SPLAY TREES

Motivation:

- An unbalanced BST, accessing a leaf node will takes $O(n)$. Splay tree provides a way to perform search, insert, delete in amortized complexity $O(\log n)$
- Splay tree follows the idea of locality, which stated that "something that is recently used will likely to be used again in the future"

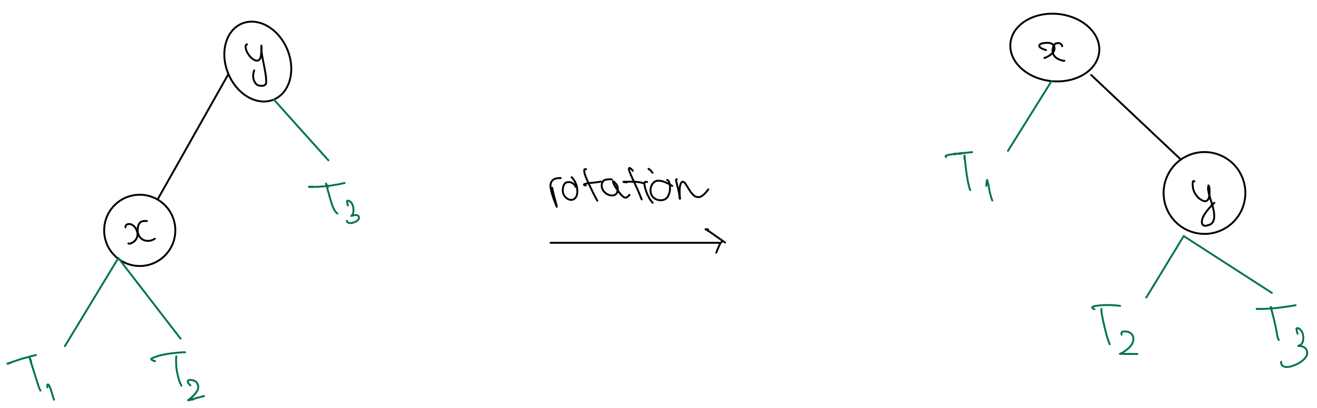
Splay tree:

- The idea is after an operation (search, insert, delete) is performed, the tree will perform a "splay" operation.
- A "splay" operation can be thought of as rotating the recently used node to the root.

Splaying:

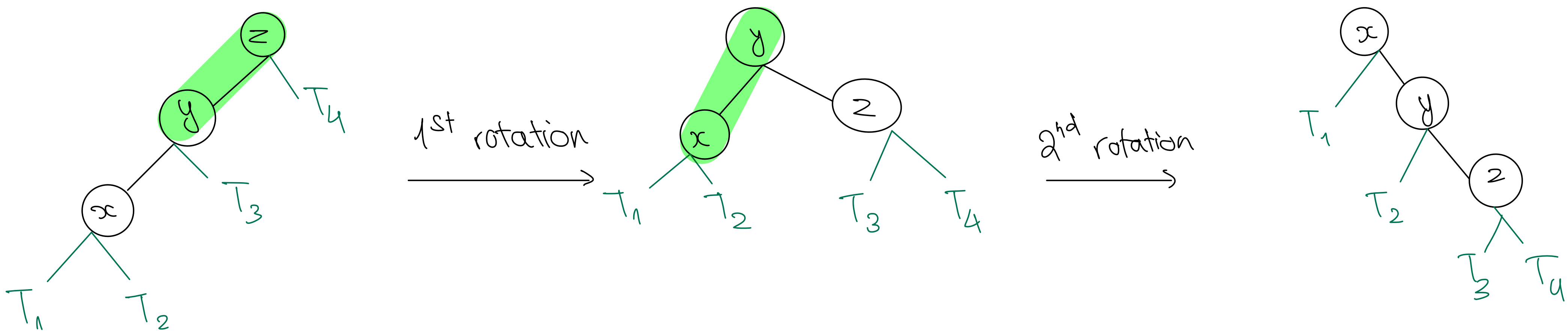
There are 3 types of splaying: zig, zig-zig, zig-zag

