# Persistent Segment Tree

CONTENTS

# Ⅰ. Persistent Segment Tree

# Ⅰ. Persistent Segment Tree

## 1. 개념

- 한글로 번역하면 **지속되는 Segment Tree**, 줄여서 **PST**라고 부른다.
- 어떠한 자료구조가 **Persistent 하다는 뜻**은 **현재 시점에서 과거의 상태에 접근이 가능하다**는 것을 의미한다.
- 즉, 기존의 Segment Tree는 update연산을 지원하고
- PST는 과거에 접근 가능한 Segment Tree임을 알 수 있다.

- 어떻게 할까??

# Ⅰ. Persistent Segment Tree

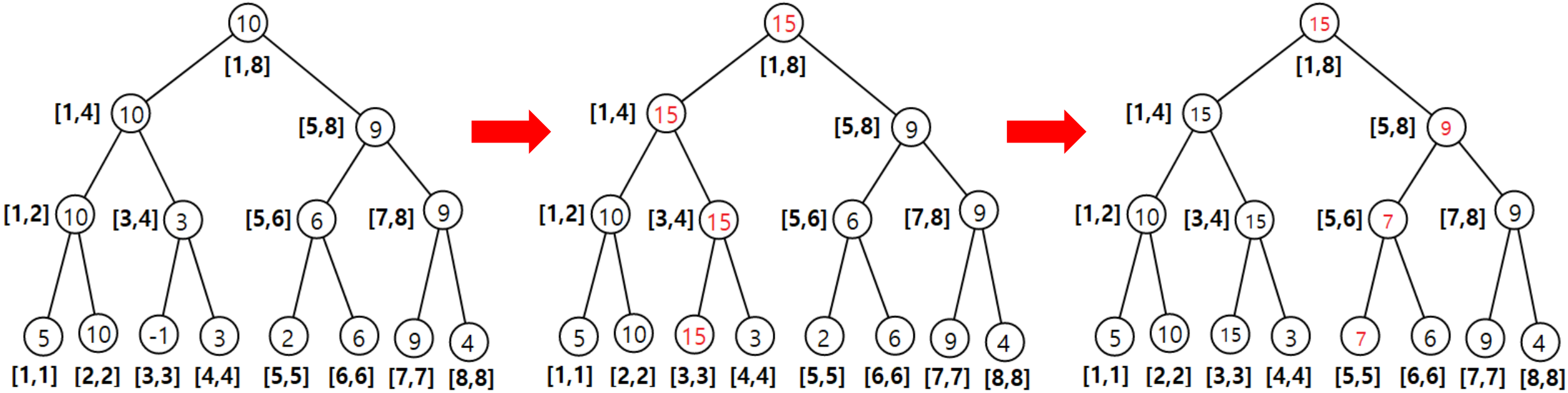
## 1. 개념

- 어떻게 할까??
- **가장 간단하게** 생각해 볼 수 있는 것은 $k$**번의 update가 이루어지면, 각각의 update된 Segment Tree를 모두 저장**하는 것이다.
- 이렇게 하면 **공간 복잡도는** $O(kN)$이 된다.
- 당연히 **공간을 너무 많이 차지**한다.
- 다음과 같은 예시를 살펴보자.

