CPM2022

Cartesian Tree Subsequence Matching

Tsubasa Oizumi[1], Takeshi Kai,[1] Takuya Mieno,[2]
Shunsuke Inenaga,[3] Hiroki Arimura[1]

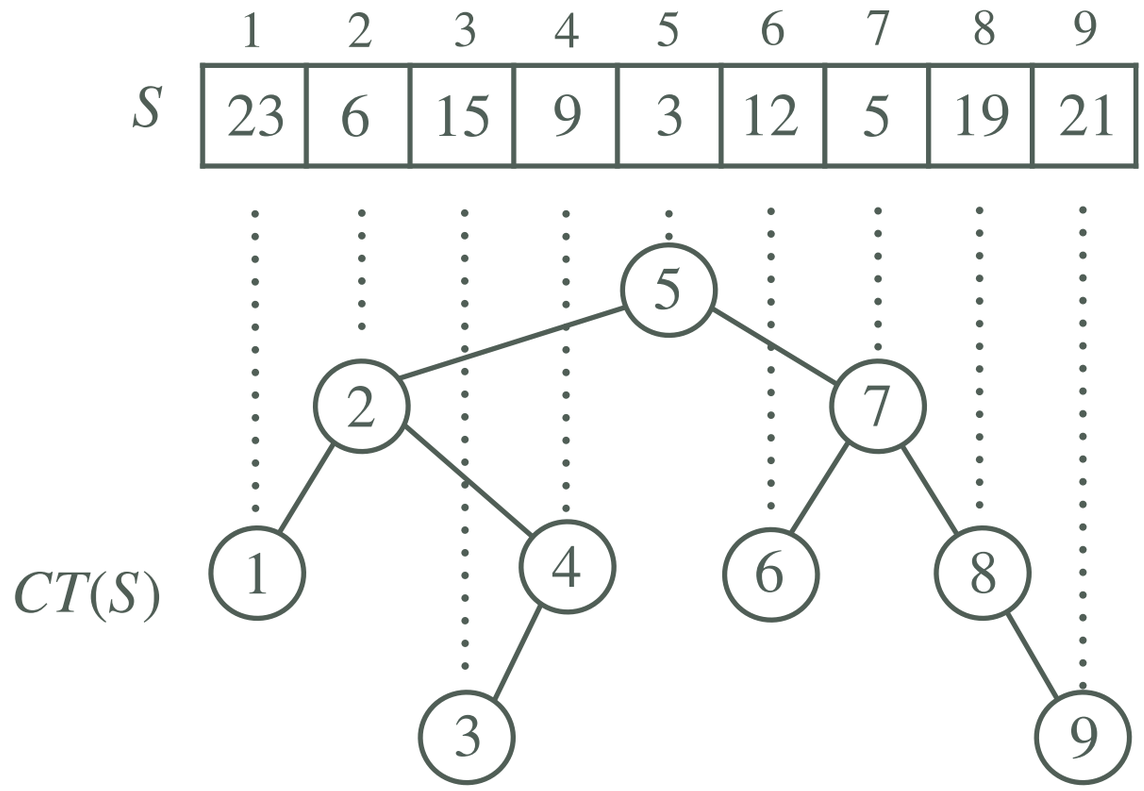
[1]Hokkaido University
[2]University of Electro-Communications
[3]Kyushu University

Cartesian tree

Cartesian Tree

The Cartesian Tree $CT(S)$ of a numeric series $S = (S[1], \ldots, S[n])$ is defined recursively as follows, where $i_{\min}$ is the leftmost index of the minimum value in S.

- The root of $CT(S)$ is $i_{\min}$,
- the left subtree of $i_{\min}$ is $CT(S[1..i_{\min} - 1])$, and
- the right subtree of $i_{\min}$ is $CT(S[i_{\min} + 1..n])$.

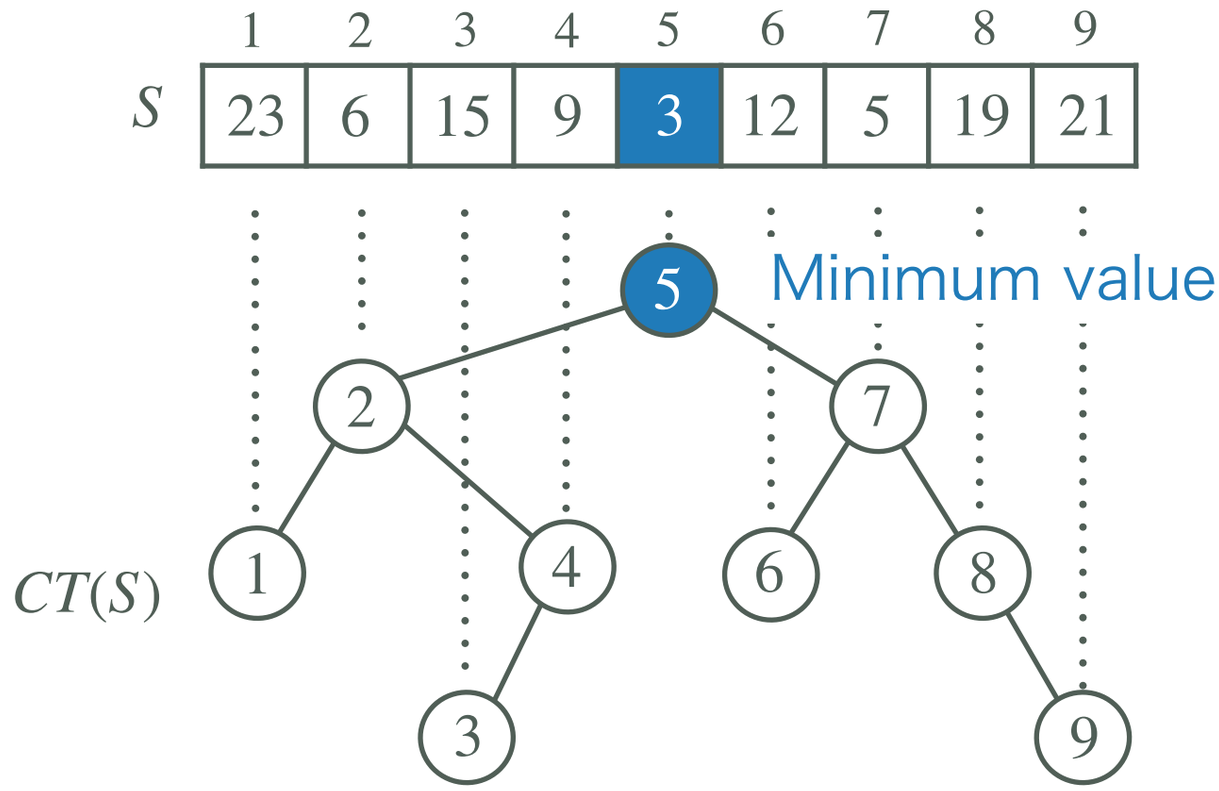


Cartesian tree

Cartesian Tree

The Cartesian Tree $CT(S)$ of a numeric series $S = (S[1], \ldots, S[n])$ is defined recursively as follows, where $i_{\min}$ is the leftmost index of the minimum value in S.

- The root of $CT(S)$ is $i_{\min}$,
- the left subtree of $i_{\min}$ is $CT(S[1..i_{\min} - 1])$, and
- the right subtree of $i_{\min}$ is $CT(S[i_{\min} + 1..n])$.

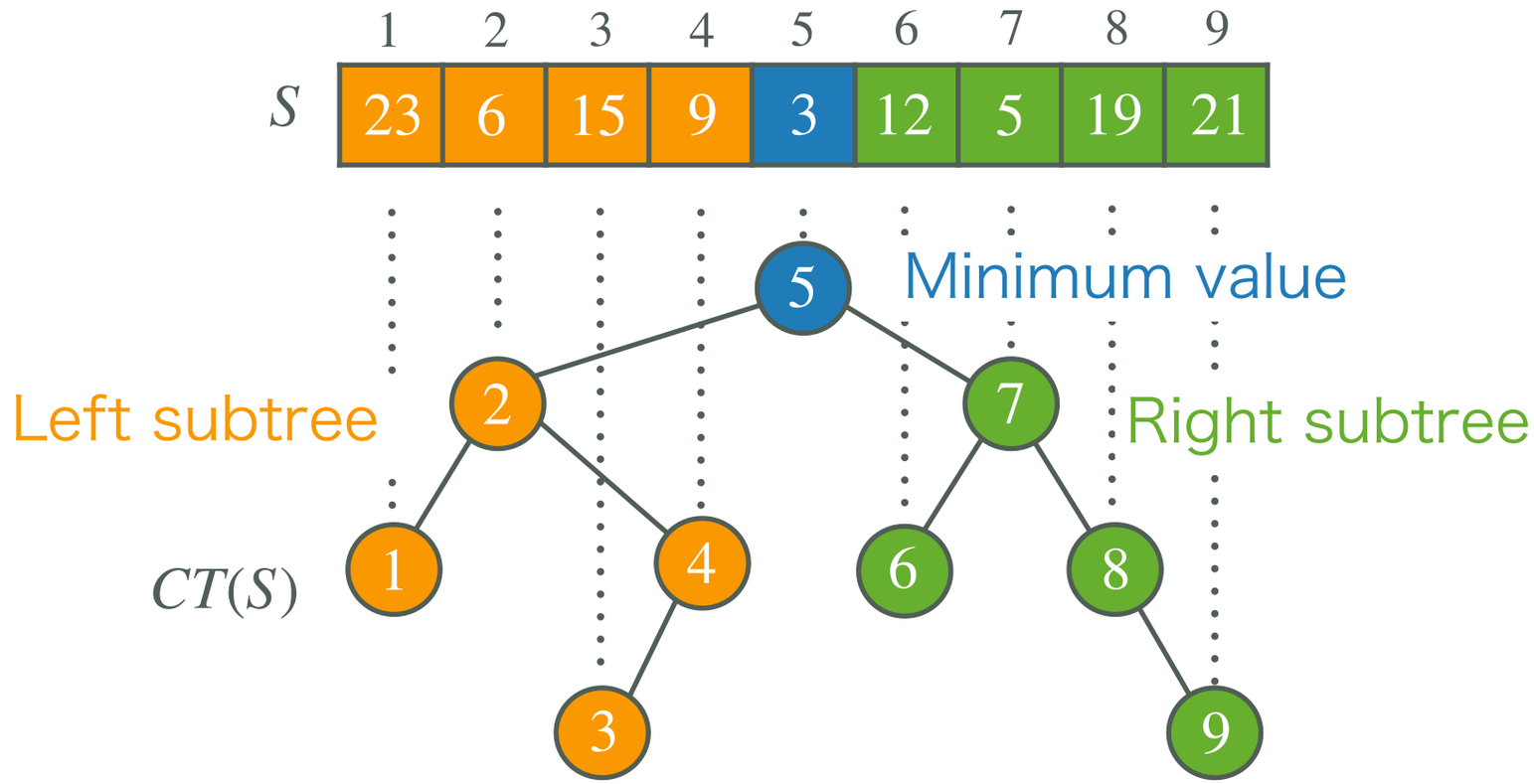


Cartesian tree

Cartesian Tree

The Cartesian Tree $CT(S)$ of a numeric series $S = (S[1], \ldots, S[n])$ is defined recursively as follows, where $i_{\min}$ is the leftmost index of the minimum value in S.

- The root of $CT(S)$ is $i_{\min}$,
- the left subtree of $i_{\min}$ is $CT(S[1..i_{\min} - 1])$, and
- the right subtree of $i_{\min}$ is $CT(S[i_{\min} + 1..n])$.



Cartesian tree matching

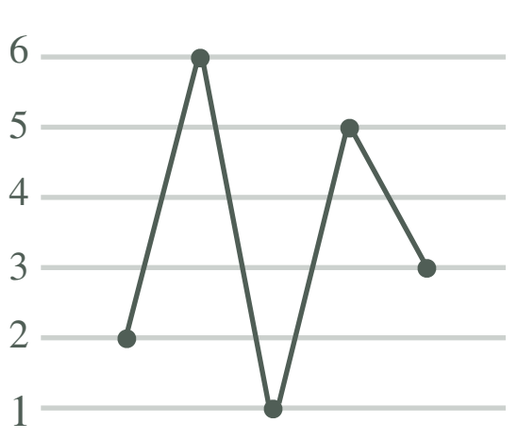
`CTMStr` problem [Park et al., 2019]

Input: Text $T[1..n]$ and pattern $P[1..m]$.

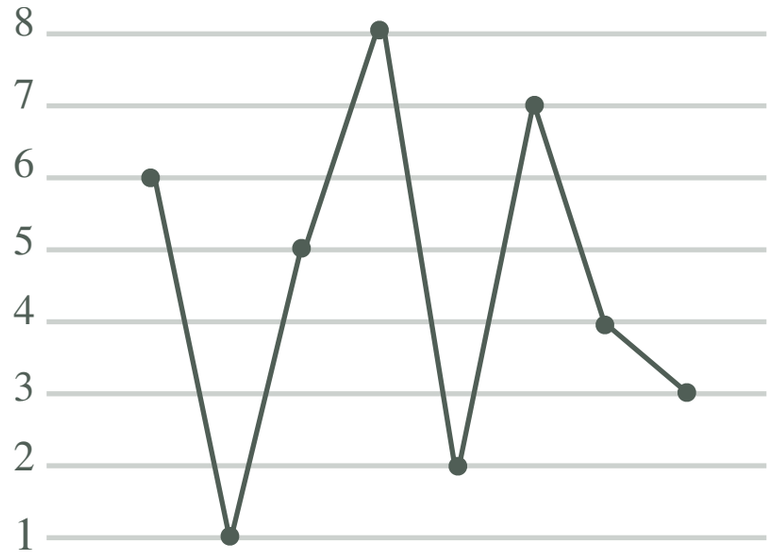
Output: Every **substring** T' of a text T such that $CT(T') = CT(P)$.

- Park et al. proved that `CTMStr` can be solved in $O(m+n)$ time and $O(n)$ space.

Pattern P



Text T



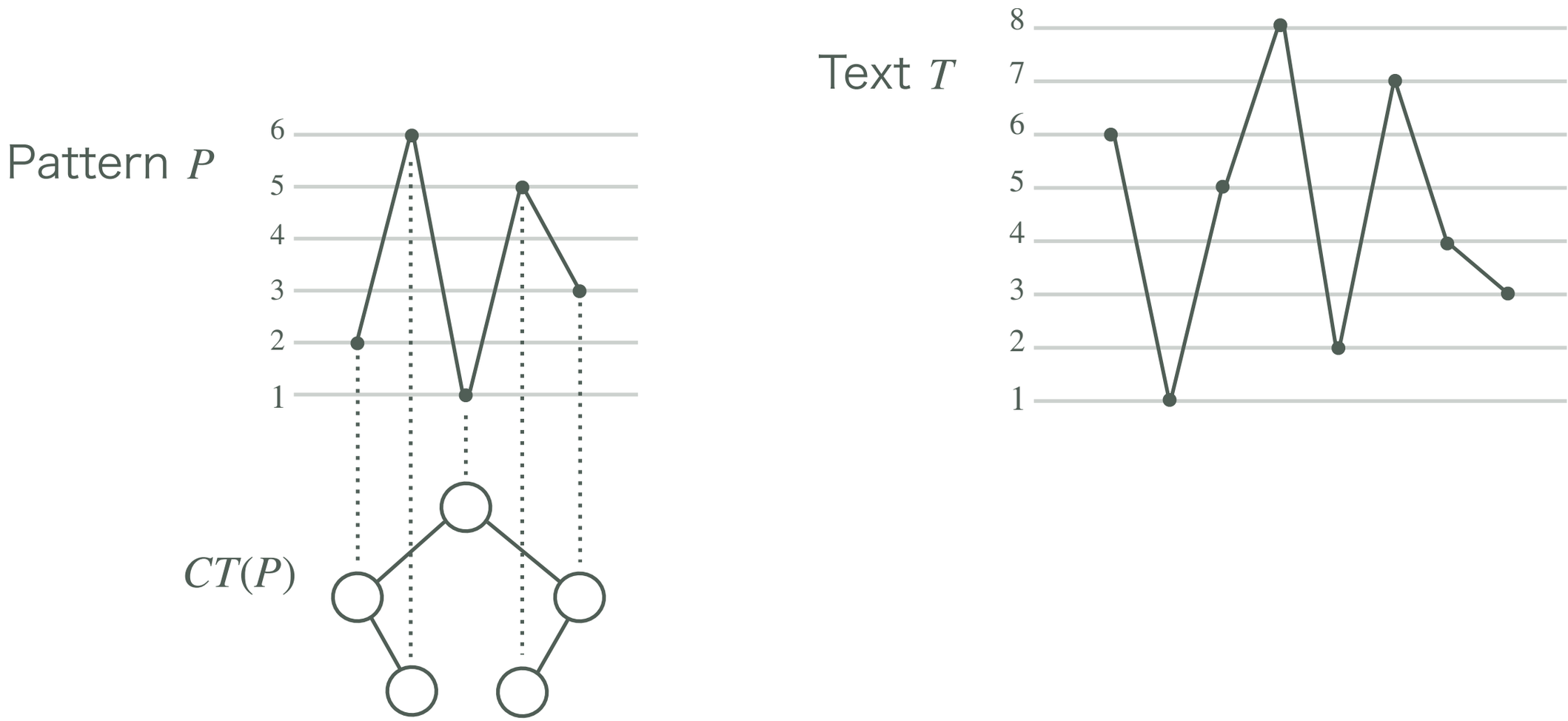
Cartesian tree matching

`CTMStr` problem [Park et al., 2019]

Input: Text $T[1..n]$ and pattern $P[1..m]$.

Output: Every **substring** T' of a text T such that $CT(T') = CT(P)$.

- Park et al. proved that `CTMStr` can be solved in $O(m+n)$ time and $O(n)$ space.



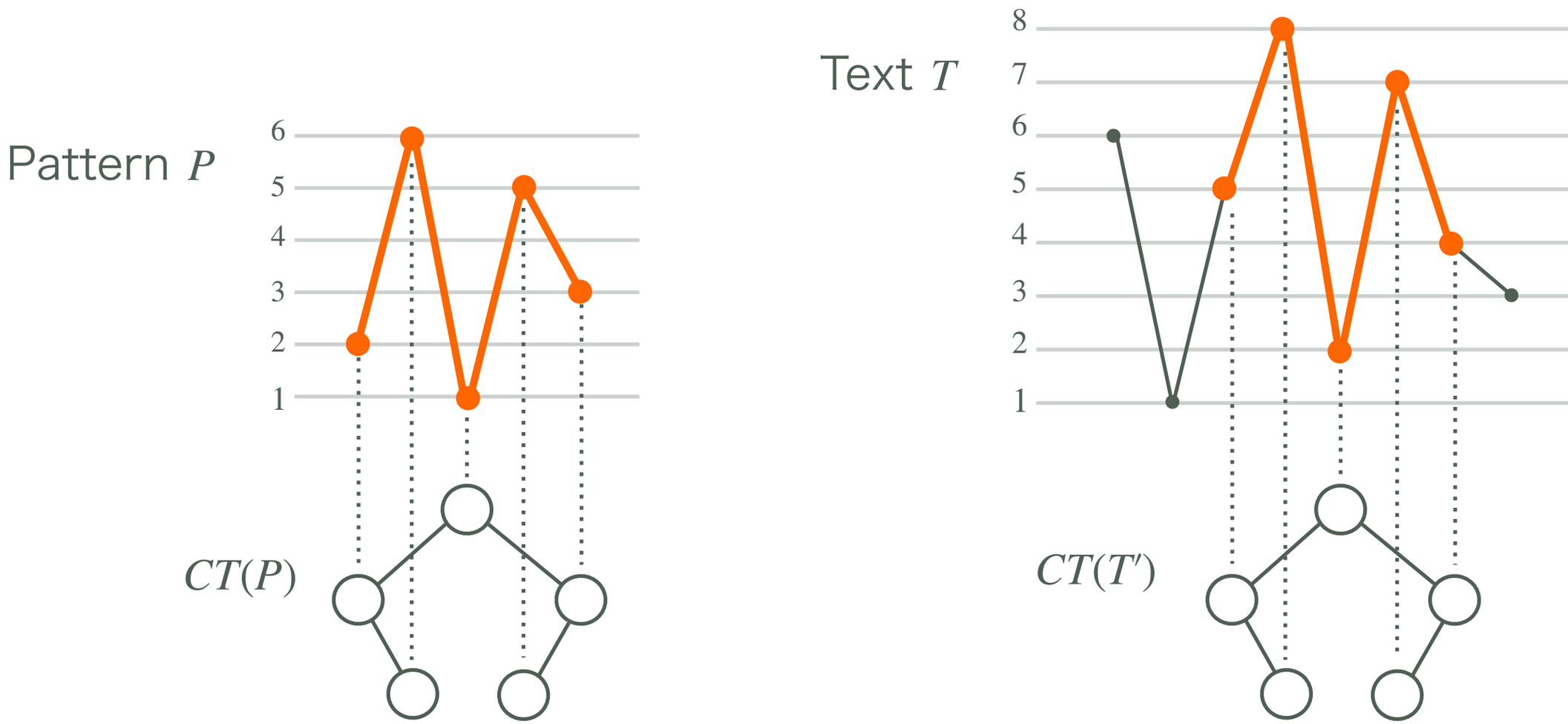
Cartesian tree matching

`CTMStr` problem [Park et al., 2019]

Input: Text $T[1..n]$ and pattern $P[1..m]$.

Output: Every **substring** T' of a text T such that $CT(T') = CT(P)$.

- Park et al. proved that `CTMStr` can be solved in $O(m+n)$ time and $O(n)$ space.

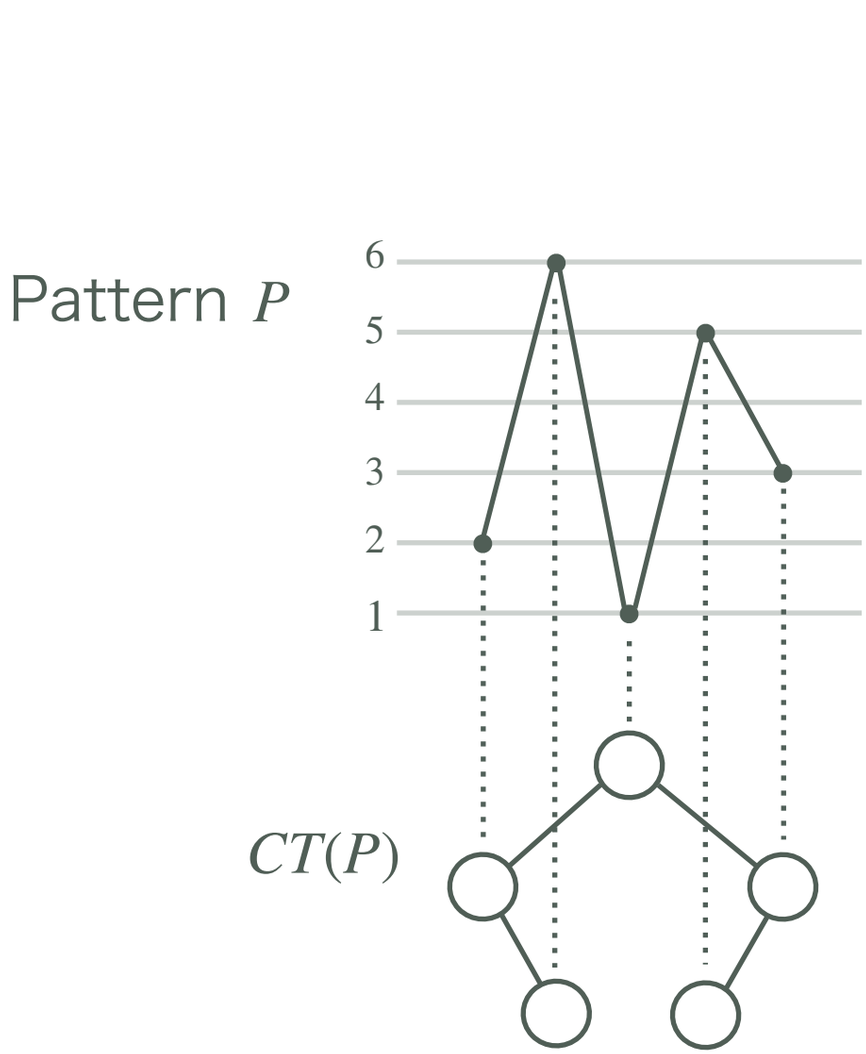


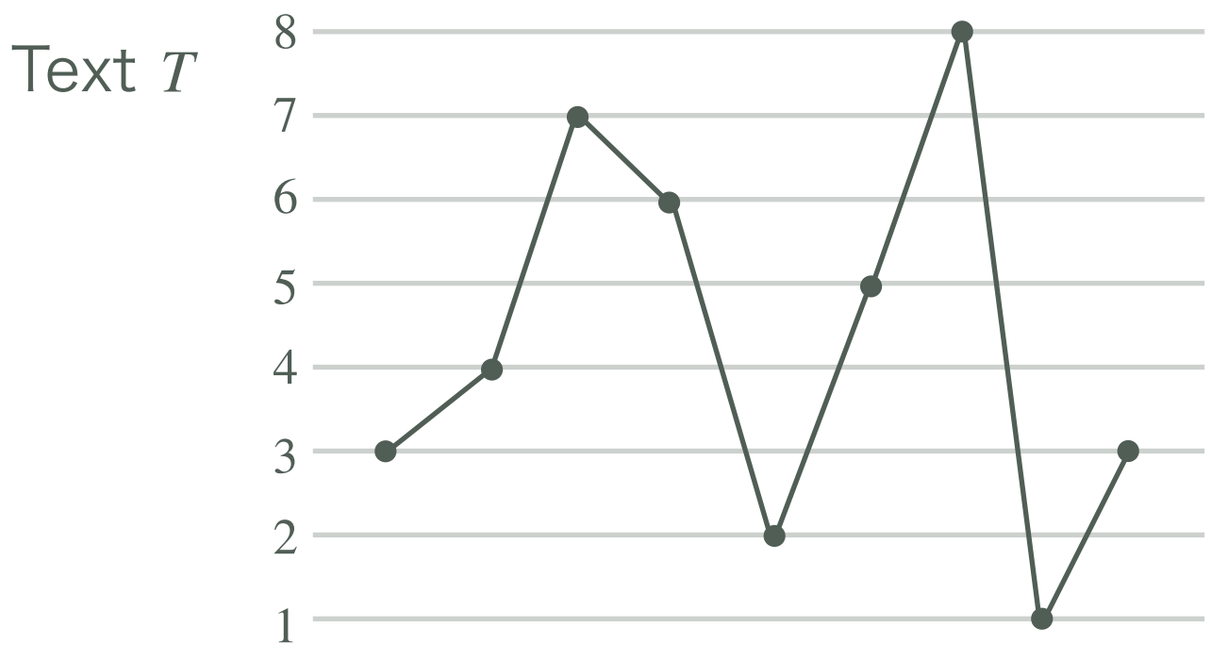
Cartesian tree **subsequence** matching

`CTMSeq` problem [This work]

Input: Text $T[1..n]$ and pattern $P[1..m]$.

Output: Every minimal **subsequence** T' of a text T such that $CT(T') = CT(P)$.



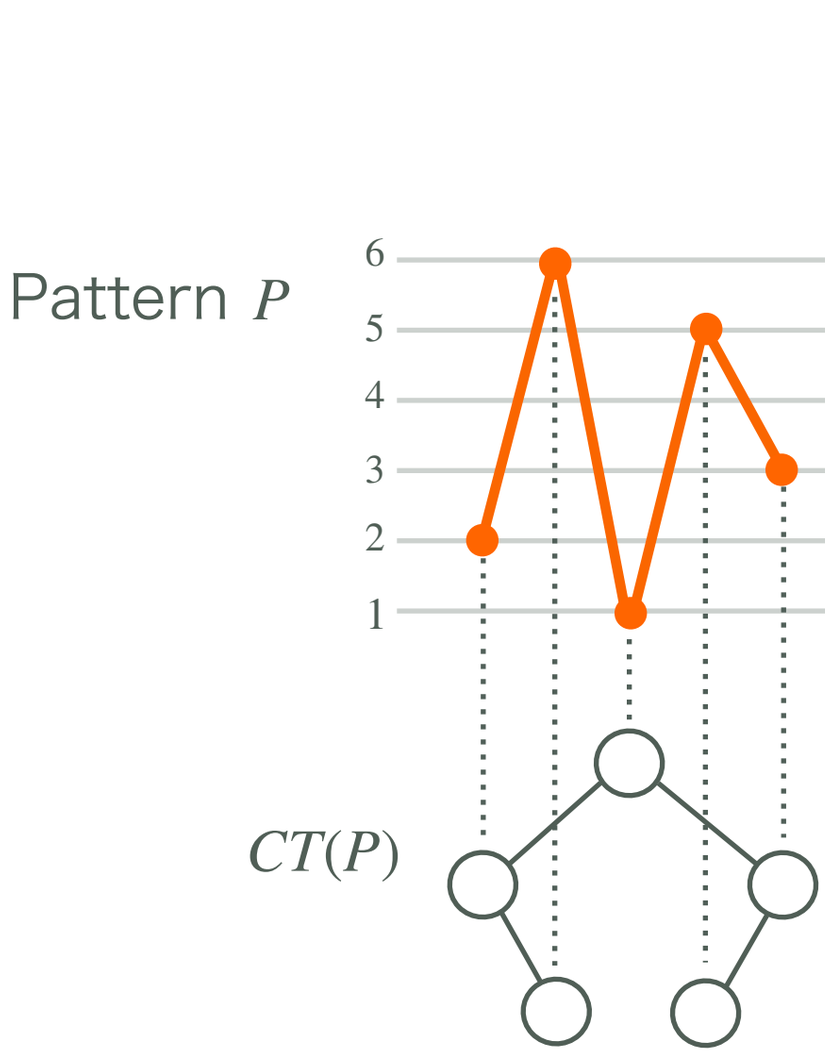


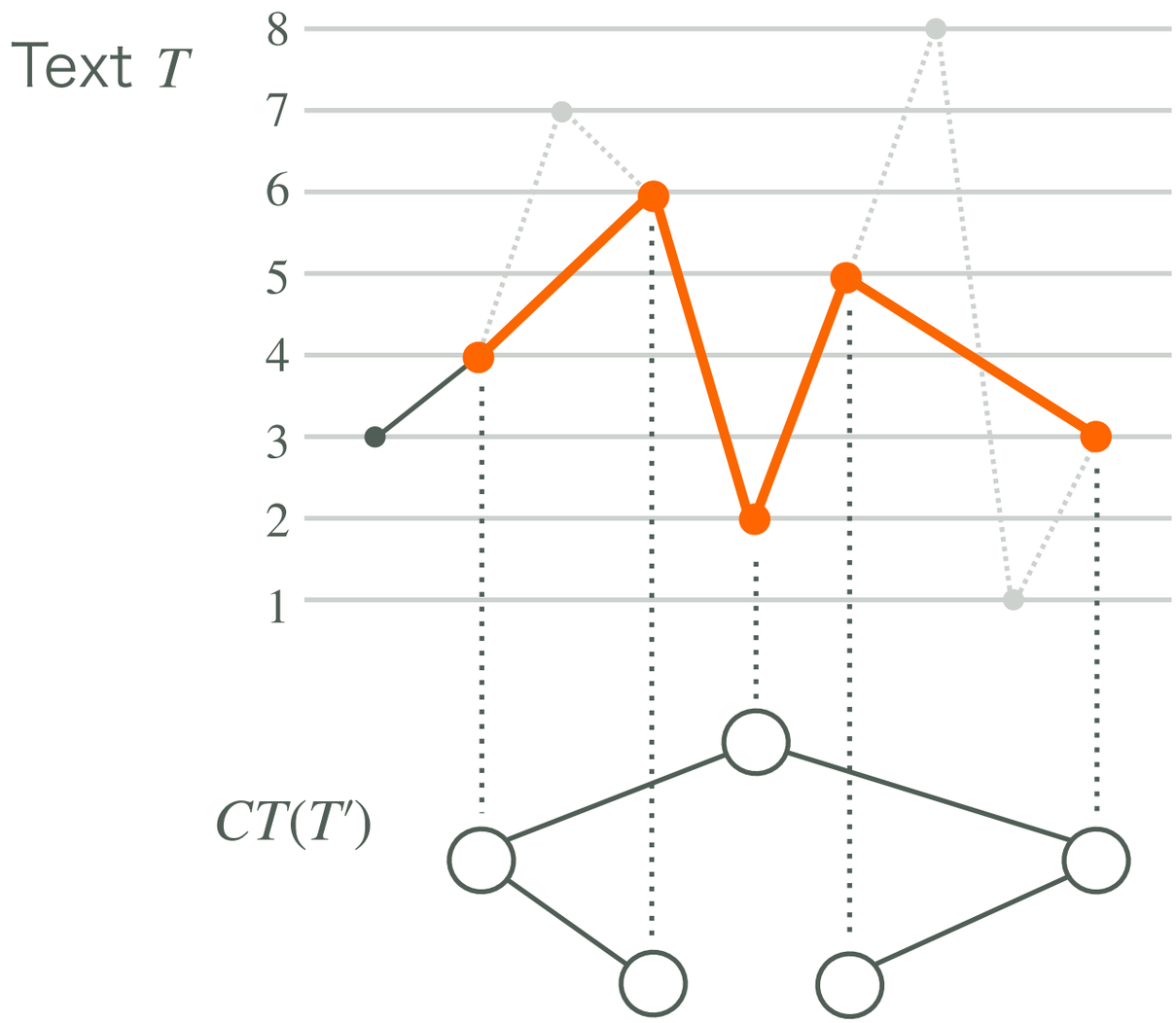
Cartesian tree **subsequence** matching

`CTMSeq` problem [This work]

Input: Text $T[1..n]$ and pattern $P[1..m]$.