- Zig



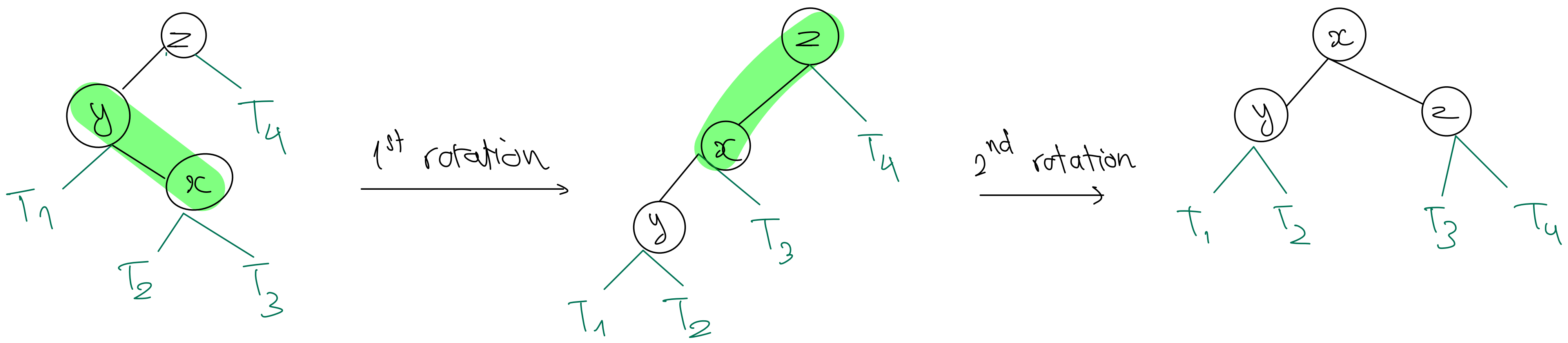
If we are splaying node x, and its parent y is the root. The zig rule is just one left/right rotation with respect to y.

- Zig-zig:



When we are splaying node x, parent y, grandparent z, and x and y are both left/right children. The zig-zig rule has 2 rotations: rotation w.r.t. z (grandparent) followed by rotation w.r.t. y (parent)

- Zig-zag



When we are splaying node x, parent y, grandparent x, and x and y are not both left or right children.
Zig-zag rule has 2 rotations: rotation w.r.t. y (parent) followed by rotation w.r.t. z (grandparent)

Node to be splayed in types of operations

- Search:
  - If key is found, splay the node containing the key.
  - If key not found, splay the parent node of the leaf position where you tried to find the key
    (Splay the last node you touch)
- Insert:
  Splay the newly inserted node.
- Delete:
  Splay the parent of the deleted node.
- Split:
  Given key x, we would like to split into 2 trees. One contains all nodes $\leq x$, the other contains all nodes $> x$.
  $\Rightarrow$ Splay the node containing key x, then remove its right subtree
- Join:
  Given 2 trees A and B, we would like to put them together with all items in A to the left of all items in B.
  $\Rightarrow$ Splay the right most node in A, then make B the right child of this node

The amortized analysis:

- Let T_0 be the initial tree structure, and T_i is the resulting tree structure after apply the i^{th} operation on T_{i-1}.

  The amortized cost after n operations is:

$$\sum^{n-1} \hat{c}_i = c_0 + c_1 + \ldots + c_{n-1} + (\phi(T_1) - \phi(T_0)) + (\phi(T_2) - \phi(T_1)) + \ldots + (\phi(T_{n-1}) - \phi(T_{n-2}))$$

$$\sum^{n-1} \hat{c}_i = \sum_i^{n-1} c_i + (\phi(T_{n-1}) - \phi(T_0))$$

  $\Rightarrow$ The total amortized cost $\sum^{n-1} \hat{c}_i$ underestimate the total actual cost $\sum^{n-1} c_i$ by at most the drop in potential $\phi(T_{n-1}) - \phi(T_0)$ over the whole sequence of operations

- Key to amortized analysis is to pick the right potential function.

  In our case of splay tree, the tree operations possible are:

  - Zig : costs 1 (rotation)
  - Zig-zig: costs 2 (rotations)
  - Zig-zag: costs 2 (rotations)

  Given these operations, a good potential function is the total ranks.

