Splay trees

Samo-modyfikujące się drzewa BST (Sleator, Tarjan 1983 rok).

Dowolny ciąg m operacji `Insert`, `Delete`, `Search` wykonywany jest w łącznym czasie $O(m \lg n)$, gdzie n oznacza maksymalny rozmiar drzewa w trakcie tych operacji.

Stąd, *zamortyzowany* czas jednej operacji: $O(\lg n)$.

Operacja `splay`

Po wykonaniu każdej z operacji `Insert`, `Delete`, `Search` wykonywana jest procedura `splay`, która przekształca drzewo tak, że najgłębszy wierzchołek x osiągnięty w trakcie wykonanej operacji staje się nowym korzeniem.

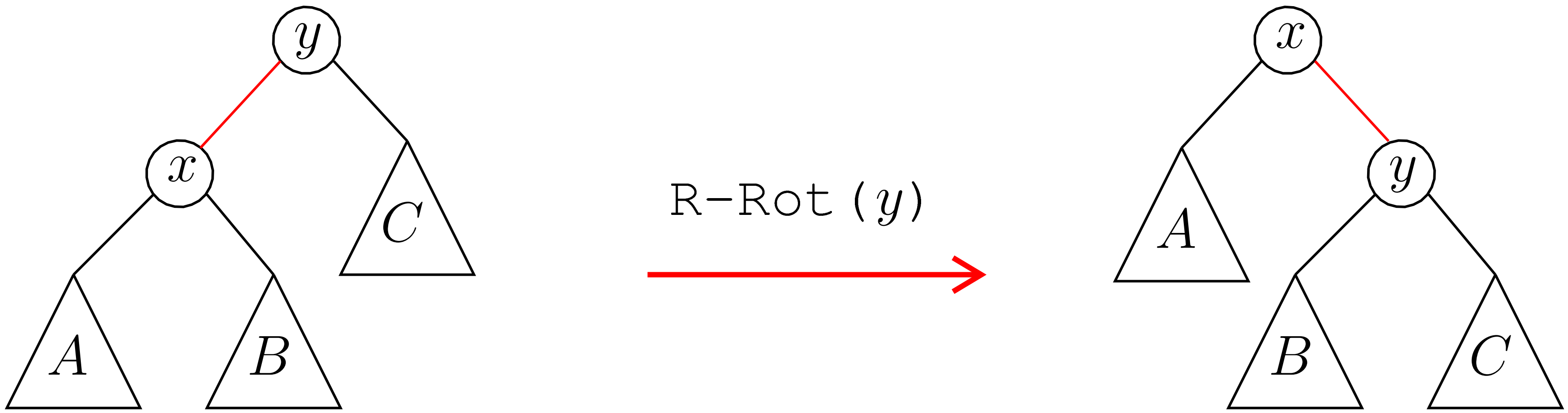
W przypadku `Insert`: x jest nowo wstawionym wierzchołkiem.

W przypadku `Search`: x jest znalezionym kluczem, lub liściem.

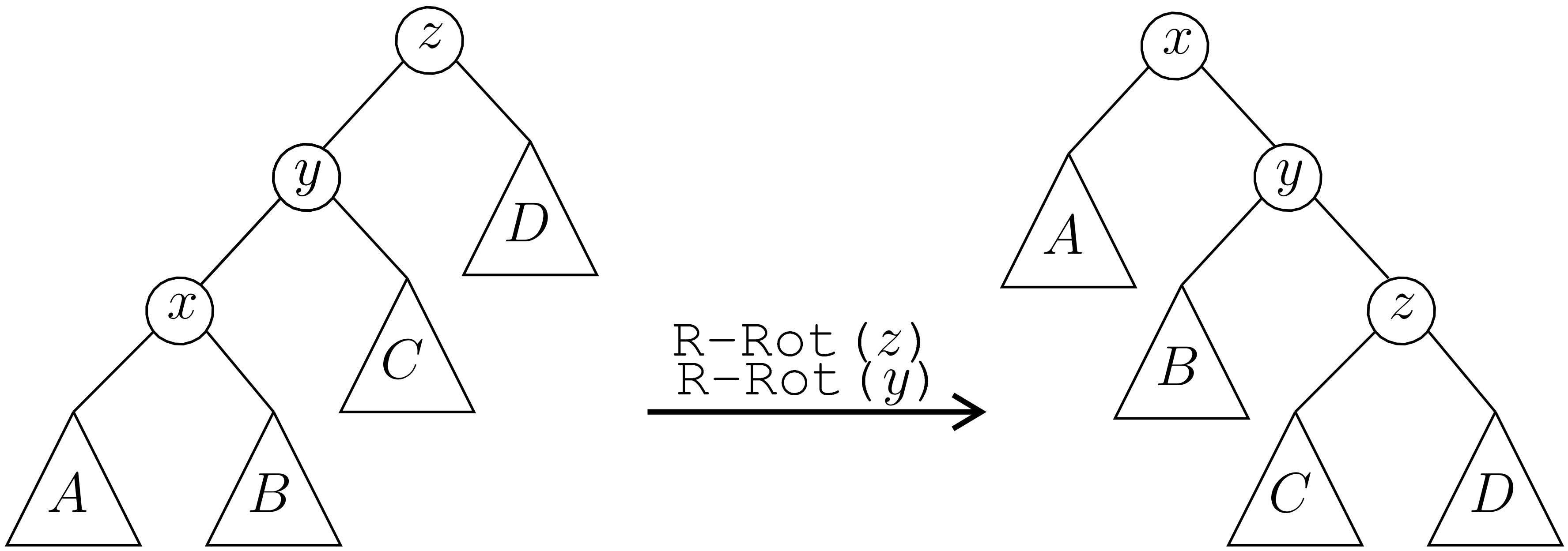
W przypadku `Delete`: x jest rodzicem *fizycznie* usuniętego węzła.