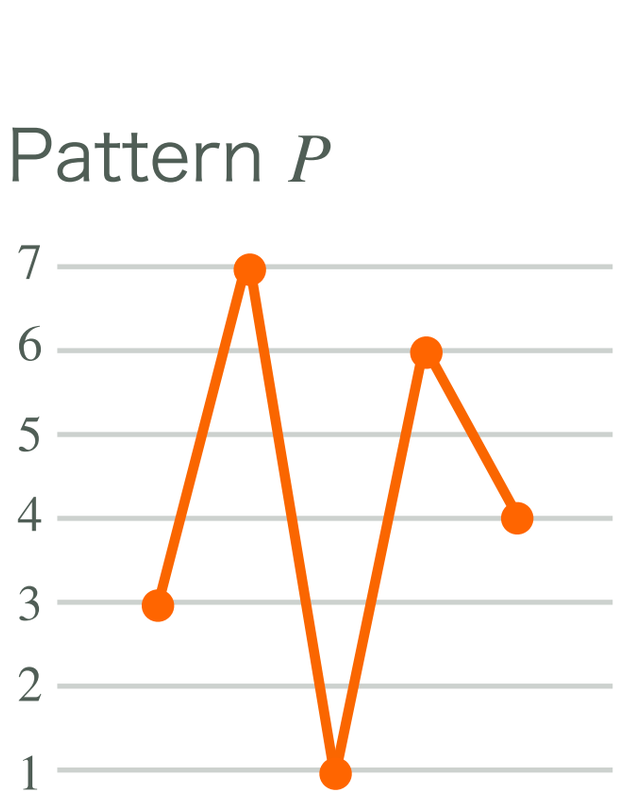
Output: Every minimal **subsequence** T' of a text T such that $CT(T') = CT(P)$.

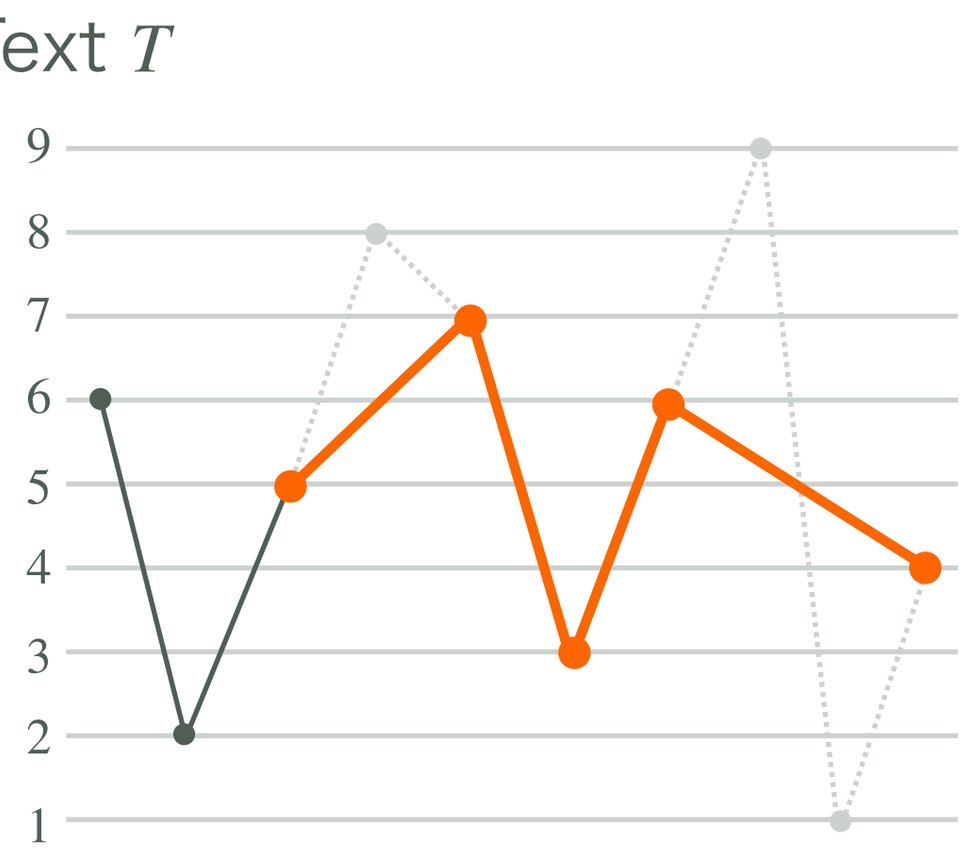




Motivation

- We extend to `CTMSeq`, which is a non-continuous subsequence version of `CTMStr`.
  - The motivation for extending to subsequence is to ignore measurement errors.
- We develop efficient algorithm for solving `CTMSeq`.

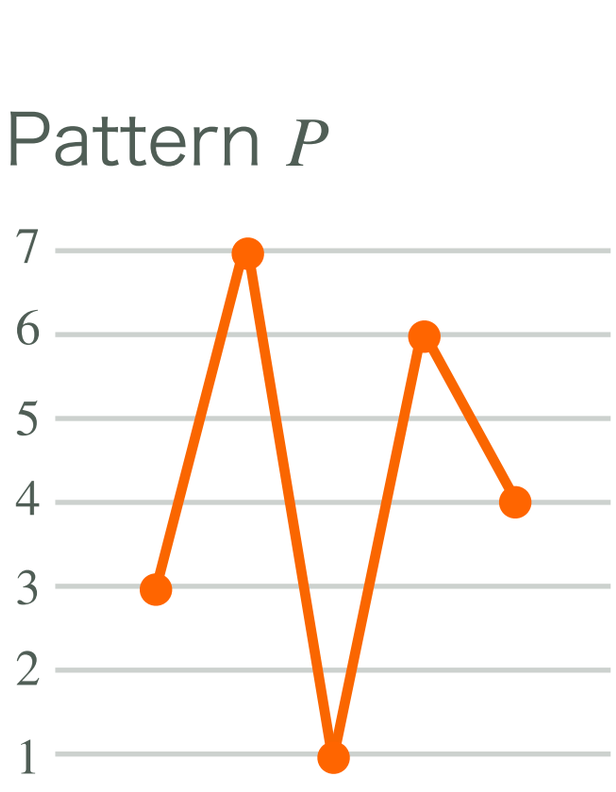


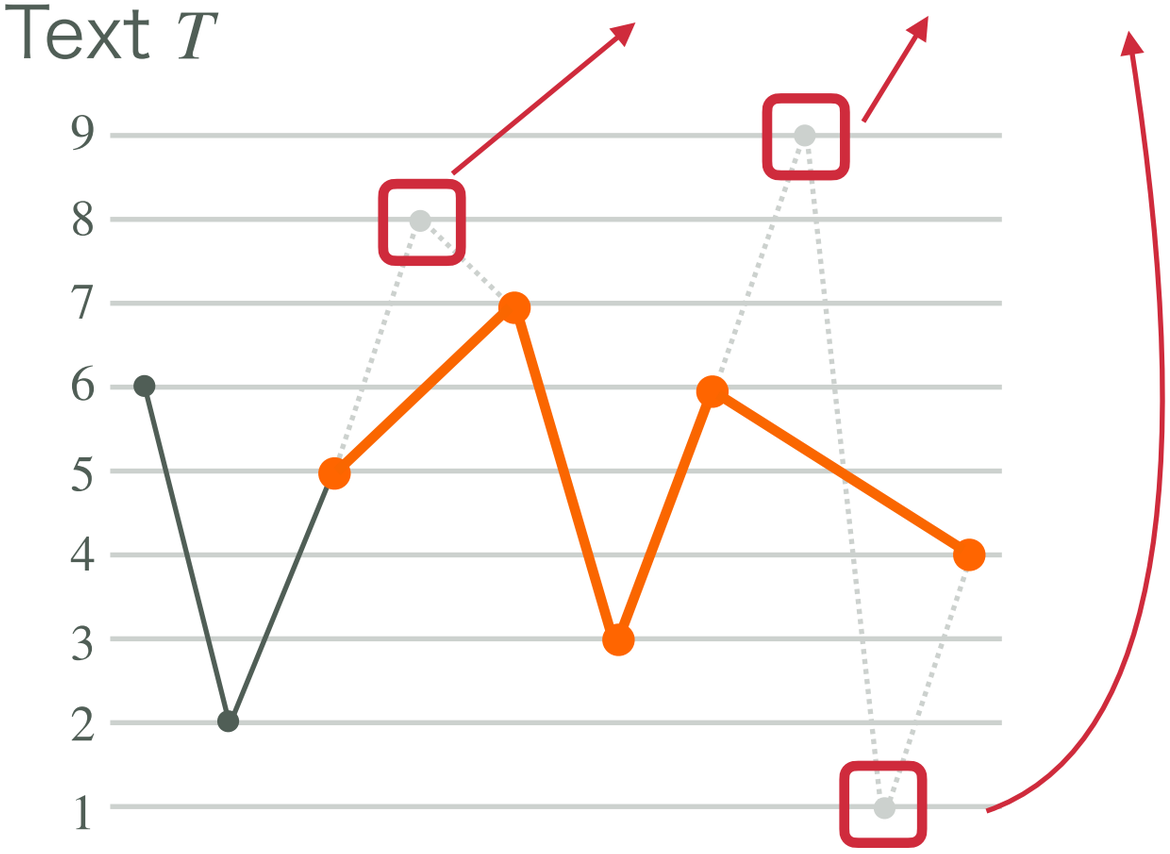


Motivation

- We extend to `CTMSeq`, which is a non-continuous subsequence version of `CTMStr`.
  - The motivation for extending to subsequence is to ignore measurement errors.
- We develop efficient algorithm for solving `CTMSeq`.

measurement errors





Related work (OPM)

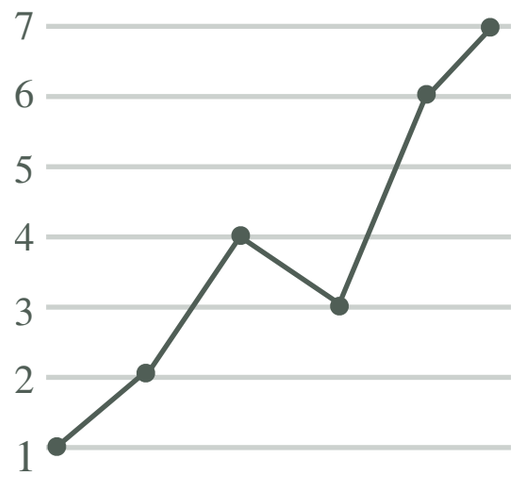
OPM problem [Kim et al., 2014]

Input: Text $T[1..n]$ and pattern $P[1..m]$.

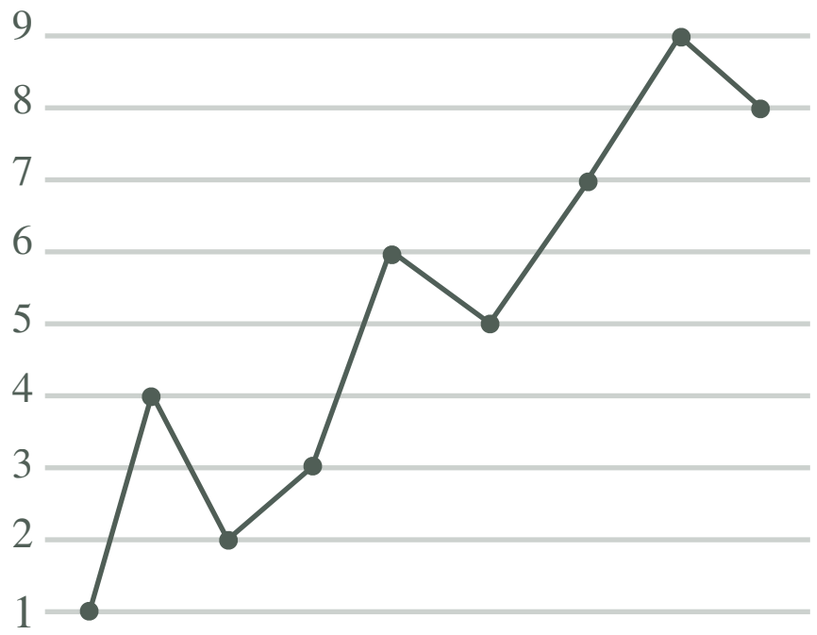
Output: Every substring T' of a text T such that the relative orders of values in T' are the same as that of a pattern P.

- OPM can be solved in $O(m+n)$ time [Kim et al., 2014].
- The subsequence version of OPM is shown to be NP-hard [Bose et al., 1998].

Pattern P



Text T



Related work (OPM)

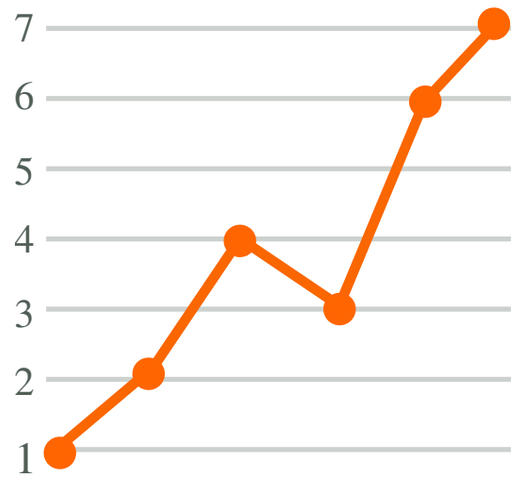
OPM problem [Kim et al., 2014]

Input: Text $T[1..n]$ and pattern $P[1..m]$.

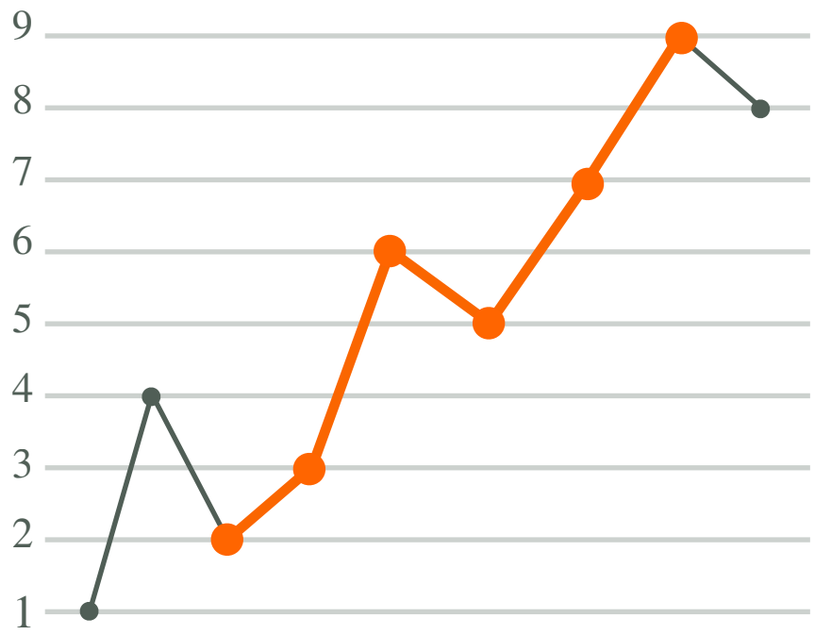
Output: Every substring T' of a text T such that the relative orders of values in T' are the same as that of a pattern P.

- OPM can be solved in $O(m+n)$ time [Kim et al., 2014].
- The subsequence version of OPM is shown to be NP-hard [Bose et al., 1998].

Pattern P



Text T



Related work (OPM)

- OPM is a problem that relaxes the matching constraints of CTM.
  - By definition, if it matches at OPM, it matches at CTM.
  - The converse doesn't always hold (you can see that in the counterexample).

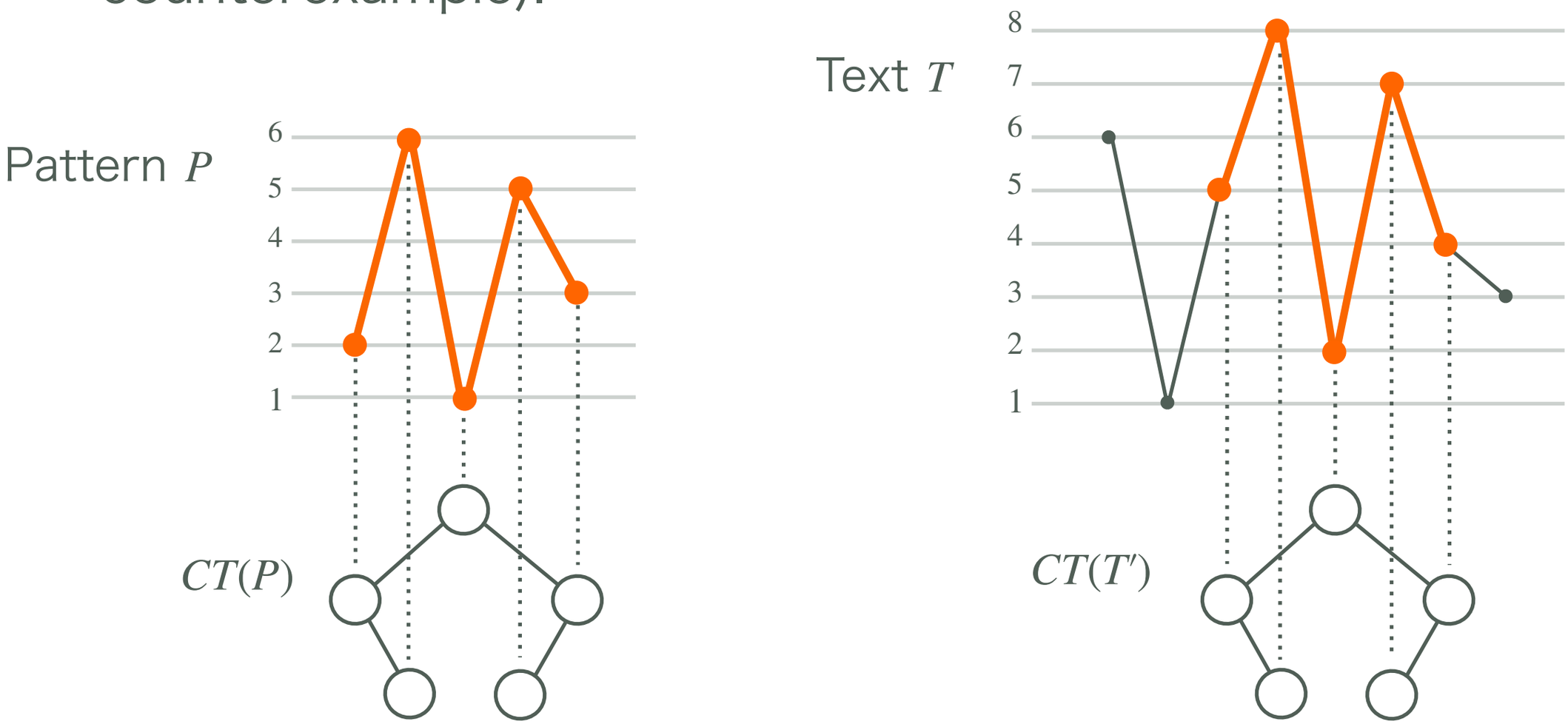


Figure: The counterexample of CTM ⇒ OPM

Related work (CTMIS)

CTMIS problem [Gawrychowski et al., 2020]

Input: Two indeterminate strings $T[1..n]$ and $P[1..n]$.

Output: Whether there exists determinate strings $T' \in \tilde{T}$ and $P' \in \tilde{P}$ such that $CT(T') = CT(P')$

- CTMIS can be solved in $O(n \log n \log \log n)$ time and $O(n \log n)$ space

$T = (\underline{2|7|10}, \underline{5|20|31}, \underline{10|17|25}, \underline{1|9|11}, \underline{1|8|18})$ $P = (\underline{2|4|7}, \underline{2|5|6}, \underline{1|4|8}, \underline{4|7|8}, \underline{3|10|16})$

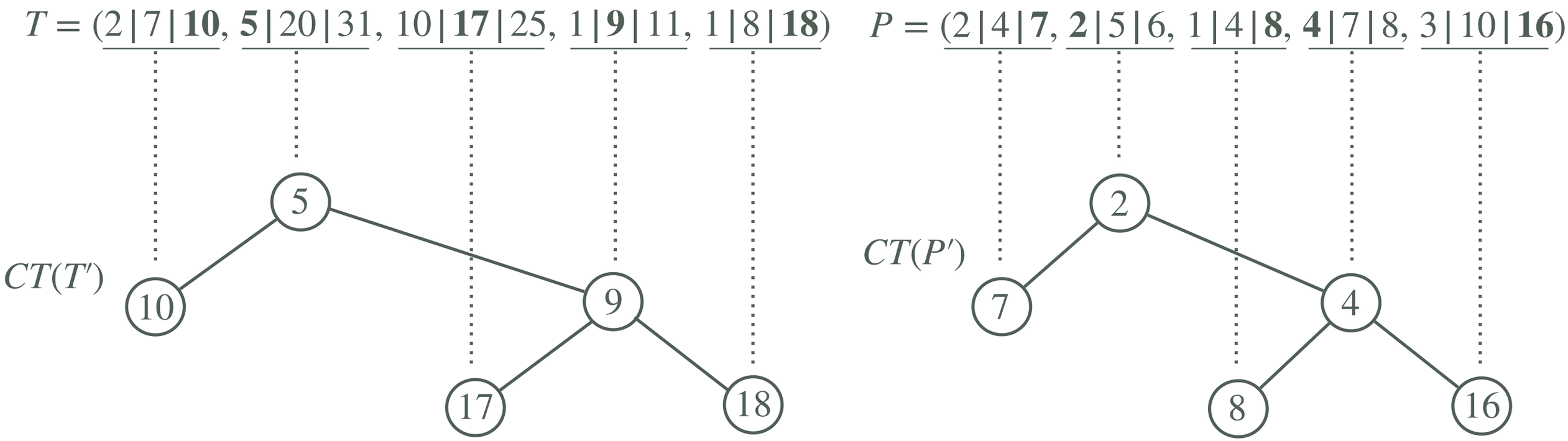
Related work (CTMIS)

`CTMIS` problem [Gawrychowski et al., 2020]

Input: Two indeterminate strings $T[1..n]$ and $P[1..n]$.

Output: Whether there exists determinate strings $T' \in \tilde{T}$ and $P' \in \tilde{P}$ such that $CT(T') = CT(P')$

- `CTMIS` can be solved in $O(n \log n \log \log n)$ time and $O(n \log n)$ space



Summary of related works

Table: time complexity for each problem

Matching model	substring	subsequence
OPM	$O(m+n)$ [Kim et al., 2014]	NP-hard [Bose et al., 2014]
CTM	$O(m+n)$ [Park et al., 2020]	$O(mn \log\log n)$ [This work]
CTMIS	$O(n \log n \log\log n)$ [Gawrychowski et al., 2020]	Open problem

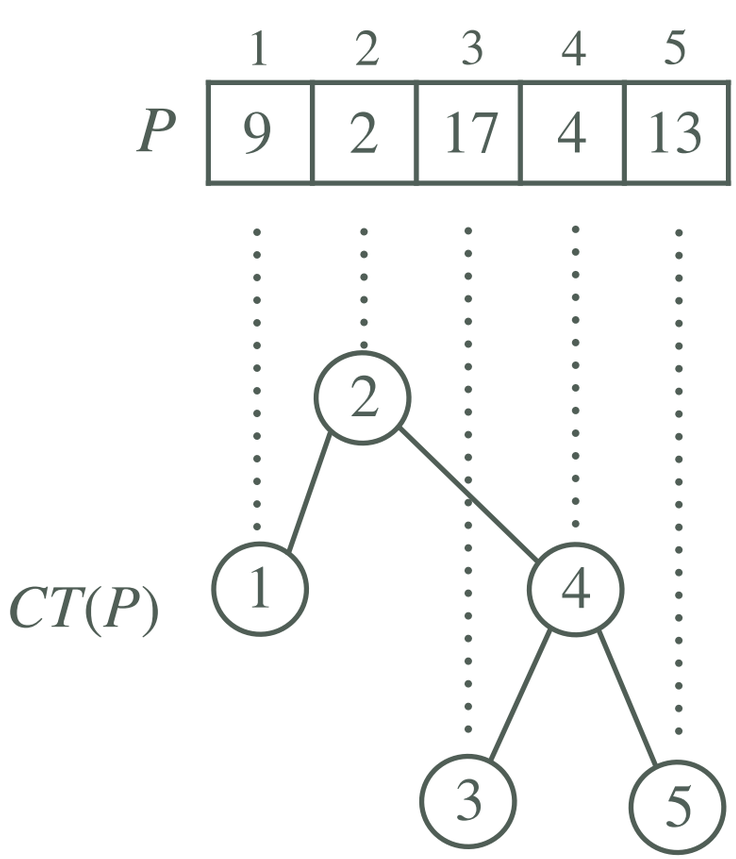
- n is the length of text T.
- m is the length of pattern P.

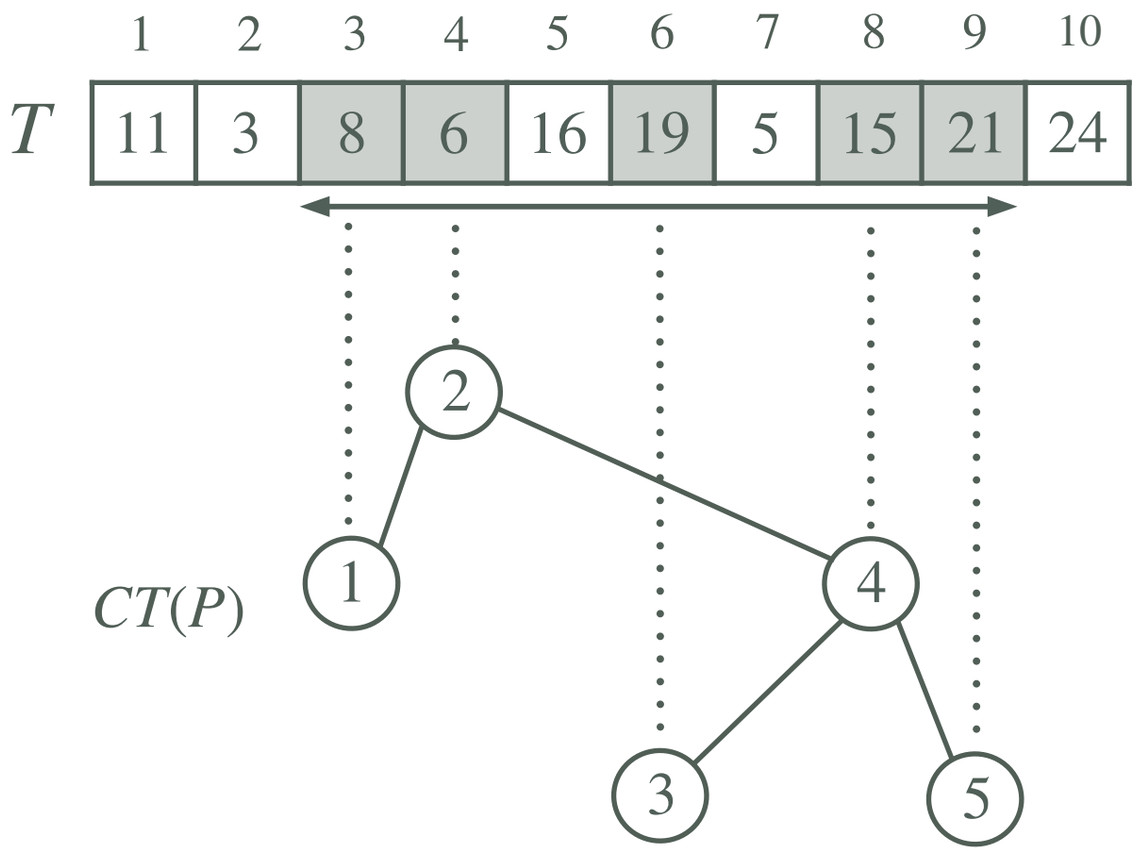
Definition of occurrence

- An interval $[\ell, r]$ is said to be an occurrence interval

  if $CT(P) = CT(T')$ for some subsequence of $T[\ell \,..\, r]$.
- An occurrence interval $[\ell, r]$ is said to be minimal

  if there is no occurrence interval $[\ell', r']$ such that $[\ell', r'] \subsetneq [\ell, r]$.