# Ⅰ. Persistent Segment Tree

## 1. 개념

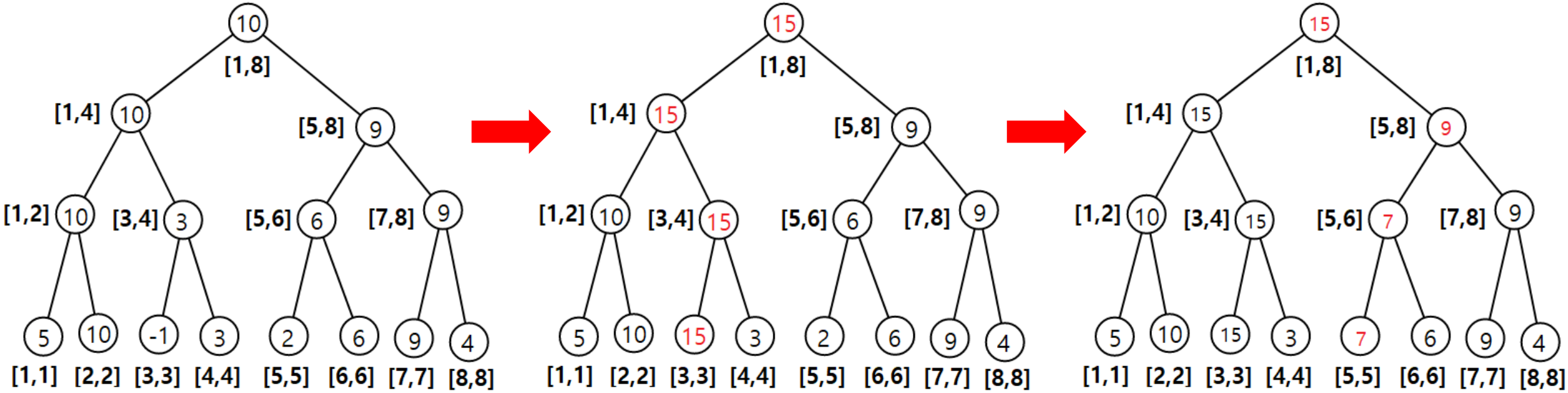
- 다음과 같은 예시를 살펴보자.
- 각 **Segment Tree는 구간의 최댓값**을 나타내고 다음과 같이 **2번의 update**가 일어났다. **빨간색은 root에서 update 되는 노드까지의 경로 상의 노드**들이다.



# Ⅰ. Persistent Segment Tree

## 1. 개념

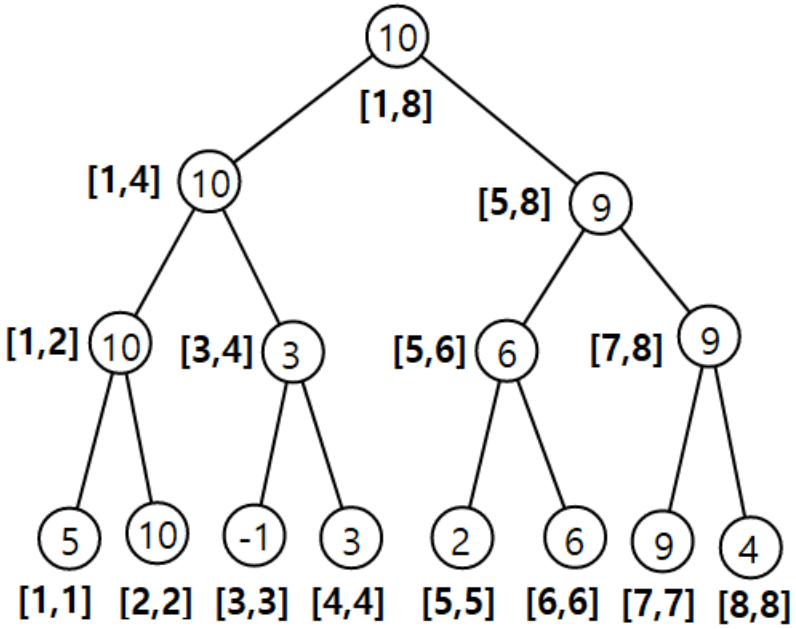
- 아래 그림에서 알 수 있듯이, **한번의 update로 바뀌는 노드의 개수는 $O(\log N)$** 이다.
- 따라서 **$O(log\ N)$개의 노드만 추가로 저장하고 pointer로 이전 노드를 가리키면 된다!!**

