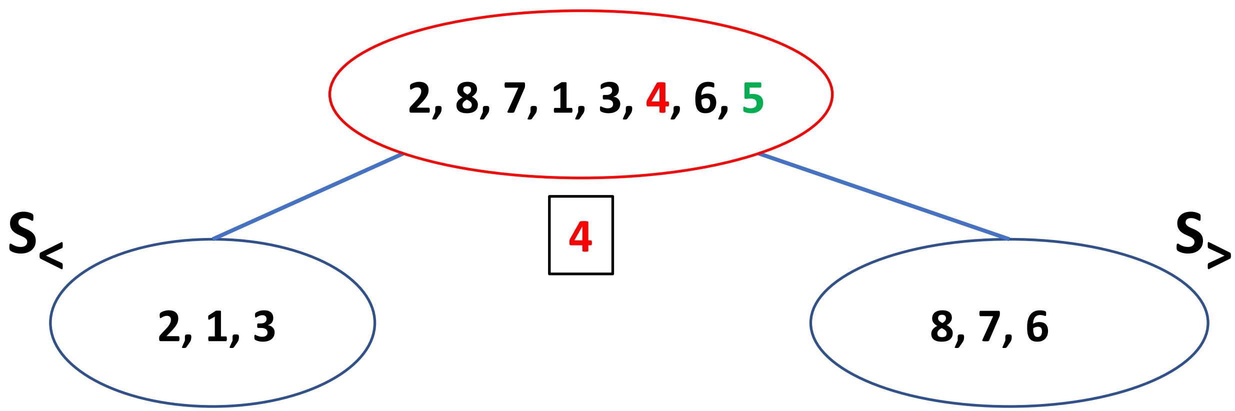
Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



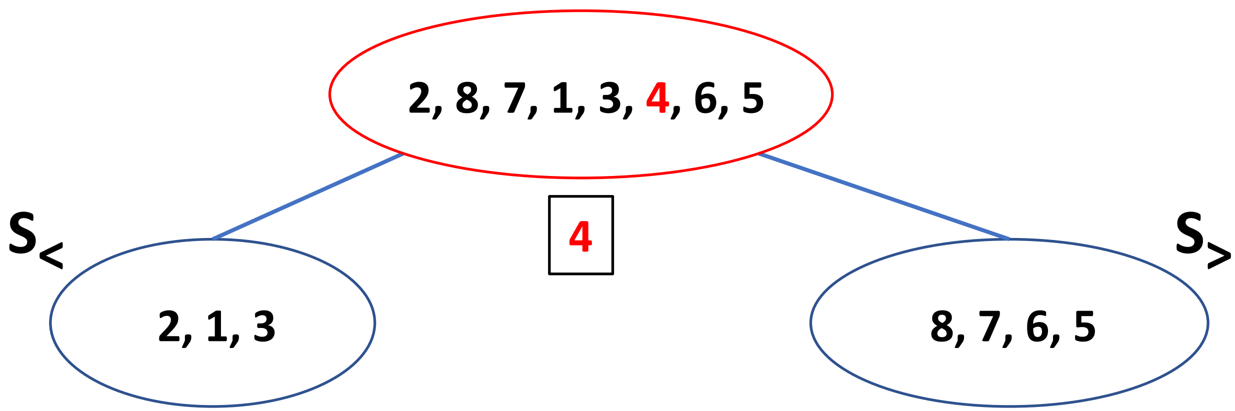
Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



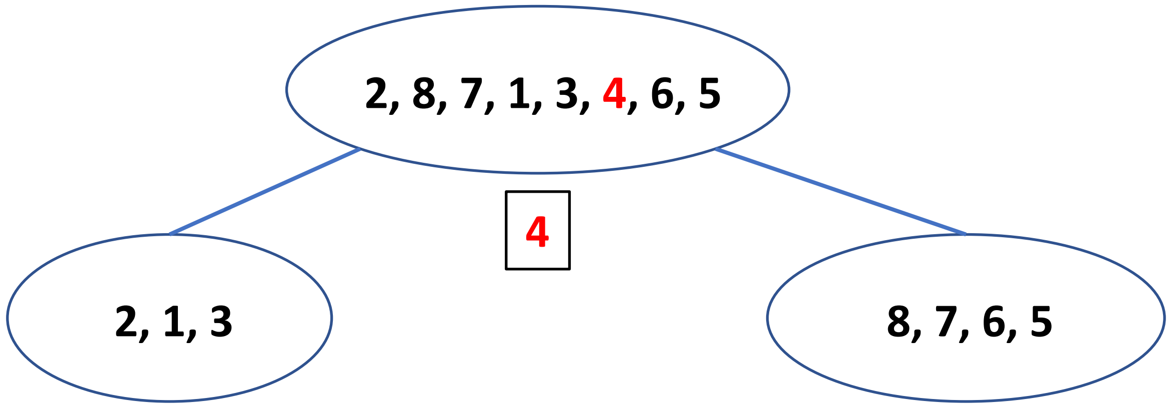
Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



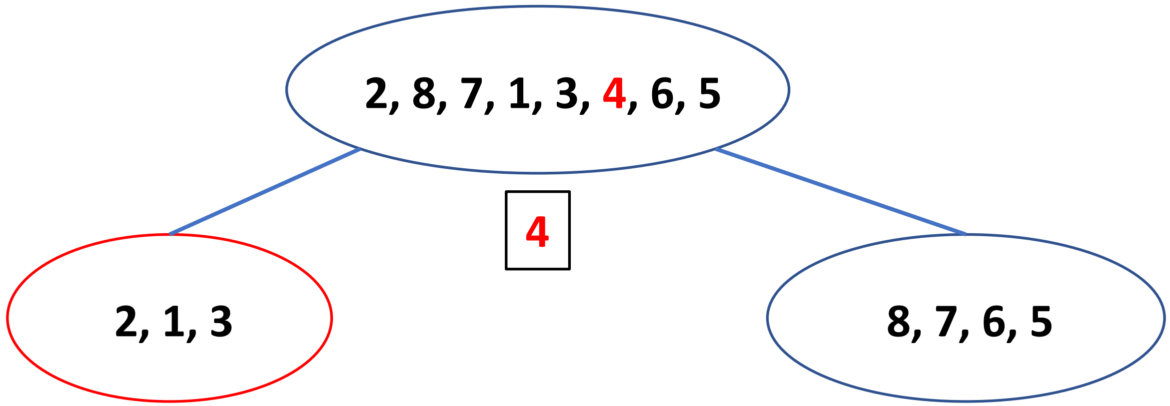
Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



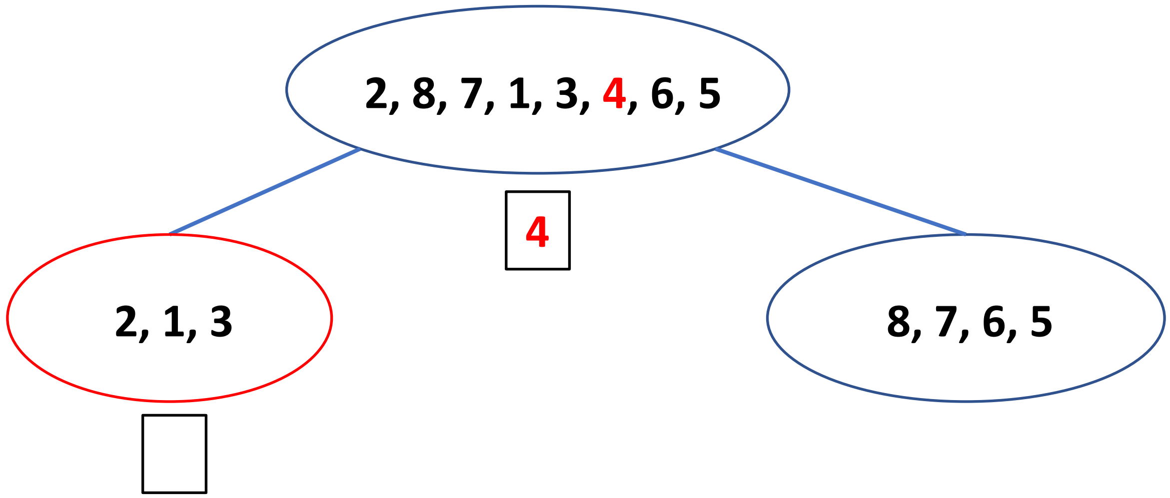
Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



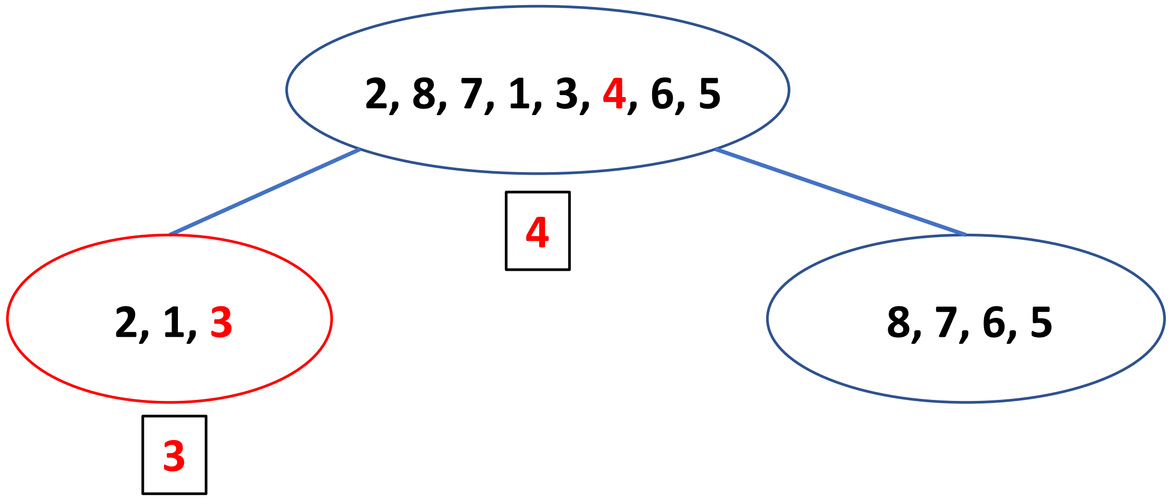
Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



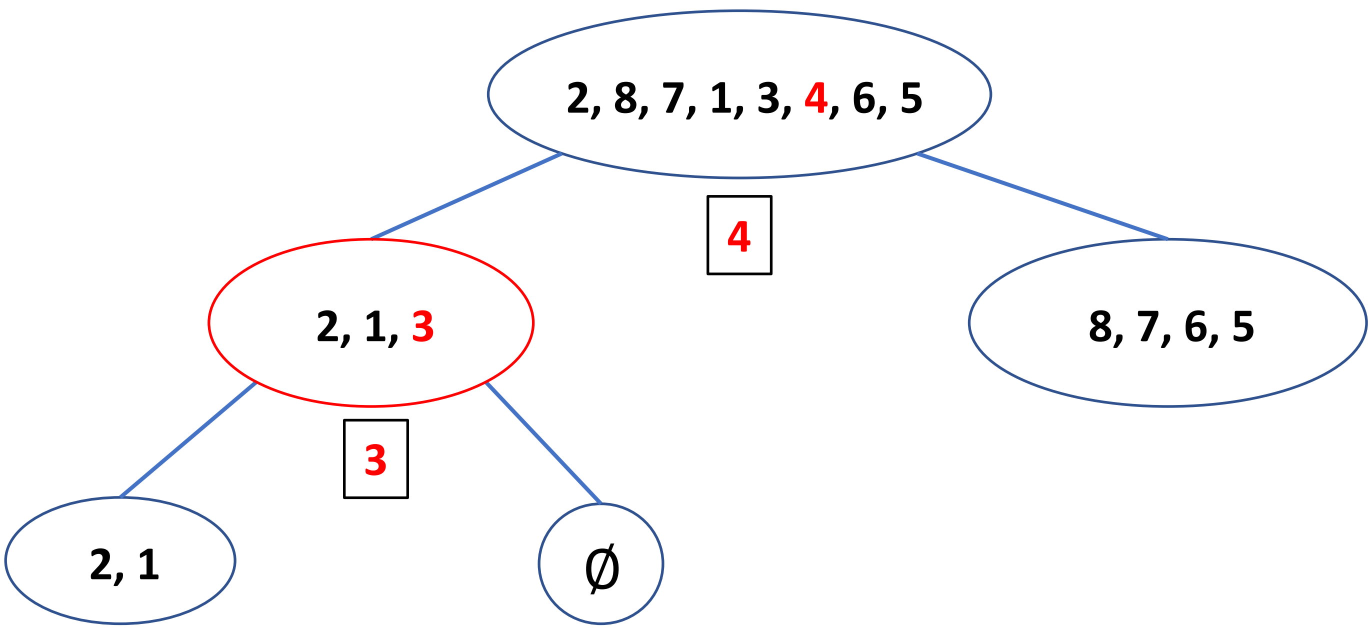
Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



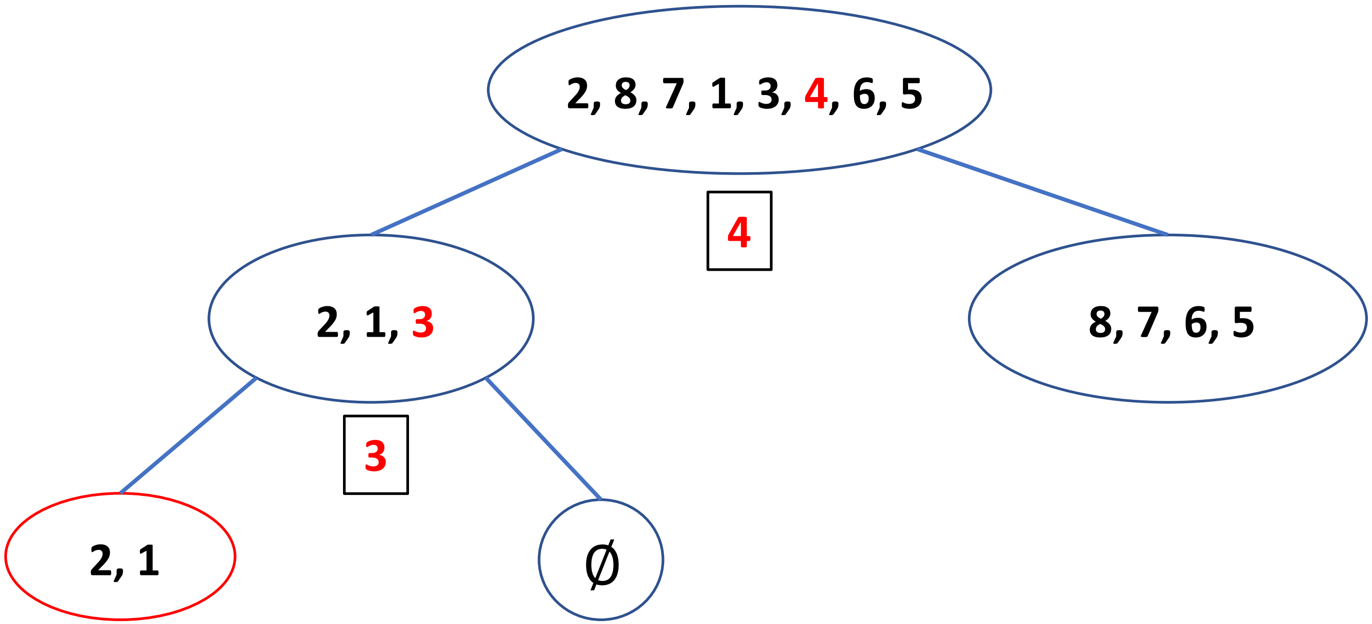
Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