  Define as follows:

  - Let $r(x)$ be total ranks of node x
    $s(x)$ be size of subtree under node x, $T(x)$
    $w(x)$ be the weight of node x, for simplicity, all nodes have weight $= 1$
  - Then: $s(x) = \sum_{y \in T(x)} w(y)$ <y is node in subtree $T(x)$>
    $r(x) = \log_2 s(x)$
  - Potential function:

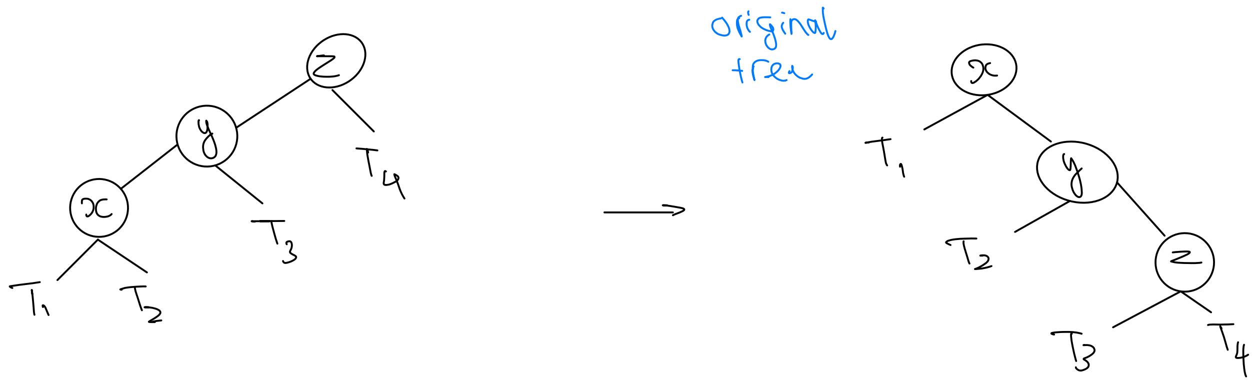
$$\phi(T) = \sum_{x \in T} r(x)$$

Some nice properties of $\phi(T)$ being the total ranks

- Rotation between 2 nodes only affect those nodes, and no other nodes in the tree. Furthermore, if node y was parent of node x before the rotation, then node x rank after the rotation is the same as node y's rank before the rotation.

- $size(x) + size'(z) = \overset{\text{subtree } x}{T(x)} + \overset{\text{subtree } z}{T'(z)} + x + z$

$$= \underset{\text{original tree}}{size(z)} - 1$$



With potential function picked, lets analyze the amortized cost of each operations:

- Zig operation:

$$\hat{c}_i = 1 + \overbrace{(r'(x) + r'(y))}^{\phi(T_i)} - \overbrace{(r(x) + r(y))}^{\phi(T_{i-1})}$$

$$= 1 + r'(x) - r(x) + \underbrace{r'(y) - r(y)}_{\leq 0}$$

$$\leq 1 + r'(x) - r(x)$$

node y dont increase rank

$$\Rightarrow \Delta\phi \leq r'(x) - r(x)$$

- Zig-Zig operation:

  - $\hat{c}_i = 2 + (\cancel{r'(x)} + r'(y) + r'(z)) - (r(x) + r(y) + \cancel{r(z)})$

$$= 2 + r'(z) - r(x) + \boxed{r'(y)} - \boxed{r(y)}$$

$$r'(y) \leq r'(x), \quad r(y) \geq r(x)$$

$$\leq 2 + r'(z) - r(x) + \boxed{r'(x)} - \boxed{r(x)}$$

$$= 2 + r'(z) - 2r(x) + r'(x) \quad (1)$$

  - Consider the term:

$$2r'(x) - r(x) - r'(z)$$

$$= \lg\left(\frac{s'(x)}{s(x)}\right) + \lg\left(\frac{s'(x)}{s'(z)}\right) \quad \text{bc } r(x) = \lg s(x)$$

$$= \lg\left(\frac{s'(x)^2}{s(x) \cdot s'(z)}\right)$$

$$\geq \lg\left(\frac{(s(x) + s'(z))^2}{s(x) \cdot s'(z)}\right) \quad \text{because } s'(x) \geq s(x) + s'(z)$$

<properties of ϕ>

$$= \lg\left(\frac{(1+1)^2}{1 \cdot 1}\right) = 2 \quad \text{because } s(x) \geq 1,\ s'(z) \geq 1$$

Therefore: $2r'(x) - r(x) - r'(z) \geq 2$ (2)