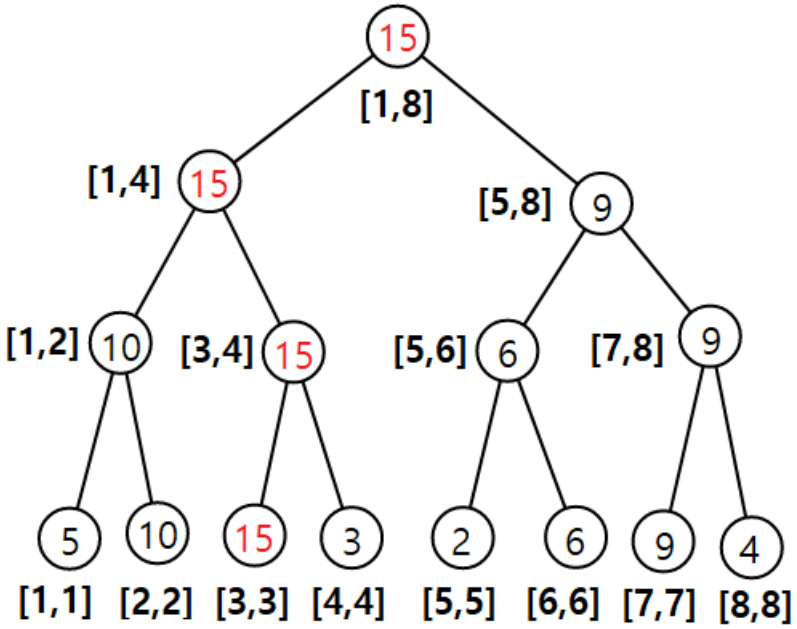


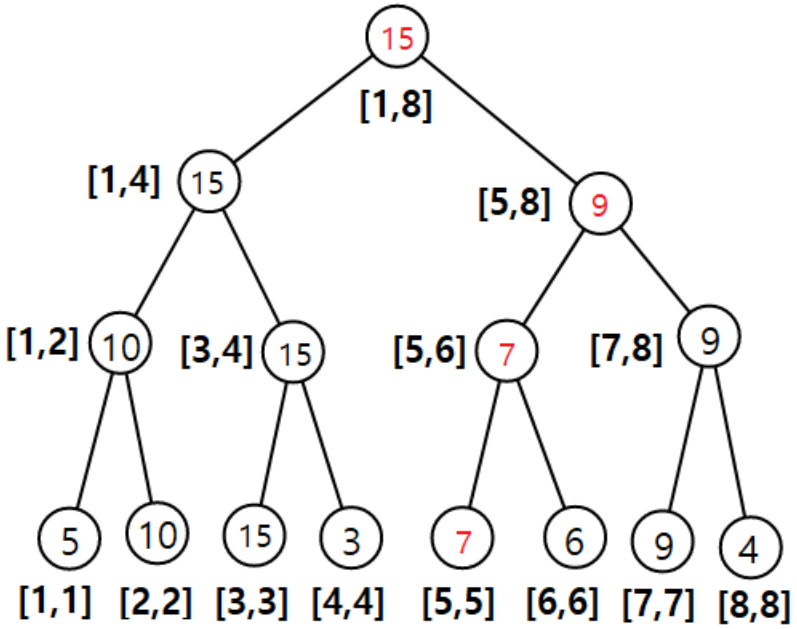
# Ⅰ. Persistent Segment Tree

## 1. 개념

- 즉, 이렇게 3개를 전부 저장하지 말고
- **$k$번 update시 공간 복잡도 $O(kN)$**



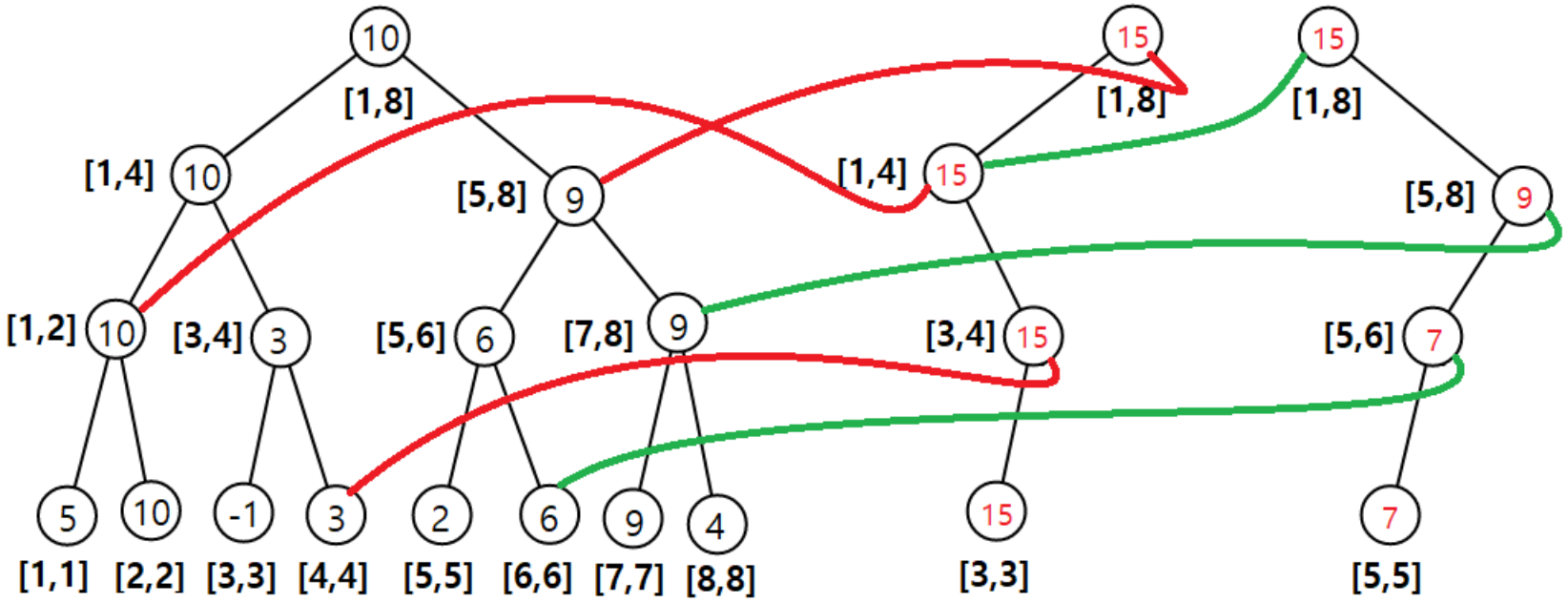



# Ⅰ. Persistent Segment Tree

## 1. 개념

- 아래와 같이 저장하도록 한다.
- 이렇게 하면, **$k$번의 update가 일어난 경우 $O(N + k \log N)$의 공간 복잡도로 이전의 모든 상태를 저장하고 접근이 가능**해 진다.



# Ⅰ. Persistent Segment Tree

## 1. 개념

- 이렇게 끝내면 아쉬우니 PST로 무엇을 할 수 있는지 알아보면
- 2차원 평면에서 x, y축과 평행한 특정 직사각형에 있는 점의 개수를 $O(\log N)$에 알아낼 수 있다.
- X축을 기준으로 PST를 update 해가면서