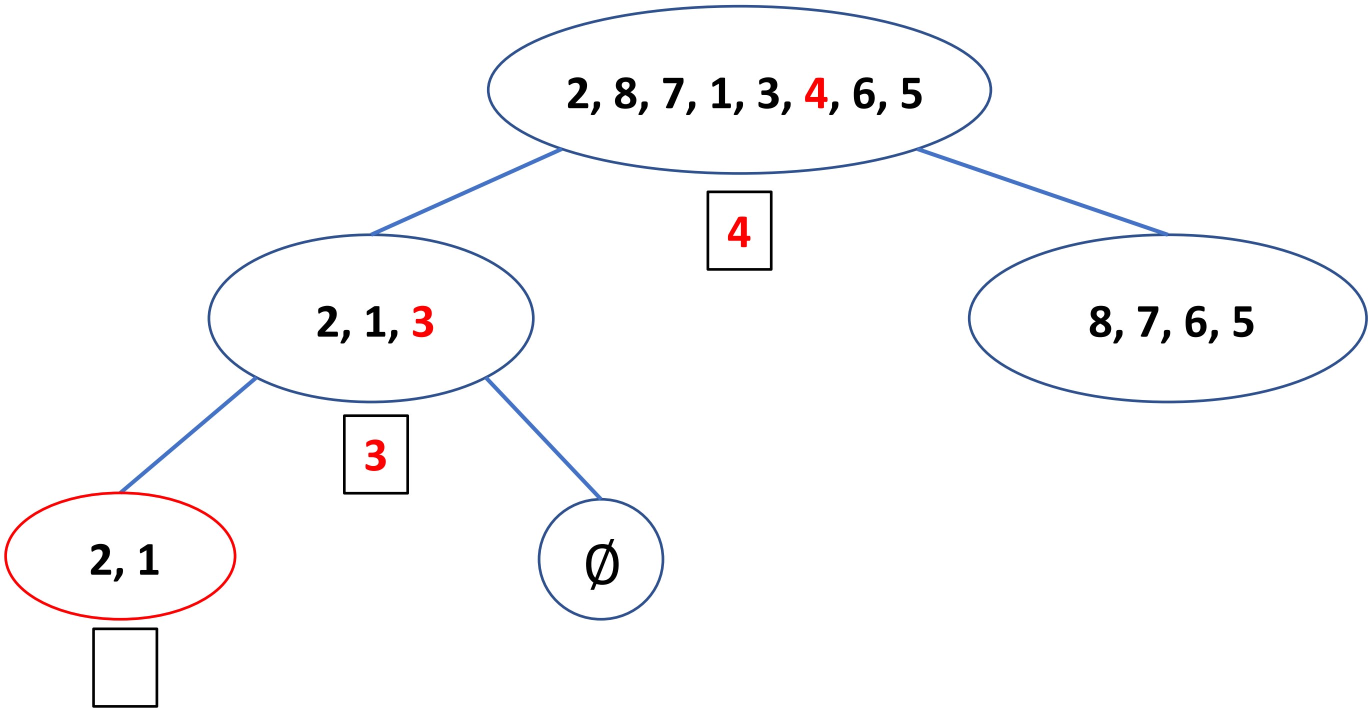
Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



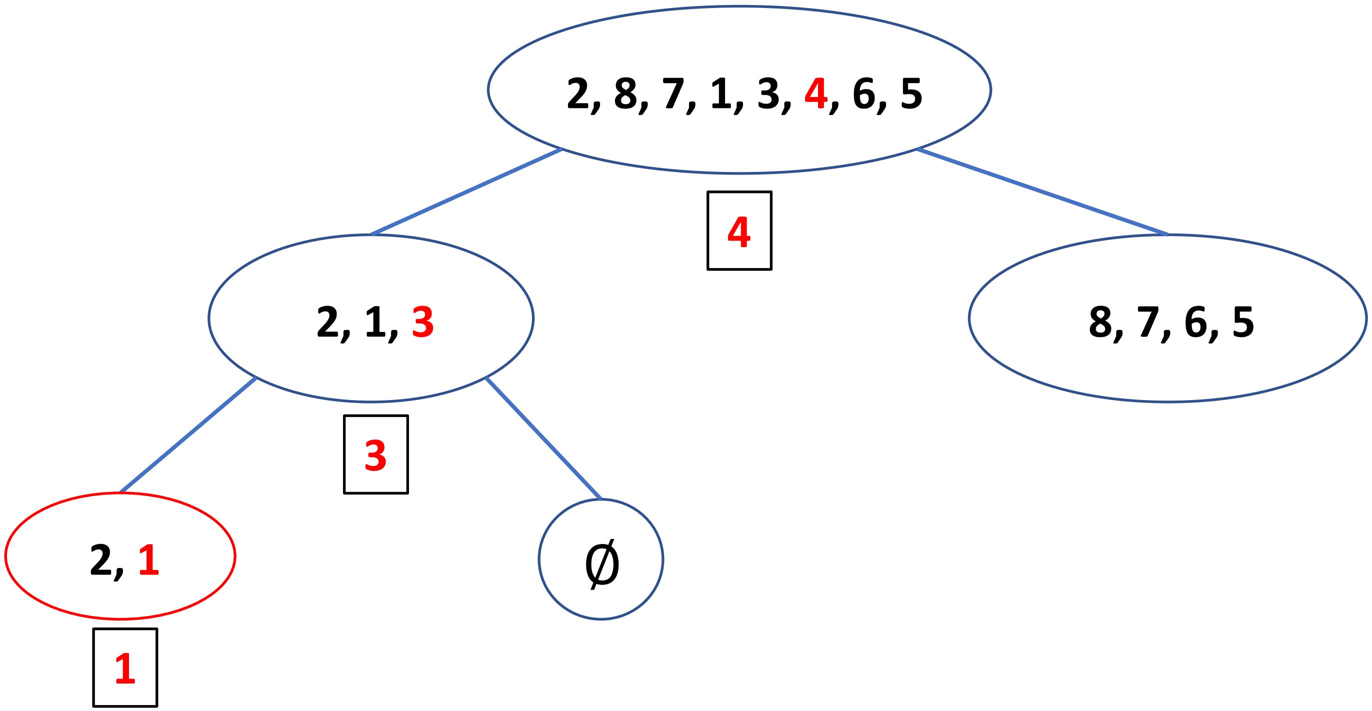
Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



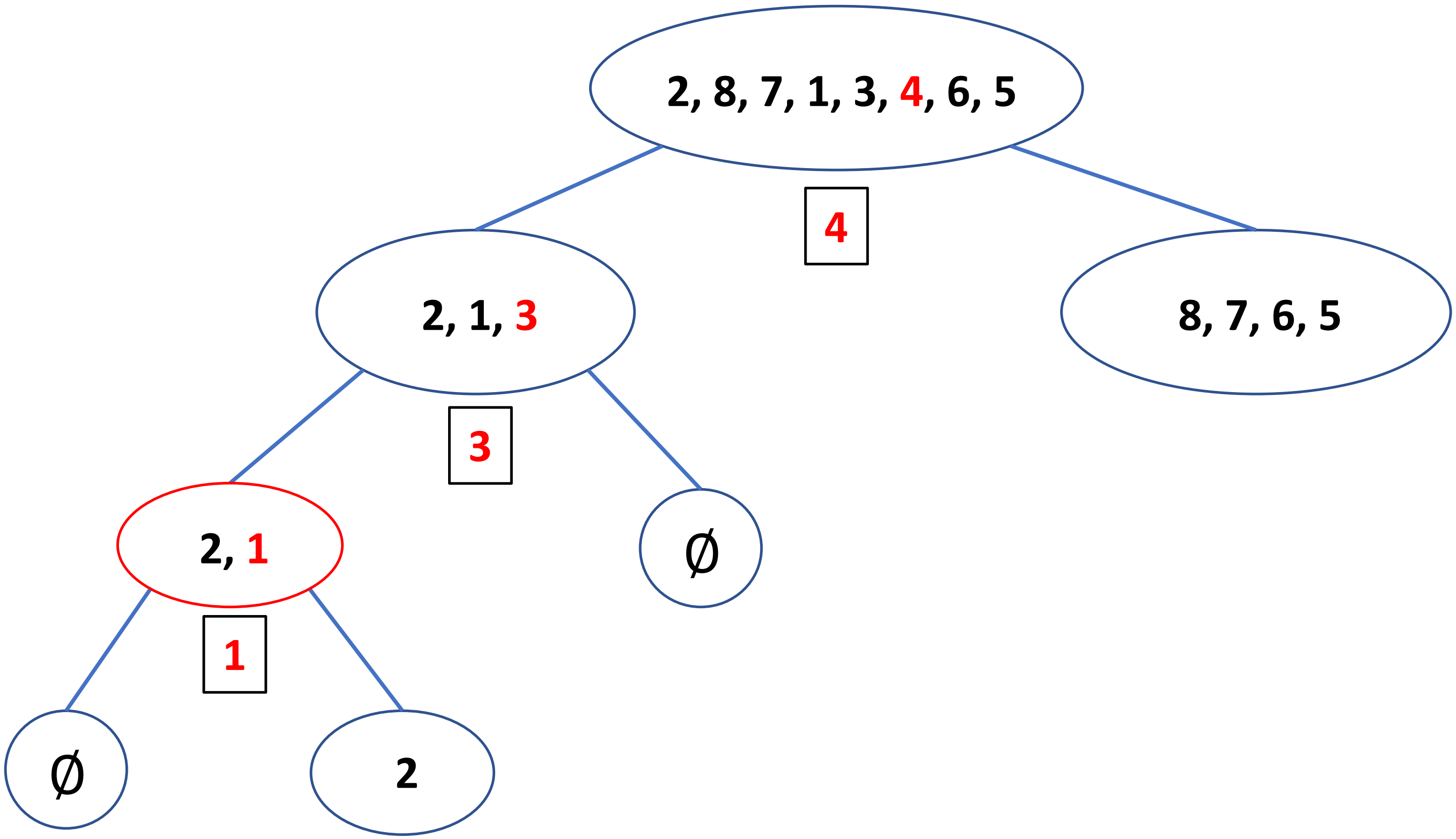
Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



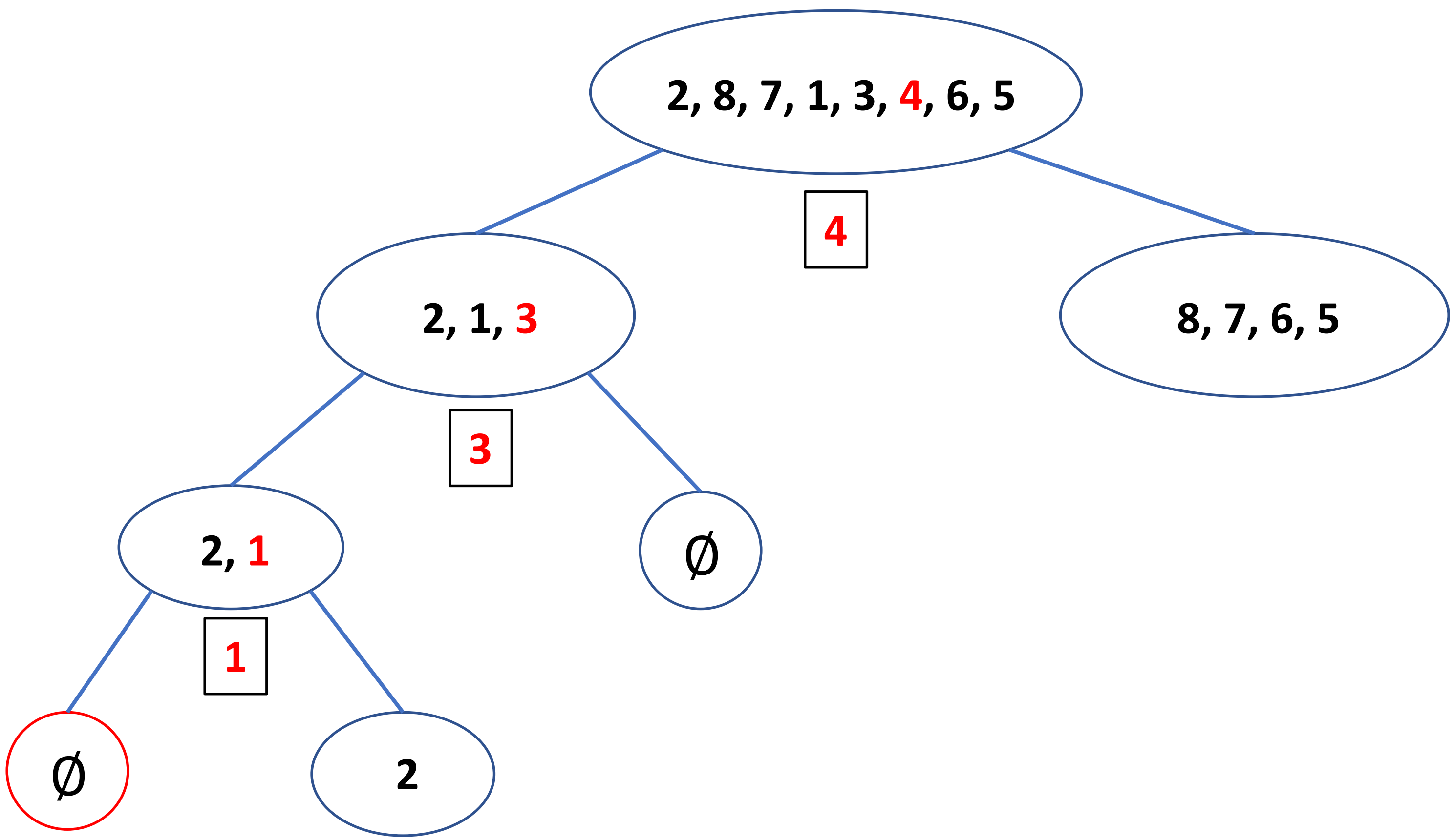
Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



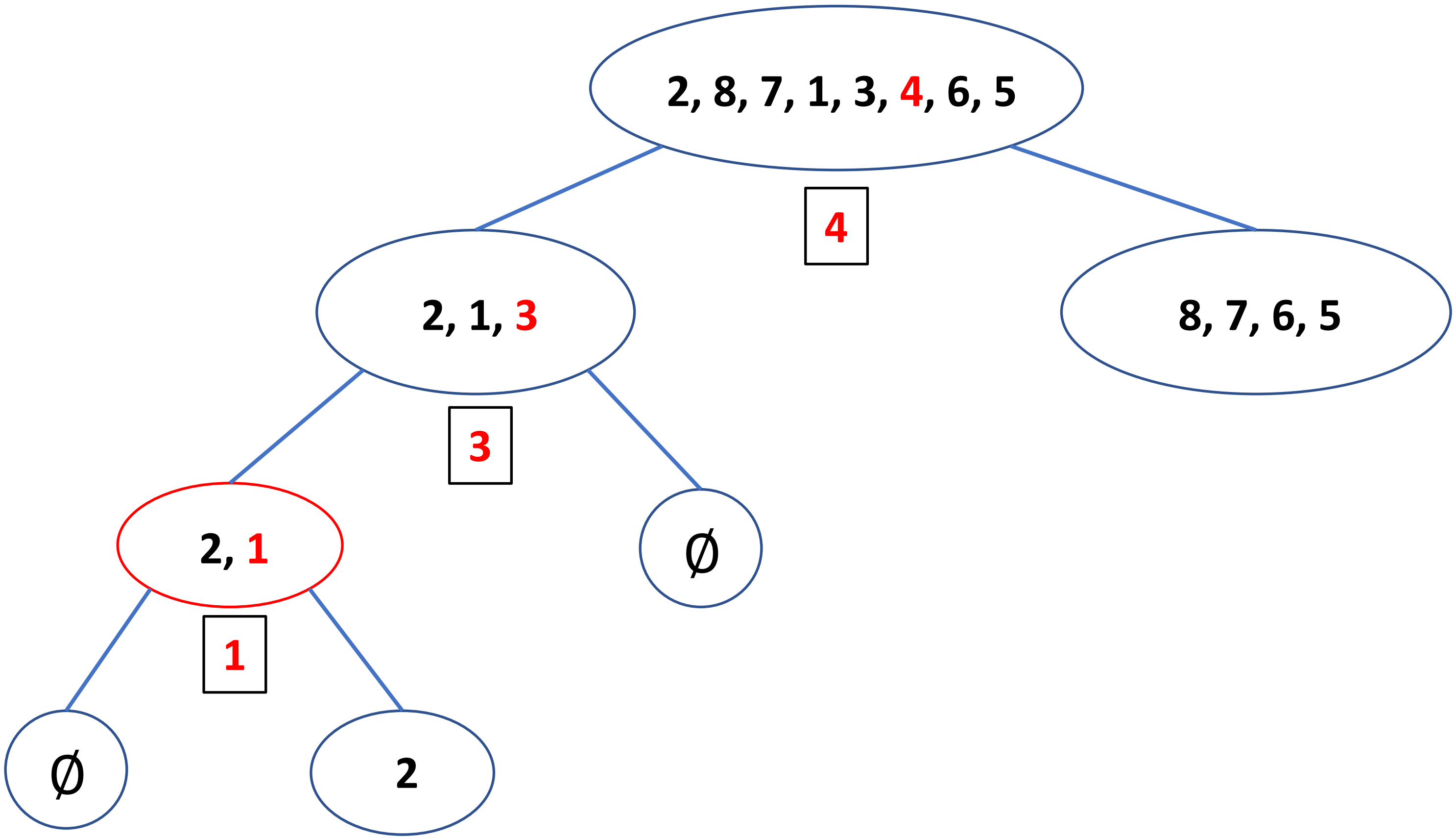
Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}