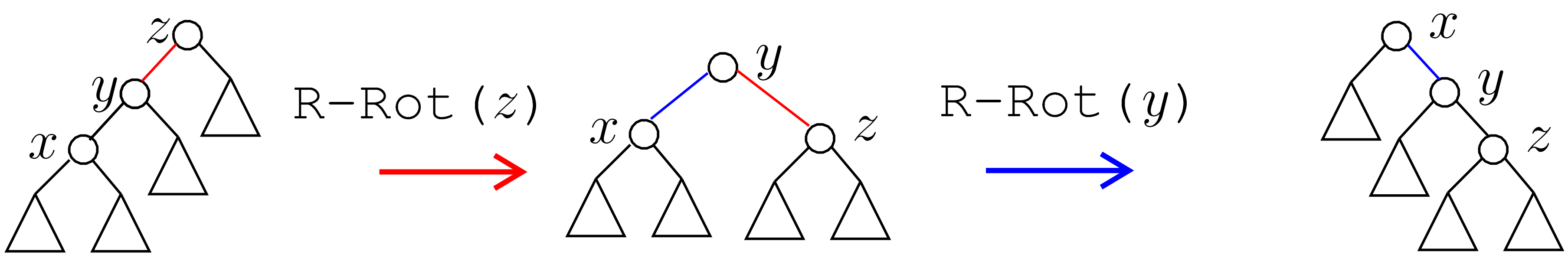
Operacja `splay` na x polega na wykonywaniu ciągu *kroków*, aż x stanie się korzeniem.
Każdy krok polega na wykonaniu jednej lub dwóch *rotacji*, zależnie od następujących przypadków:

Przypadek 1 (*zig*): x jest lewym synem korzenia y

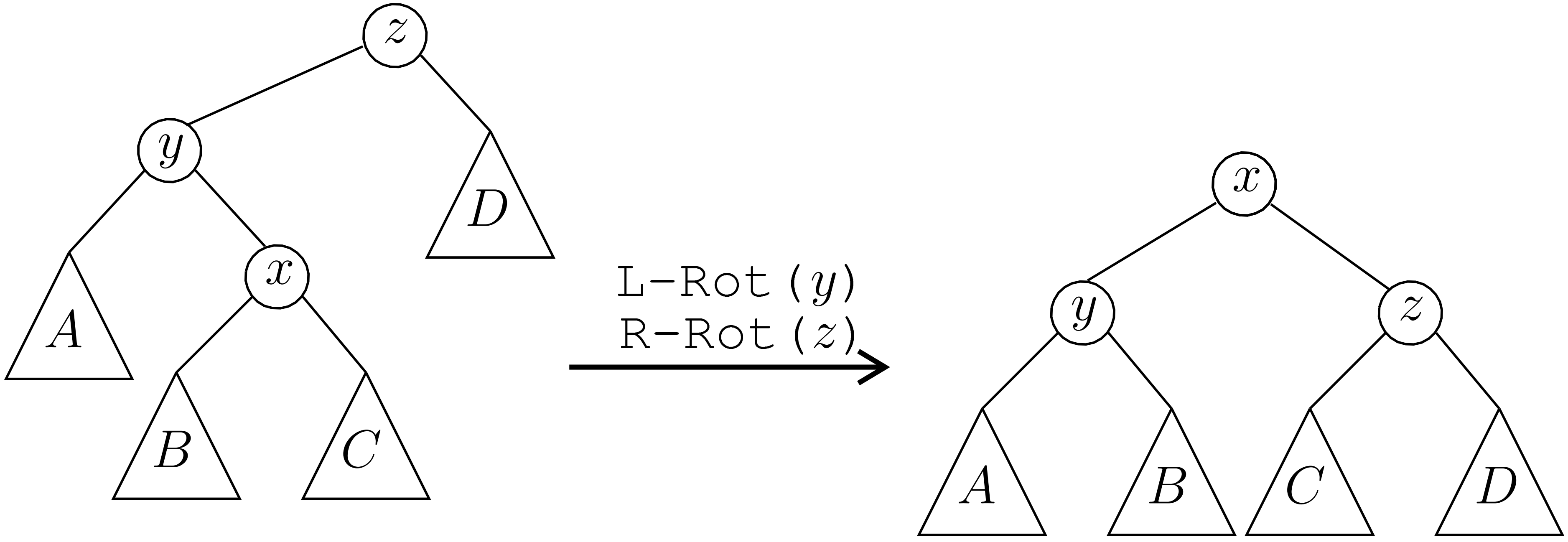


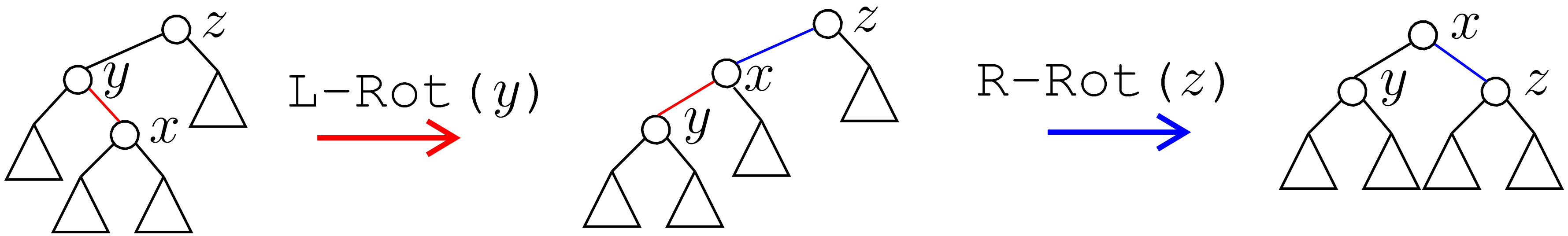
Przypadek 2 (*zig-zig*): x jest lewym synem y i y jest lewym synem z





Przypadek 3 (*zig-zag*): x jest prawym synem y i y jest lewym synem z





Symetryczne przypadki są przetwarzane symetrycznie.

Motywacje

Dla wierzchołka u drzewa BST T niech $l_T(u)$ oznacza liczbę wierzchołków na ścieżce od korzenia do u.

Niech $L(T) = \sum_{u \in T} l_T(u)$. (dla T pustego $L(T) = 0$).

Intuicyjnie: im większe $L(T)$ tym gorzej zbalansowane jest T.

Jeśli drzewo o n wierzchołkach jest zrównoważone to $L(T) = O(n \lg n)$, a jeśli $L(T) = \Omega(n^2)$, to drzewo jest niezrównoważone.

Dla $u \in T$, *wagą* $w_T(u)$ nazywamy liczbę wierzchołków w drzewie o korzeniu w.

Niech $W(T) = \sum_{u \in T} w_T(u)$.

Można sprawdzić, że $W(T) = L(T)$. (Ćwiczenie)

Niech $r_T(u) = \lg w_T(u)$ (*ranga* u).

Potencjałem drzewa T nazywamy $\Phi(T) = \sum_{u \in T} r_{T(u)}$.

Można sprawdzić, że spośród drzew o ustalonej liczbie wierzchołków, lepiej zrównoważone są te o mniejszym potencjale.

Zadaniem procedury `splay` jest zmniejszenie potencjału drzewa.

Analiza

Zauważmy, że łączny czas wszystkich operacji jest proporcjonalny do łącznej liczby kroków we wszystkich wywołaniach `splay`:

Każdy krok procedury `splay` wykonuje stałą liczbę rotacji a czas działania `Insert`, `Delete`, albo `Search` jest zdominowany przez czas wywoływanej przez siebie procedury `splay`.

Zatem szacowanym przez nas kosztem będzie liczba wykonanych rotacji procedury `splay`.