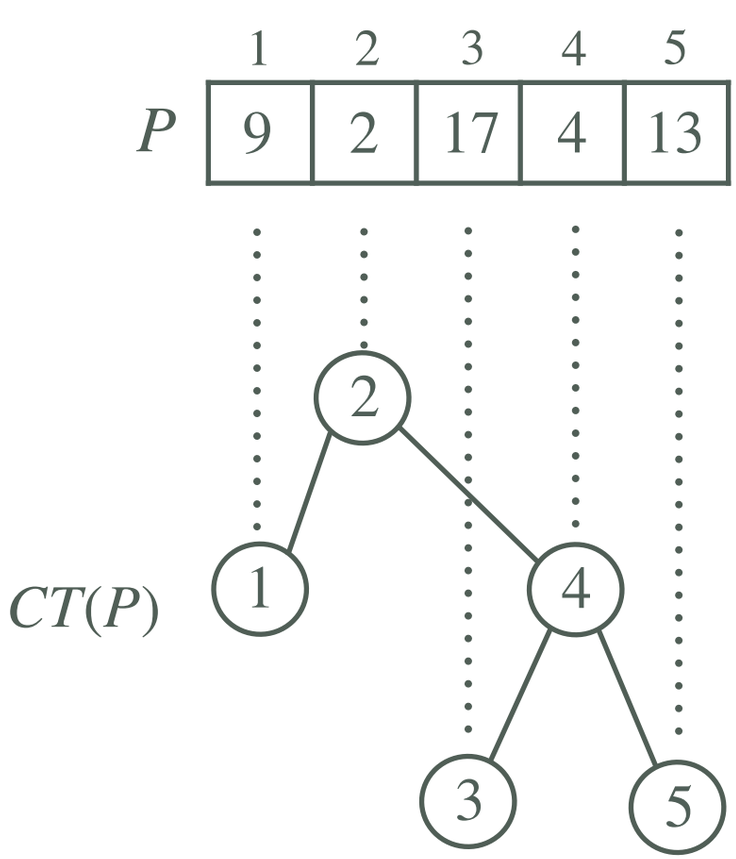


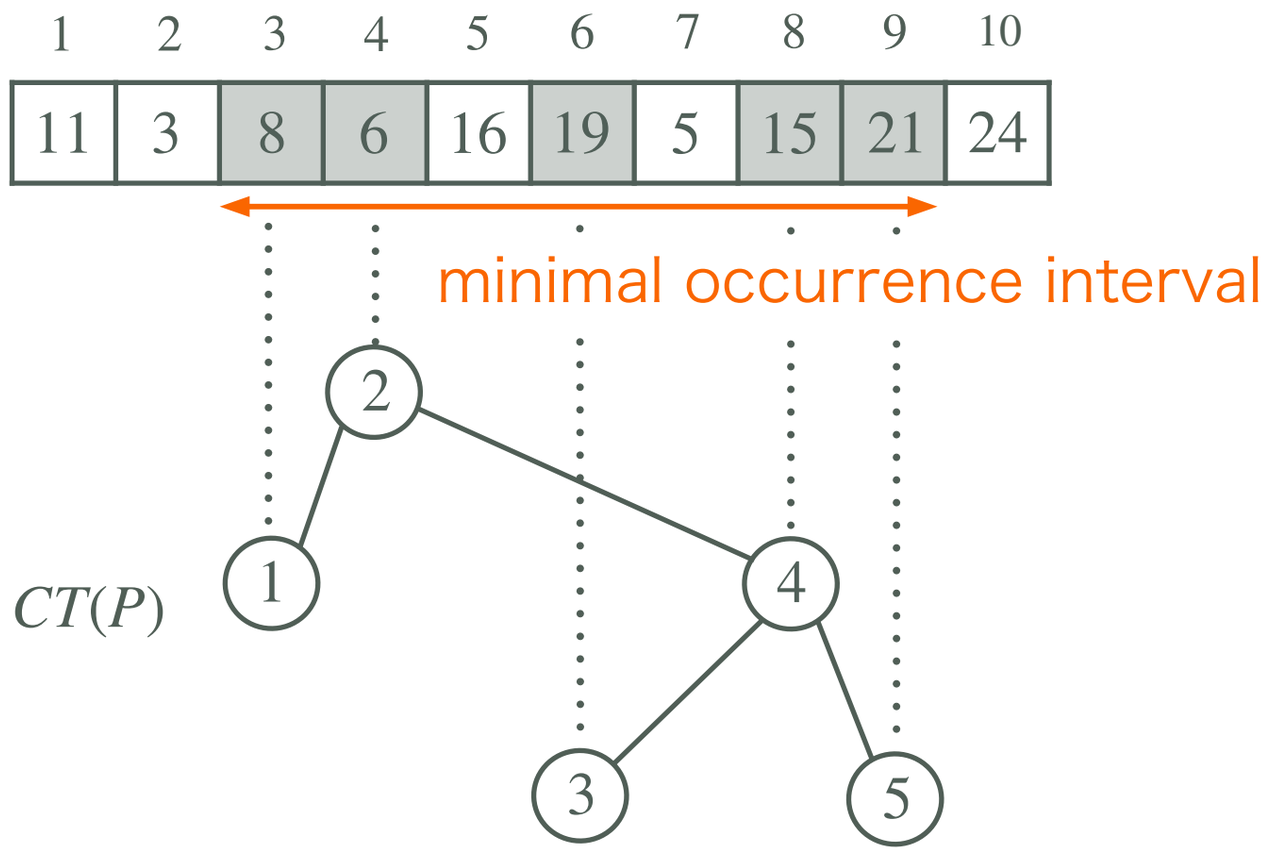
Definition of occurrence

- An interval $[\ell, r]$ is said to be an occurrence interval

  if $CT(P) = CT(T')$ for some subsequence of $T[\ell \mathinner{..} r]$.
- An occurrence interval $[\ell, r]$ is said to be minimal

  if there is no occurrence interval $[\ell', r']$ such that $[\ell', r'] \subsetneq [\ell, r]$.



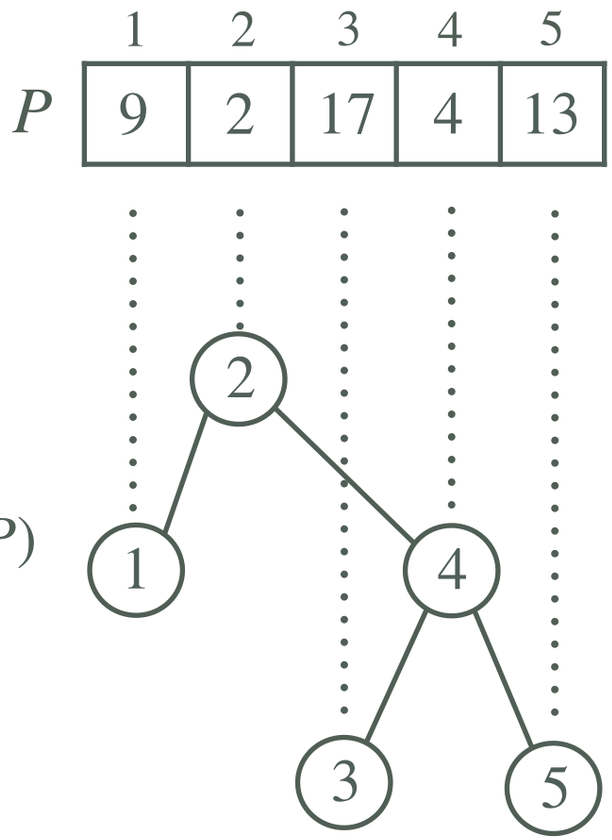


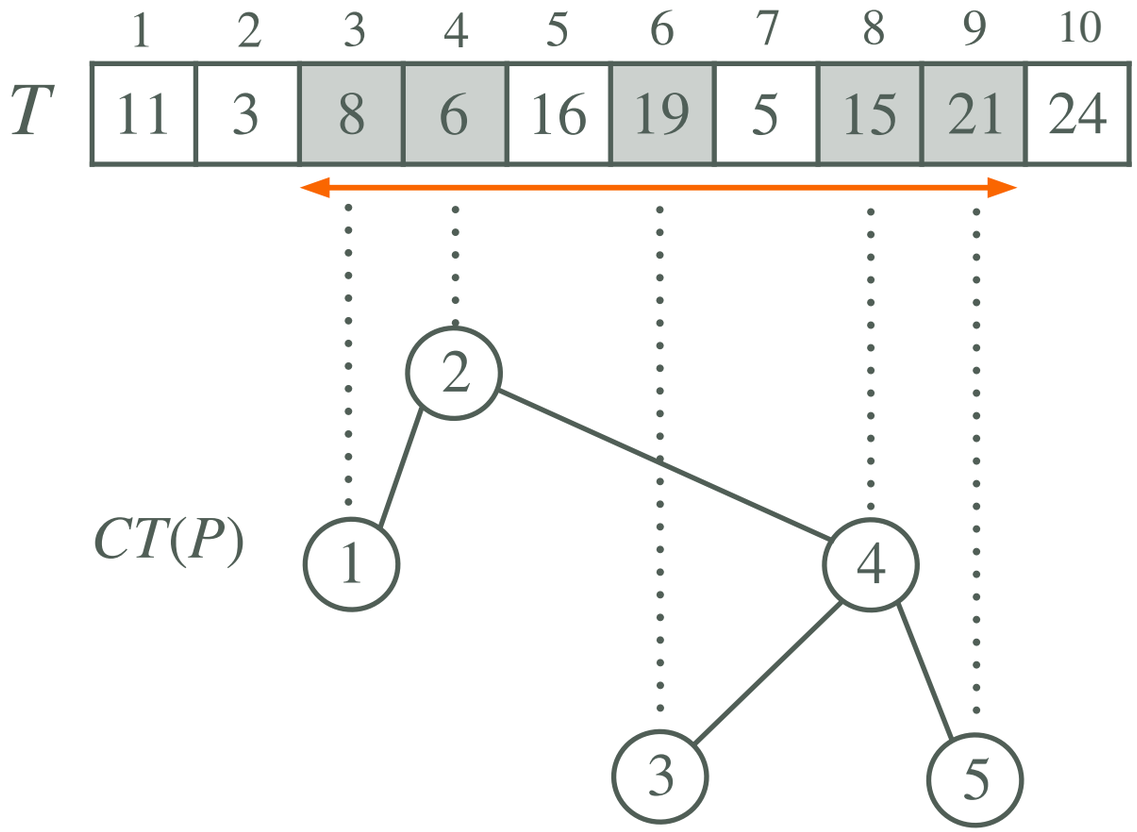
Our problem

CTMSeq problem [This work]

Input: Text $T[1..n]$ and pattern $P[1..m]$.

Output: Every minimal occurrence intervals.





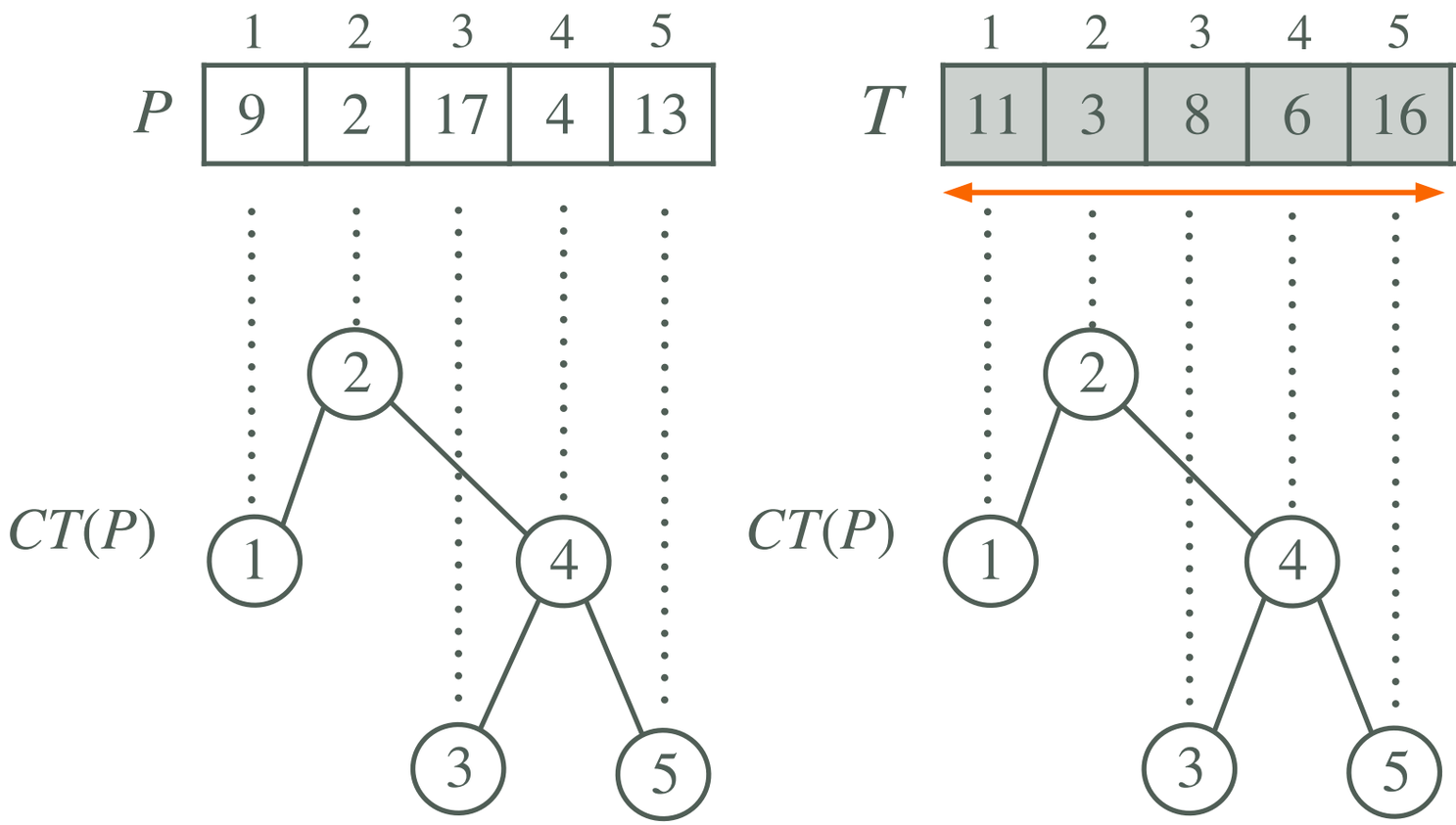
Output: [3,9]

Our problem

CTMSeq problem [This work]

Input: Text $T[1..n]$ and pattern $P[1..m]$.

Output: Every minimal occurrence intervals.



Output: [3,9], [1,5]

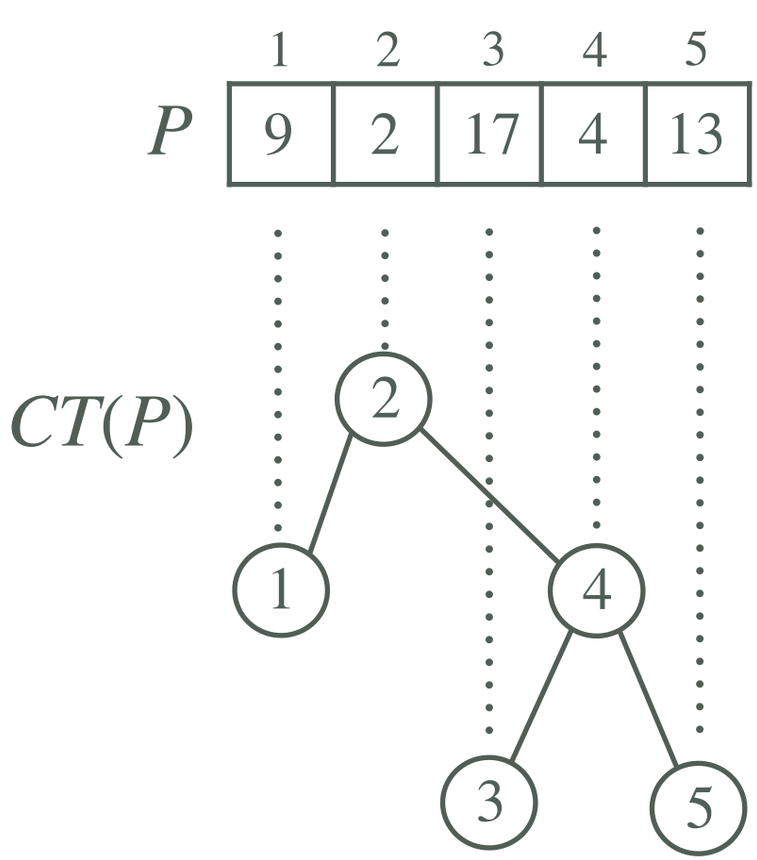
Our results

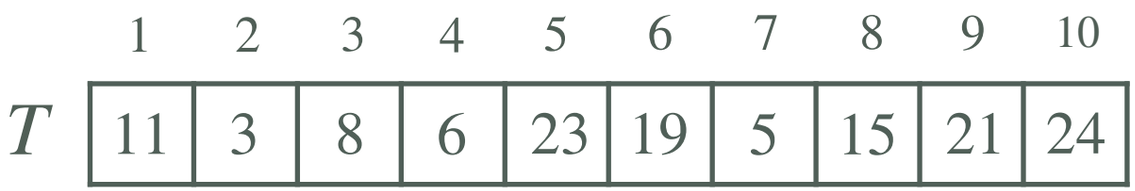
algorithm	time	space
`simple`	$O(mn^2)$	$O(mn)$
`vEB-HL`	$O(mn \log\log n)$	$O(n \log m)$

- n is the length of text T.
- m is the length of pattern P.

Observation

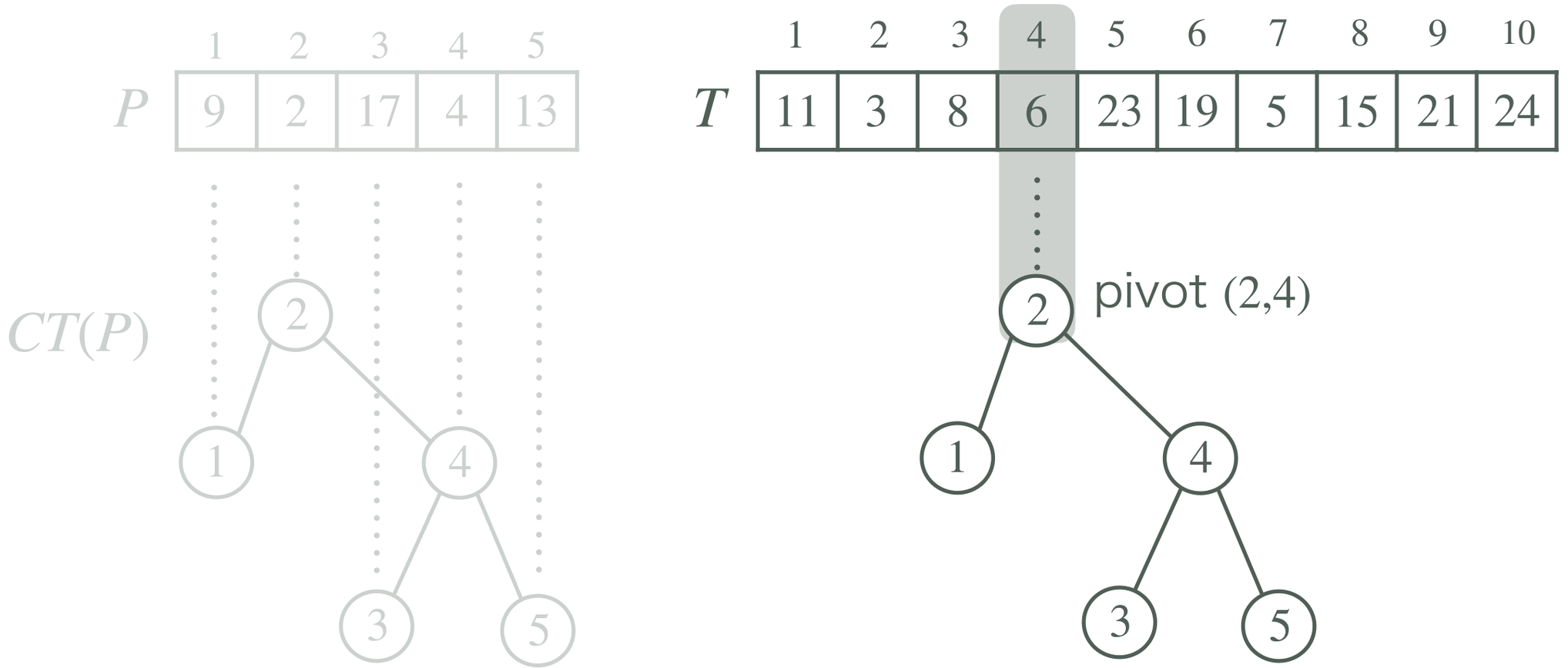
- For a vertex $v \in [m]$ and an index $i \in [n]$, we call a pair (v, i) pivot.





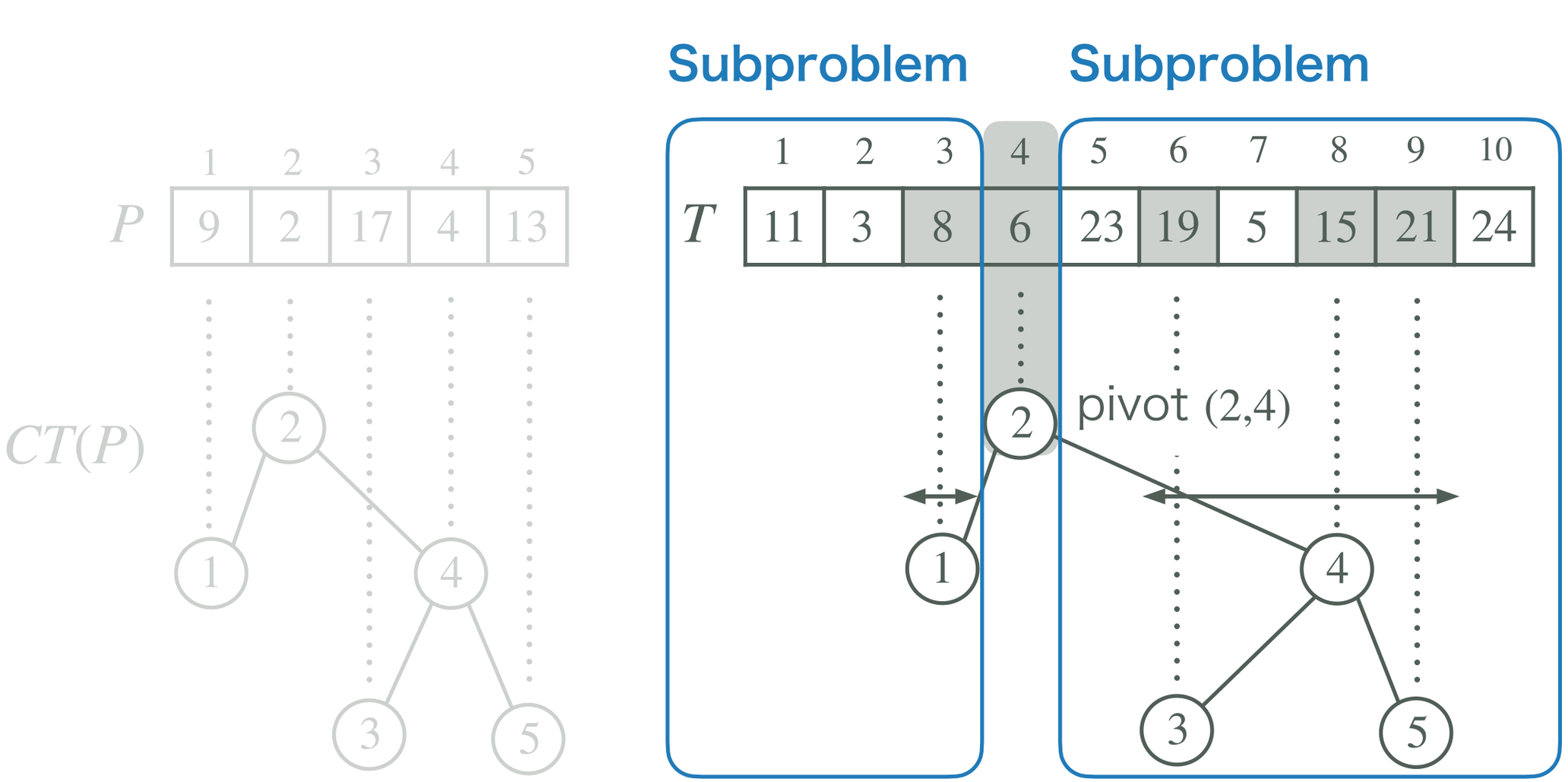
Observation

- For a vertex $v \in [m]$ and an index $i \in [n]$, we call a pair (v, i) pivot.
- Fix a position between the root of $CT(P)$ and an index of T by a pivot.



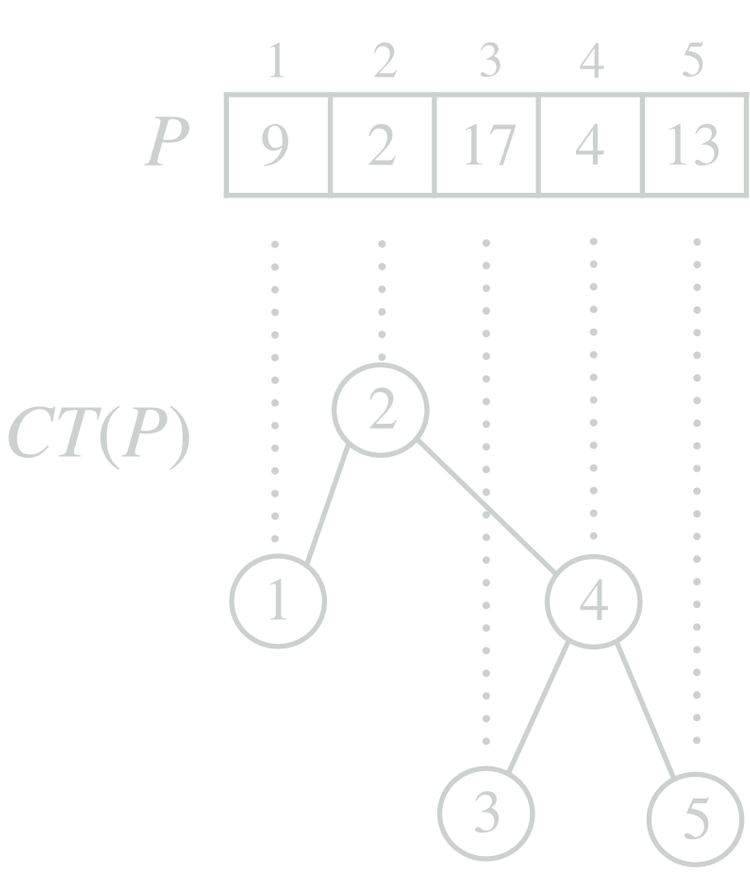
Observation

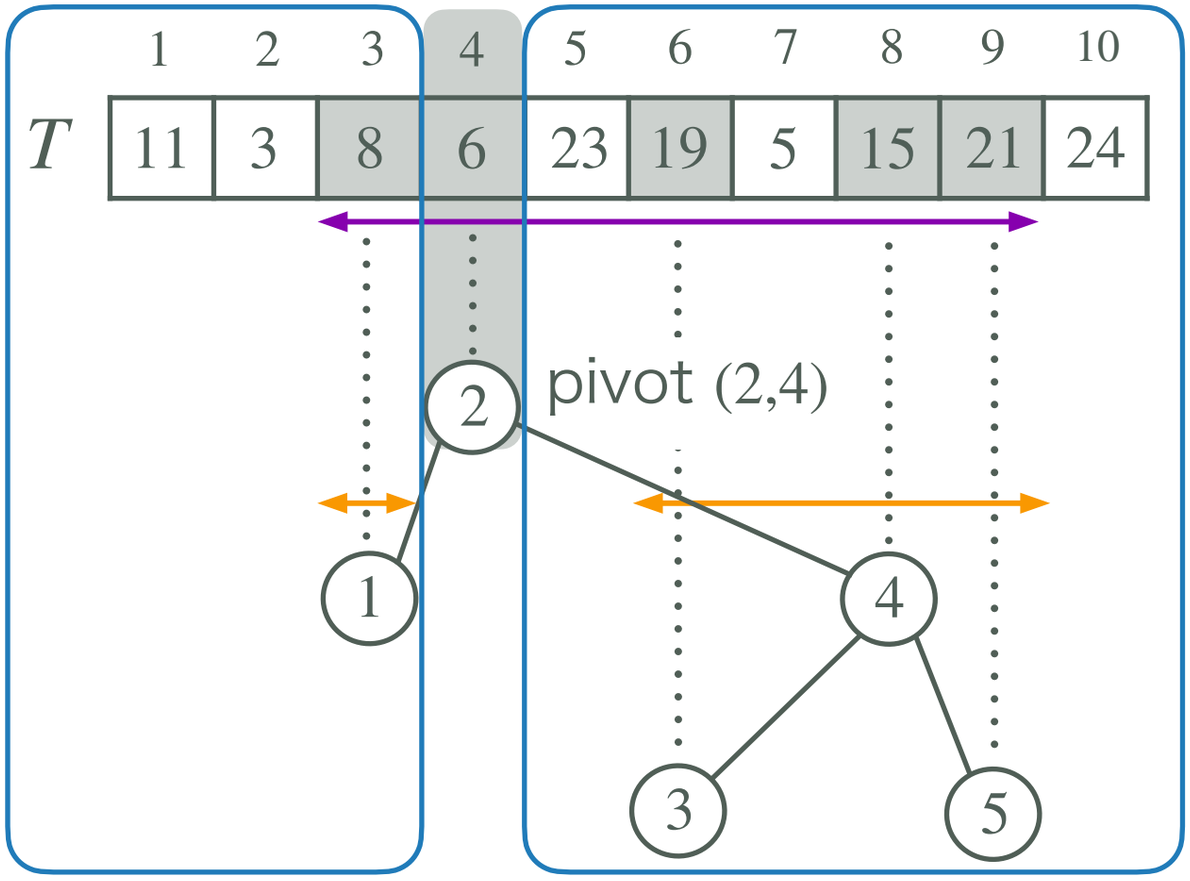
- For a vertex $v \in [m]$ and an index $i \in [n]$, we call a pair (v, i) pivot.
- Fix a position between the root of $CT(P)$ and an index of T by a pivot.
- Independent subproblems appear in the left and right intervals with respect to the root position.



Observation

- For a vertex $v \in [m]$ and an index $i \in [n]$, we call a pair (v, i) pivot.
- Fix a position between the root of $CT(P)$ and an index of T by a pivot.
- Independent subproblems appear in the left and right intervals with respect to the root position.
- Occurrence intervals for the left and right subtrees can be used to construct the overall occurrence interval.





Fixed-interval

Definition (fixed-interval)

A fixed-interval with the pivot (v, i) is an occurrence interval such that the corresponding locations between v and i are fixed.