Replace ② into ①, we have:

$$\hat{c}_i \leq \left(\underbrace{2r'(x) - r(x) - r'(z)}_{②} \right) + r'(z) - 2r(x) + r'(x)$$

$$= 3r'(x) - 3r(x)$$

$$= 3\left(r'(x) - r(x) \right)$$

- Zig-zag operation:
  - $\hat{c}_i = 2 + \cancel{r'(x)} - r(x) + r'(y) - r(y) + r'(z) - \cancel{r(z)}$

$$= 2 - r(x) + r'(y) - r(y) + r'(z)$$

$$\leq 2 + r'(x) - 2r(x) + r'(x) \quad ①$$

By the same argument as before (Zig-zig operation), we have

$$2r'(x) - r(x) - r'(z) \geq 2 \quad ②$$

From ① and ②, we have:

$$\hat{c}_i \leq 3\left(r'(x) - r(x) \right)$$

Access Lemma

$$\boxed{\begin{array}{l} \text{The amortized time to splay a splay tree with root } t \text{ at node } x: \\ \text{amortized splay cost} \leq 3\left(r(t) - r(x)\right) + 1 \\ \qquad = O\left(1 + \log \frac{s(t)}{s(x)}\right) \end{array}}$$

Why $s'(x) \geq s(x) + s'(z)$?

Let $p = \frac{s(x)}{s'(x)}$ and $q = \frac{s'(z)}{s'(x)}$

We know that, under Zig-zig and Zig-zag operations:

$$\begin{cases} s'(x) \geq s(x) \\ s'(x) \geq s'(z) \end{cases} \Rightarrow \begin{cases} 0 \leq p \leq 1 \\ 0 \leq q \leq 1 \end{cases}$$

But also, both $s(x)$ and $s'(z)$ count the total weights of subtree $T(x)$ or subtree $T'(z)$, both does not include weight of y

$$\Rightarrow p + q \leq 1$$

$$\Leftrightarrow \frac{s(x)}{s'(x)} + \frac{s'(z)}{s'(x)} \leq 1$$

$$\Leftrightarrow s(x) + s'(z) \leq s'(x)$$

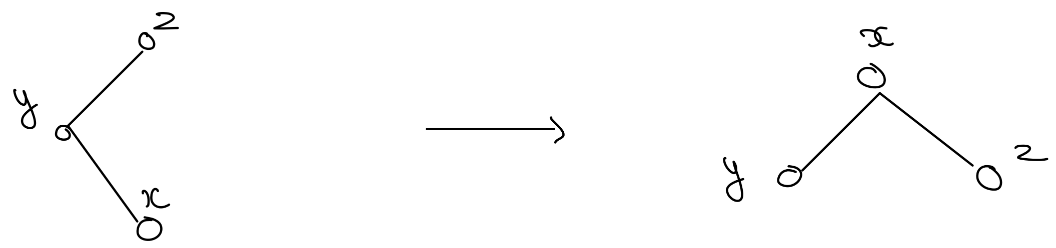
Potential Bound for each splay operation

1. Zig Operation

$$\Delta \Phi = \cancel{r'(x)} + r'(y) - r(x) - \cancel{r(y)}$$
$$= r'(y) - r(x)$$
$$\le r(y) - r(x) \qquad r(y) \ge r'(y)$$

2. Zig-Zag Operation:

Case 1: ~~rank doesn't increase between starting node and ending node~~



$r(z) > r(y) > r(x)$

$r'(x) = r(z)$
$r'(y) = ?$
$r'(z) = ?$

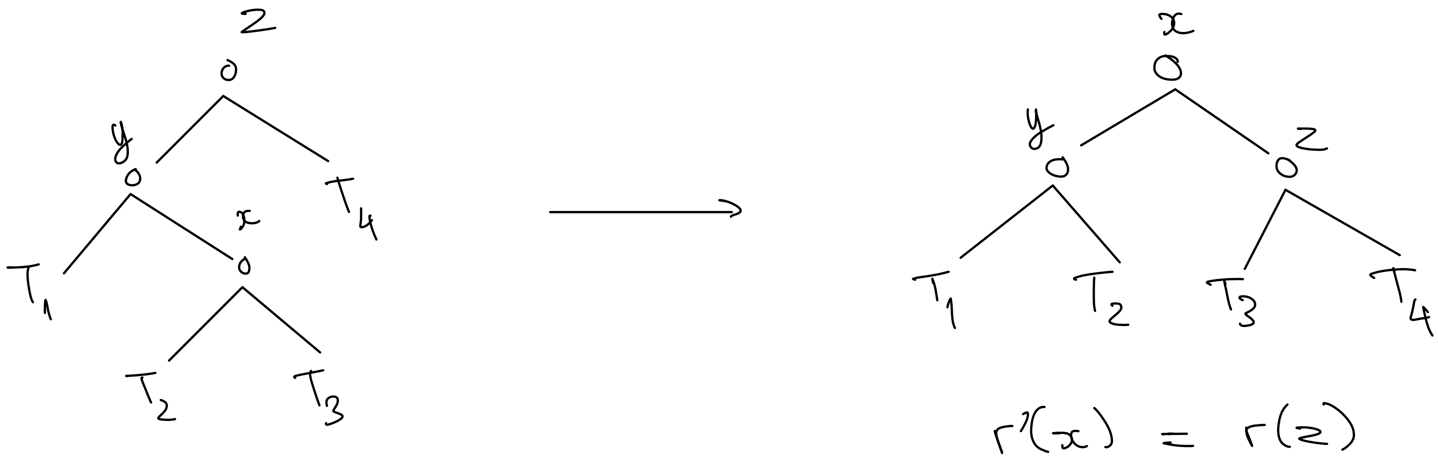
$$\Delta \phi = \cancel{r'(x)} + r'(y) + r'(z) - r(x) - r(y) - \cancel{r(z)}$$
$$= r'(y) + r'(z) - r(x) - r(y)$$
$$= r'(y) + r'(z) - 2\,r(z) \qquad r(z) = r(x) = r(y)$$