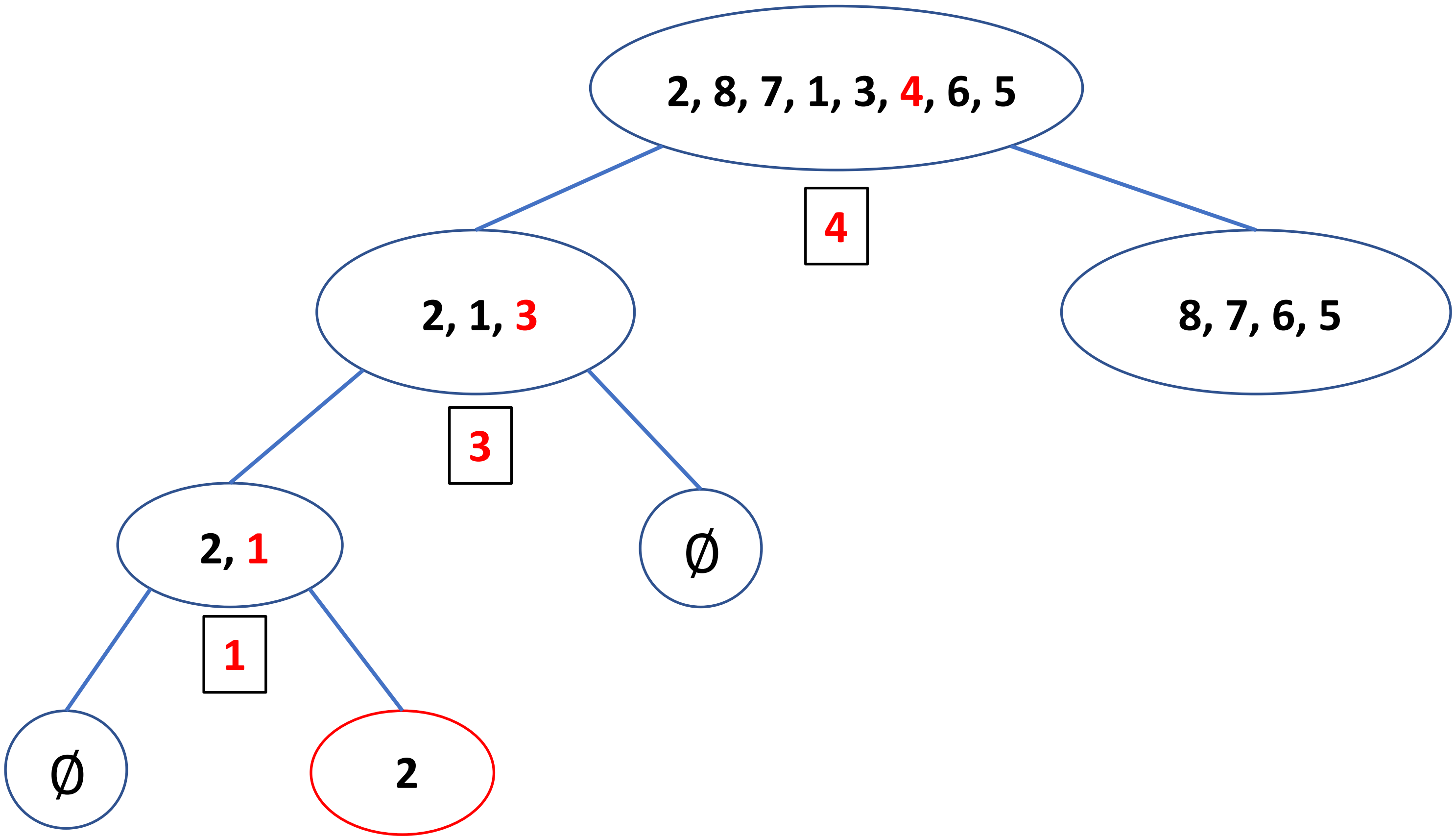


Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



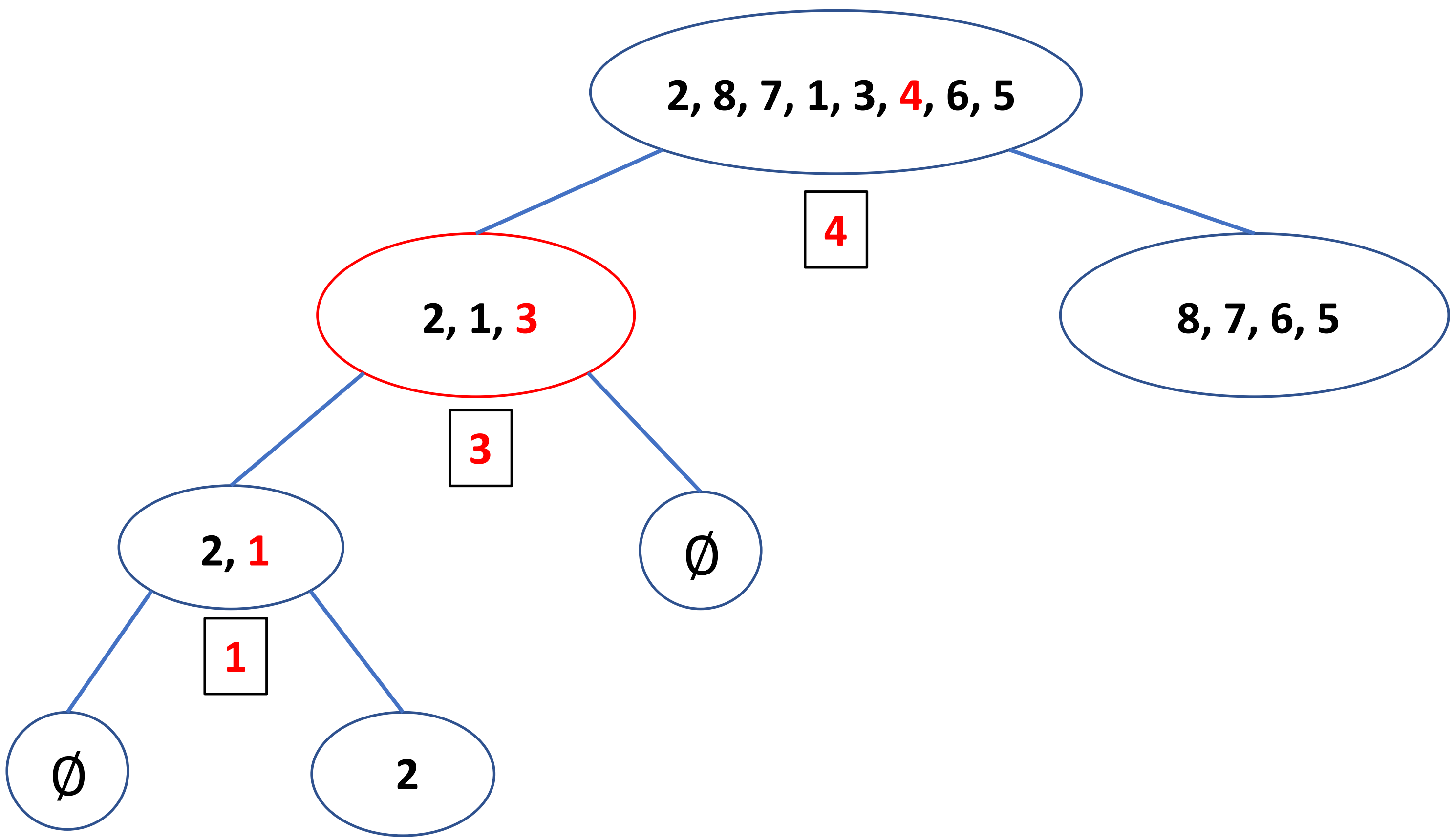
1

Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



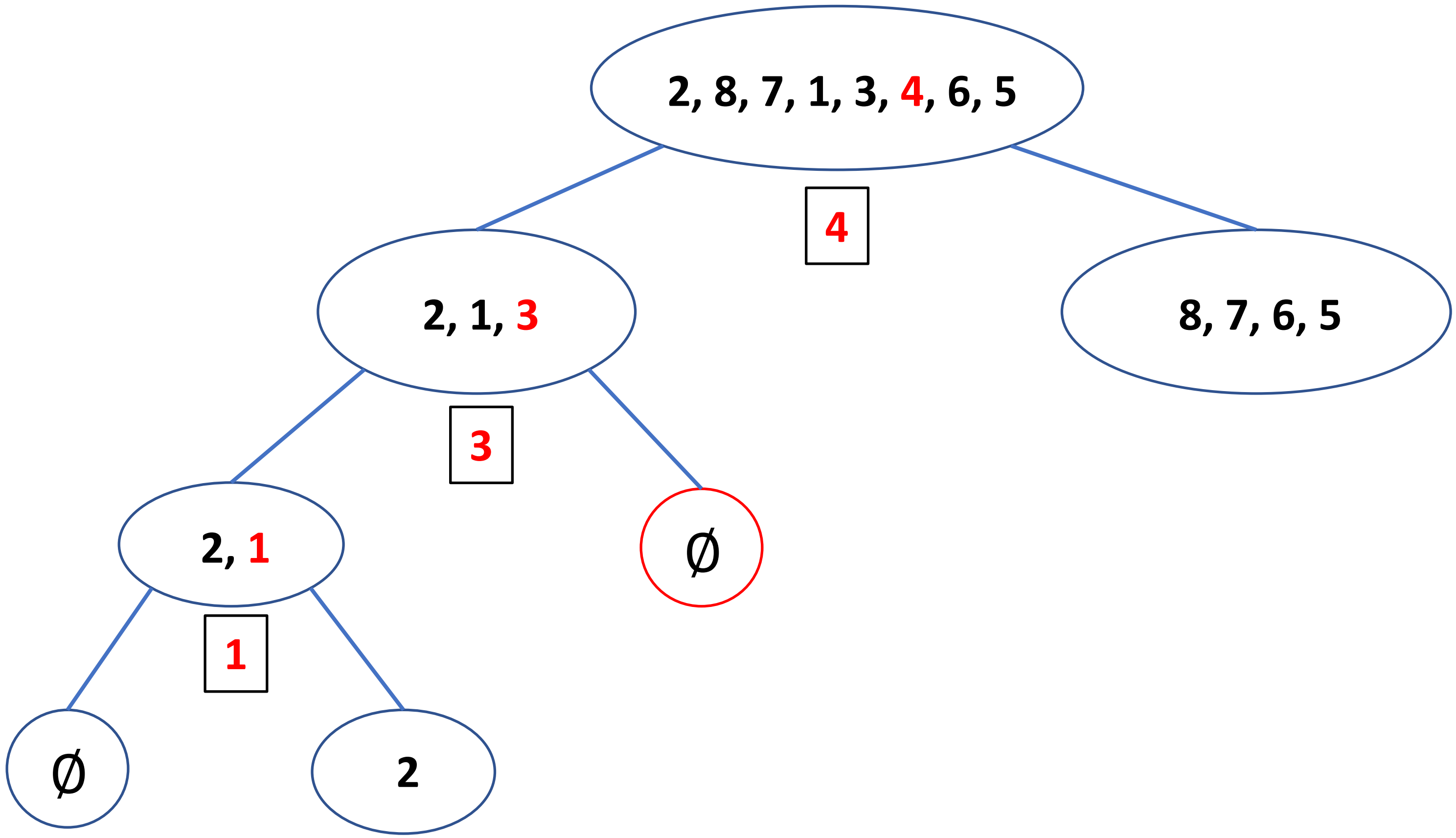
1, 2

Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



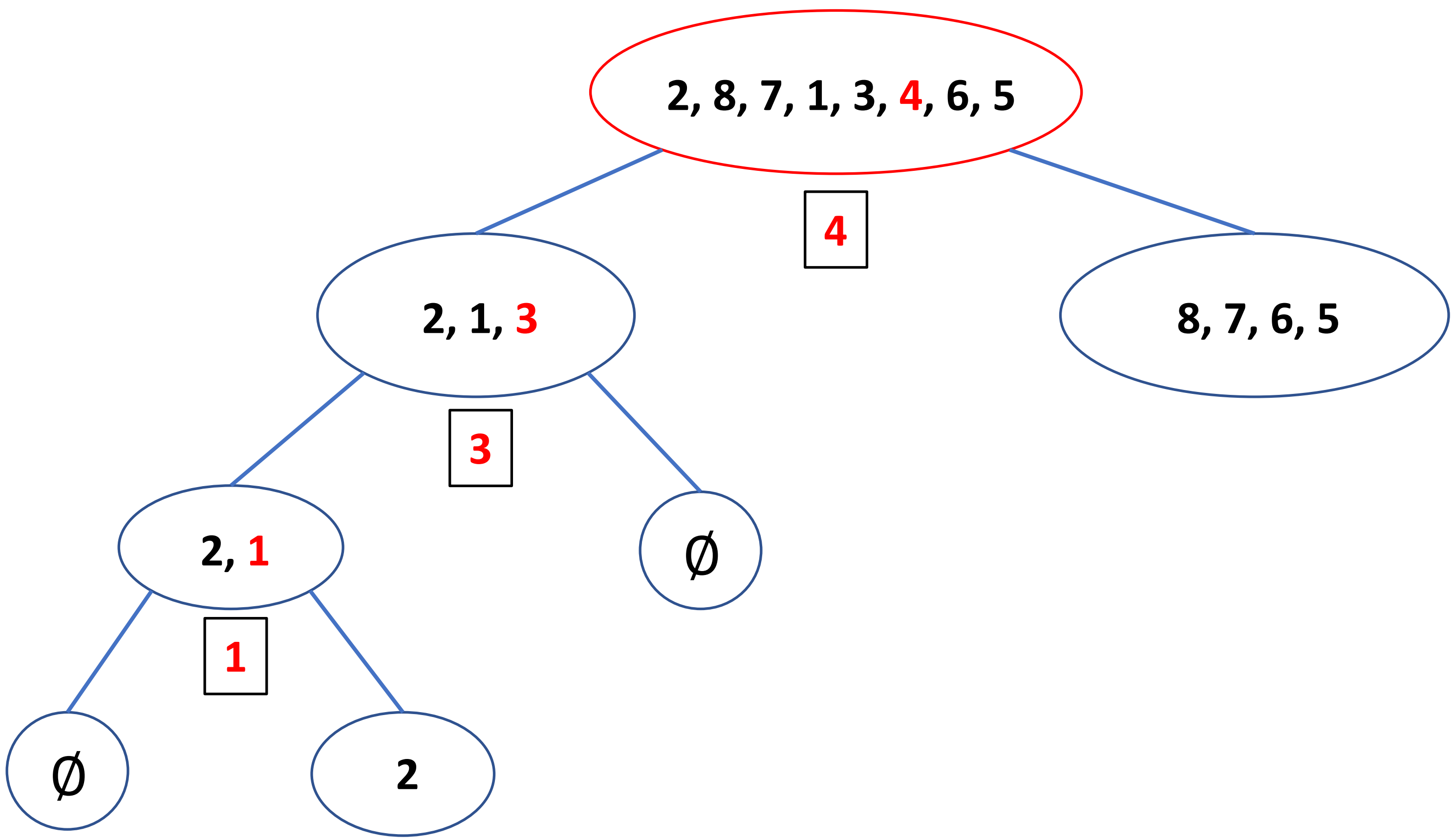
1, 2, 3

Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



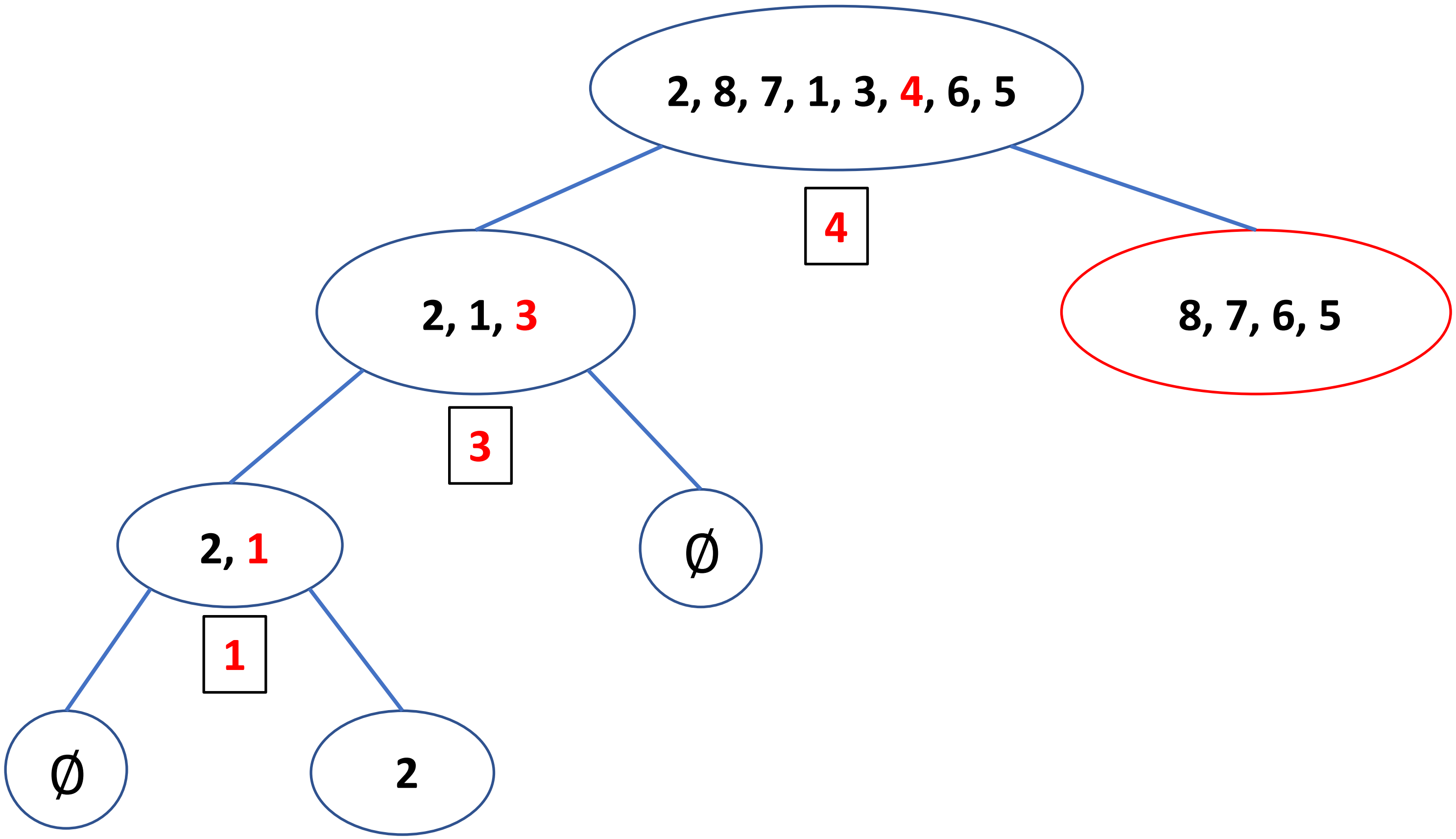
1, 2, 3

Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



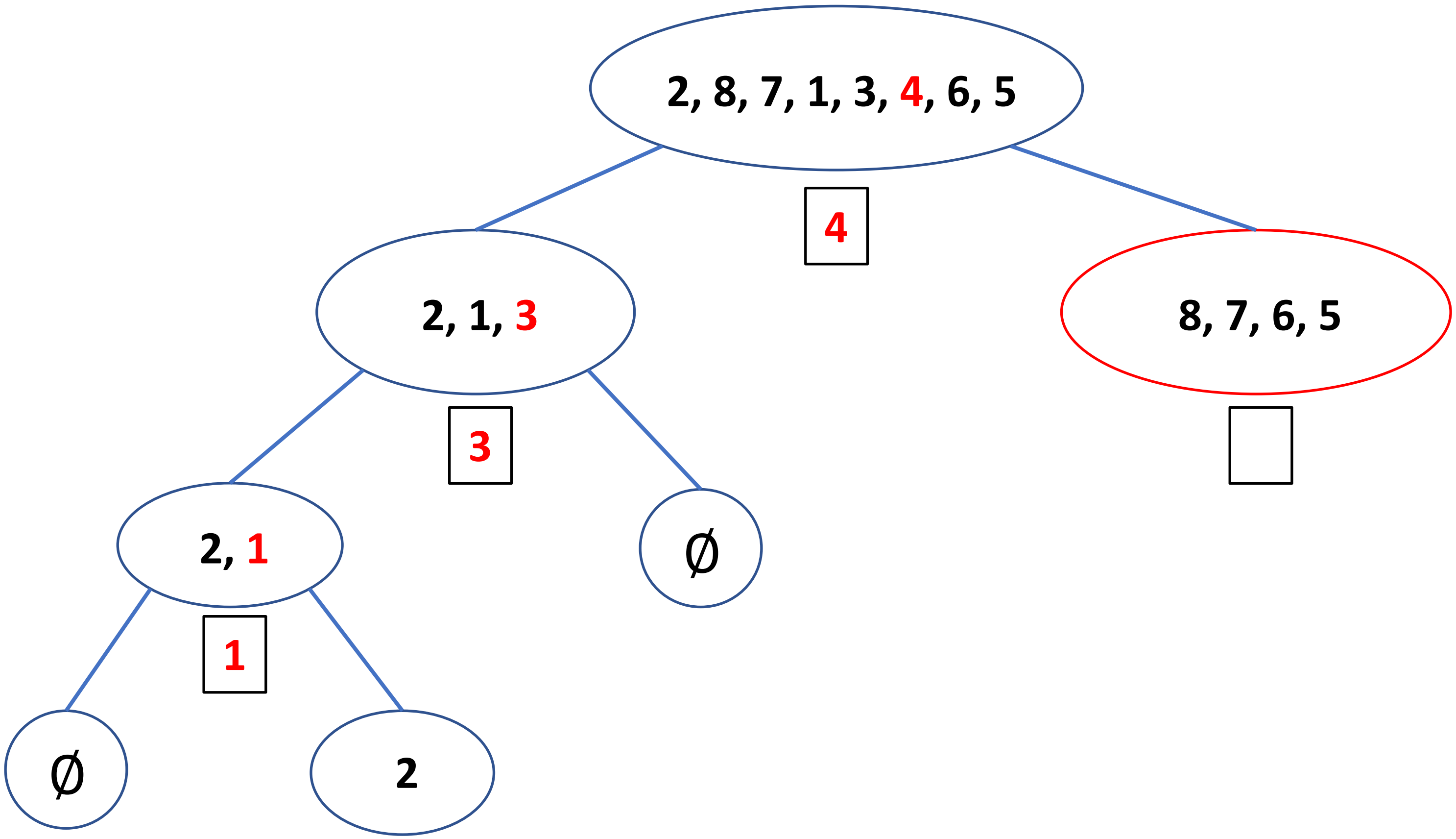
1, 2, 3, 4

Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



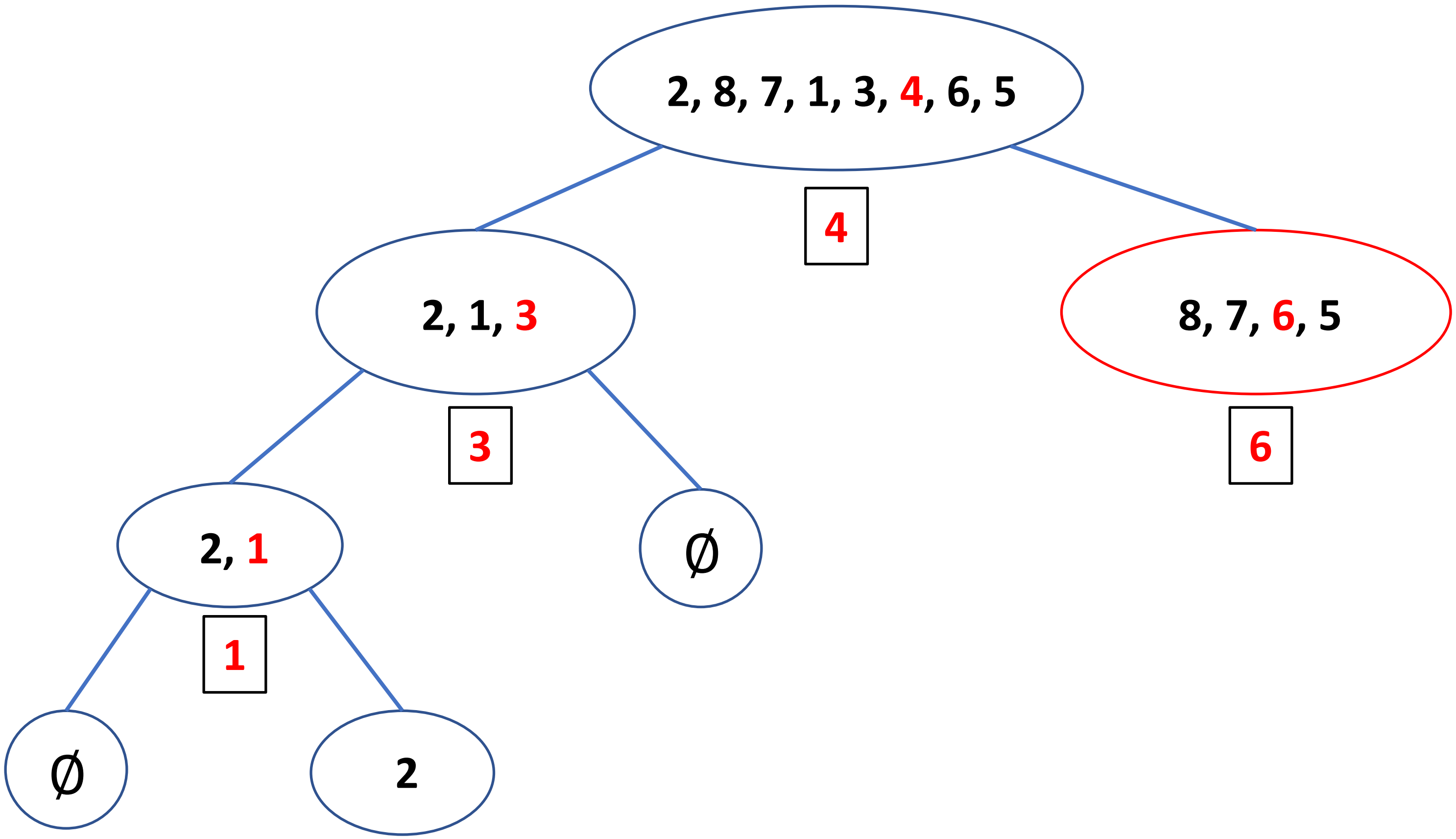
1, 2, 3, 4

Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



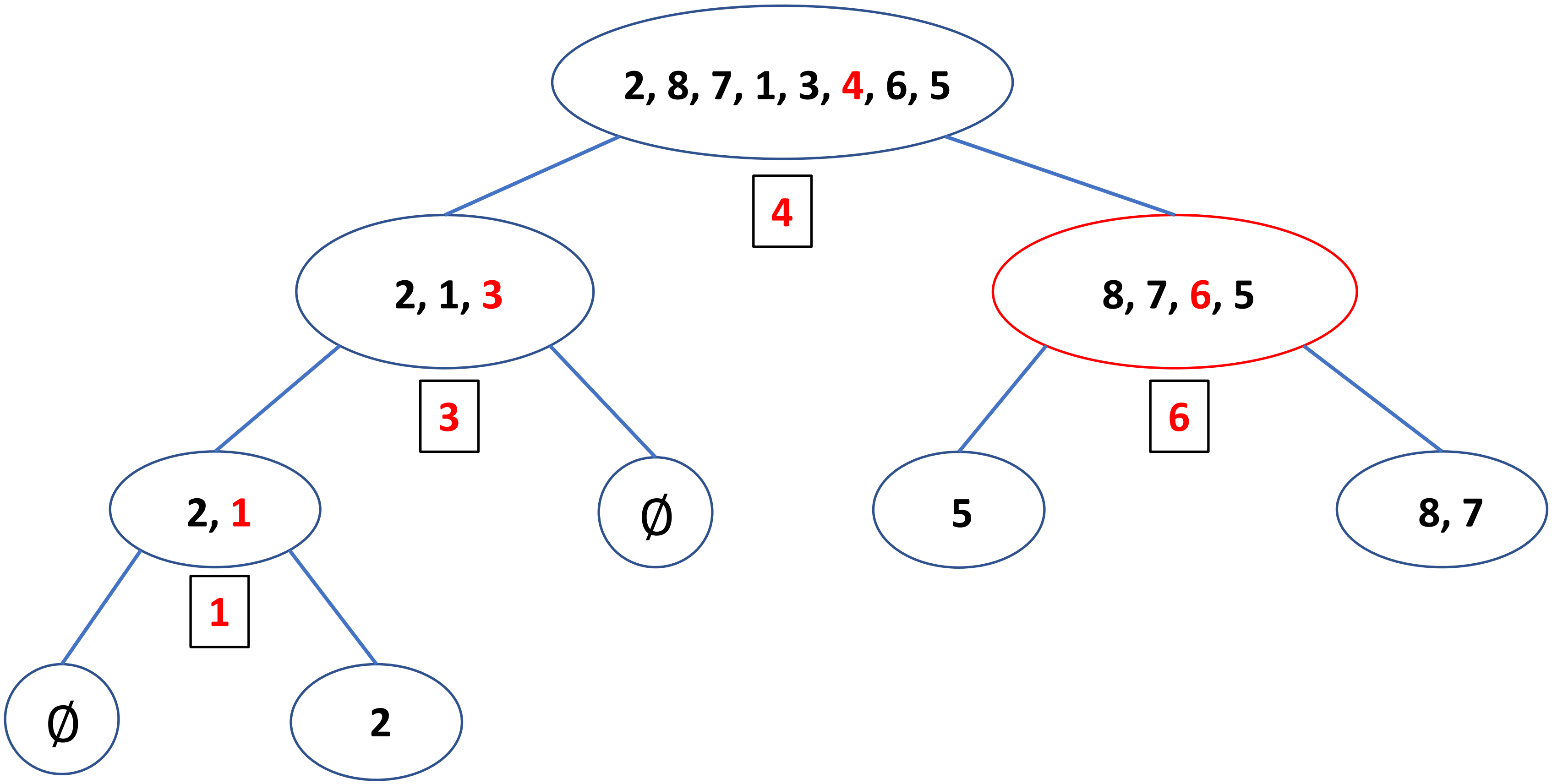
1, 2, 3, 4

Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



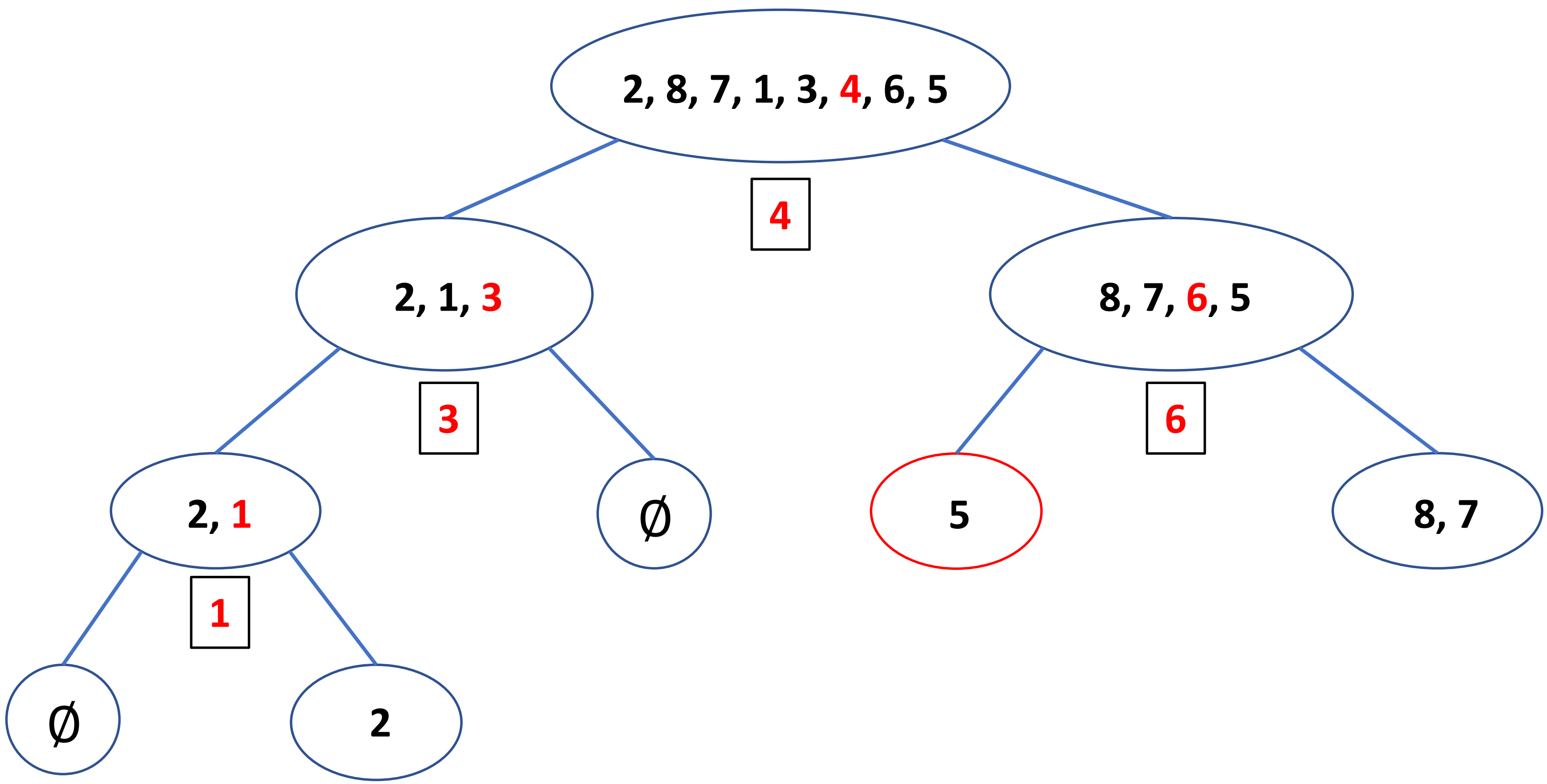
1, 2, 3, 4

Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



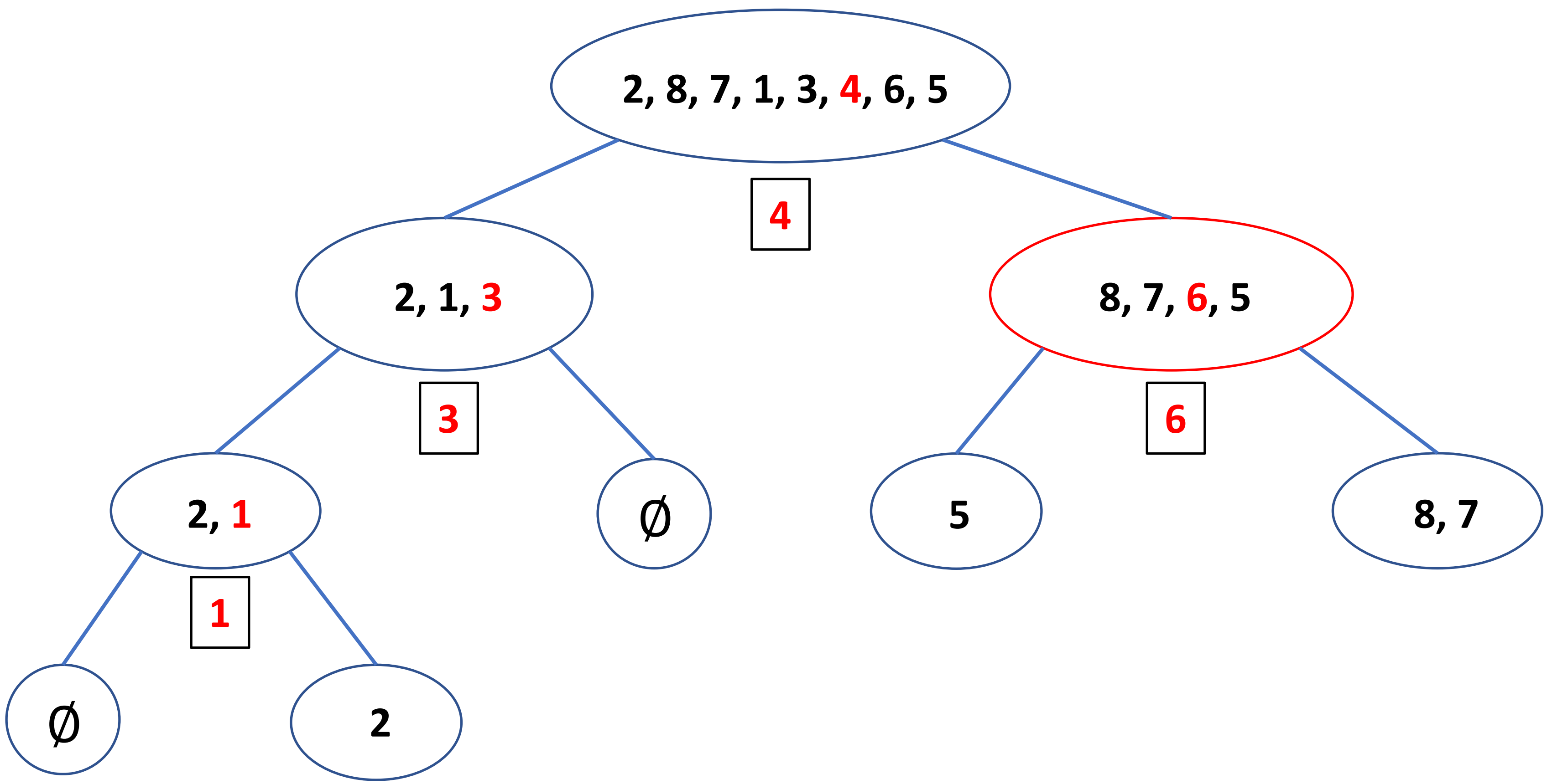
1, 2, 3, 4

Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



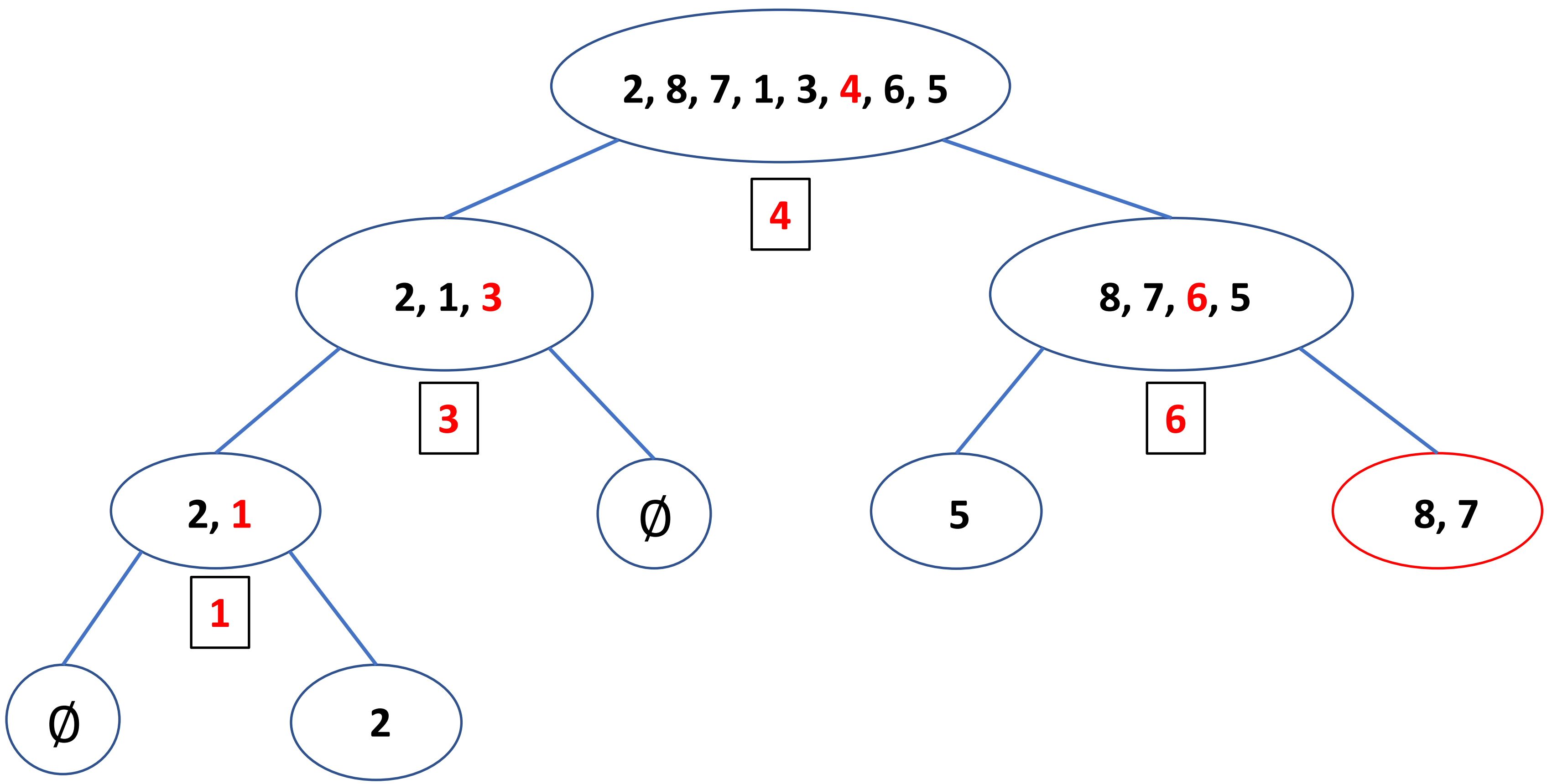
1, 2, 3, 4, 5

Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



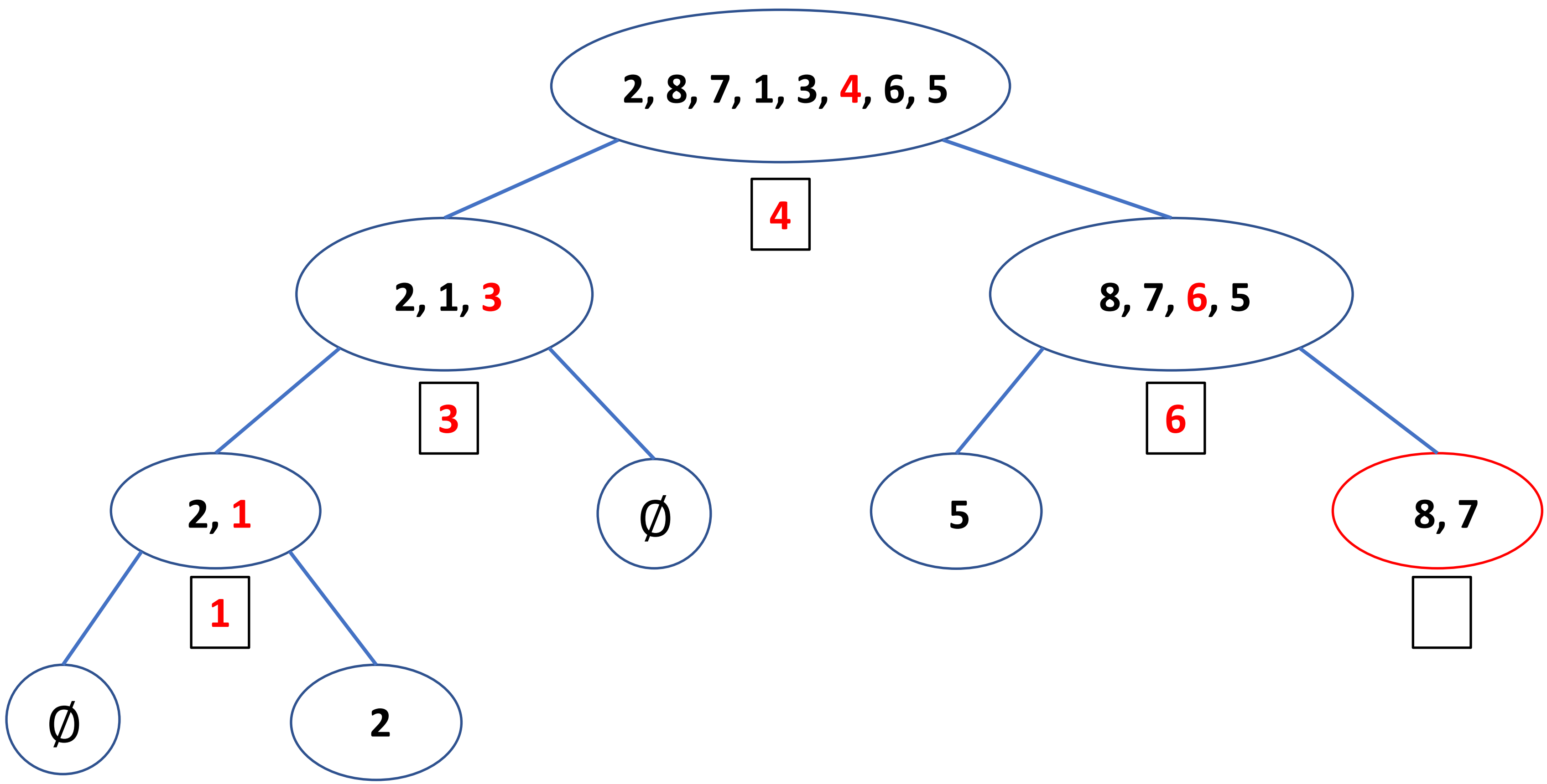
1, 2, 3, 4, 5, 6

Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



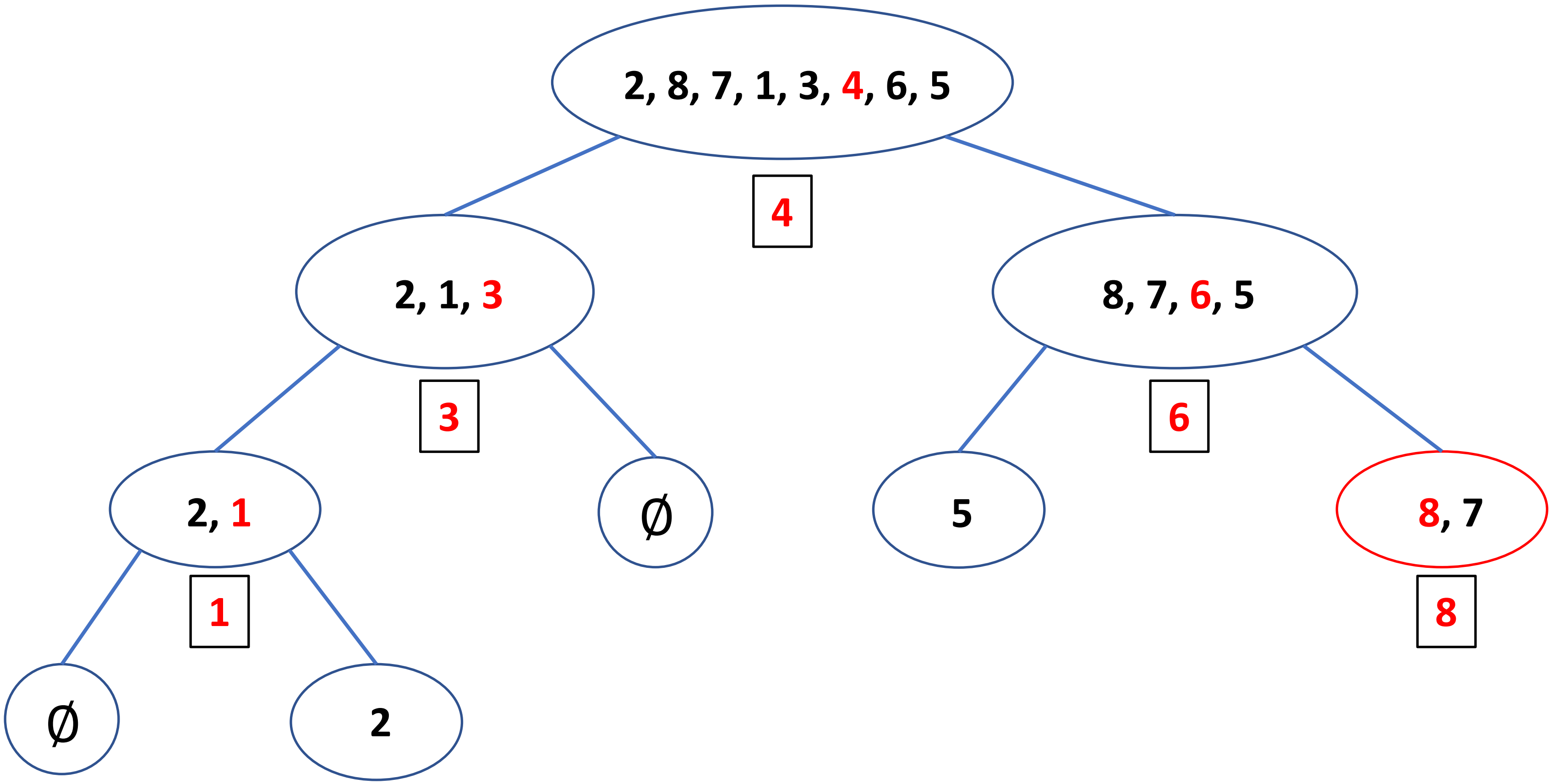
1, 2, 3, 4, 5, 6

Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



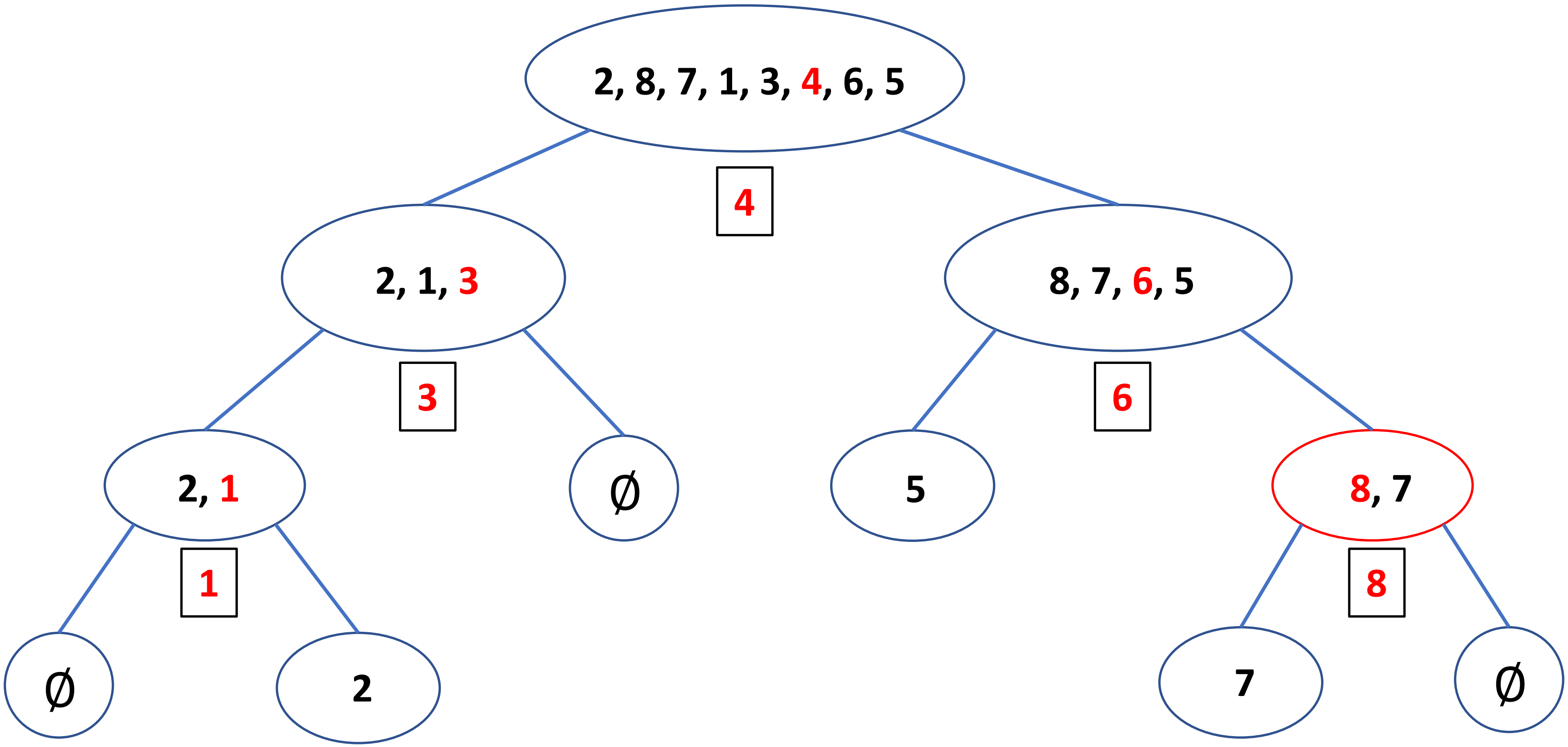
1, 2, 3, 4, 5, 6

Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



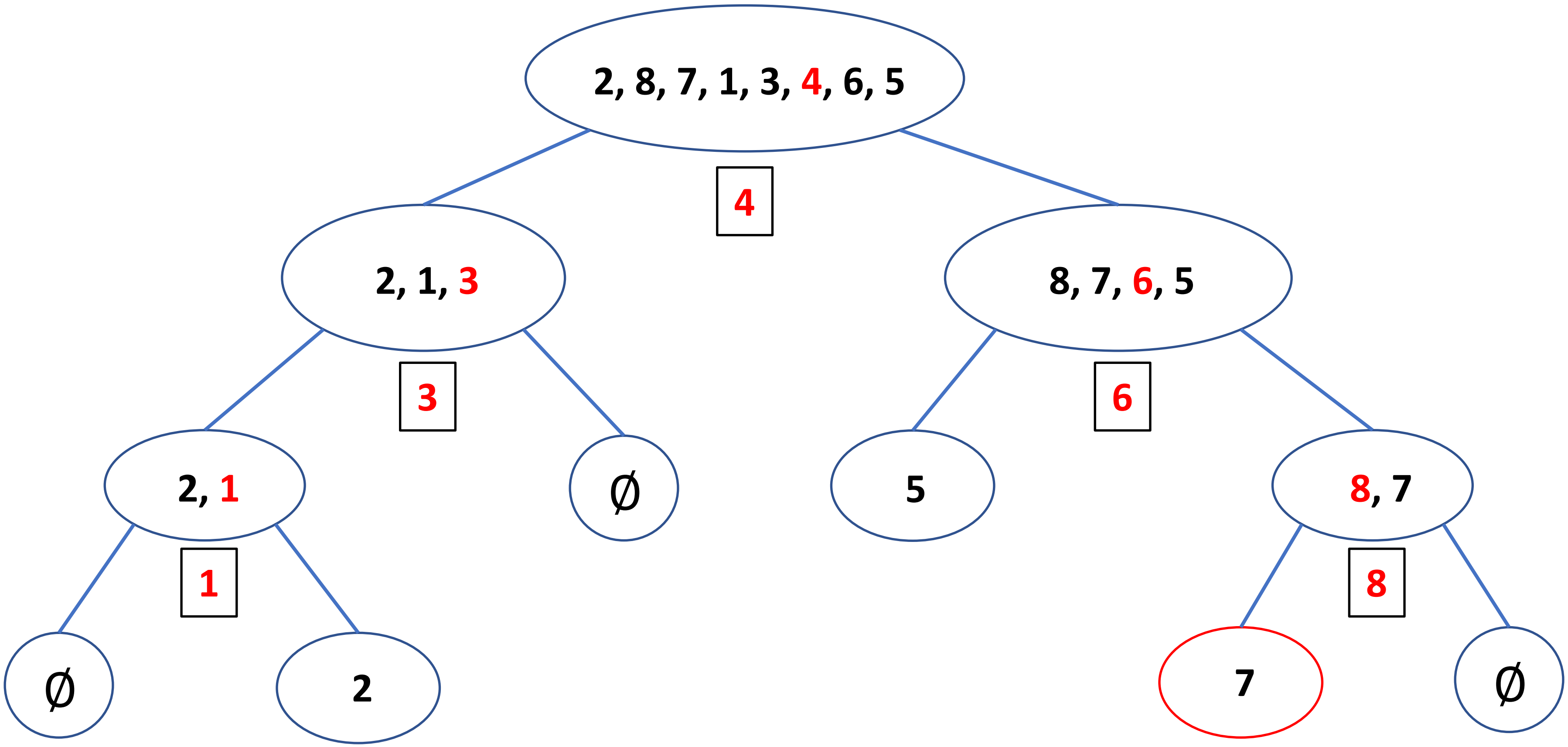
1, 2, 3, 4, 5, 6

Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



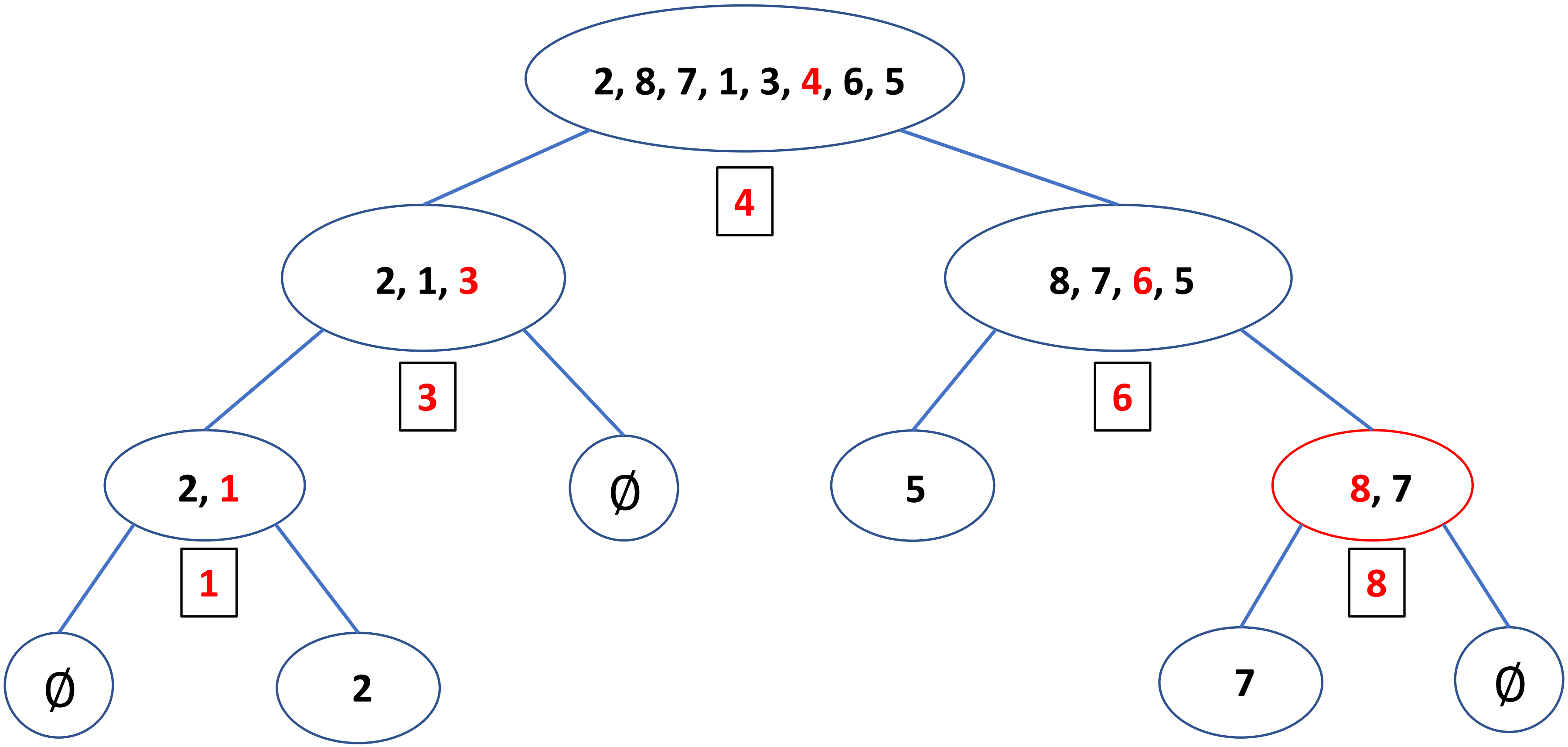
1, 2, 3, 4, 5, 6

Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



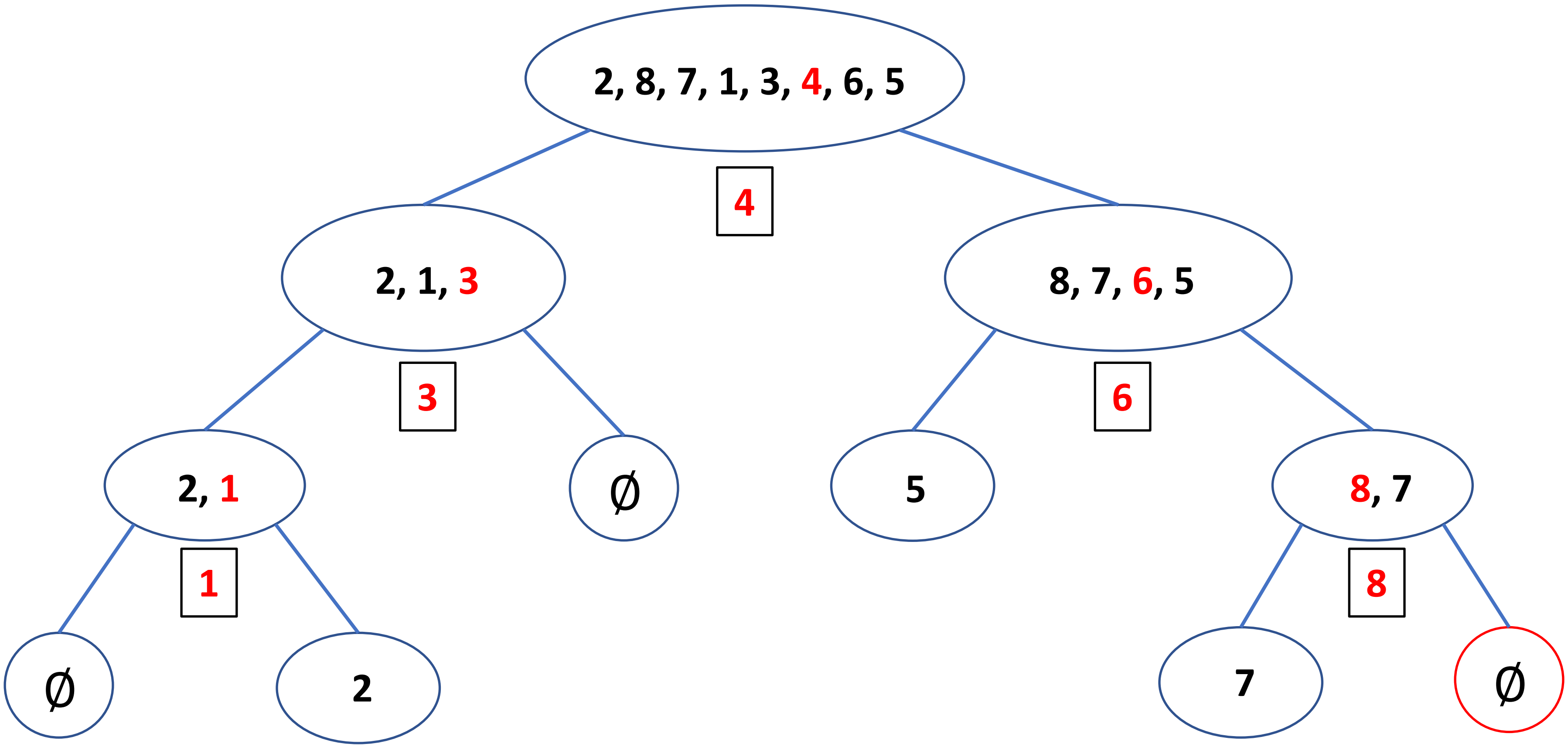
1, 2, 3, 4, 5, 6, 7

Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



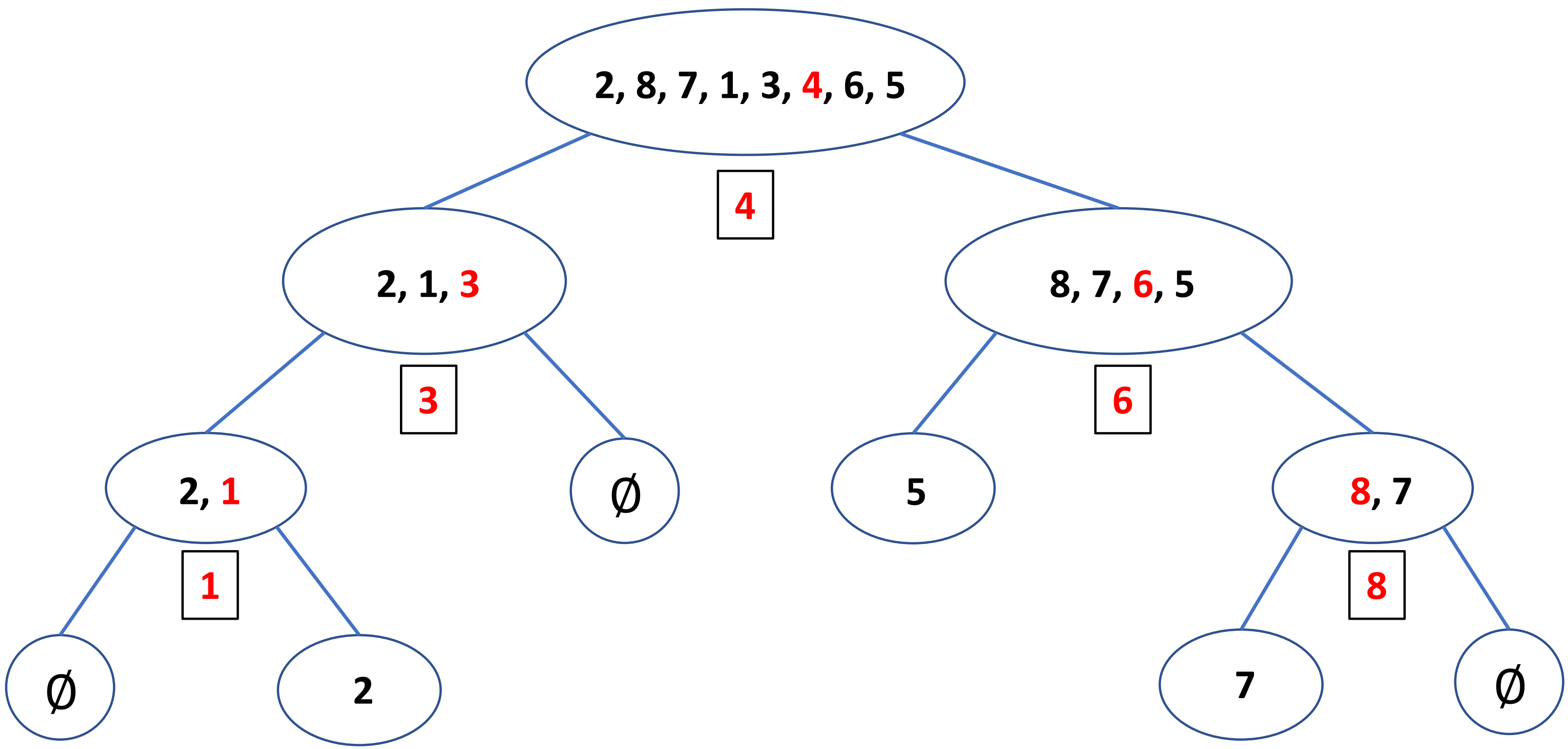
1, 2, 3, 4, 5, 6, 7, 8

Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



1, 2, 3, 4, 5, 6, 7, 8

Example execution of **RQS** on S = {**2, 8, 7, 1, 3, 4, 6, 5**}



1, 2, 3, 4, 5, 6, 7, 8

Basic Observations

In every $RQS(S)$ execution:

Basic Observations

In every $RQS(S)$ execution:

- Two keys are compared if and only if one of them is selected as a pivot.

Basic Observations

In every $RQS(S)$ execution:

- Two keys are compared if and only if one of them is selected as a pivot.
- Two keys are compared at most once.

Basic Observations

In every $RQS(S)$ execution:

- Two keys are compared if and only if one of them is selected as a pivot.
- Two keys are compared at most once.
- If two keys are "split apart" in different sets by a pivot (like 2 and 7 are split apart by pivot 4) then they are never compared.