Lemat 0. Dla dodatnich liczb rzeczywistych a, b i c jeśli $a+b \leqslant c$, to $\lg a + \lg b \leqslant 2\lg c - 2$.

D-d. $\lg$ jest funkcją niewypukłą.

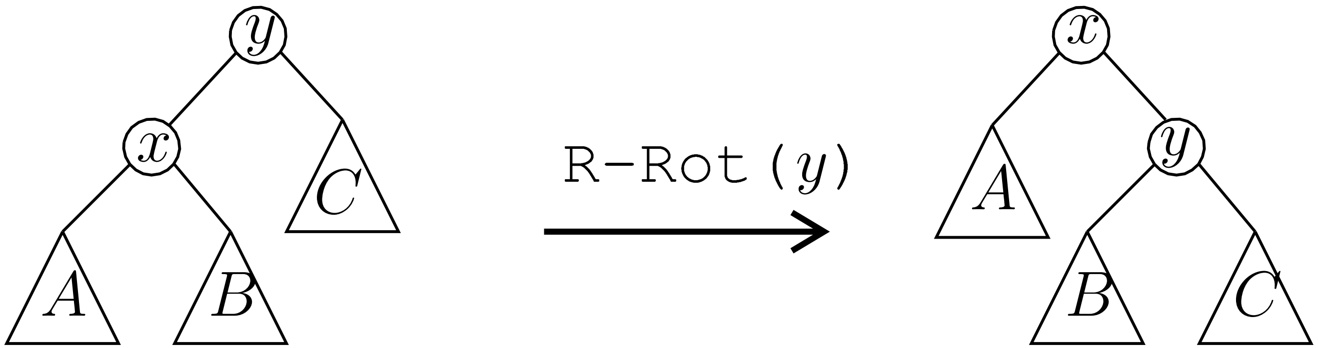
Stąd $\frac{\lg a + \lg b}{2} \leqslant \lg\left(\frac{a+b}{2}\right) \leqslant \lg \frac{c}{2} = \lg c - 1$.

Pomnóżmy obie strony przez dwa. $\square$

Lemat 1. *Zamortyzowany* koszt *kroku zig* procedury `splay`(x) przekształcającego drzewo T w T' wynosi:

$$a = 1 + \Phi(T') - \Phi(T) \leqslant 1 + 3[r_{T'}(x) - r_T(x)]$$

D-d.



Zmieniają się jedynie rangi x i y. Zatem:

$a = 1 + \Phi(T') - \Phi(T) = 1 + [r_{T'}(x) + r_{T'}(y)] - [r_T(x) + r_T(y)]$.

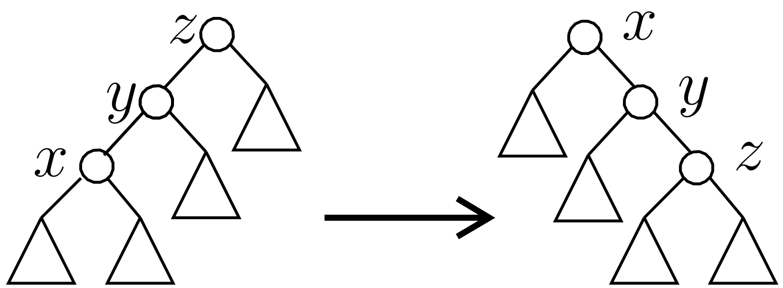
Ponieważ $r_{T'}(y) \leqslant r_T(y)$: $\quad a \leqslant 1 + [r_{T'}(x) - r_T(x)]$.

Ponieważ $r_{T'}(x) \geqslant r_T(x)$: $\quad a \leqslant 1 + 3[r_{T'}(x) - r_T(x)]$. $\square$

Lemat 2. *Zamortyzowany* czas *kroku zig-zig* procedury $\texttt{splay}(x)$ przekształcającego drzewo T w T' wynosi:

$$a = 2 + \Phi(T') - \Phi(T) \leqslant 3[r_{T'}(x) - r_T(x)]$$

D-d.