Rank rule: either $\left[\begin{array}{l} r'(y) < r(z) \\ r'(z) < r(z) \end{array}\right.$

$$\Rightarrow \Delta \phi \begin{cases} < r'(z) - r(z) & \text{if } r'(y) < r(z) \\ < r'(y) - r(z) & \text{otherwise} \end{cases}$$

$\Rightarrow$ $\Delta \Phi \downarrow$ at least d

Case 2: Rank increases between starting node and ending node



$r(x) < r(y) < r(z)$

$r'(x) = r(z)$
$r'(y) \le r(y)$
$r'(z) \le r(z)$

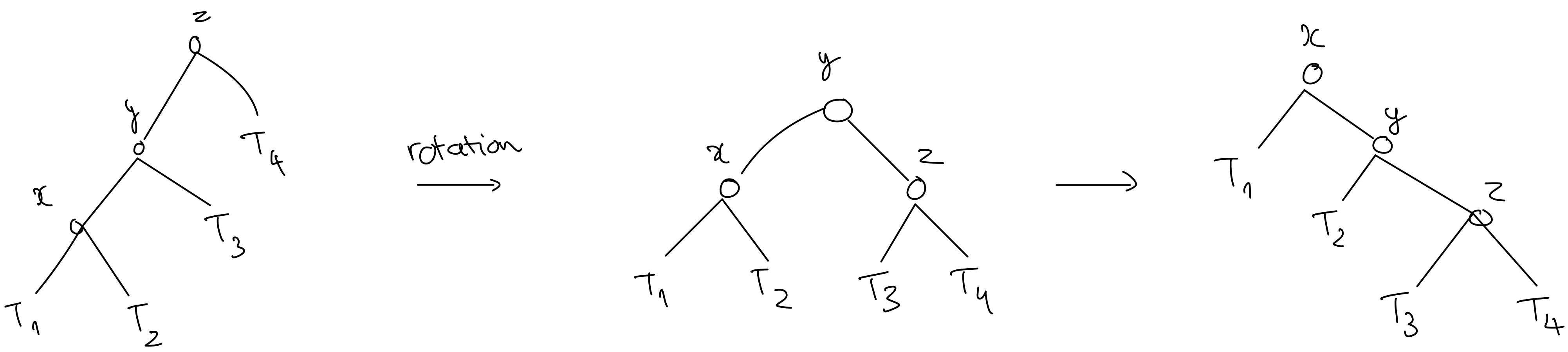
$$
\begin{aligned}
\Delta \phi &= \cancel{r'(x)} + r'(y) + r'(z) - r(x) - r(y) - \cancel{r(z)} \\
&= r'(y) + r'(z) - r(x) - r(y) \\
&\leq r'(z) - r(x) \\
&\leq r(z) - r(x)
\end{aligned}
$$

$r'(y) \leq r(y)$

$r'(z) \leq r(z)$

$r(z) \geq r(x) + 1$

3. Zig-Zig Operation:



$r(z) > r(y) > r(x)$

$r'(x) = r(x)$

$r'(y) = r(z)$

$r'(z) < r(z)$

$r''(z) < r(z)$

$r''(x) = r(z)$