Complexity Analysis of Randomized Quicksort

- Fix some input S with n *distinct* keys.

Complexity Analysis of Randomized Quicksort

- Fix some input S with n *distinct* keys.
- Run $RQS(S)$

Complexity Analysis of Randomized Quicksort

- Fix some input S with n *distinct* keys.
- Run $RQS(S)$
- Let $C =$ number of key comparisons done by $RQS(S)$

Complexity Analysis of Randomized Quicksort

- Fix some input S with n *distinct* keys.
- Run $RQS(S)$
- Let $C =$ number of key comparisons done by $RQS(S)$
- What is the worst-case value of C?

Complexity Analysis of Randomized Quicksort

- Fix some input S with n *distinct* keys.
- Run $RQS(S)$
- Let $C =$ number of key comparisons done by $RQS(S)$
- What is the worst-case value of C?

$$C = (n - 1)$$

Complexity Analysis of Randomized Quicksort

- Fix some input S with n *distinct* keys.
- Run $RQS(S)$
- Let $C =$ number of key comparisons done by $RQS(S)$
- What is the worst-case value of C?

$$C = (n-1) + (n-2)$$

Complexity Analysis of Randomized Quicksort

- Fix some input S with n *distinct* keys.
- Run $RQS(S)$
- Let C = number of key comparisons done by $RQS(S)$
- What is the worst-case value of C?

$$C = (n-1) + (n-2) + \ldots + 2 + 1$$

Complexity Analysis of Randomized Quicksort

- Fix some input S with n *distinct* keys.
- Run $RQS(S)$
- Let $C =$ number of key comparisons done by $RQS(S)$
- What is the worst-case value of C?

$$C = (n-1) + (n-2) + \ldots + 2 + 1 = \frac{n(n-1)}{2}$$

Complexity Analysis of Randomized Quicksort

- Fix some input S with n *distinct* keys.
- Run $RQS(S)$
- Let $C =$ number of key comparisons done by $RQS(S)$
- What is the worst-case value of C?

$$C = (n-1) + (n-2) + \ldots + 2 + 1 = \frac{n(n-1)}{2} \text{ i.e, } \Theta(n^2)$$

Complexity Analysis of Randomized Quicksort

- Fix some input S with n *distinct* keys.
- Run $RQS(S)$
- Let C = number of key comparisons done by $RQS(S)$
- What is the worst-case value of C?

$$C = (n-1) + (n-2) + \ldots + 2 + 1 = \frac{n(n-1)}{2} \text{ i.e, } \Theta(n^2)$$

- What is the expected ("average") value of C?

Complexity Analysis of Randomized Quicksort

- Fix some input S with n *distinct* keys.
- Run $RQS(S)$
- Let $C =$ number of key comparisons done by $RQS(S)$
- What is the worst-case value of C?

$$C = (n-1) + (n-2) + \ldots + 2 + 1 = \frac{n(n-1)}{2} \text{ i.e, } \Theta(n^2)$$

- What is the expected ("average") value of C?

  (over all the possible random pivot selections by $RQS(S)$)

Complexity Analysis of Randomized Quicksort

- Fix some input S with n *distinct* keys.
- Run $RQS(S)$
- Let C = number of key comparisons done by $RQS(S)$
- What is the worst-case value of C?