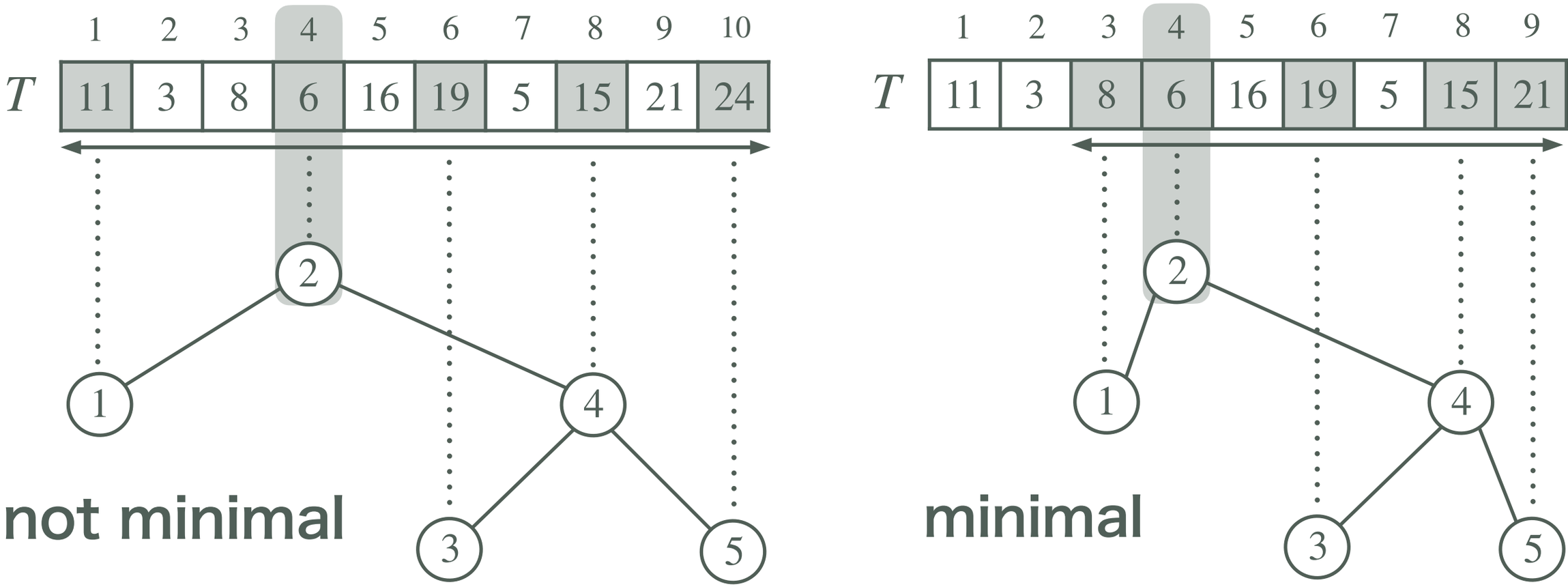


Figure: Example of fixed-intervals with the pivot (2,4)

Uniqueness

Lemma (uniqueness)

For any pivot $(v, i) \in [m] \times [n]$, there exists at most one minimal fixed-interval with the pivot (v, i)

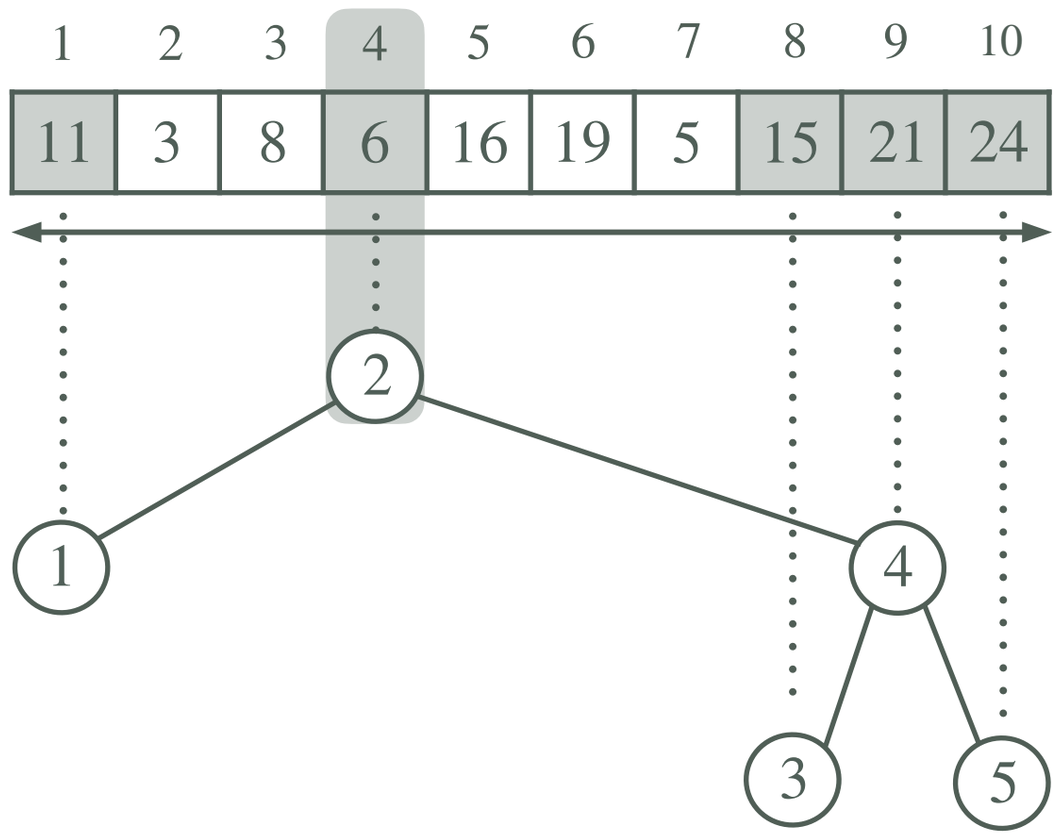


Figure: The minimal fixed-intervals [3,9] with the pivot (2,4)

Uniqueness

Lemma (uniqueness)

For any pivot $(v, i) \in [m] \times [n]$, there exists at most one minimal fixed-interval with the pivot (v, i)

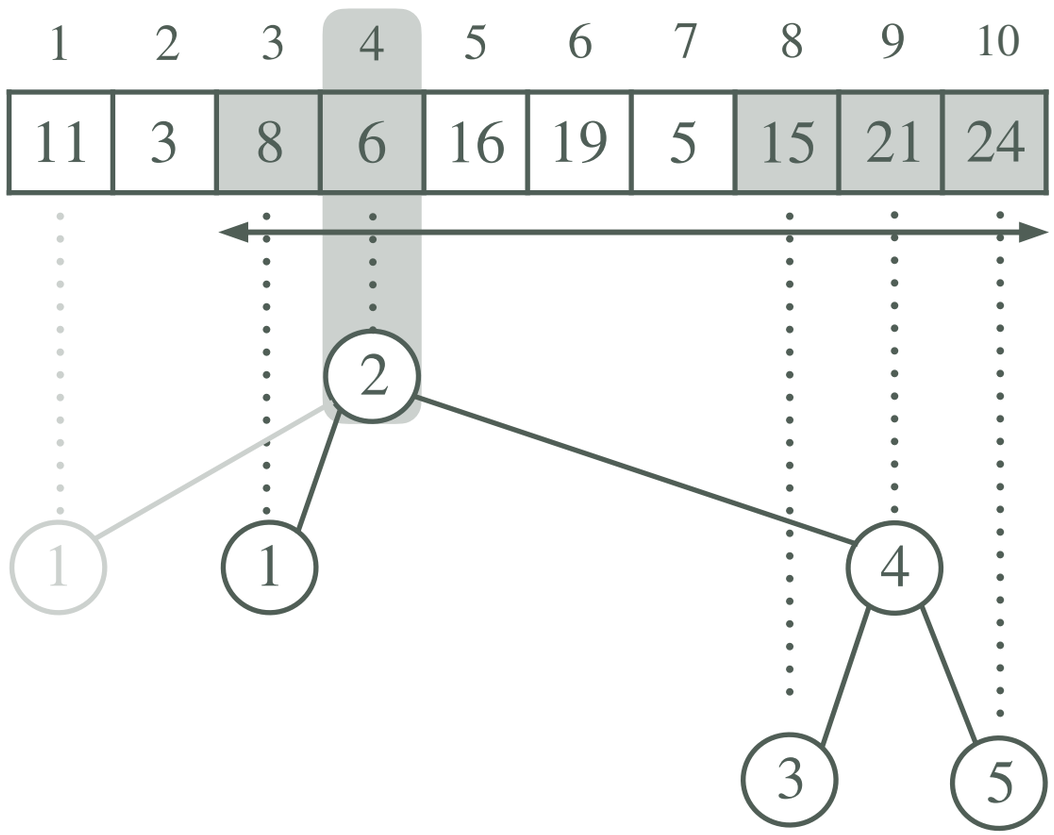


Figure: The minimal fixed-intervals [3,9] with the pivot (2,4)

Uniqueness

Lemma (uniqueness)

For any pivot $(v, i) \in [m] \times [n]$, there exists at most one minimal fixed-interval with the pivot (v, i)

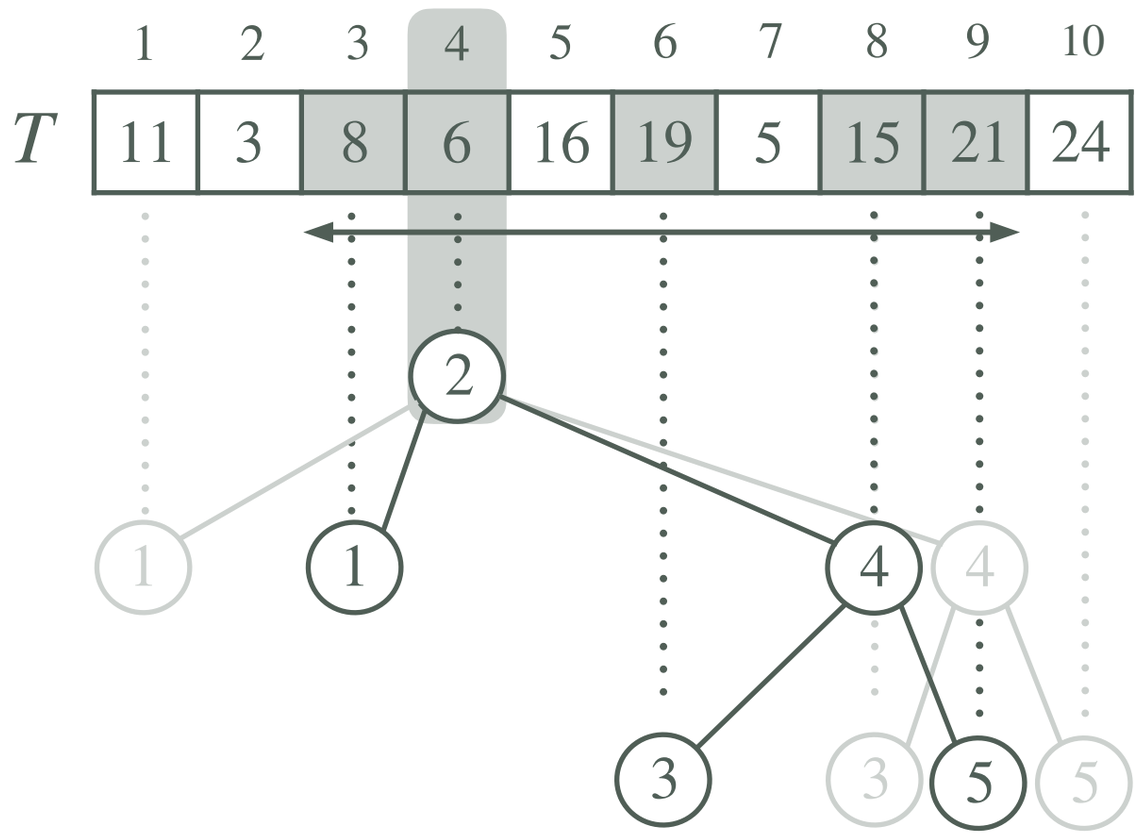


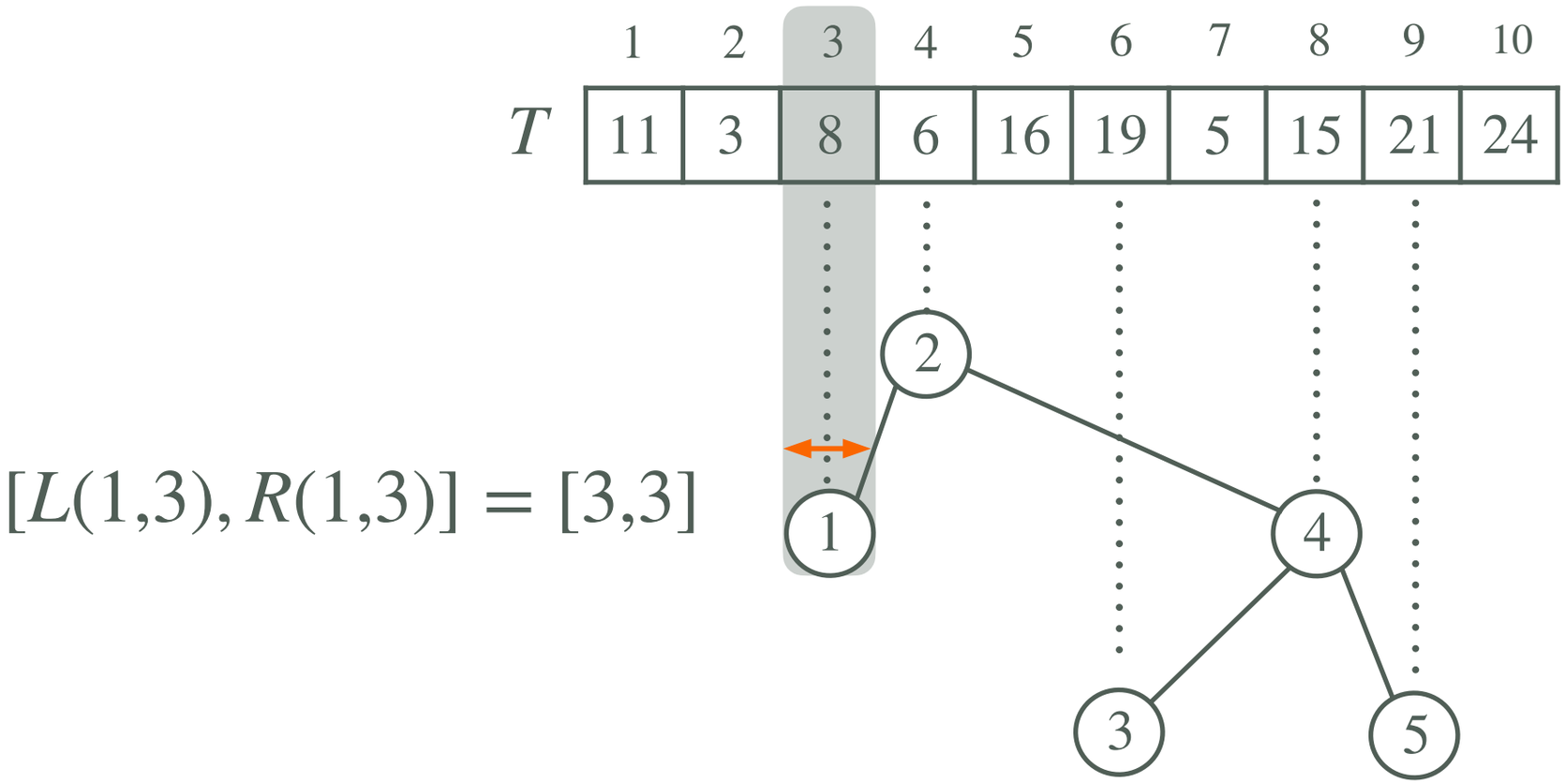
Figure: The minimal fixed-intervals [3,9] with the pivot (2,4)

The main Idea of our algorithm

Definition (DP table)

Let $[L(v,i), R(v,i)]$ be the minimal fixed-interval with the pivot (v,i).

- Compute minimal fixed-intervals for all pivot (v,i) in a bottom-up manner from the leaves using dynamic programming.

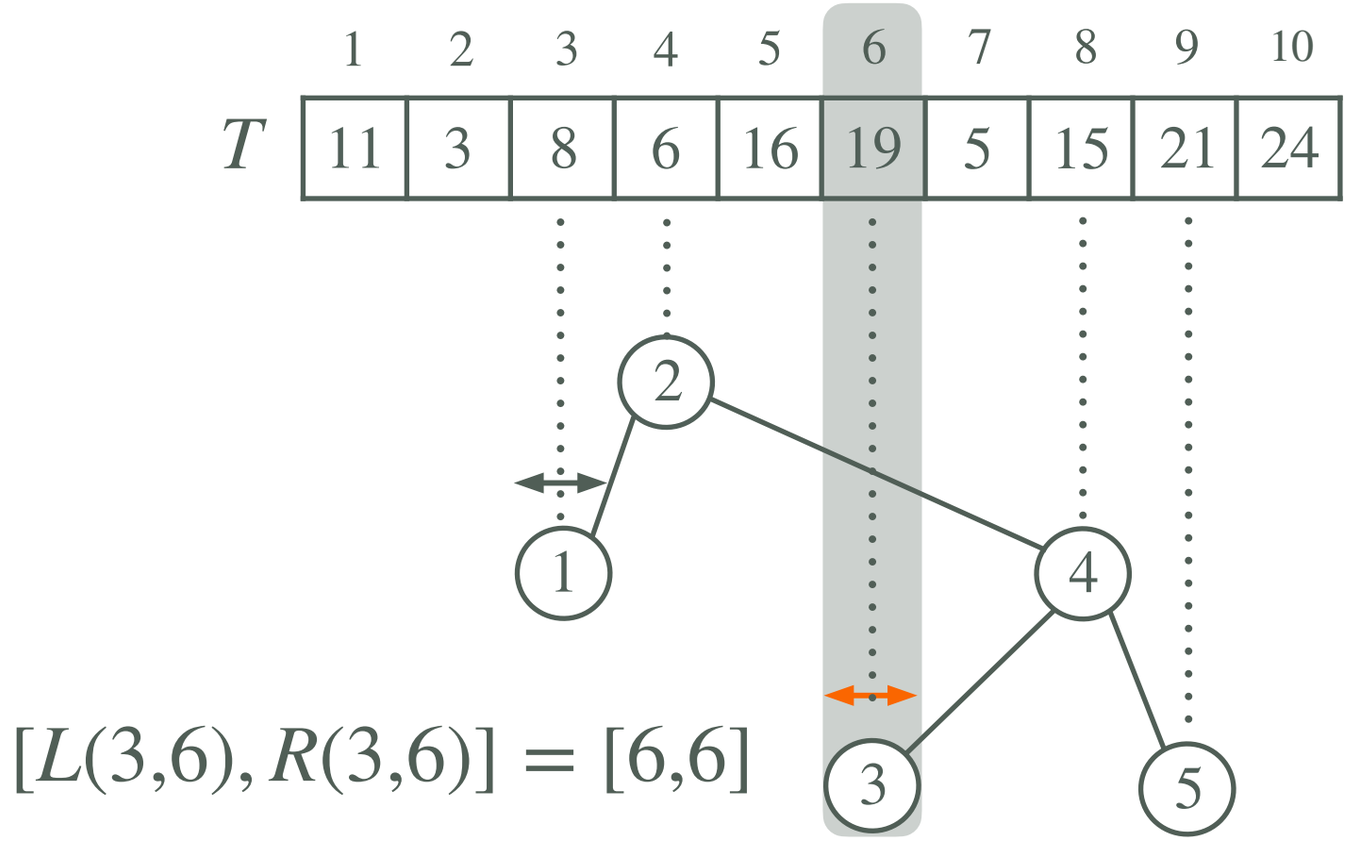


The main Idea of our algorithm

Definition (DP table)

Let $[L(v,i), R(v,i)]$ be the minimal fixed-interval with the pivot (v,i).

- Compute minimal fixed-intervals for all pivot (v,i) in a bottom-up manner from the leaves using dynamic programming.

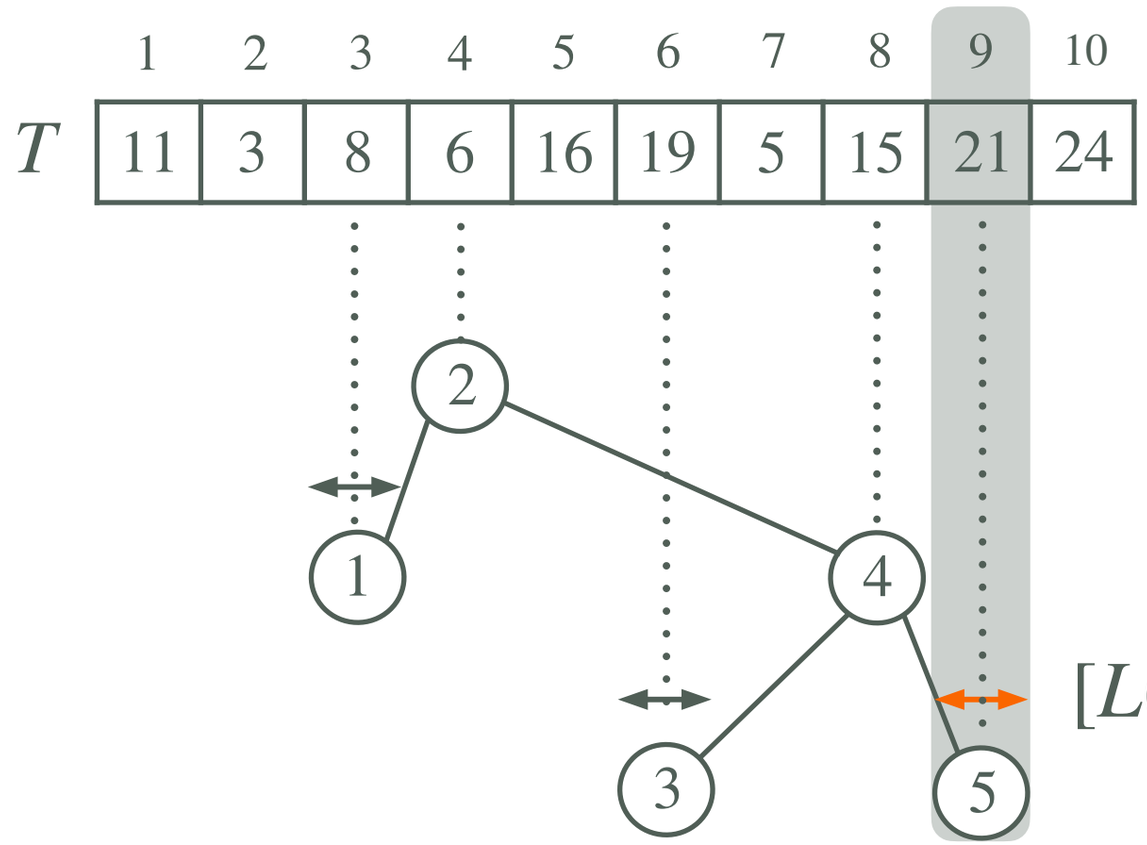


The main Idea of our algorithm

Definition (DP table)

Let $[L(v,i), R(v,i)]$ be the minimal fixed-interval with the pivot (v,i).

- Compute minimal fixed-intervals for all pivot (v,i) in a bottom-up manner from the leaves using dynamic programming.

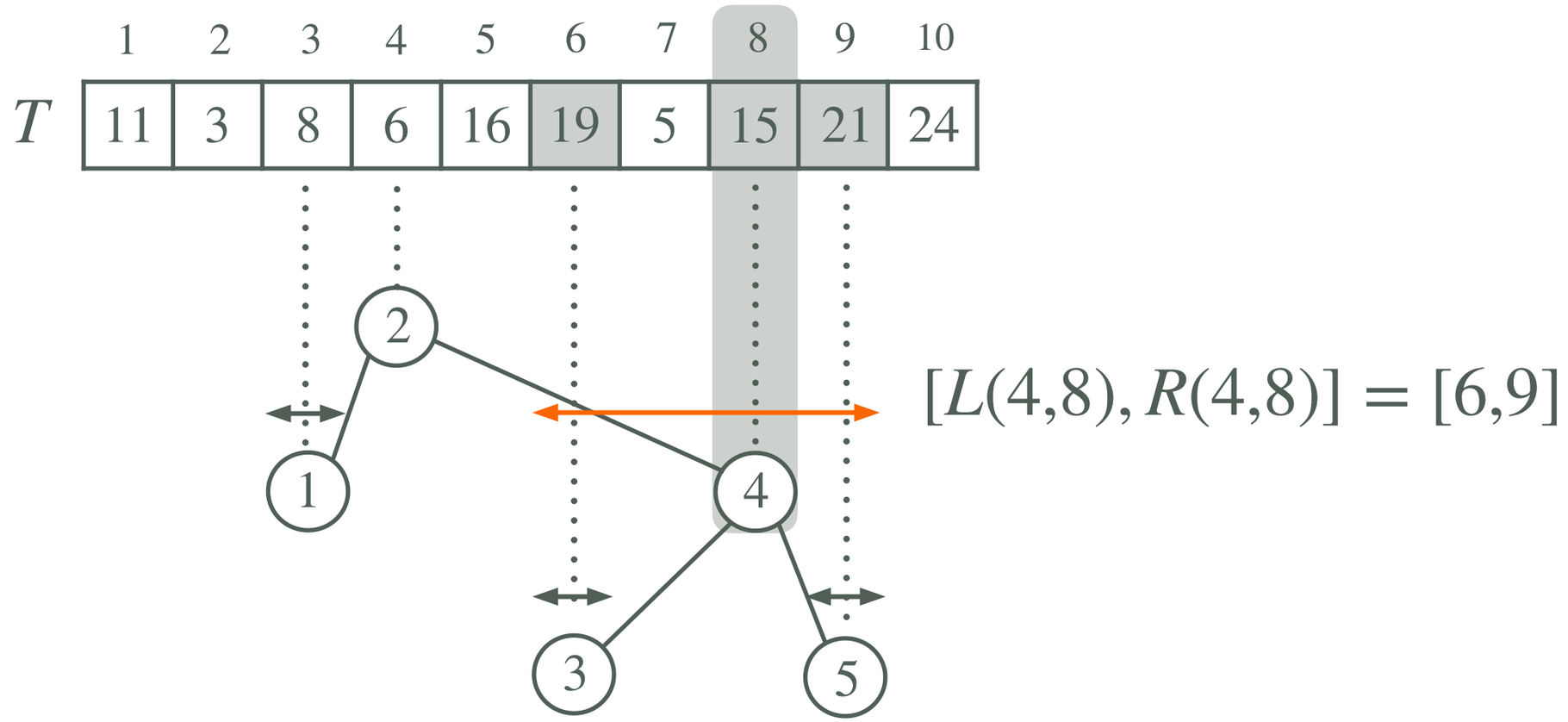


The main Idea of our algorithm

Definition (DP table)

Let $[L(v,i), R(v,i)]$ be the minimal fixed-interval with the pivot (v,i).

- Compute minimal fixed-intervals for all pivot (v,i) in a bottom-up manner from the leaves using dynamic programming.

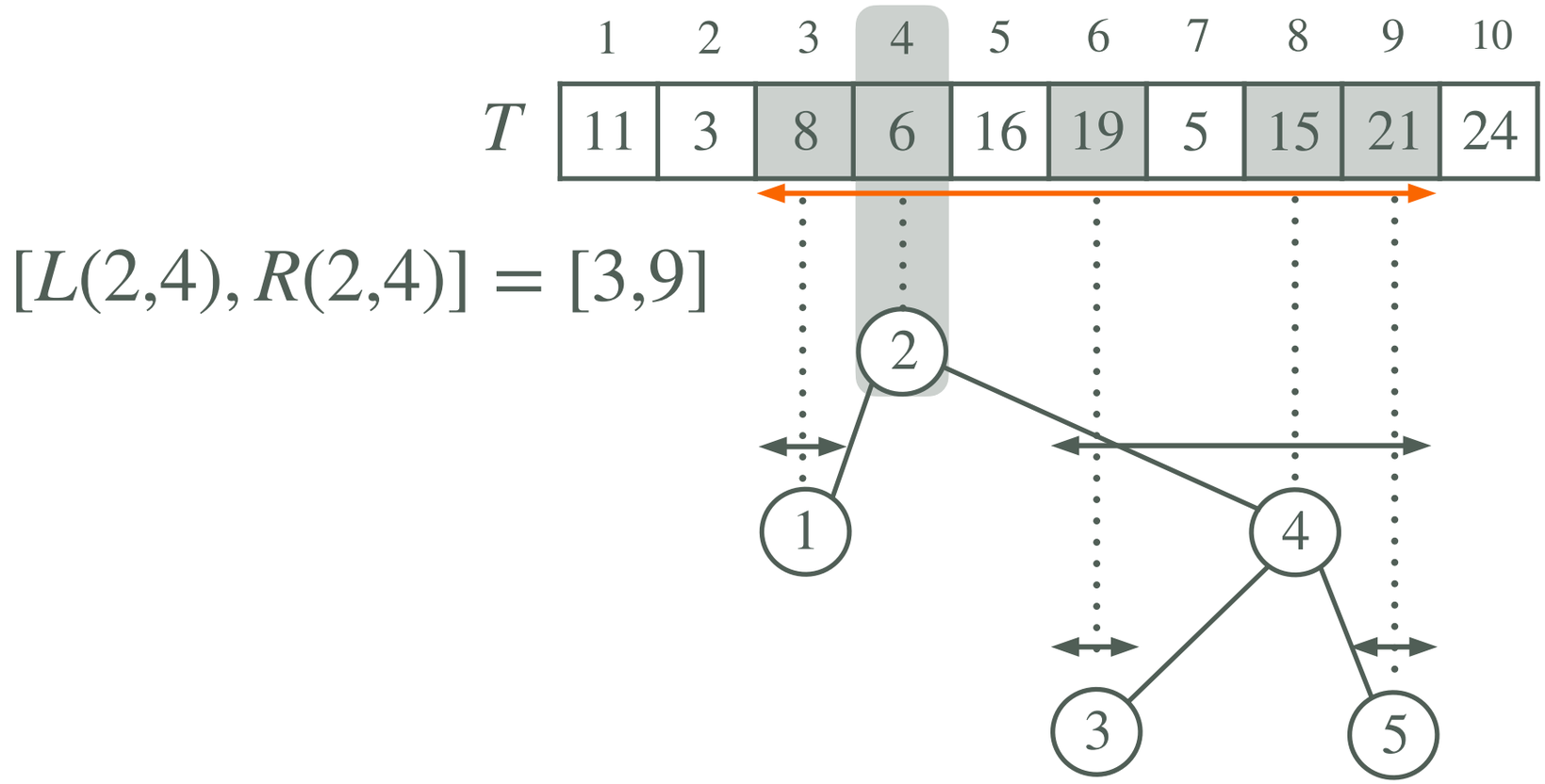


The main Idea of our algorithm

Definition (DP table)

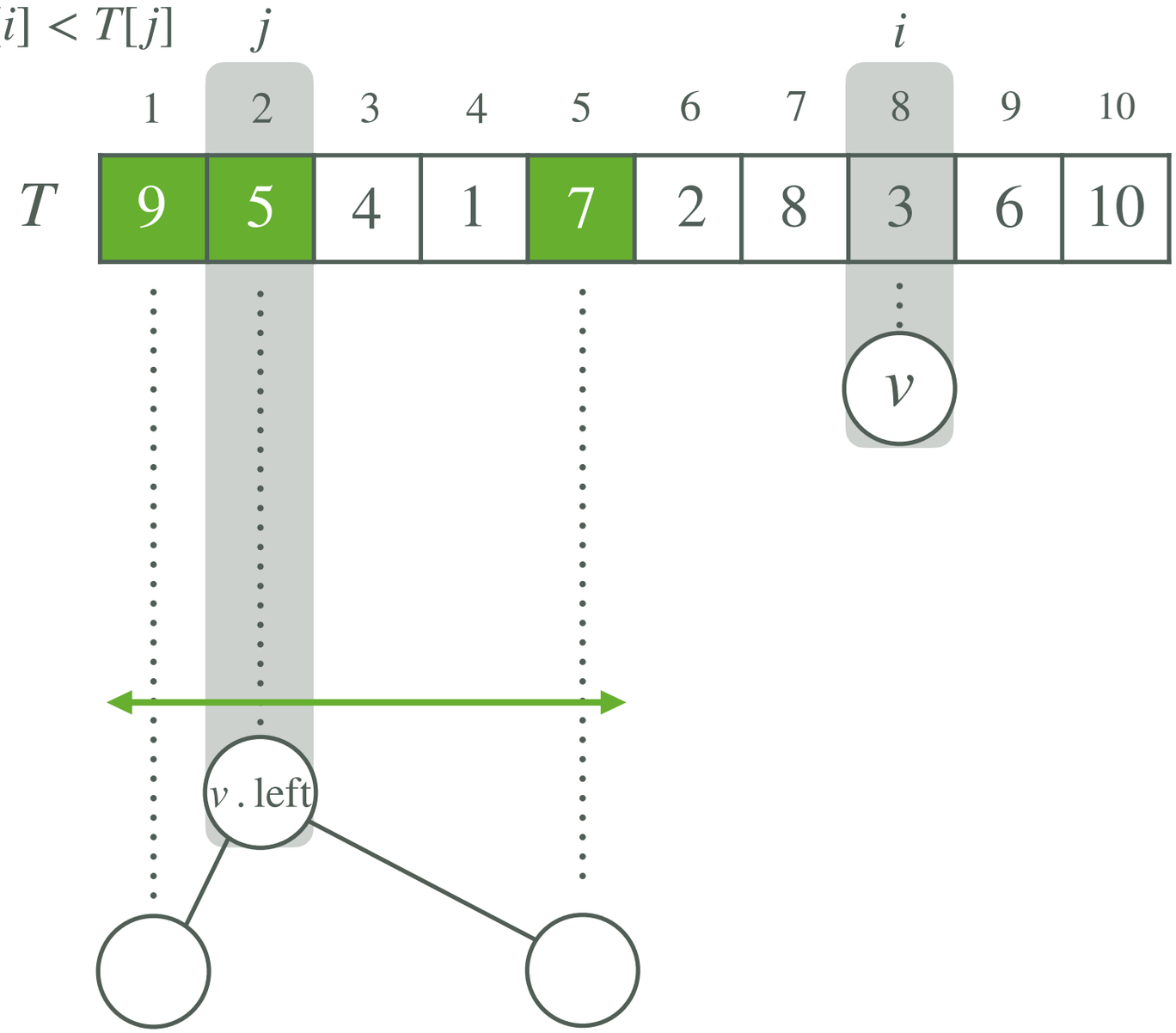
Let $[L(v,i), R(v,i)]$ be the minimal fixed-interval with the pivot (v,i).