- Y축의 데이터를 PST의 구간에 대응 시켜 update 한다.
- 즉, 처음에는 비어있는 구간 합 Segment Tree에 (편의상 $T_0$)
- 같은 x 값을 가진 모든 점들을 구간 합 Segment Tree에 update를 한다.
- 이때 저장 방식을 PST로 하는 것이다.

# Ⅰ. Persistent Segment Tree

## 1. 개념

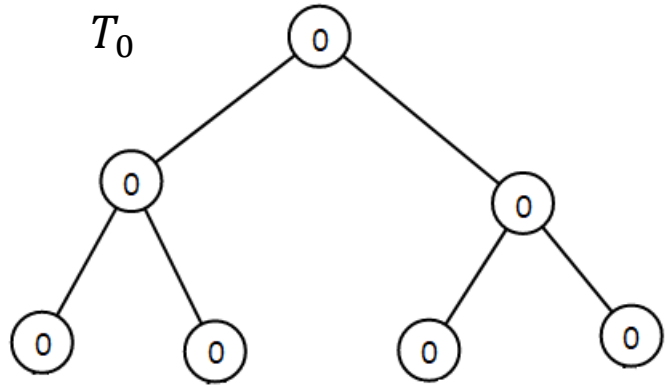
- 즉, 처음에는 비어있는 구간 합 Segment Tree에 (편의상 $T_0$)
- 같은 x 값을 가진 모든 점들을 구간 합 Segment Tree에 update를 한다.
- 이때 저장 방식을 PST로 하는 것이다.

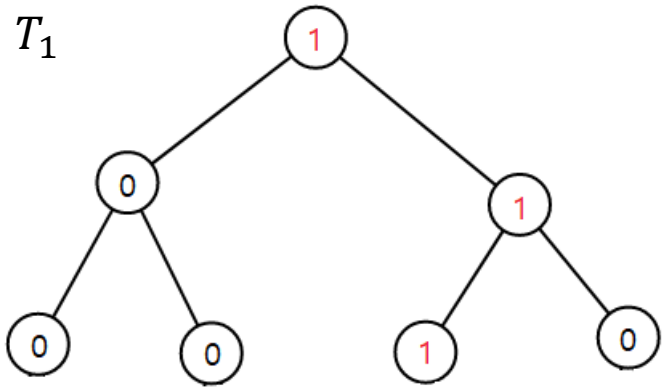
| $T_0$ | $T_1$ | $T_2$ | $T_2$ | $T_3$ |
|---|---|---|---|---|
| 4 | | . | | |
| 3 | . | | | . |
| 2 | | | | . |
| 1 | | . | | |
| 0 | 1 | 2 | 3 | 4 |