$$C = (n-1) + (n-2) + \ldots + 2 + 1 = \frac{n(n-1)}{2} \text{ i.e, } \Theta(n^2)$$

- What is the expected ("average") value of C?

  (over all the possible random pivot selections by $RQS(S)$)
- More precisely: what is $E(C)$?

Complexity Analysis of Randomized Quicksort

- Let $z_1 < z_2 < \ldots < z_i < \ldots < z_j < \ldots < z_n$ be the keys of S in ascending order

Complexity Analysis of Randomized Quicksort

- Let $z_1 < z_2 < \ldots < z_i < \ldots < z_j < \ldots < z_n$ be the keys of S in ascending order
- Run $RQS(S)$

Complexity Analysis of Randomized Quicksort

- Let $z_1 < z_2 < \ldots < z_i < \ldots < z_j < \ldots < z_n$ be the keys of S in ascending order
- Run $RQS(S)$
- Note that $RQS(S)$ compares z_i and z_j at most once (and only if, at some point, it selects z_i or z_j as a pivot)

Complexity Analysis of Randomized Quicksort

- Let $z_1 < z_2 < \ldots < z_i < \ldots < z_j < \ldots < z_n$ be the keys of S in ascending order
- Run $RQS(S)$
- Note that $RQS(S)$ compares z_i and z_j at most once (and only if, at some point, it selects z_i or z_j as a pivot)

$$\text{Let } c_{ij} = \begin{cases} 1 & RQS(S) \text{ compares } z_i \text{ and } z_j \\ & \end{cases}$$

Complexity Analysis of Randomized Quicksort

- Let $z_1 < z_2 < \ldots < z_i < \ldots < z_j < \ldots < z_n$ be the keys of S in ascending order
- Run $RQS(S)$
- Note that $RQS(S)$ compares z_i and z_j at most once (and only if, at some point, it selects z_i or z_j as a pivot)

$$\text{Let } c_{ij} = \begin{cases} 1 & RQS(S) \text{ compares } z_i \text{ and } z_j \\ 0 & \text{otherwise} \end{cases}$$

Complexity Analysis of Randomized Quicksort

- Let $z_1 < z_2 < \ldots < z_i < \ldots < z_j < \ldots < z_n$ be the keys of S in ascending order
- Run $RQS(S)$
- Note that $RQS(S)$ compares z_i and z_j at most once (and only if, at some point, it selects z_i or z_j as a pivot)

$$\text{Let } c_{ij} = \begin{cases} 1 & RQS(S) \text{ compares } z_i \text{ and } z_j \\ 0 & \text{otherwise} \end{cases}$$

This is an indicator random variable

Complexity Analysis of Randomized Quicksort

- Let $z_1 < z_2 < \ldots < z_i < \ldots < z_j < \ldots < z_n$ be the keys of S in ascending order
- Run $RQS(S)$
- Note that $RQS(S)$ compares z_i and z_j at most once (and only if, at some point, it selects z_i or z_j as a pivot)

$$\text{Let } c_{ij} = \begin{cases} 1 & RQS(S) \text{ compares } z_i \text{ and } z_j \\ 0 & \text{otherwise} \end{cases}$$

This is an indicator random variable

- The total number of key comparisons done by $RQS(S)$ is:

Complexity Analysis of Randomized Quicksort

- Let $z_1 < z_2 < \ldots < z_i < \ldots < z_j < \ldots < z_n$ be the keys of S in ascending order
- Run $RQS(S)$