- Compute minimal fixed-intervals for all pivot (v,i) in a bottom-up manner from the leaves using dynamic programming.



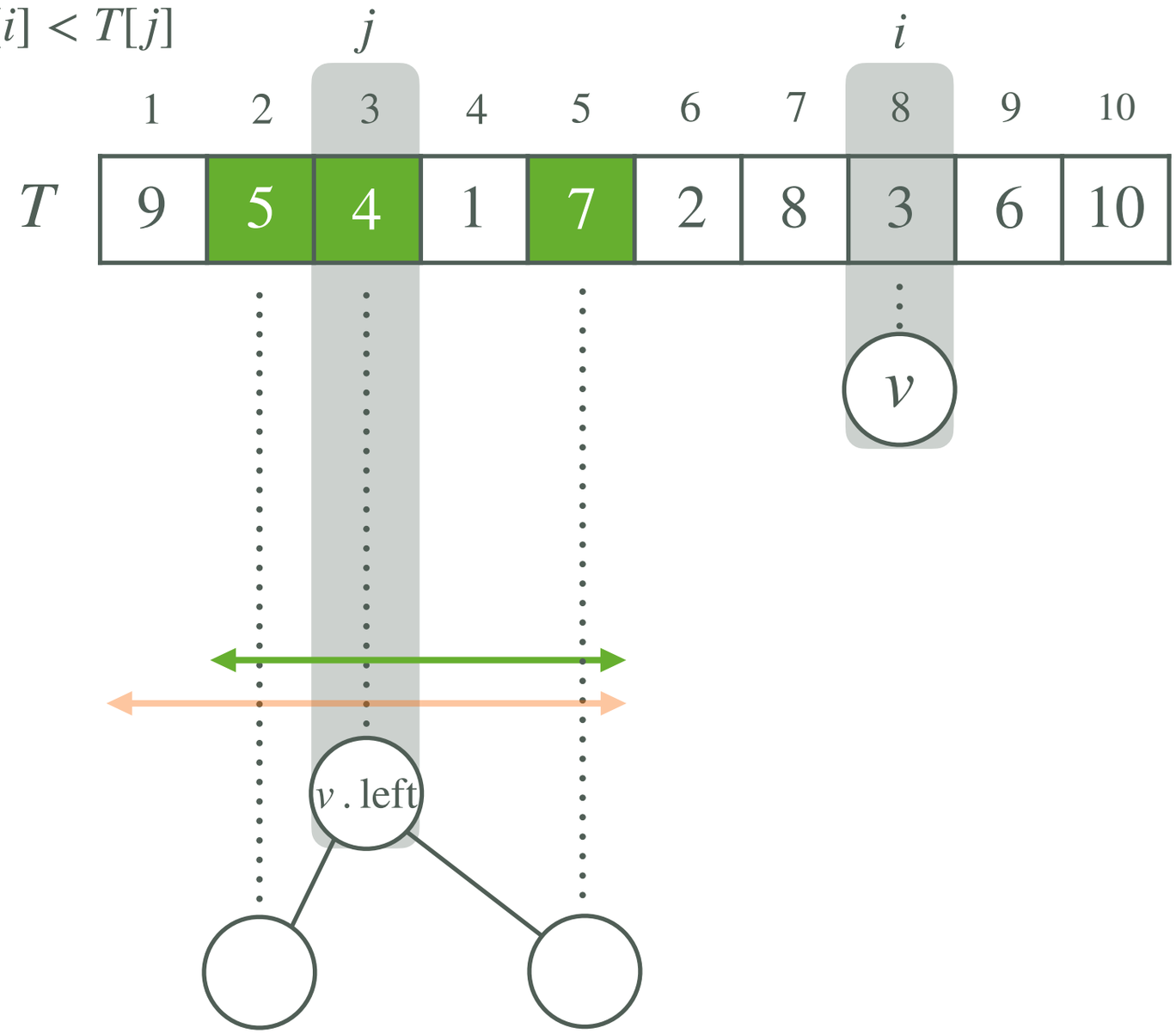
Recurrence formula

$$L(v,i) = \begin{cases} i & \text{if } v\,.\,\text{left} = \text{null}, \\ \max\limits_{\substack{1 \le j \le i-1 \\ T[i] < T[j]}} \{L(v\,.\,\text{left}, j) \mid R(v\,.\,\text{left}, j) < i\} & \text{otherwise}\,. \end{cases}$$



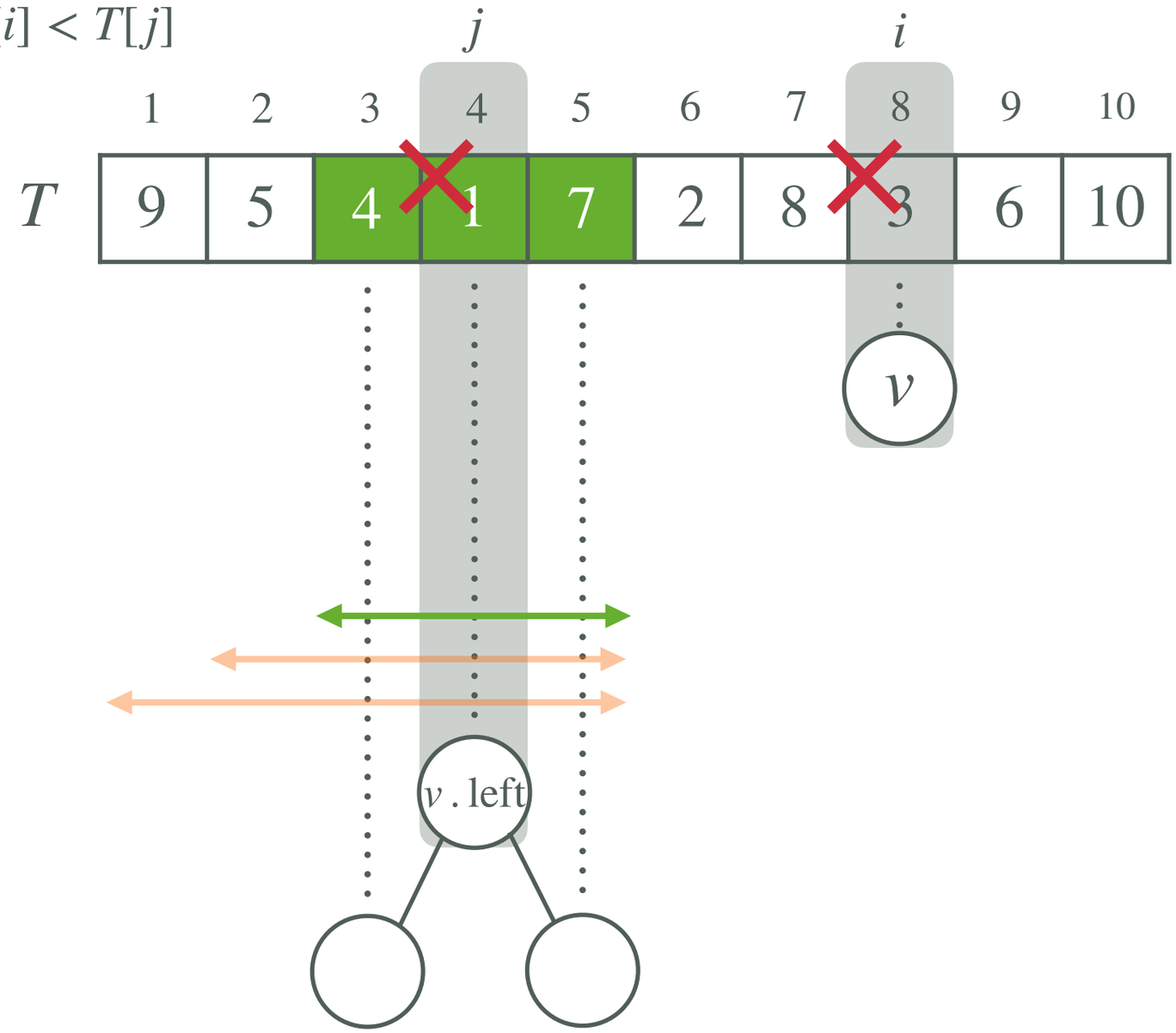
Recurrence formula

$$L(v,i) = \begin{cases} i & \text{if } v\,.\,\text{left} = \text{null}, \\ \max\limits_{\substack{1 \le j \le i-1 \\ T[i] < T[j]}} \{L(v\,.\,\text{left}, j) \mid R(v\,.\,\text{left}, j) < i\} & \text{otherwise}. \end{cases}$$



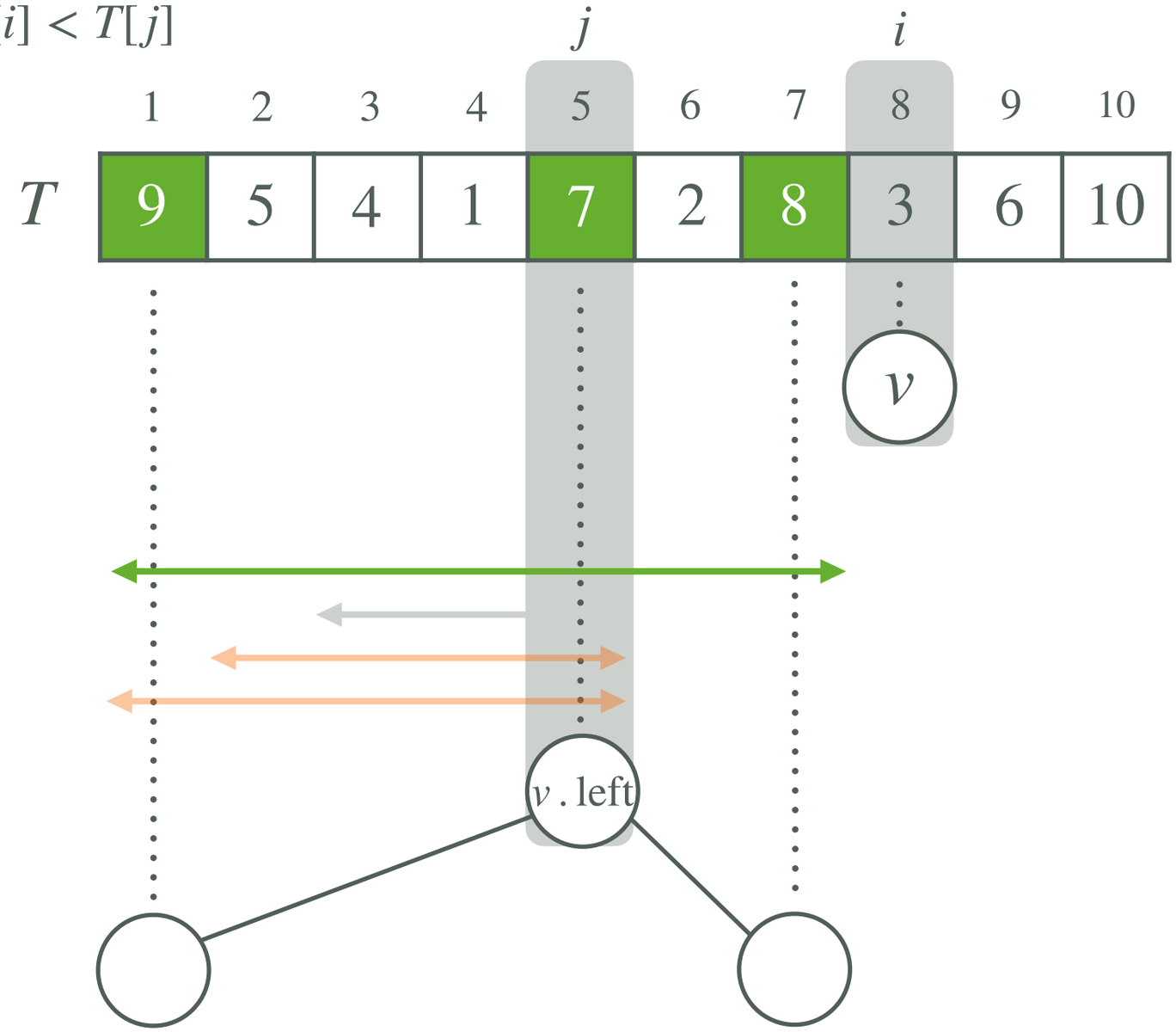
Recurrence formula

$$L(v,i) = \begin{cases} i & \text{if } v\,.\,\text{left} = \text{null}, \\ \max\limits_{1 \le j \le i-1} \{L(v\,.\,\text{left}, j) \mid R(v\,.\,\text{left}, j) < i\} & \text{otherwise}. \\ \times\ T[i] < T[j] & \end{cases}$$



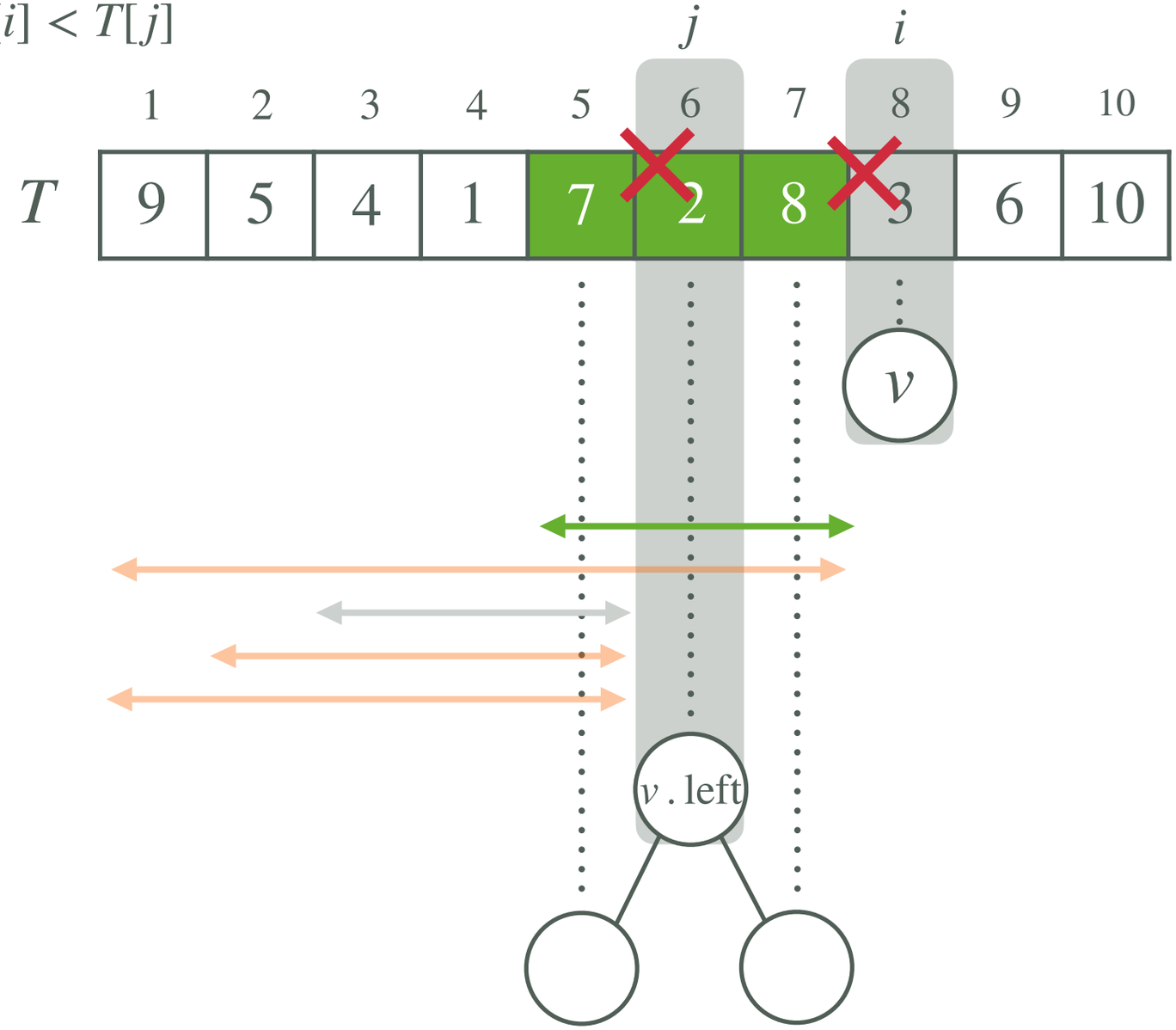
Recurrence formula

$$L(v,i) = \begin{cases} i & \text{if } v\,.\,\text{left} = \text{null}, \\ \max\limits_{\substack{1 \le j \le i-1 \\ T[i] < T[j]}} \{L(v\,.\,\text{left}, j) \mid R(v\,.\,\text{left}, j) < i\} & \text{otherwise}\,. \end{cases}$$



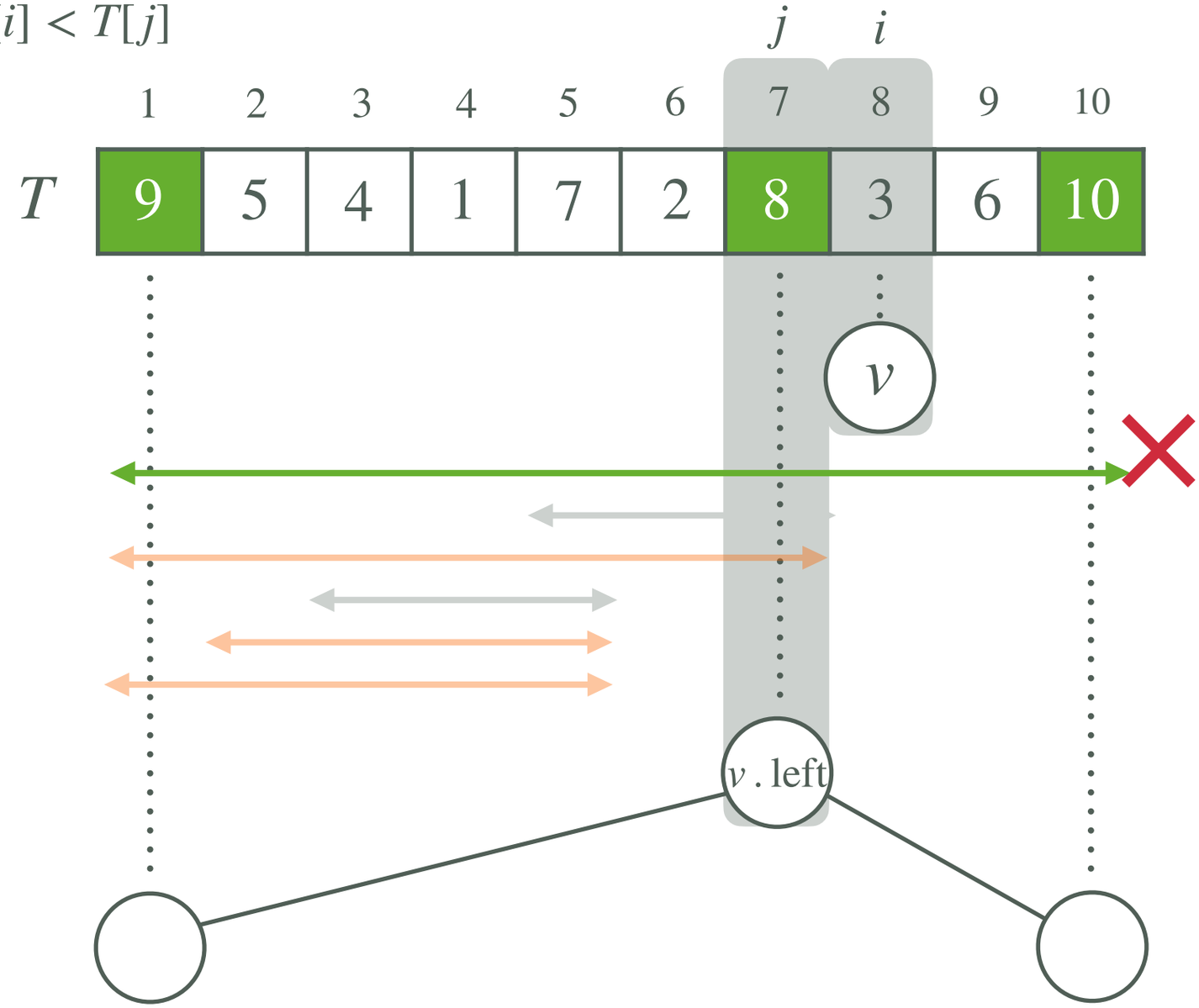
Recurrence formula

$$L(v,i) = \begin{cases} i & \text{if } v\,.\,\text{left} = \text{null}, \\ \max\limits_{1 \le j \le i-1} \{L(v\,.\,\text{left}, j) \mid R(v\,.\,\text{left}, j) < i\} & \text{otherwise}. \\ \text{~~}T[i] < T[j]\text{~~} \end{cases}$$



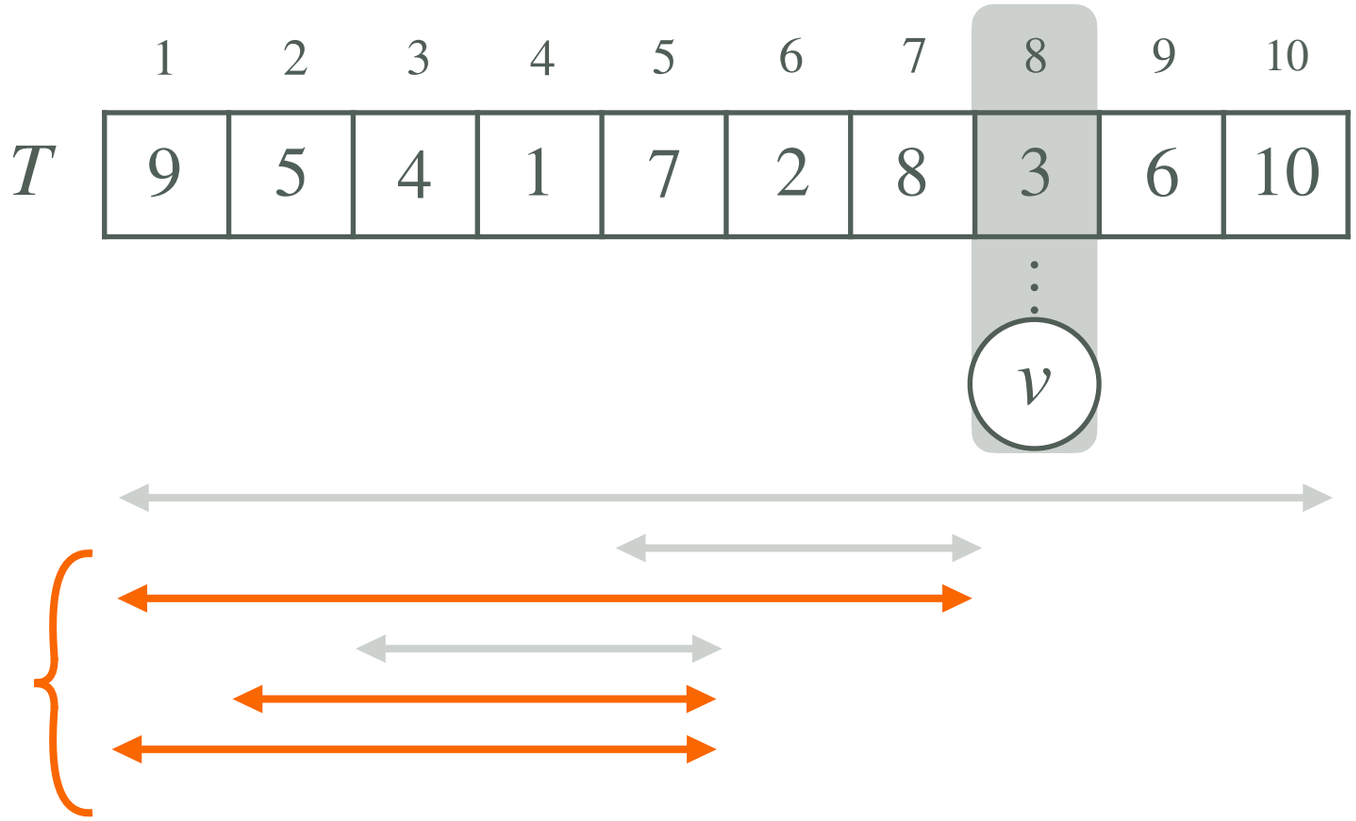
Recurrence formula

$$L(v,i) = \begin{cases} i & \text{if } v\,.\,\text{left} = \text{null}, \\ \max\limits_{\substack{1 \le j \le i-1 \\ T[i] < T[j]}} \{L(v\,.\,\text{left}, j) \mid R(v\,.\,\text{left}, j) < i\} & \text{otherwise}\,. \end{cases}$$



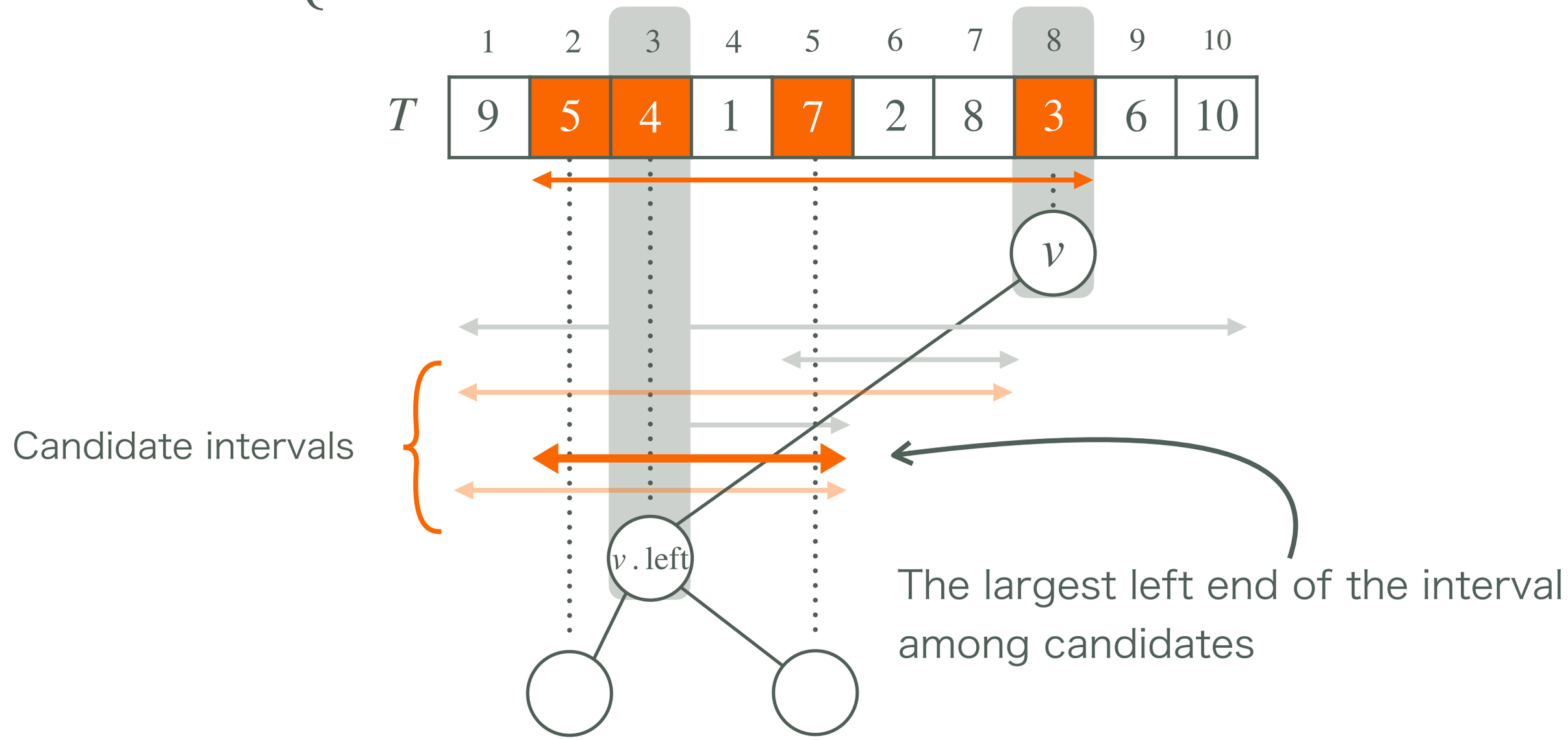
Recurrence formula

$$L(v,i) = \begin{cases} i & \text{if } v\,.\,\text{left} = \text{null}, \\ \max\limits_{\substack{1 \le j \le i-1 \\ T[i] < T[j]}} \{L(v\,.\,\text{left}, j) \mid R(v\,.\,\text{left}, j) < i\} & \text{otherwise}\,. \end{cases}$$



Recurrence formula

$$L(v,i) = \begin{cases} i & \text{if } v\,.\,\text{left} = \text{null}, \\ \max\limits_{\substack{1 \le j \le i-1 \\ T[i] < T[j]}} \{L(v\,.\,\text{left}, j) \mid R(v\,.\,\text{left}, j) < i\} & \text{otherwise}\,. \end{cases}$$



Computational complexity

Theorem. 1

The `CTMSeq` problem can be solved in $O(mn^2)$ time using $O(mn)$ space.