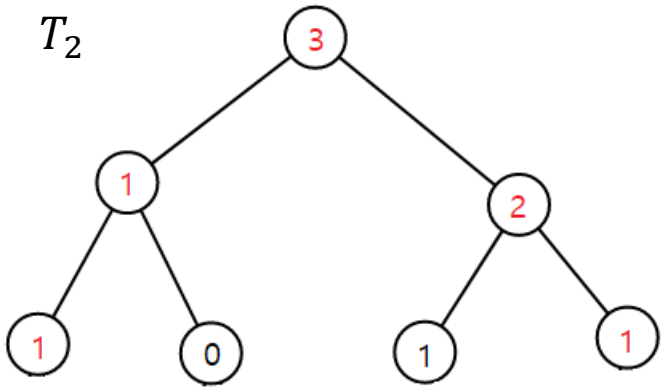
# Ⅰ. Persistent Segment Tree

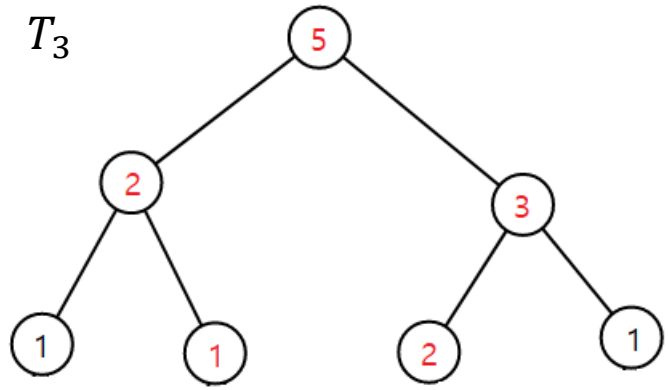
## 1. 개념

| $T_0$ | $T_1$ | $T_2$ | $T_2$ | $T_3$ |
|---|---|---|---|---|
| 4 | | . | | |
| 3 | . | | | . |
| 2 | | | | . |
| 1 | | . | | |
| 0 | 1 | 2 | 3 | 4 |



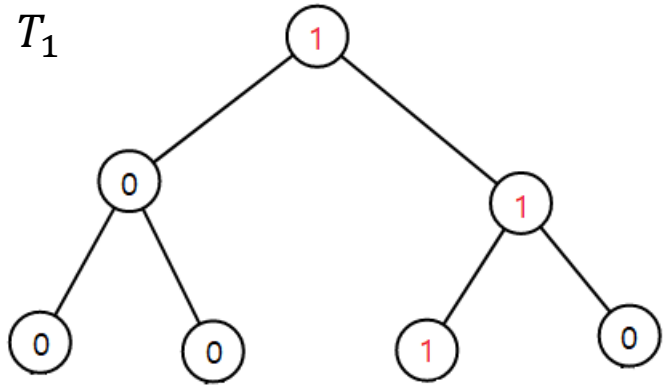



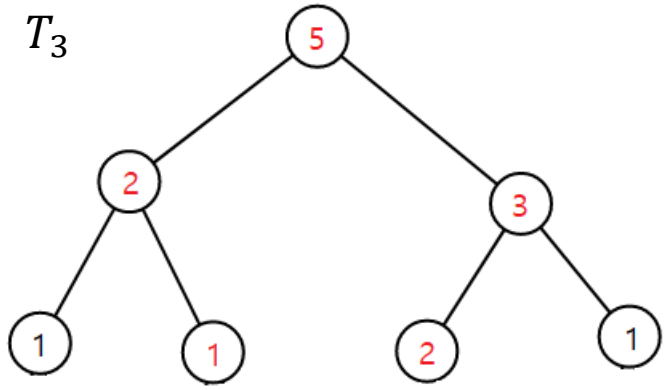



# Ⅰ. Persistent Segment Tree

## 1. 개념

| $T_0$ | $T_1$ | $T_2$ | $T_2$ | $T_3$ |
|---|---|---|---|---|
| 4 | | . | | |
| 3 | . | | | . |
| 2 | | | | . |
| 1 | | . | | |
| 0 | 1 | 2 | 3 | 4 |




- 만약 [2,4] x [1,3] 안의 점들의 개수를 찾는다고 하면
- x=4에 대응되는 $T_3$의 [1,3] 합 – x=1에 대응되는 $T_1$의 [1,3] 합

  = 4 – 1 = 3 이렇게 구할 수 있다.
- 이해가 안되면 아래 링크를 참조하자.

  (http://kks227.blog.me/221411526404)