- Note that $RQS(S)$ compares z_i and z_j at most once (and only if, at some point, it selects z_i or z_j as a pivot)

$$\text{Let } c_{ij} = \begin{cases} 1 & RQS(S) \text{ compares } z_i \text{ and } z_j \\ 0 & \text{otherwise} \end{cases}$$

This is an indicator random variable

- The total number of key comparisons done by $RQS(S)$ is:

$$C = \sum_{1 \le i < j \le n} c_{ij}$$

Complexity Analysis of Randomized Quicksort

- Let $z_1 < z_2 < \ldots < z_i < \ldots < z_j < \ldots < z_n$ be the keys of S in ascending order
- Run $RQS(S)$
- Note that $RQS(S)$ compares z_i and z_j at most once (and only if, at some point, it selects z_i or z_j as a pivot)

$$\text{Let } c_{ij} = \begin{cases} 1 & RQS(S) \text{ compares } z_i \text{ and } z_j \\ 0 & \text{otherwise} \end{cases}$$

This is an indicator random variable

- The total number of key comparisons done by $RQS(S)$ is:

$$C = \sum_{1 \leq i < j \leq n} c_{ij}$$

- We want to compute $E(C)$

Complexity Analysis of Randomized Quicksort

Number of comparisons done by $RQS(S)$: $C = \sum_{1 \leq i < j \leq n} c_{ij}$

Complexity Analysis of Randomized Quicksort

Number of comparisons done by $RQS(S)$: $C = \sum_{1 \leq i < j \leq n} c_{ij}$

To compute $E(C)$, we first compute $E(c_{ij})$:

$$E(c_{ij}) \quad =$$

Complexity Analysis of Randomized Quicksort

Number of comparisons done by $RQS(S)$: $C = \sum_{1 \le i < j \le n} c_{ij}$

To compute $E(C)$, we first compute $E(c_{ij})$:

$$E(c_{ij}) = 1 \cdot \Pr[c_{ij} = 1]$$

Complexity Analysis of Randomized Quicksort

Number of comparisons done by $RQS(S)$: $C = \sum_{1 \le i < j \le n} c_{ij}$

To compute $E(C)$, we first compute $E(c_{ij})$:

$$E(c_{ij}) = 1 \cdot \Pr[c_{ij} = 1] + 0 \cdot \Pr[c_{ij} = 0]$$

Complexity Analysis of Randomized Quicksort

Number of comparisons done by $RQS(S)$: $C = \sum_{1 \le i < j \le n} c_{ij}$

To compute $E(C)$, we first compute $E(c_{ij})$:

$$\begin{aligned} E(c_{ij}) &= 1 \cdot \Pr[c_{ij} = 1] + 0 \cdot \Pr[c_{ij} = 0] \\ &= \Pr[c_{ij} = 1] \end{aligned}$$

Complexity Analysis of Randomized Quicksort

Number of comparisons done by $RQS(S)$: $C = \sum_{1 \le i < j \le n} c_{ij}$

To compute $E(C)$, we first compute $E(c_{ij})$:

$$\begin{aligned} E(c_{ij}) &= 1 \cdot \Pr[c_{ij} = 1] + 0 \cdot \Pr[c_{ij} = 0] \\ &= \Pr[c_{ij} = 1] \\ &= \Pr[z_i \text{ and } z_j \text{ are compared}] \end{aligned}$$

Complexity Analysis of Randomized Quicksort

Number of comparisons done by $RQS(S)$: $C = \sum_{1 \le i < j \le n} c_{ij}$

To compute $E(C)$, we first compute $E(c_{ij})$:

$$\begin{aligned} E(c_{ij}) &= 1 \cdot \Pr[c_{ij} = 1] + 0 \cdot \Pr[c_{ij} = 0] \\ &= \Pr[c_{ij} = 1] \\ &= \Pr[z_i \text{ and } z_j \text{ are compared}] \end{aligned}$$

$$E(C) = E\Big(\sum_{1 \le i < j \le n} c_{ij}\Big)$$