- The size of the tables $L(v, i)$ and $R(v, i)$ is $\Theta(mn)$ and the time complexity to compute one cell is $O(n)$.
- By tracing these tables, we can obtain a concrete subsequence of T in $O(m)$ time for each minimal occurrence interval.

$$L(v,i) = \begin{cases} i & \text{if } v\,.\,\text{left} = \text{null}, \\ \max\limits_{\substack{1 \le j \le i-1 \\ T[i] < T[j]}} \{L(v\,.\,\text{left}, j) \mid R(v\,.\,\text{left}, j) < i\} & \text{otherwise}\,. \end{cases}$$

Improve time complexity

Theorem. 2

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(mn)$ space.

- Manage a set of the right end of candidate intervals with a predecessor dictionary for fast finding the target interval.

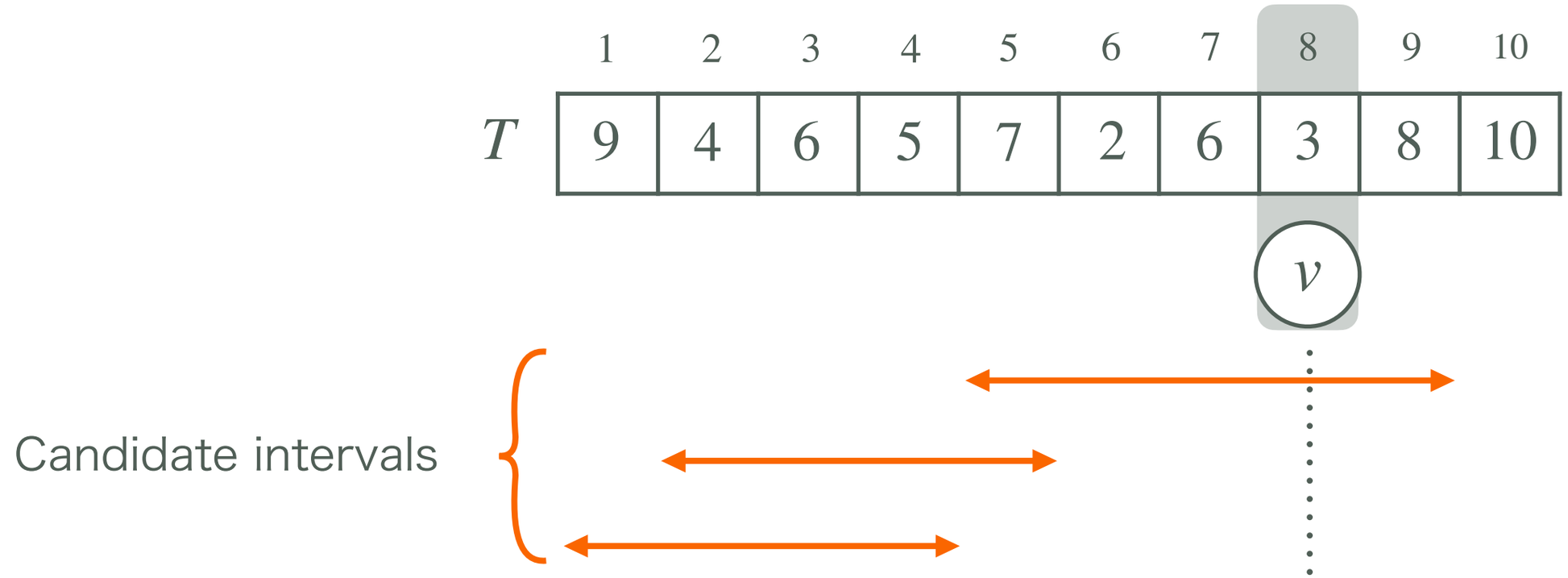


Figure: Find the target interval [2,5] by executing predecessor(8)

Improve time complexity

Theorem. 2

The `CTMSeq` problem can be solved in $O(mn \log \log n)$ time using $O(mn)$ space.

- Manage a set of the right end of candidate intervals with a predecessor dictionary for fast finding **the target interval**.

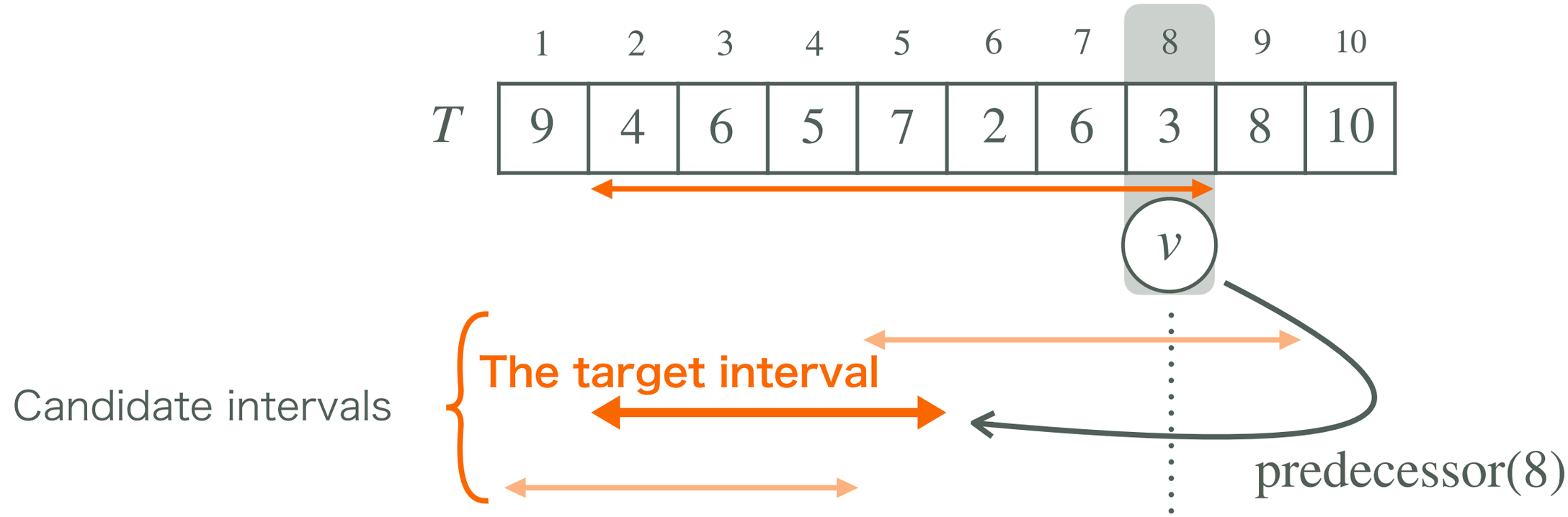


Figure: Find the target interval [2,5] by executing predecessor(8)

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.

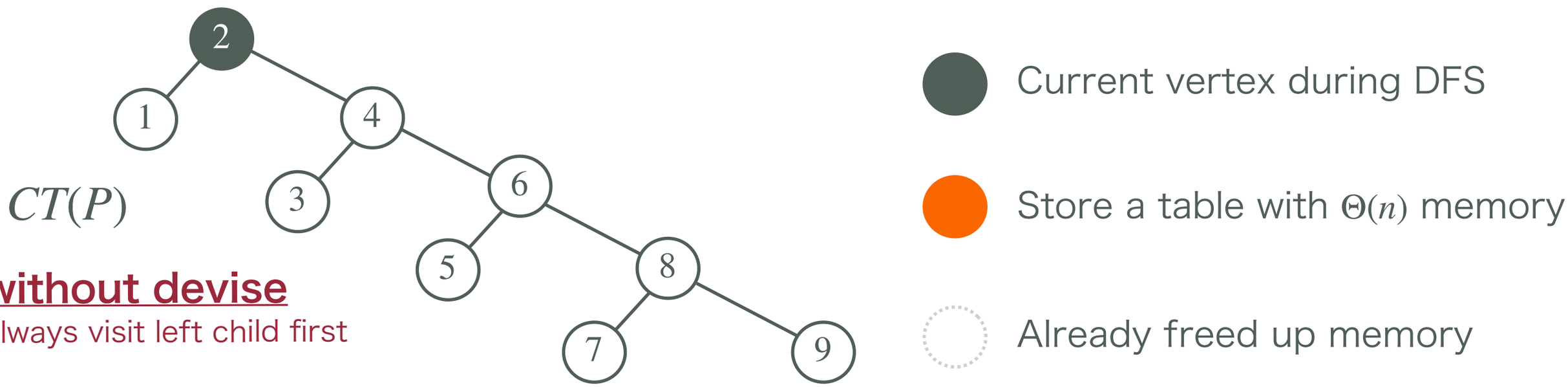


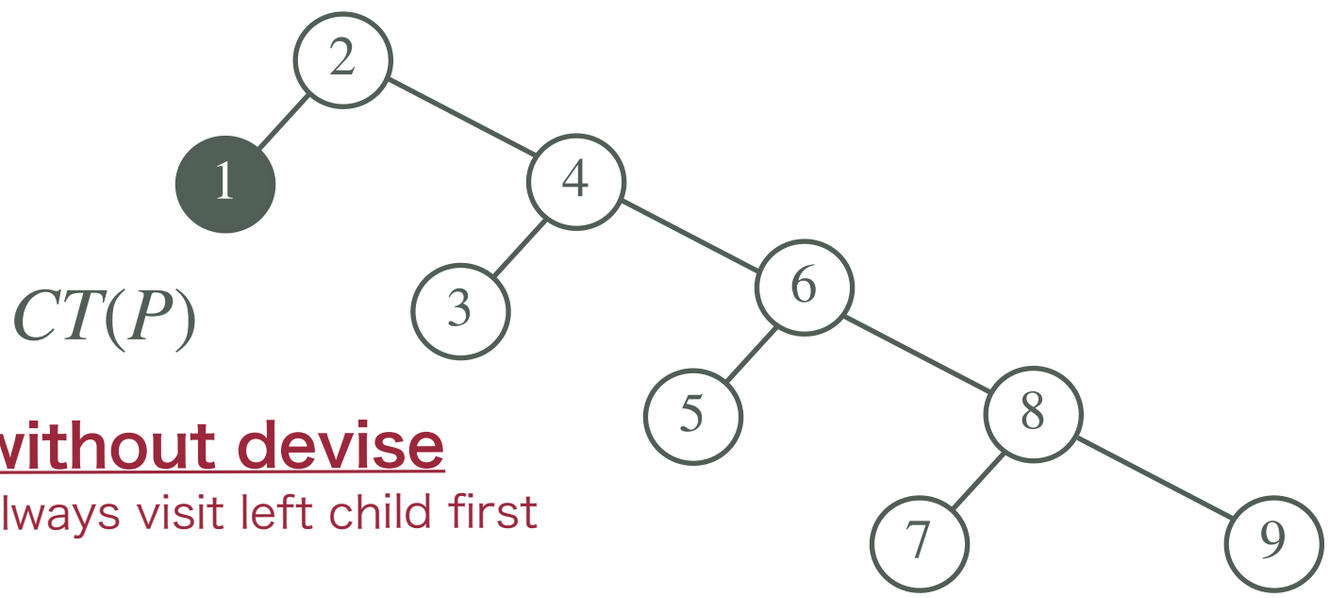
Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.



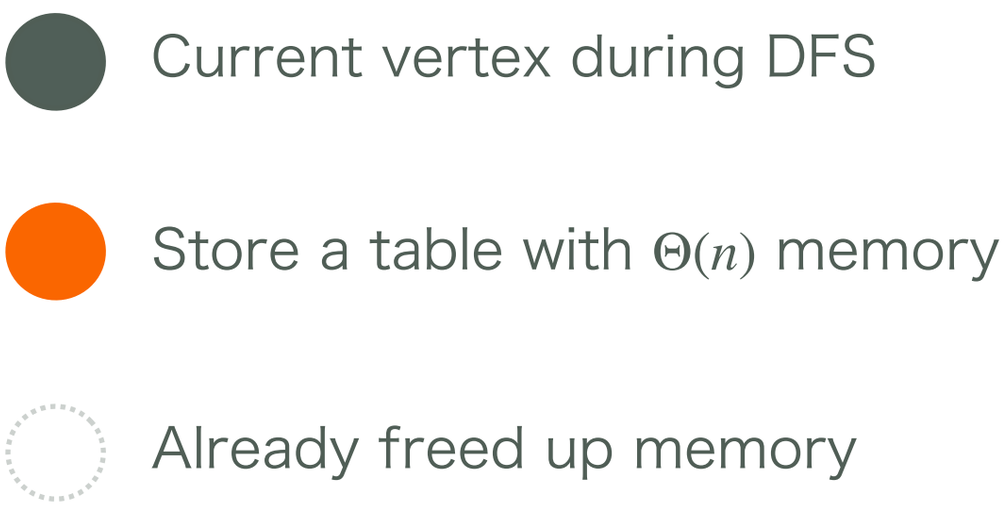


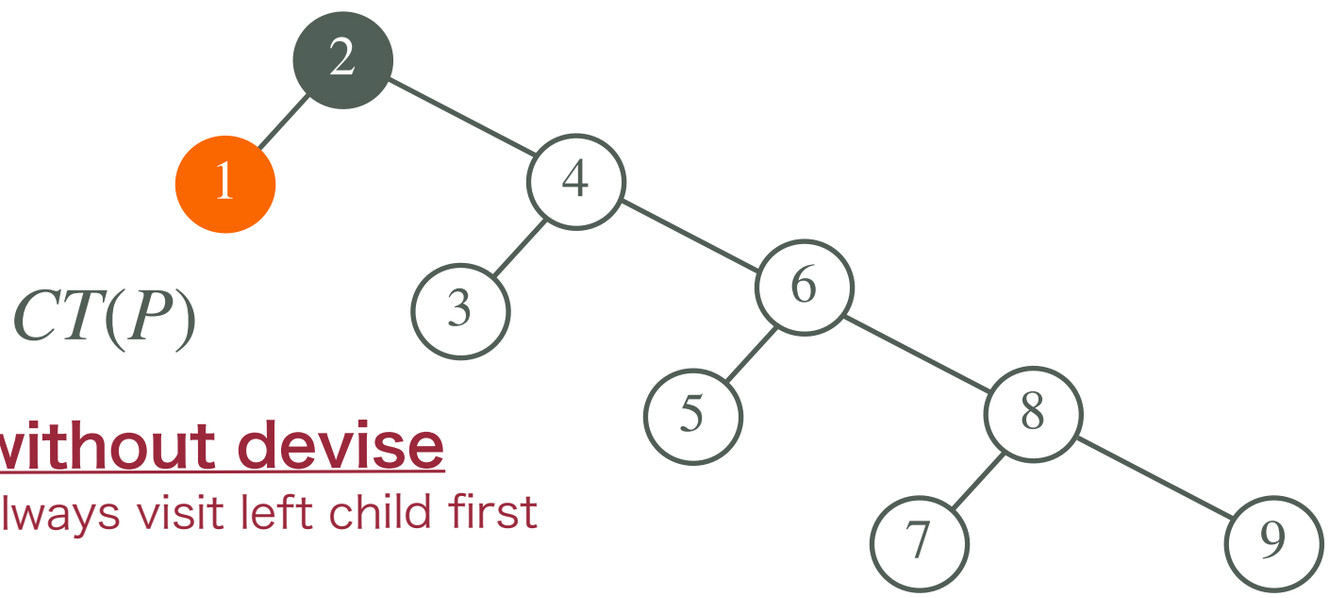
Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.



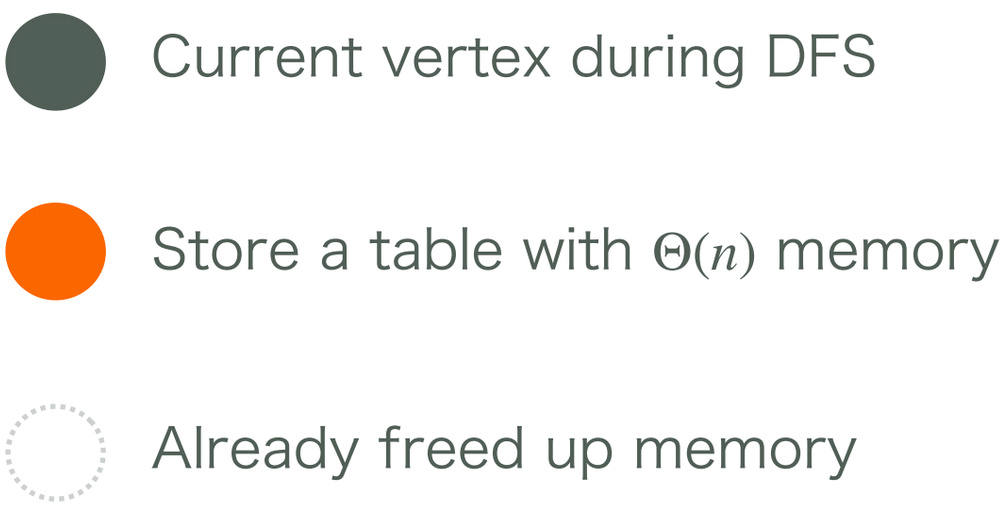


Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.

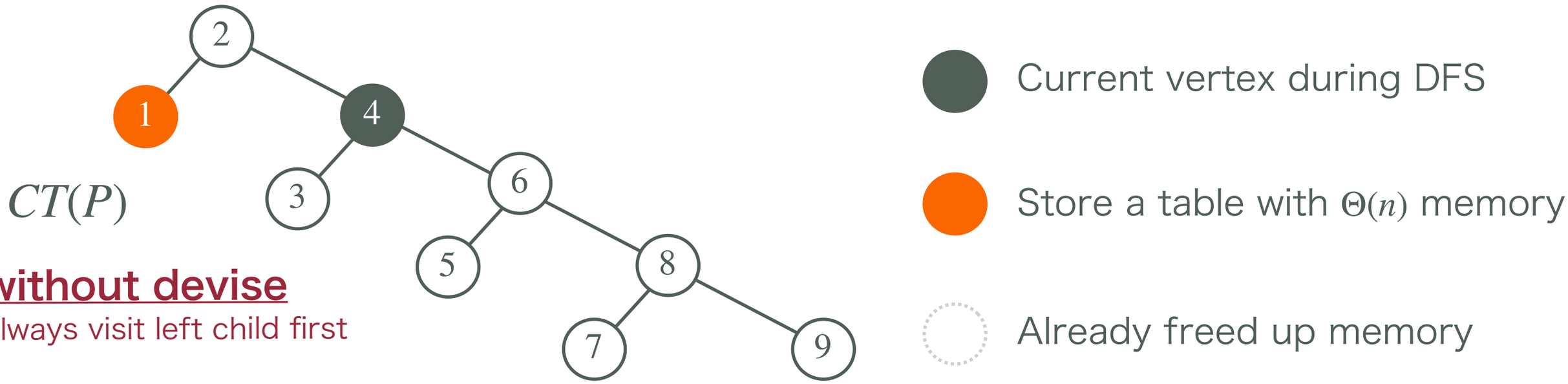


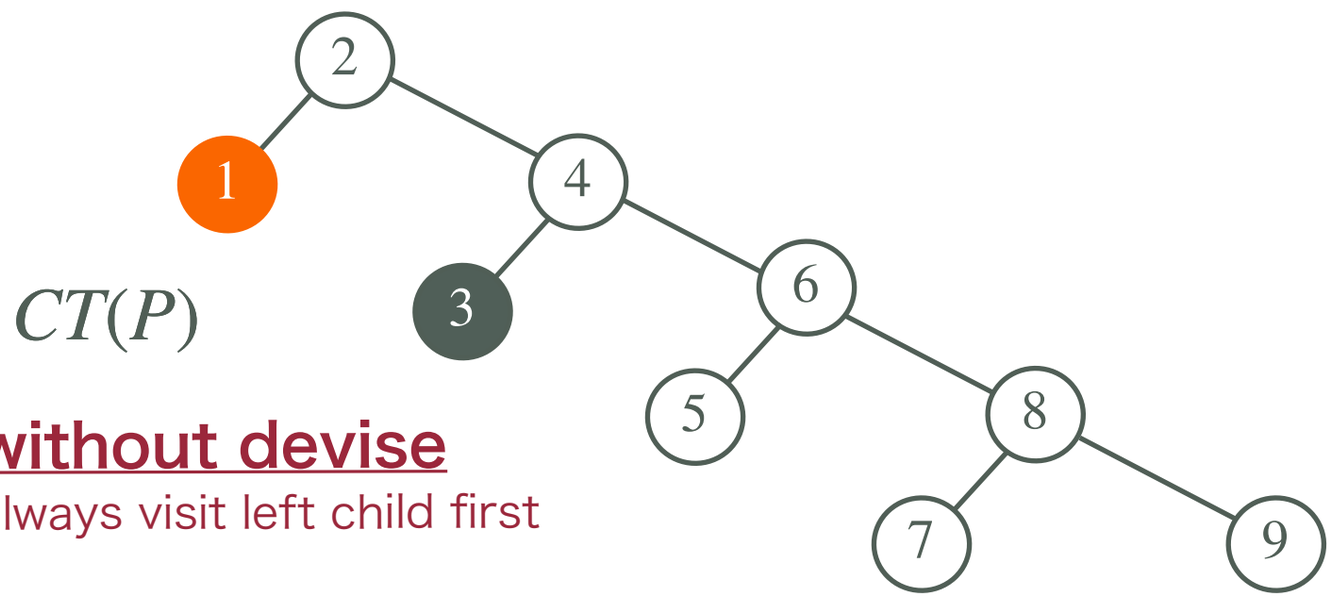
Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.



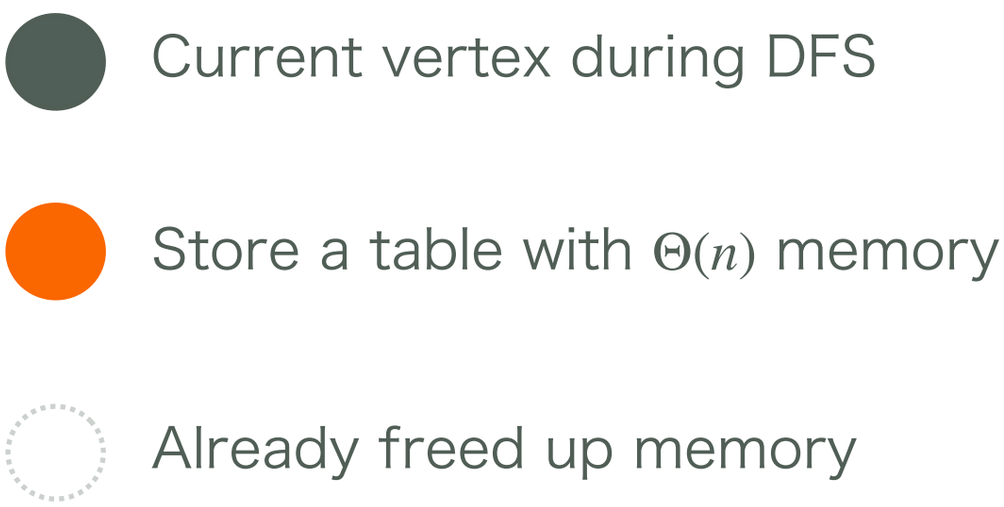


Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.

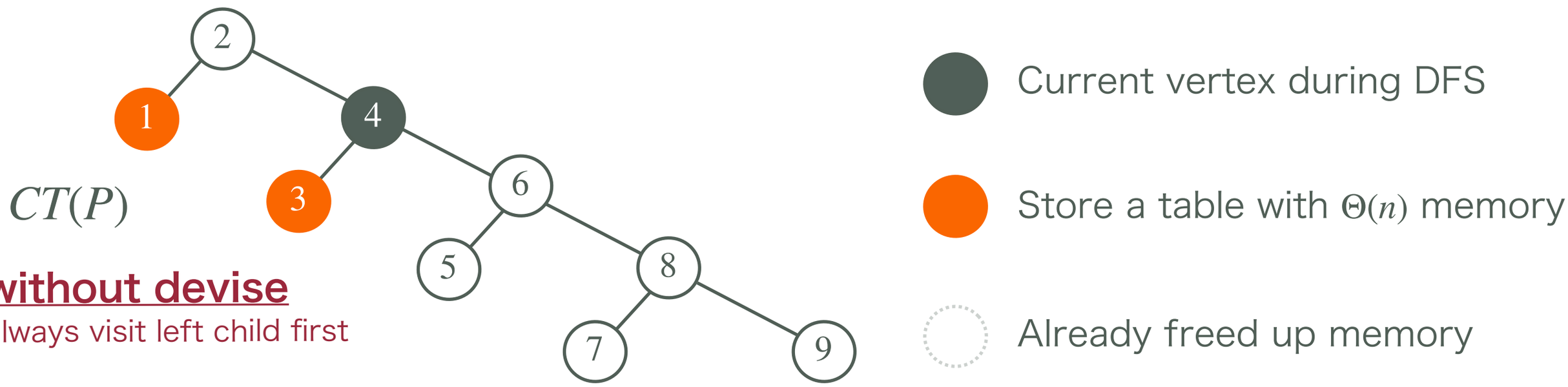


Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.

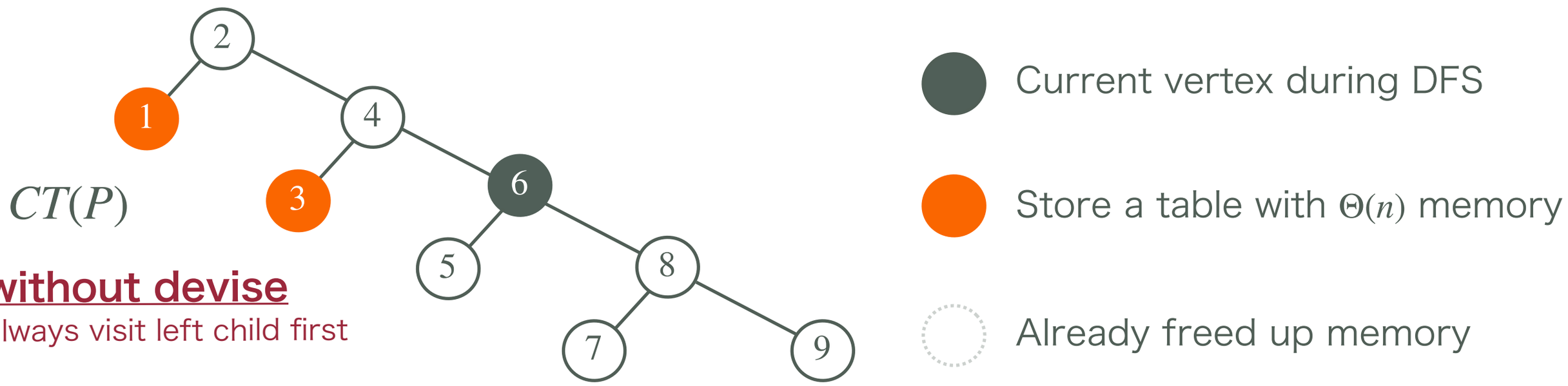


Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.

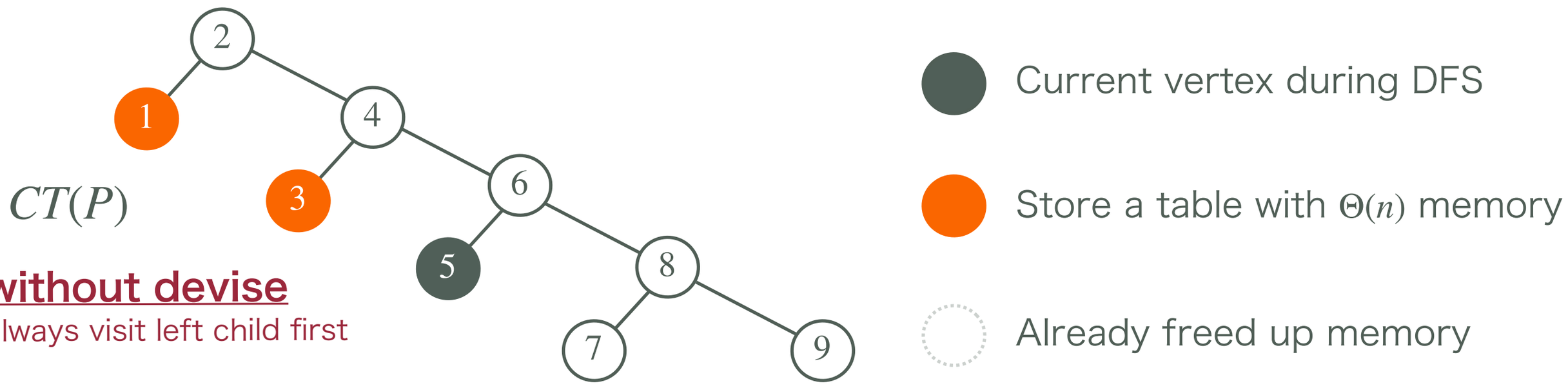


Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.