Complexity Analysis of Randomized Quicksort

Number of comparisons done by $RQS(S)$: $C = \sum_{1 \leq i < j \leq n} c_{ij}$

To compute $E(C)$, we first compute $E(c_{ij})$:

$$\begin{aligned} E(c_{ij}) &= 1 \cdot \Pr[c_{ij} = 1] + 0 \cdot \Pr[c_{ij} = 0] \\ &= \Pr[c_{ij} = 1] \\ &= \Pr[z_i \text{ and } z_j \text{ are compared}] \end{aligned}$$

$$\begin{aligned} E(C) &= E\Big(\sum_{1 \leq i < j \leq n} c_{ij}\Big) \\ &= \sum_{1 \leq i < j \leq n} E(c_{ij}) \qquad \text{by linearity of expectations} \end{aligned}$$

Complexity Analysis of Randomized Quicksort

Number of comparisons done by $RQS(S)$: $C = \sum_{1 \le i < j \le n} c_{ij}$

To compute $E(C)$, we first compute $E(c_{ij})$:

$$\begin{aligned} E(c_{ij}) &= 1 \cdot \Pr[c_{ij} = 1] + 0 \cdot \Pr[c_{ij} = 0] \\ &= \Pr[c_{ij} = 1] \\ &= \Pr[z_i \text{ and } z_j \text{ are compared}] \end{aligned}$$

$$\begin{aligned} E(C) &= E\Big(\sum_{1 \le i < j \le n} c_{ij}\Big) \\ &= \sum_{1 \le i < j \le n} E(c_{ij}) \qquad \text{by linearity of expectations} \\ &= \sum_{1 \le i < j \le n} \Pr[z_i \text{ and } z_j \text{ are compared}] \end{aligned}$$

Complexity Analysis of Randomized Quicksort

- We just proved:

  Lemma: $E(C) = \sum_{1 \le i < j \le n} \Pr[z_i \text{ and } z_j \text{ are compared}]$

Complexity Analysis of Randomized Quicksort

- We just proved:

  Lemma: $E(C) = \sum_{1 \le i < j \le n} \Pr[z_i \text{ and } z_j \text{ are compared}]$

- We will later prove:

  Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Complexity Analysis of Randomized Quicksort

- We just proved:

  Lemma: $E(C) = \sum_{1 \le i < j \le n} \Pr[z_i \text{ and } z_j \text{ are compared}]$

- We will later prove:

  Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

- By combining these two lemmas:

  $E(C) = \sum_{1 \le i < j \le n} \frac{2}{j-i+1}$

Complexity Analysis of Randomized Quicksort

- We just proved:

  Lemma: $E(C) = \sum_{1 \le i < j \le n} \Pr[z_i \text{ and } z_j \text{ are compared}]$

- We will later prove:

  Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

- By combining these two lemmas:

  $$E(C) = \sum_{1 \le i < j \le n} \frac{2}{j-i+1} = \ldots = 2n\left(1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}\right)$$

Complexity Analysis of Randomized Quicksort

- We just proved:

  Lemma: $E(C) = \sum_{1 \le i < j \le n} \Pr[z_i \text{ and } z_j \text{ are compared}]$

- We will later prove:

  Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

- By combining these two lemmas:

  $$E(C) = \sum_{1 \le i < j \le n} \frac{2}{j-i+1} = \ldots = 2n(1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n})$$

- It is known that $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$ is $O(\log n)$.