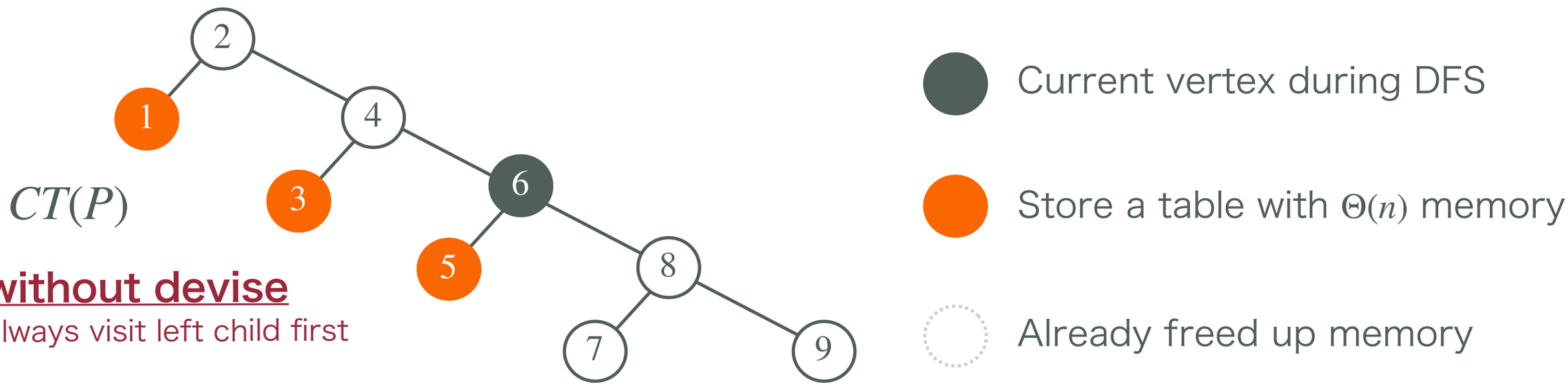


Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.

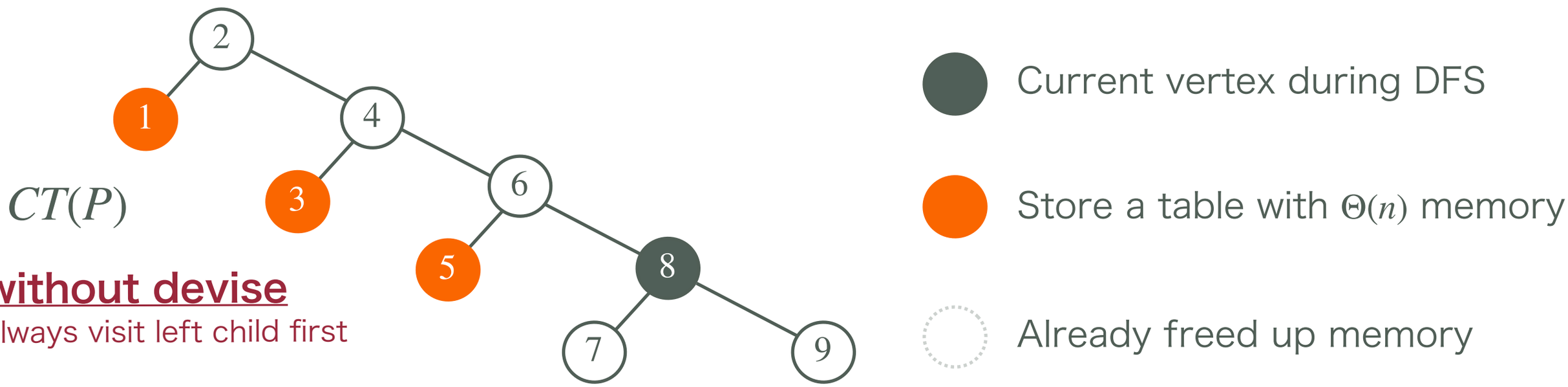


Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.

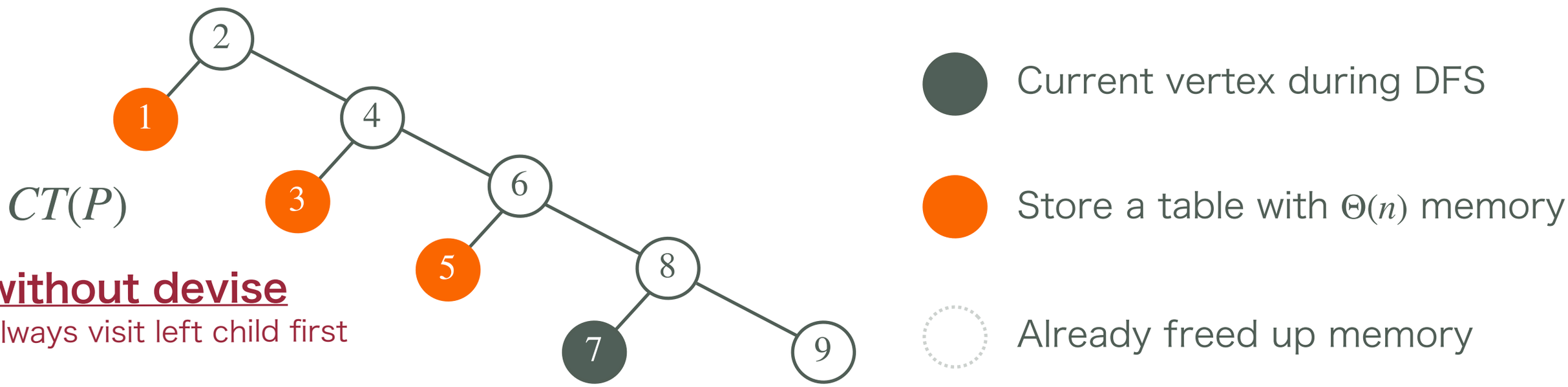


Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.

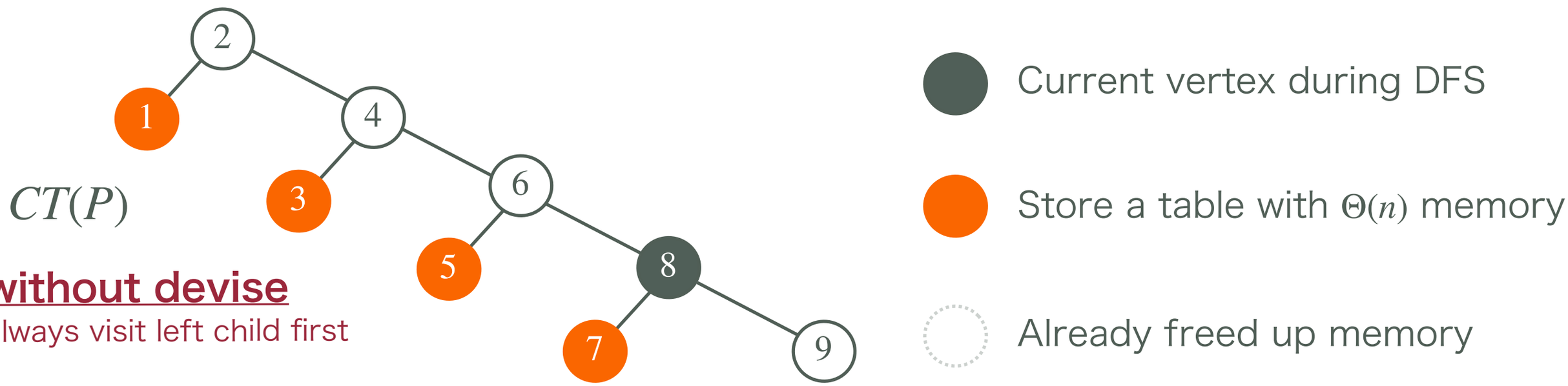


Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.

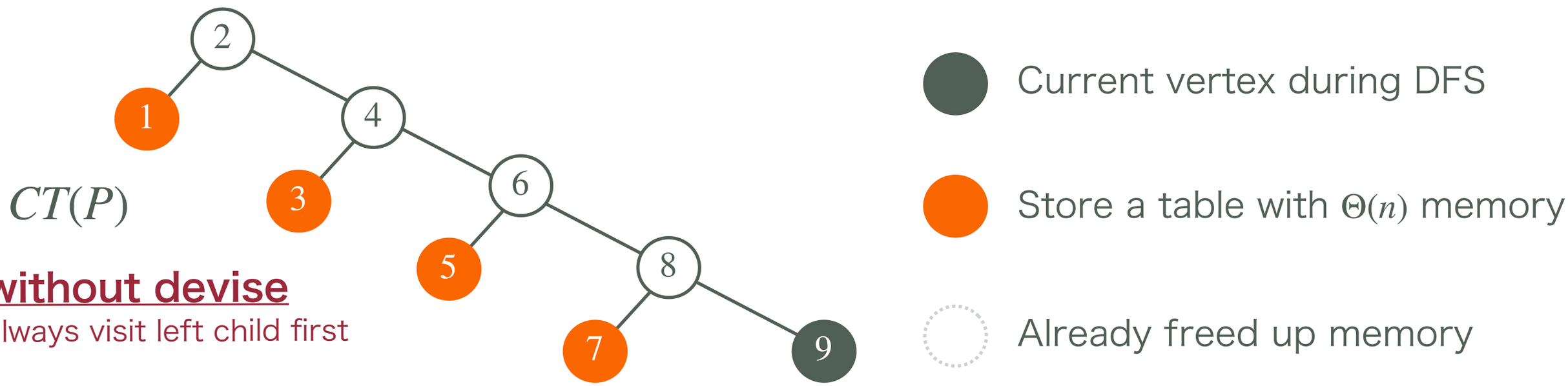


Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.

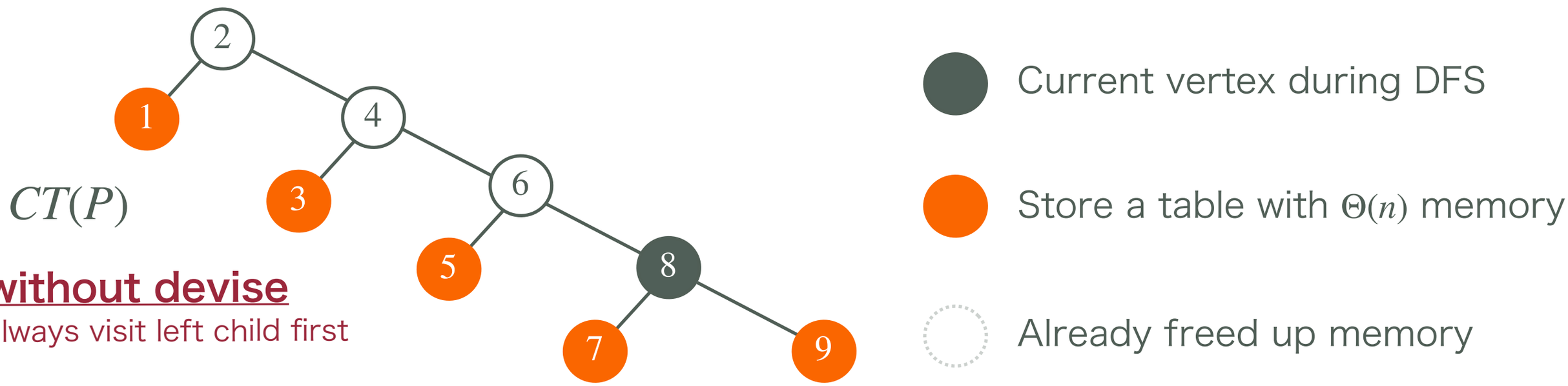


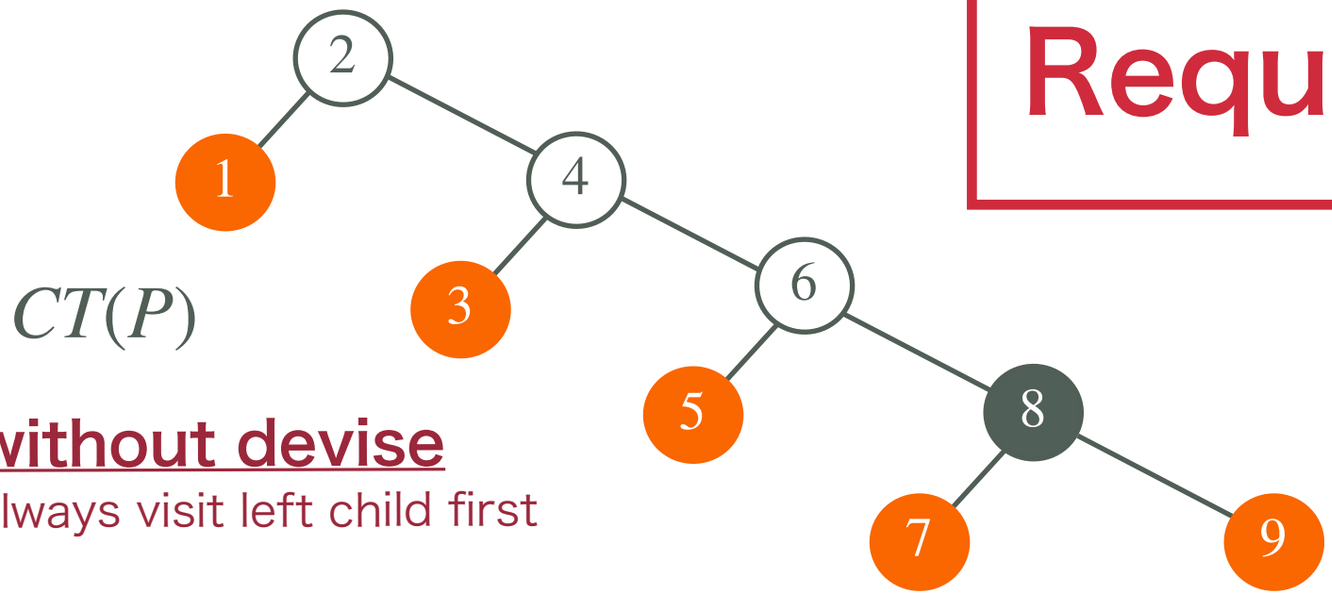
Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.



Require $\Theta(mn)$ memory

Store a table with $\Theta(n)$ memory

Already freed up memory

Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.

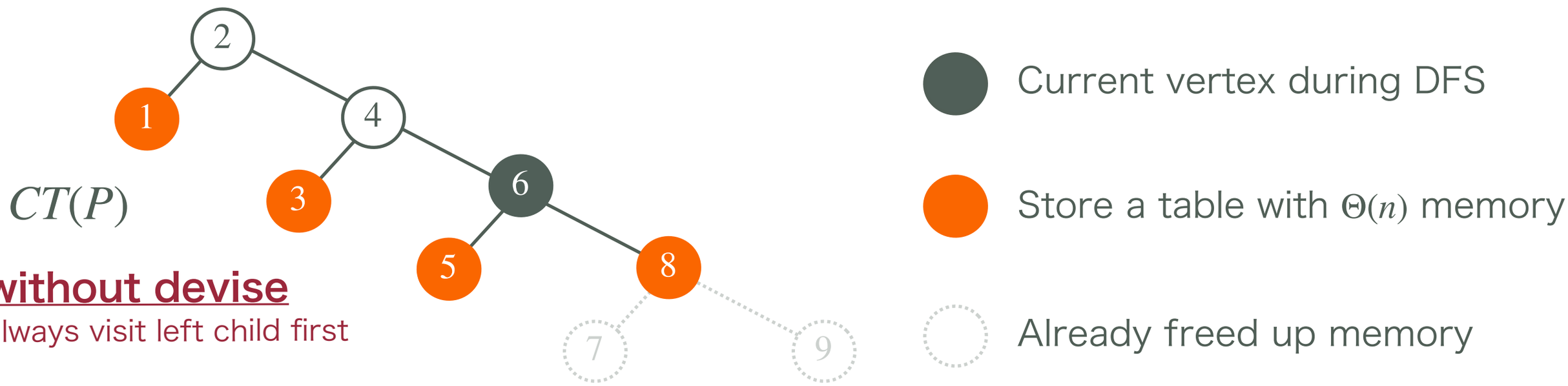


Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.

Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.

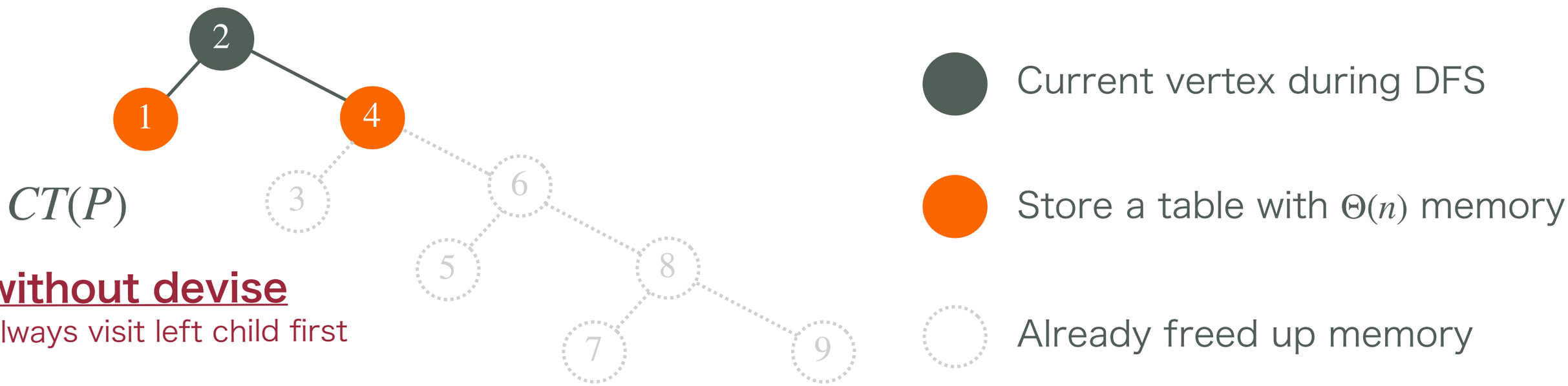


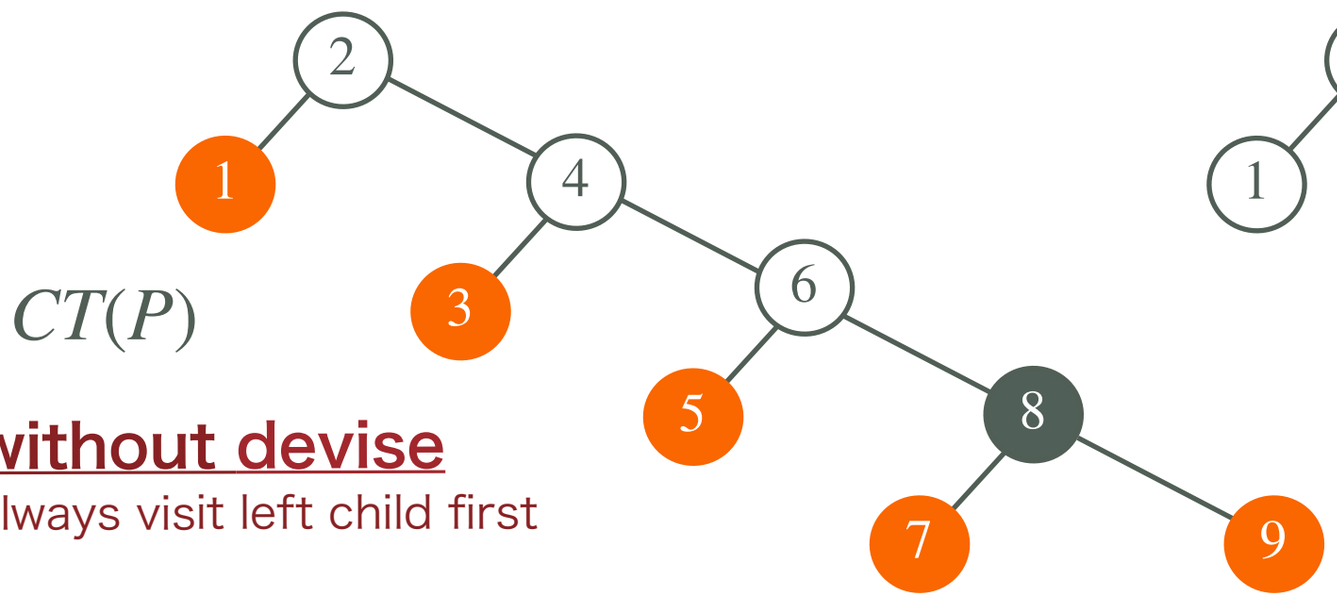
Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.
2. Go down to the **larger** subtree first.



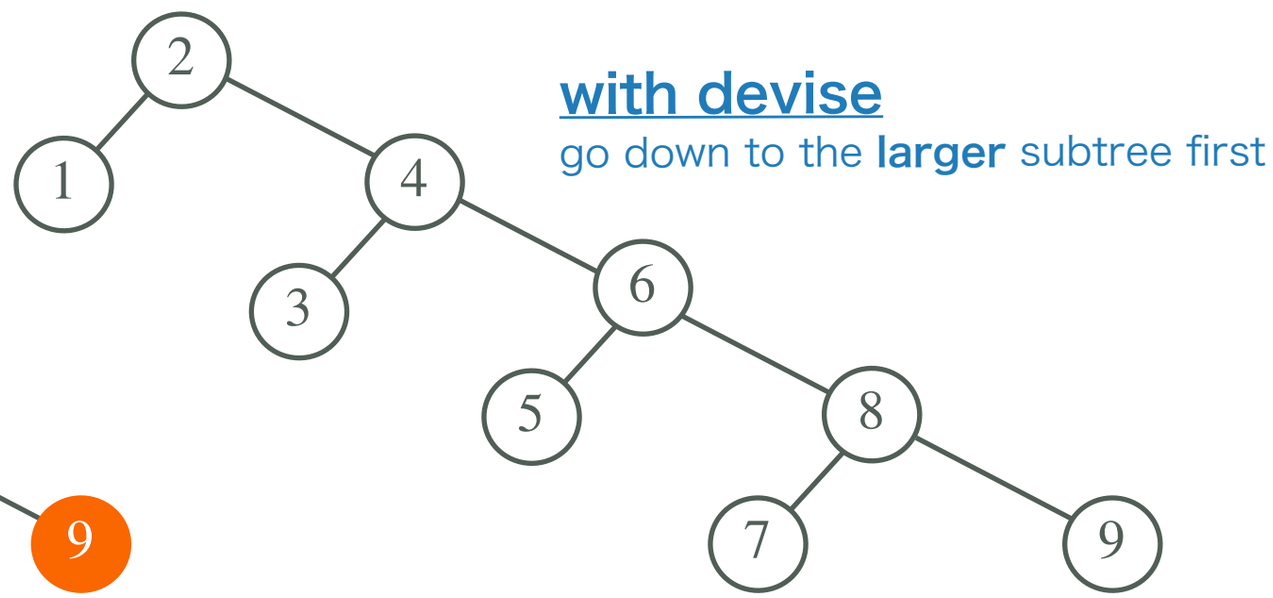


Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.
2. Go down to the **larger** subtree first.

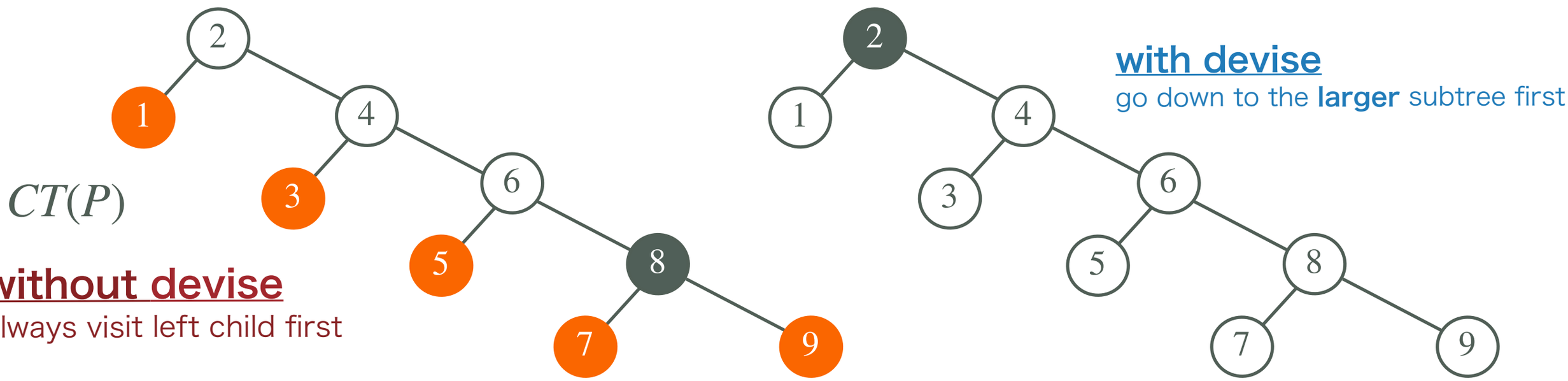


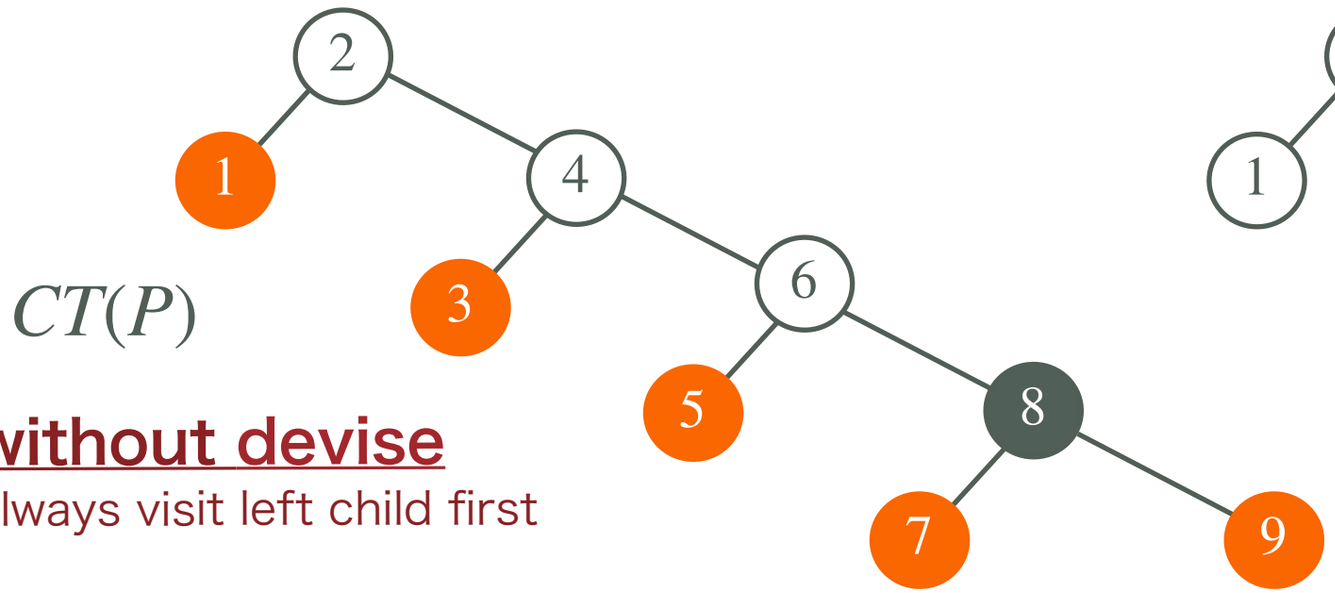
Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.
2. Go down to the **larger** subtree first.



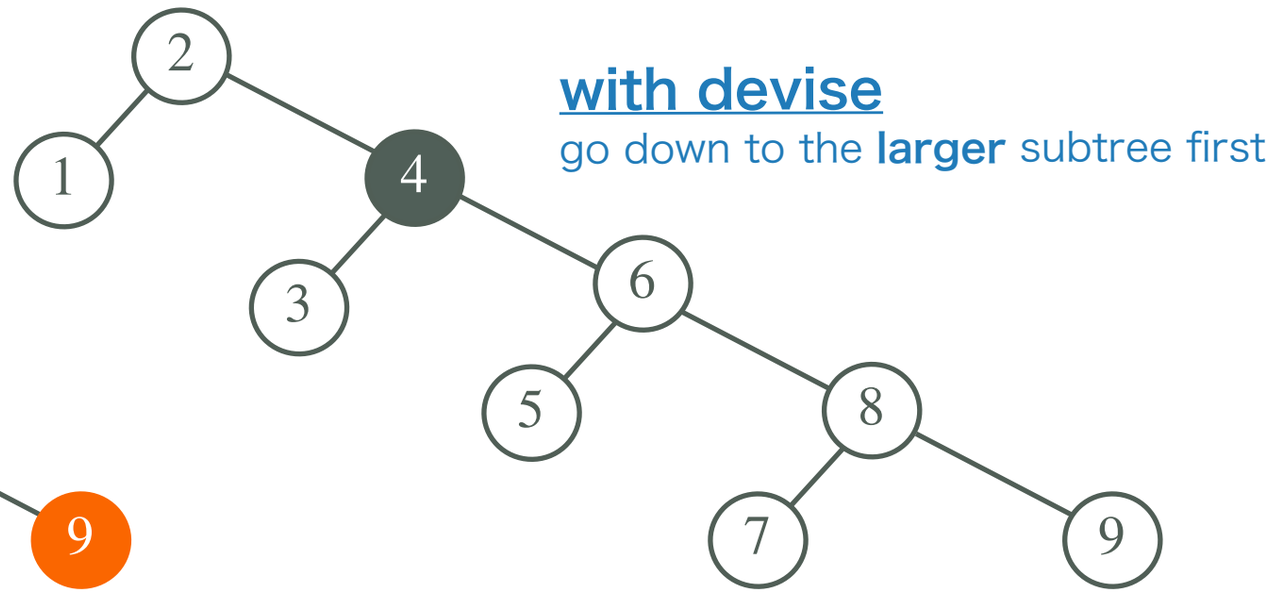


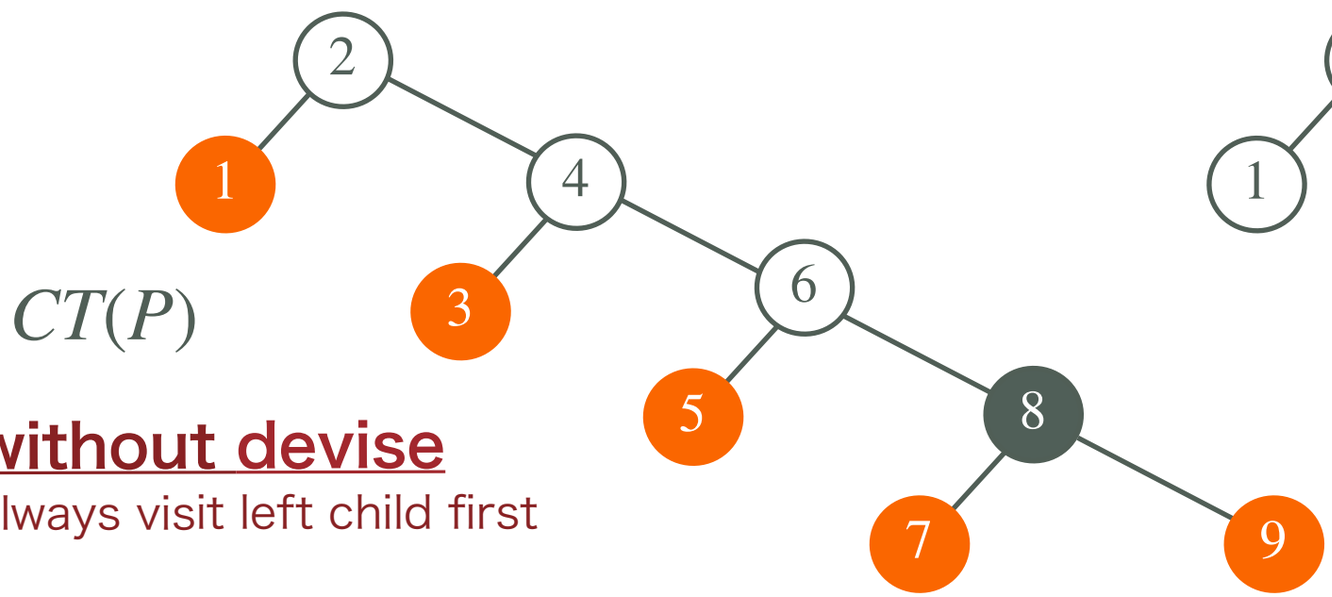
Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.
2. Go down to the **larger** subtree first.