Complexity Analysis of Randomized Quicksort

- We just proved:

  Lemma: $E(C) = \sum_{1 \le i < j \le n} \Pr[z_i \text{ and } z_j \text{ are compared}]$

- We will later prove:

  Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

- By combining these two lemmas:

  $$E(C) = \sum_{1 \le i < j \le n} \frac{2}{j-i+1} = \ldots = 2n(1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n})$$

- It is known that $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$ is $O(\log n)$.

  [In fact, for $n \ge 2$, $H_n \le (\log_e n) + 1$.]

Complexity Analysis of Randomized Quicksort

- We just proved:

  Lemma: $E(C) = \sum_{1 \le i < j \le n} \Pr[z_i \text{ and } z_j \text{ are compared}]$

- We will later prove:

  Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

- By combining these two lemmas:

  $$E(C) = \sum_{1 \le i < j \le n} \frac{2}{j-i+1} = \ldots = 2n(1 + \tfrac{1}{2} + \tfrac{1}{3} + \ldots + \tfrac{1}{n})$$

- It is known that $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$ is $O(\log n)$.

  [In fact, for $n \ge 2$, $H_n \le (\log_e n) + 1$.]

- Therefore:

  Theorem: $E(C)$ is $O(n \log n)$

Complexity Analysis of Randomized Quicksort

It now remains to prove:

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Complexity Analysis of Randomized Quicksort

It now remains to prove:

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Some intuition first:

- For $i = 1$ and $j = n$, $\Pr[z_1 \text{ and } z_n \text{ are compared}] =$

Complexity Analysis of Randomized Quicksort

It now remains to prove:

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Some intuition first:

- For $i = 1$ and $j = n$, $\Pr[z_1 \text{ and } z_n \text{ are compared}] = \frac{2}{n}$

  Since z_1 and z_n are the smallest and largest element in S,

Complexity Analysis of Randomized Quicksort

It now remains to prove:

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Some intuition first:

- For $i = 1$ and $j = n$, $\Pr[z_1 \text{ and } z_n \text{ are compared}] = \frac{2}{n}$

  Since z_1 and z_n are the smallest and largest element in S, they will be compared by $RQS(S)$ if and only if the first pivot that $RQS(S)$ selects among the n elements of S is

Complexity Analysis of Randomized Quicksort

It now remains to prove:

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Some intuition first:

- For $i = 1$ and $j = n$, $\Pr[z_1 \text{ and } z_n \text{ are compared}] = \frac{2}{n}$

  Since z_1 and z_n are the smallest and largest element in S, they will be compared by $RQS(S)$ if and only if the first pivot that $RQS(S)$ selects among the n elements of S is z_1 or z_n.

Complexity Analysis of Randomized Quicksort

It now remains to prove:

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Some intuition first:

- For $i = 1$ and $j = n$, $\Pr[z_1 \text{ and } z_n \text{ are compared}] = \frac{2}{n}$

  Since z_1 and z_n are the smallest and largest element in S, they will be compared by $RQS(S)$ if and only if the first pivot that $RQS(S)$ selects among the n elements of S is z_1 or z_n.

  Note that $j - i + 1 = n$, so the lemma is correct in this case!

Complexity Analysis of Randomized Quicksort

It now remains to prove:

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Some intuition first:

- For $i = 1$ and $j = n$, $\Pr[z_1 \text{ and } z_n \text{ are compared}] = \frac{2}{n}$

  Since z_1 and z_n are the smallest and largest element in S, they will be compared by $RQS(S)$ if and only if the first pivot that $RQS(S)$ selects among the n elements of S is z_1 or z_n.

  Note that $j - i + 1 = n$, so the lemma is correct in this case!

  In our example: $n = 8$, $z_1 = 1$ and $z_n = 8$, and $\Pr[z_1 \text{ and } z_n \text{ are compared}] = \frac{2}{8} = \frac{1}{4}$.

- For any i and $j = i + 1$, $\Pr[z_i \text{ and } z_j \text{ are compared}] =$

Complexity Analysis of Randomized Quicksort

It now remains to prove:

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Some intuition first:

- For $i = 1$ and $j = n$, $\Pr[z_1 \text{ and } z_n \text{ are compared}] = \frac{2}{n}$

  Since z_1 and z_n are the smallest and largest element in S, they will be compared by $RQS(S)$ if and only if the first pivot that $RQS(S)$ selects among the n elements of S is z_1 or z_n.

  Note that $j - i + 1 = n$, so the lemma is correct in this case!

  In our example: $n = 8$, $z_1 = 1$ and $z_n = 8$, and $\Pr[z_1 \text{ and } z_n \text{ are compared}] = \frac{2}{8} = \frac{1}{4}$.

- For any i and $j = i + 1$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = 1$!

Complexity Analysis of Randomized Quicksort

It now remains to prove:

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Some intuition first:

- For $i = 1$ and $j = n$, $\Pr[z_1 \text{ and } z_n \text{ are compared}] = \frac{2}{n}$

  Since z_1 and z_n are the smallest and largest element in S, they will be compared by $RQS(S)$ if and only if the first pivot that $RQS(S)$ selects among the n elements of S is z_1 or z_n.

  Note that $j - i + 1 = n$, so the lemma is correct in this case!

  In our example: $n = 8$, $z_1 = 1$ and $z_n = 8$, and $\Pr[z_1 \text{ and } z_n \text{ are compared}] = \frac{2}{8} = \frac{1}{4}$.

- For any i and $j = i + 1$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = 1!$

  (do you see why?)

Complexity Analysis of Randomized Quicksort

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Complexity Analysis of Randomized Quicksort

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Proof: Consider the set of keys $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$

Complexity Analysis of Randomized Quicksort

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Proof: Consider the set of keys $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$

- $|Z_{ij}| =$

Complexity Analysis of Randomized Quicksort

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Proof: Consider the set of keys $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$

- $|Z_{ij}| = j - i + 1$.

Complexity Analysis of Randomized Quicksort

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Proof: Consider the set of keys $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$

- $|Z_{ij}| = j - i + 1$.
- Initially, Z_{ij} is entirely contained in (the set of keys) S.

Complexity Analysis of Randomized Quicksort

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Proof: Consider the set of keys $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$

- $|Z_{ij}| = j - i + 1$.
- Initially, Z_{ij} is entirely contained in (the set of keys) S.
- $RQS(S)$ keeps selecting pivots and uses them to split S into smaller and smaller subsets until it gets subsets of size one or zero.

Complexity Analysis of Randomized Quicksort

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Proof: Consider the set of keys $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$

- $|Z_{ij}| = j - i + 1$.
- Initially, Z_{ij} is entirely contained in (the set of keys) S.
- $RQS(S)$ keeps selecting pivots and uses them to split S into smaller and smaller subsets until it gets subsets of size one or zero.
- As long as $RQS(S)$ selects pivots that are not in Z_{ij}, then Z_{ij} remains contained in one of the subsets formed by the pivots,

Complexity Analysis of Randomized Quicksort

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Proof: Consider the set of keys $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$

- $|Z_{ij}| = j - i + 1$.
- Initially, Z_{ij} is entirely contained in (the set of keys) S.
- $RQS(S)$ keeps selecting pivots and uses them to split S into smaller and smaller subsets until it gets subsets of size one or zero.
- As long as $RQS(S)$ selects pivots that are not in Z_{ij}, then Z_{ij} remains contained in one of the subsets formed by the pivots, and z_i and z_j remain uncompared.

Complexity Analysis of Randomized Quicksort

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Proof: Consider the set of keys $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$

- $|Z_{ij}| = j - i + 1$.
- Initially, Z_{ij} is entirely contained in (the set of keys) S.
- $RQS(S)$ keeps selecting pivots and uses them to split S into smaller and smaller subsets until it gets subsets of size one or zero.
- As long as $RQS(S)$ selects pivots that are not in Z_{ij}, then Z_{ij} remains contained in one of the subsets formed by the pivots, and z_i and z_j remain uncompared.
- At some point, $RQS(S)$ must select a pivot p in Z_{ij}, because it must split Z_{ij} into smaller sets.

Complexity Analysis of Randomized Quicksort

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Proof: Consider the set of keys $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$

- $|Z_{ij}| = j - i + 1$.
- Initially, Z_{ij} is entirely contained in (the set of keys) S.
- $RQS(S)$ keeps selecting pivots and uses them to split S into smaller and smaller subsets until it gets subsets of size one or zero.
- As long as $RQS(S)$ selects pivots that are not in Z_{ij}, then Z_{ij} remains contained in one of the subsets formed by the pivots, and z_i and z_j remain uncompared.
- At some point, $RQS(S)$ must select a pivot p in Z_{ij}, because it must split Z_{ij} into smaller sets.
- Consider the first time $RQS(S)$ selects a pivot p in Z_{ij}.

Complexity Analysis of Randomized Quicksort

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Proof: Let $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$. $|Z_{ij}| = j - i + 1$

- ...
- Consider the first time $RQS(S)$ selects a pivot p in Z_{ij}.

Complexity Analysis of Randomized Quicksort

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Proof: Let $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$. $|Z_{ij}| = j - i + 1$

- ...
- Consider the first time $RQS(S)$ selects a pivot p in Z_{ij}.
- There are 2 possible cases:

Complexity Analysis of Randomized Quicksort

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Proof: Let $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$. $|Z_{ij}| = j - i + 1$

- ...
- Consider the first time $RQS(S)$ selects a pivot p in Z_{ij}.
- There are 2 possible cases:
  1. If $z_i < p < z_j$ then

Complexity Analysis of Randomized Quicksort

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Proof: Let $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$. $|Z_{ij}| = j - i + 1$

- ...
- Consider the first time $RQS(S)$ selects a pivot p in Z_{ij}.
- There are 2 possible cases:
  1. If $z_i < p < z_j$ then z_i and z_j are never compared

Complexity Analysis of Randomized Quicksort

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Proof: Let $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$. $|Z_{ij}| = j - i + 1$

- ...
- Consider the first time $RQS(S)$ selects a pivot p in Z_{ij}.
- There are 2 possible cases:
  1. If $z_i < p < z_j$ then z_i and z_j are never compared
  2. If $p = z_i$ or $p = z_j$ then

Complexity Analysis of Randomized Quicksort

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Proof: Let $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$. $|Z_{ij}| = j - i + 1$

- ...
- Consider the first time $RQS(S)$ selects a pivot p in Z_{ij}.
- There are 2 possible cases:
  1. If $z_i < p < z_j$ then z_i and z_j are never compared
  2. If $p = z_i$ or $p = z_j$ then z_i and z_j are compared.

Complexity Analysis of Randomized Quicksort

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Proof: Let $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$. $|Z_{ij}| = j - i + 1$

- ...
- Consider the first time $RQS(S)$ selects a pivot p in Z_{ij}.
- There are 2 possible cases:
  1. If $z_i < p < z_j$ then z_i and z_j are never compared
  2. If $p = z_i$ or $p = z_j$ then z_i and z_j are compared.
- So:

$$\Pr[z_i \text{ and } z_j \text{ are compared}] = \Pr[p = z_i \text{ or } p = z_j$$

Complexity Analysis of Randomized Quicksort

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Proof: Let $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$. $|Z_{ij}| = j - i + 1$

- ...
- Consider the first time $RQS(S)$ selects a pivot p in Z_{ij}.
- There are 2 possible cases:
  1. If $z_i < p < z_j$ then z_i and z_j are never compared
  2. If $p = z_i$ or $p = z_j$ then z_i and z_j are compared.
- So:

$$\Pr[z_i \text{ and } z_j \text{ are compared}] = \Pr[p = z_i \text{ or } p = z_j \mid p \in Z_{ij}]$$

Complexity Analysis of Randomized Quicksort

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Proof: Let $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$. $|Z_{ij}| = j - i + 1$

- ...
- Consider the first time $RQS(S)$ selects a pivot p in Z_{ij}.
- There are 2 possible cases:
  1. If $z_i < p < z_j$ then z_i and z_j are never compared
  2. If $p = z_i$ or $p = z_j$ then z_i and z_j are compared.
- So:

$$\begin{aligned} \Pr[z_i \text{ and } z_j \text{ are compared}] &= \Pr[p = z_i \text{ or } p = z_j \mid p \in Z_{ij}] \\ &= \frac{2}{j-i+1} \end{aligned}$$

Complexity Analysis of Randomized Quicksort

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Proof: Let $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$. $|Z_{ij}| = j - i + 1$

- ...
- Consider the first time $RQS(S)$ selects a pivot p in Z_{ij}.
- There are 2 possible cases:
  1. If $z_i < p < z_j$ then z_i and z_j are never compared
  2. If $p = z_i$ or $p = z_j$ then z_i and z_j are compared.
- So:

$$\begin{aligned}\Pr[z_i \text{ and } z_j \text{ are compared}] &= \Pr[p = z_i \text{ or } p = z_j \mid p \in Z_{ij}] \\ &= \frac{2}{j-i+1}\end{aligned}$$

The last equality holds because:
(1) $|Z_{ij}| = j - i + 1$

Complexity Analysis of Randomized Quicksort

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Proof: Let $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$. $|Z_{ij}| = j - i + 1$

- ...
- Consider the first time $RQS(S)$ selects a pivot p in Z_{ij}.
- There are 2 possible cases:
  1. If $z_i < p < z_j$ then z_i and z_j are never compared
  2. If $p = z_i$ or $p = z_j$ then z_i and z_j are compared.
- So:

$$\begin{aligned}\Pr[z_i \text{ and } z_j \text{ are compared}] &= \Pr[p = z_i \text{ or } p = z_j \mid p \in Z_{ij}] \\ &= \frac{2}{j-i+1}\end{aligned}$$

The last equality holds because:
(1) $|Z_{ij}| = j - i + 1$, and (2) each key in Z_{ij} is equaly likely to be the selected pivot $p \in Z_{ij}$.

Complexity Analysis of Randomized Quicksort

Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

Proof: Let $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$. $|Z_{ij}| = j - i + 1$

- ...
- Consider the first time $RQS(S)$ selects a pivot p in Z_{ij}.
- There are 2 possible cases:
  1. If $z_i < p < z_j$ then z_i and z_j are never compared
  2. If $p = z_i$ or $p = z_j$ then z_i and z_j are compared.
- So:

$$\begin{aligned}\Pr[z_i \text{ and } z_j \text{ are compared}] &= \Pr[p = z_i \text{ or } p = z_j \mid p \in Z_{ij}] \\ &= \frac{2}{j-i+1}\end{aligned}$$

The last equality holds because:
(1) $|Z_{ij}| = j - i + 1$, and (2) each key in Z_{ij} is equaly likely to be the selected pivot $p \in Z_{ij}$.

Note: The exact argument uses conditional probabilities.

Complexity Analysis of Randomized Quicksort

- We just proved:

  Lemma: $E(C) = \sum_{1 \le i < j \le n} \Pr[z_i \text{ and } z_j \text{ are compared}]$

- We will later prove:

  Lemma: For $i < j$, $\Pr[z_i \text{ and } z_j \text{ are compared}] = \frac{2}{j-i+1}$

- By combining these two lemmas:

  $E(C) = \sum_{1 \le i < j \le n} \frac{2}{j-i+1} = \ldots = 2n(1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n})$

- It is known that $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$ is $O(\log n)$.

  [In fact, for $n \ge 2$, $H_n \le (\log_e n) + 1$.]

- Therefore:

  Theorem: $E(C)$ is $O(n \log n)$