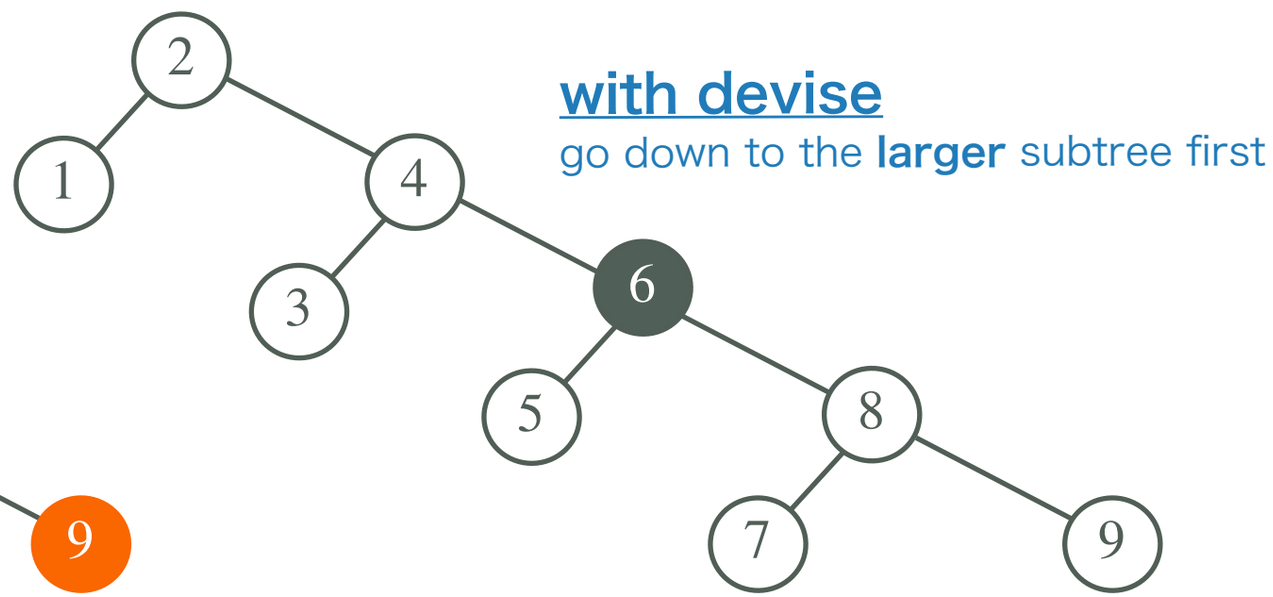


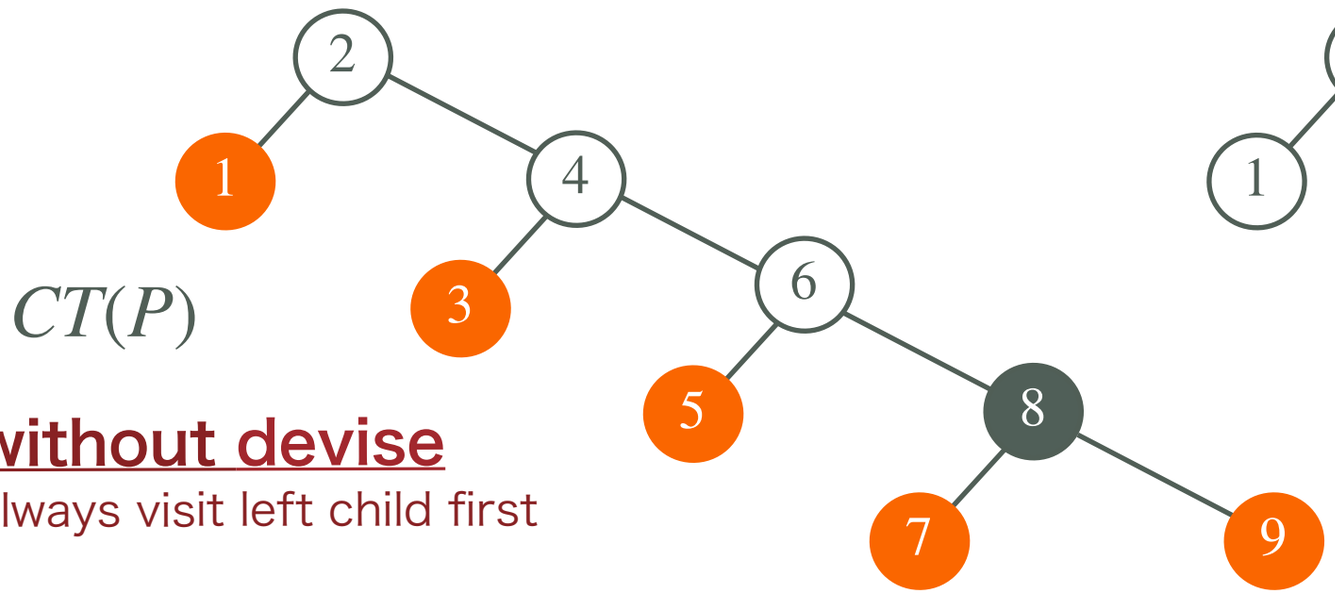
Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.
2. Go down to the **larger** subtree first.



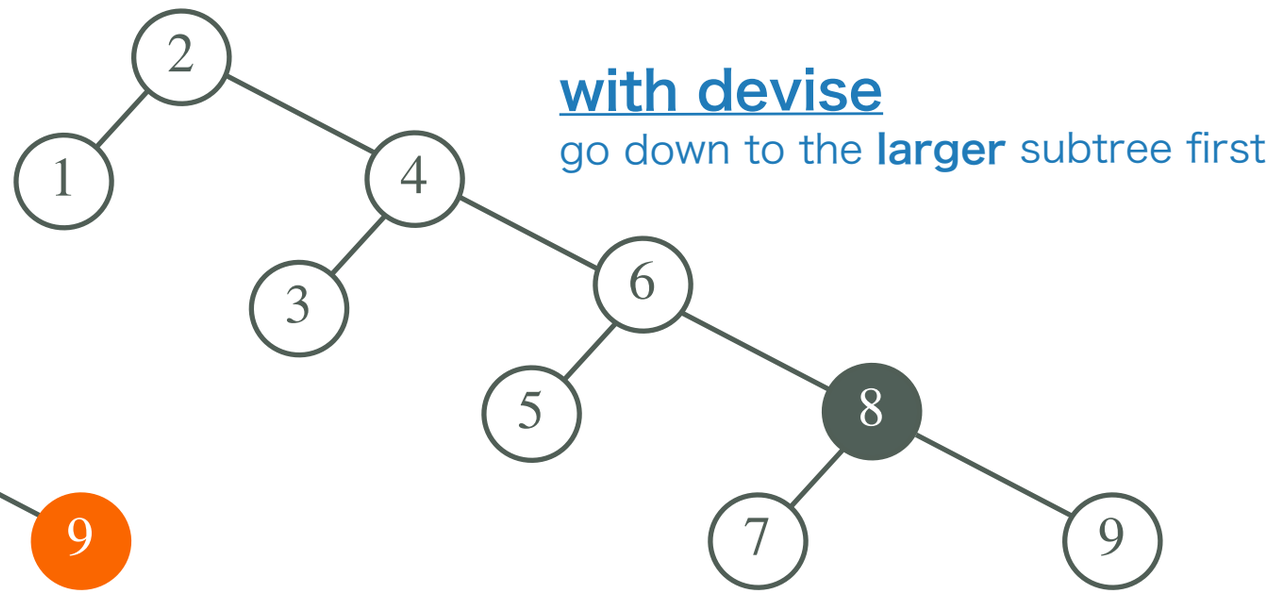


Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.
2. Go down to the **larger** subtree first.

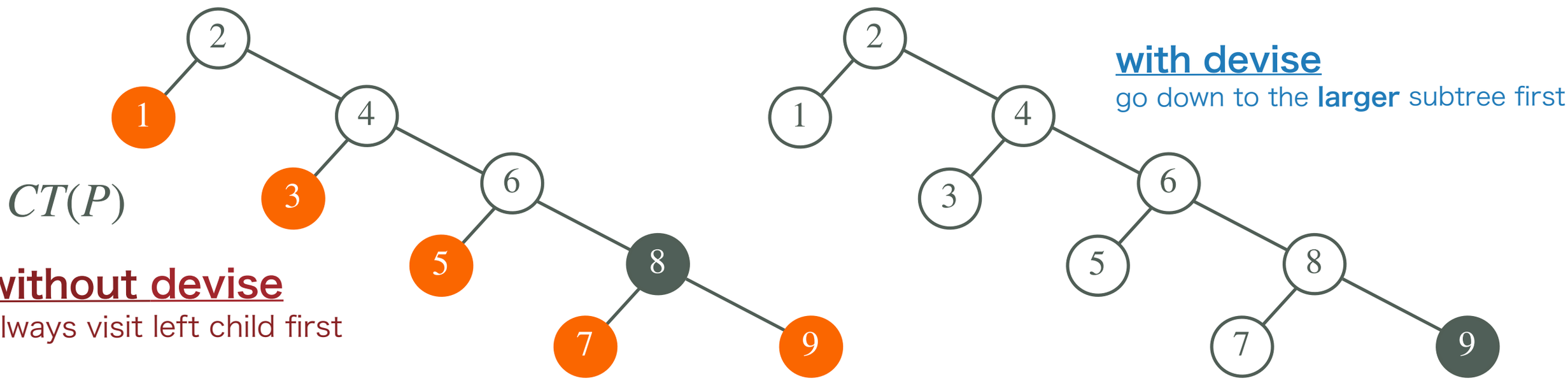


Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.
2. Go down to the **larger** subtree first.

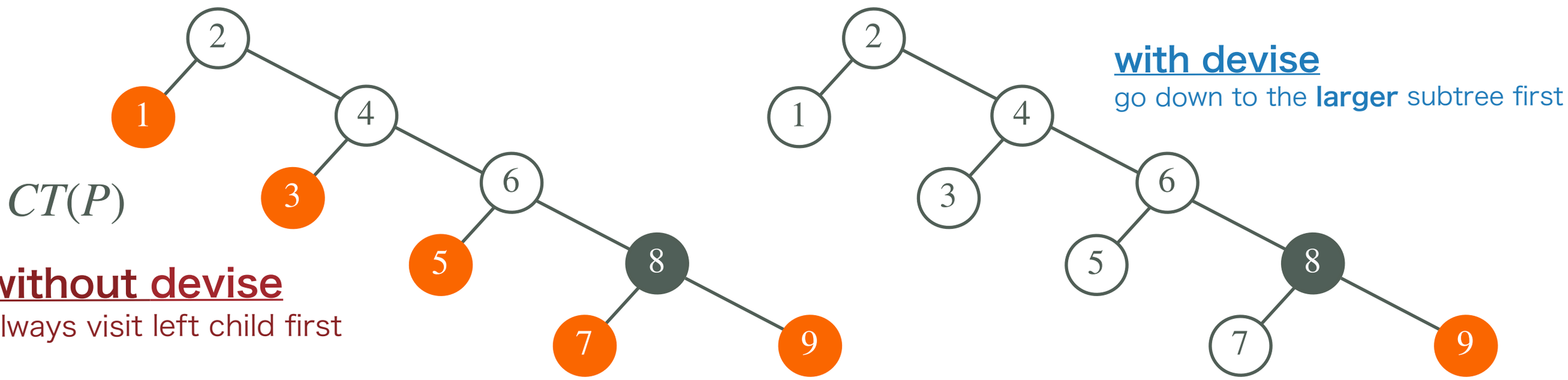


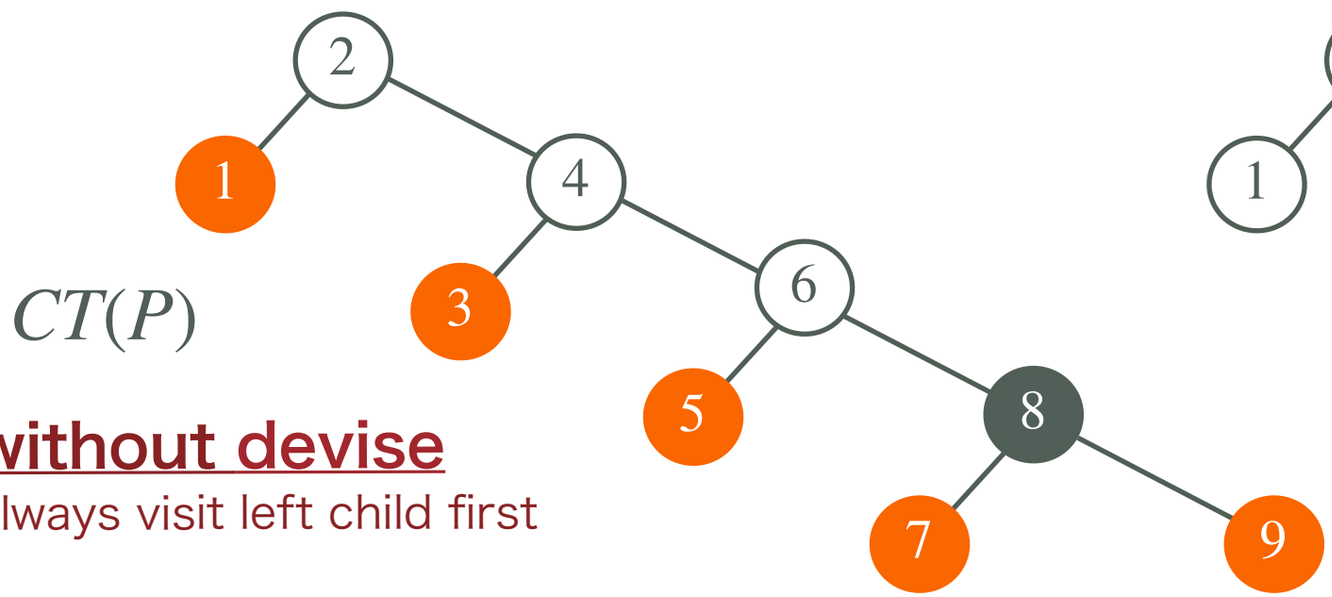
Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.
2. Go down to the **larger** subtree first.



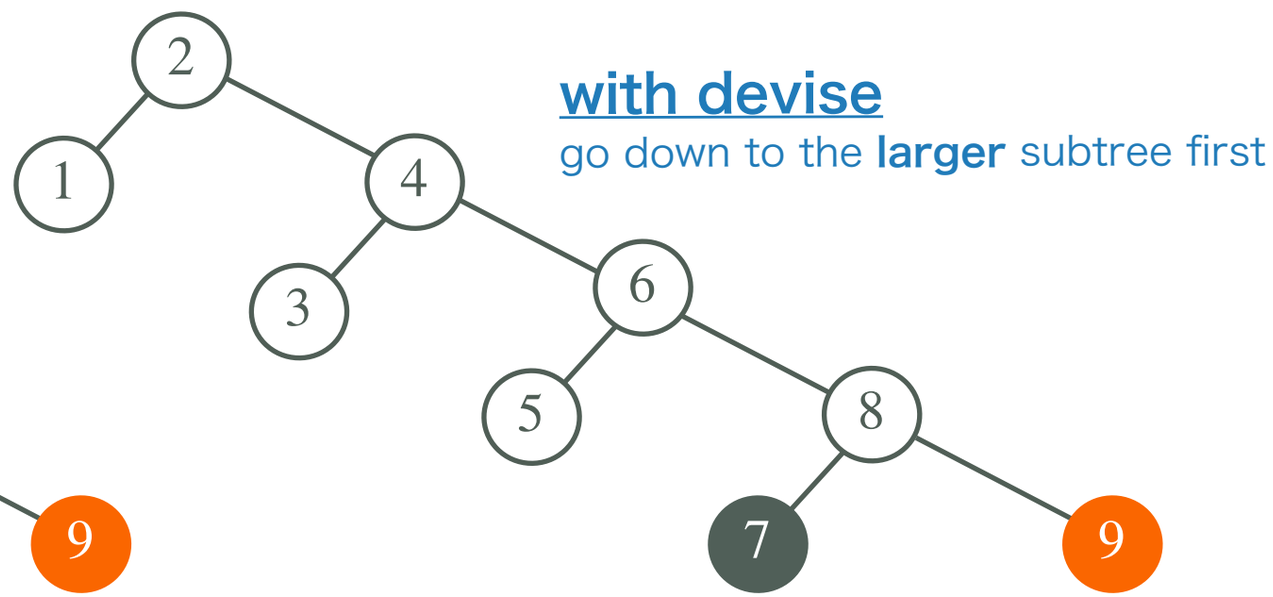


Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.
2. Go down to the **larger** subtree first.

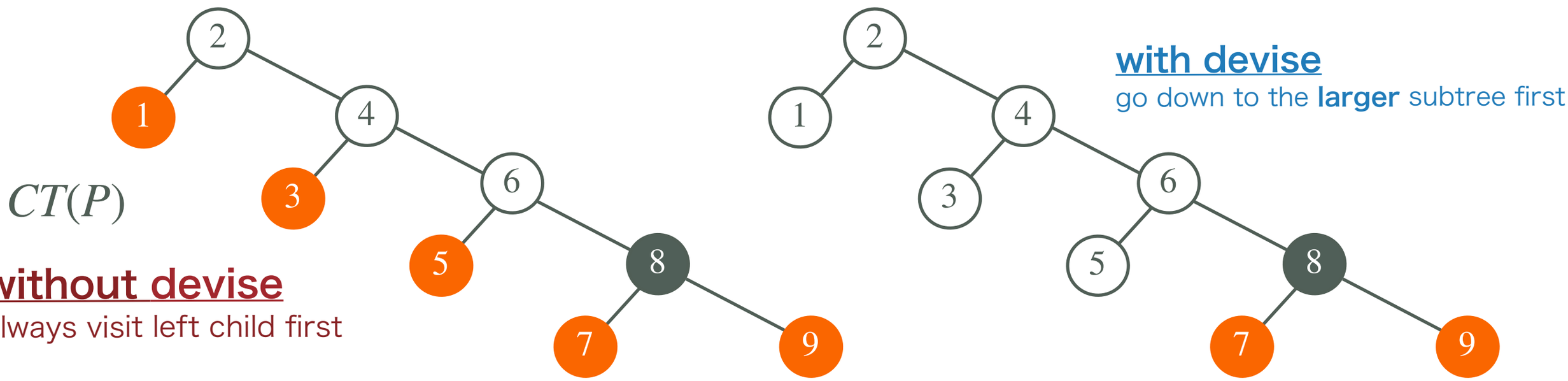


Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.
2. Go down to the **larger** subtree first.

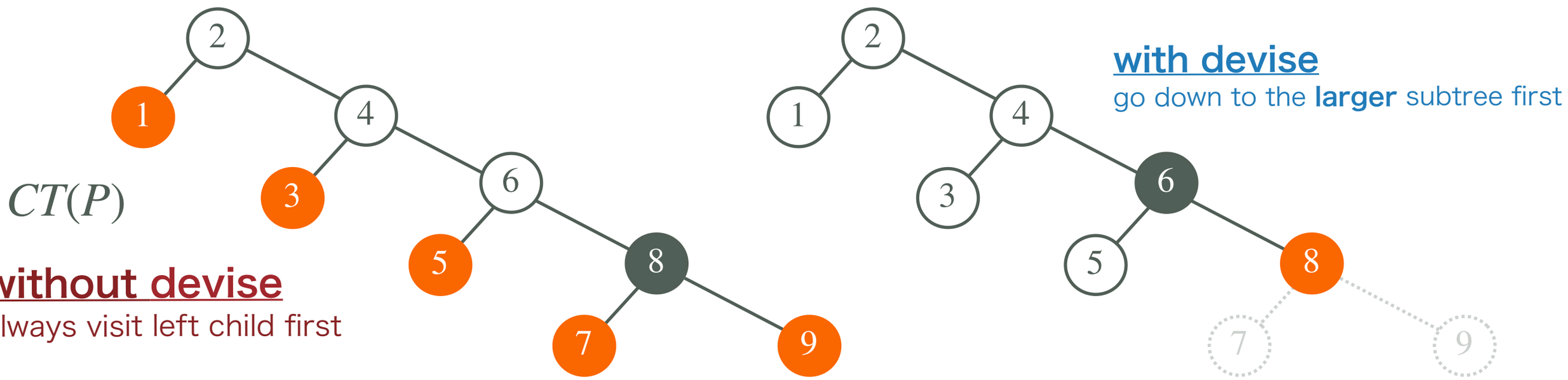


Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.
2. Go down to the **larger** subtree first.

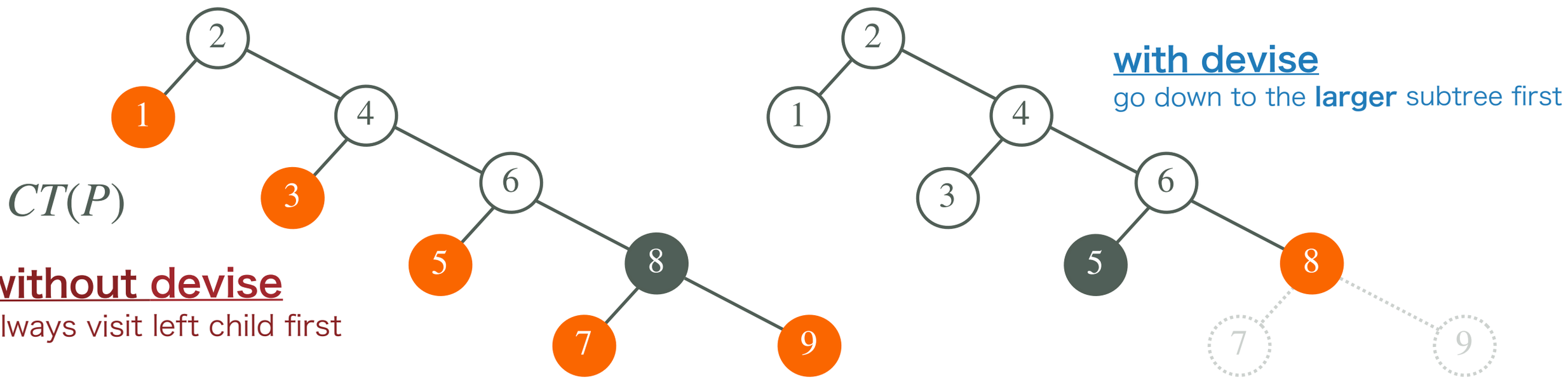


Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.
2. Go down to the **larger** subtree first.

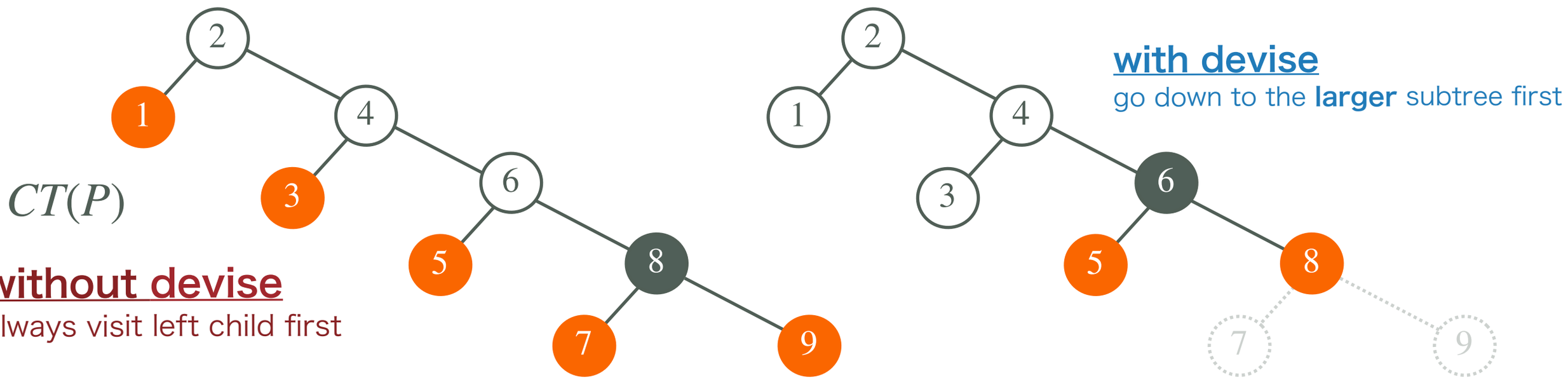


Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.
2. Go down to the **larger** subtree first.

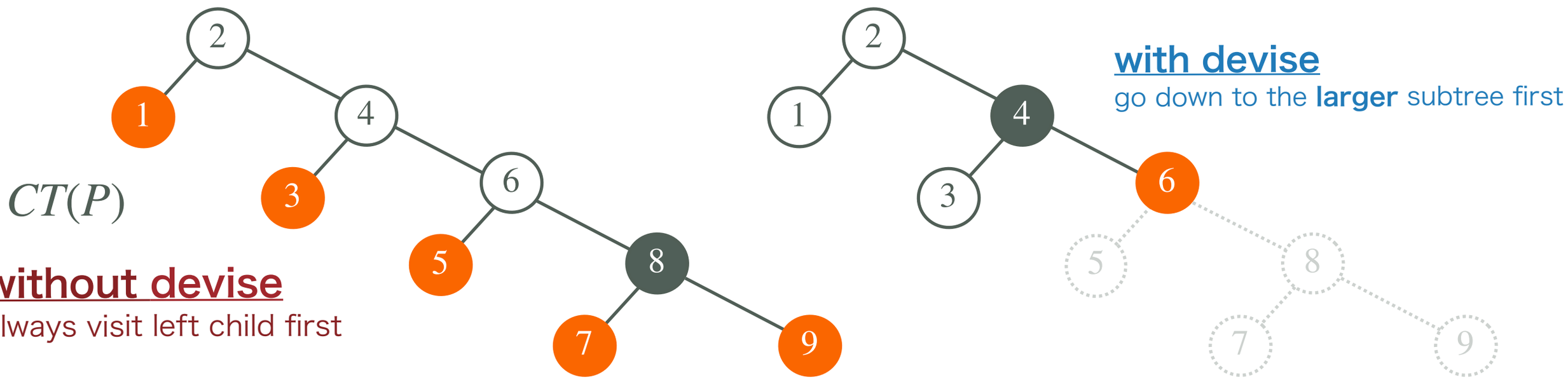


Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Improve space complexity

Theorem. 3

The `CTMSeq` problem can be solved in $O(mn \log\log n)$ time using $O(n \log m)$ space.

1. Free up memory for vertices of $CT(P)$ that are no longer needed.
   - If always go down to the left subtree first, there are worst case examples to lead the space complexity $\Theta(mn)$.
2. Go down to the **larger** subtree first.

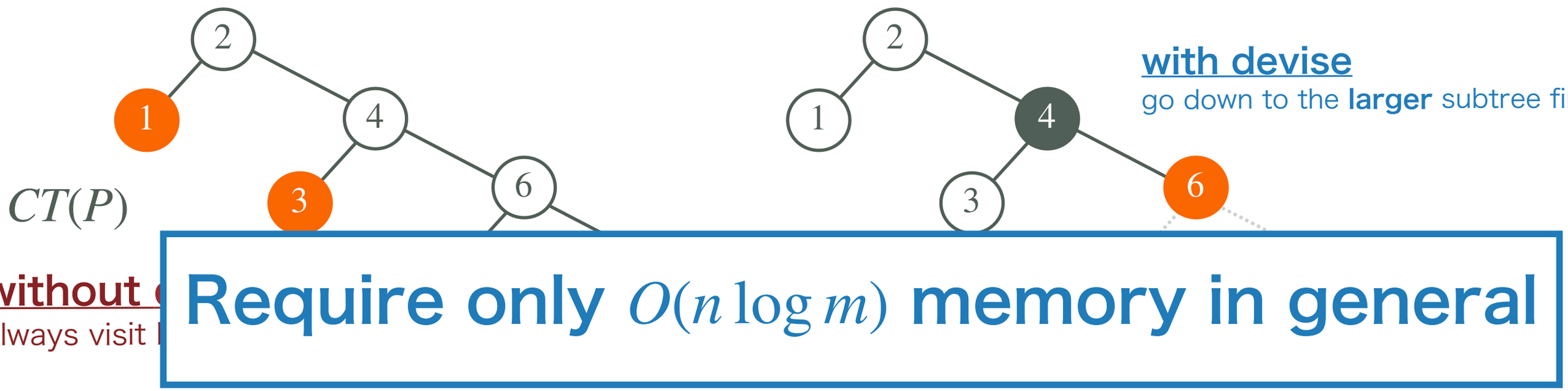


Figure: worst case example of $CT(P)$ which causes the space complexity $\Theta(mn)$.

Experiments

- The most theoretically superior algorithm `vEB-HL` outperforms the other algorithms.
- The fastest and the second fastest algorithms are **highlighted**.
- The shortest and the second shortest memory usage algorithms are **highlighted**

		simple		simple-HL		vEB		vEB-HL	
n	m	time	space	time	space	time	space	time	space
5000	50	2.03	**1980**	0.09	3148	**0.03**	2496	**0.03**	2124
5000	500	19.20	2788	19.86	**2168**	**0.37**	3272	**0.37**	2596
5000	1000	40.62	2932	40.34	**2236**	**0.73**	3520	**0.73**	2604
5000	2500	96.27	3124	96.23	**2368**	**1.84**	3532	**1.84**	2816
10000	50	7.77	2128	7.74	**1804**	**0.07**	2504	**0.07**	2188
10000	1000	159.82	2740	159.70	**1960**	**1.38**	3128	**1.38**	2352
10000	2000	321.07	2920	323.09	**2068**	**3.08**	3312	3.09	2452
10000	5000	841.85	3252	835.29	**2212**	**7.22**	3644	7.23	2592
50000	50	206.49	4976	211.24	**3836**	**0.39**	6076	0.40	4920
50000	5000	NA	NA	NA	NA	39.98	13040	**39.70**	**6576**
50000	10000	NA	NA	NA	NA	**79.42**	12684	80.20	**7044**
50000	25000	NA	NA	NA	NA	199.14	13900	**197.71**	**7340**

The most theoretically superior algorithm

- terminate the program if the execution time exceed 60 sec.
- Text T is a uniform random permutation, and pattern P is a uniform random subsequence of T.
- Unit is [sec], [KB].

Conclusion & Open Problem

algorithm	time	space
`simple`	$O(mn^2)$	$O(mn)$
`vEB-HLD`	$O(mn \log\log n)$	$O(n \log m)$

- n is the length of text T
- m is the length of pattern P

- Open problems
  - How can we improve computational complexity?
  - How can we show the conditional lower bound?