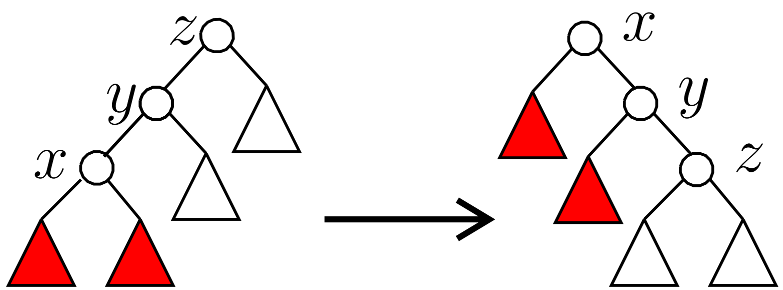


$$a = 2 - \Phi(T') - \Phi(T)$$
$$= 2 + [r_{T'}(x) + r_{T'}(y) + r_{T'}(z)] - [r_T(x) + r_T(y) + r_T(z)]$$

Ponieważ $r_{T'}(x) = r_T(z)$: $\qquad a = 2 + r_{T'}(y) + r_{T'}(z) - r_T(x) - r_T(y)$

Ponieważ $r_T(x) \leqslant r_T(y)$ oraz $r_{T'}(x) \geqslant r_{T'}(y)$:

$$a \leqslant 2 + r_{T'}(x) + r_{T'}(z) - 2r_T(x)$$



Zauważmy, że $\quad w_T(x) + w_{T'}(z) < w_{T'}(x)$

Z Lematu 0 mamy: $\quad r_T(x) + r_{T'}(z) \leqslant 2r_{T'}(x) - 2$

Stąd: $\quad r_{T'}(z) \leqslant 2r_{T'}(x) - r_T(x) - 2$.

Wstawiając to do: $a \leqslant 2 + r_{T'}(x) + r_{T'}(z) - 2r_T(x)$ otrzymujemy:

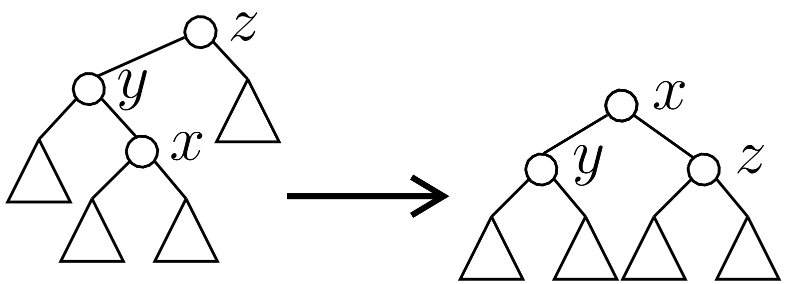
$$\begin{aligned} a &\leqslant 2 + r_{T'}(x) + 2r_{T'}(x) - r_T(x) - 2 - 2r_T(x) \\ &= 3[r_{T'}(x) - r_T(x)] \end{aligned}$$

$\square$

Lemat 2. *Zamortyzowany* czas *kroku zig-zag* procedury $\texttt{splay}(x)$ przekształcającego drzewo T w T' wynosi:

$$a = 2 + \Phi(T') - \Phi(T) \leqslant 3[r_{T'}(x) - r_T(x)]$$

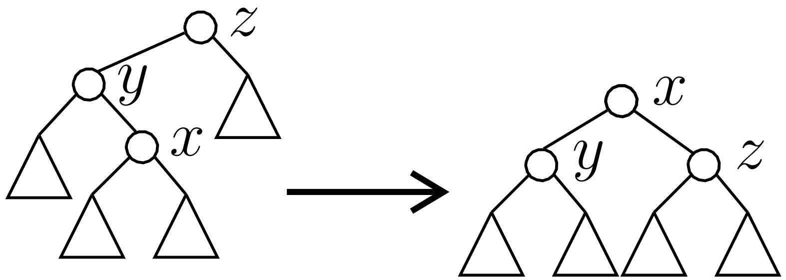
D-d.



$$a = 2 - \Phi(T') - \Phi(T)$$
$$= 2 + [r_{T'}(x) + r_{T'}(y) + r_{T'}(z)] - [r_T(x) + r_T(y) + r_T(z)]$$

Ponieważ $r_{T'}(x) = r_T(z)$: $a = 2 + r_{T'}(y) + r_{T'}(z) - r_T(x) - r_T(y)$

Ponieważ $r_T(x) \leqslant r_T(y)$: $a \leqslant 2 + [r_{T'}(y) + r_{T'}(z)] - 2r_T(x)$



Zauważmy, że: $w_{T'}(y) + w_{T'}(z) < w_{T'}(x)$

Z Lematu 0: $r_{T'}(y) + r_{T'}(z) \leqslant 2r_{T'}(x) - 2$

Wstawiając to do $a \leqslant 2 + [r_{T'}(y) + r_{T'}(z)] - 2r_T(x)$ otrzymujemy:

$$a \leqslant 2 + [2r_{T'}(x) - 2] - 2r_T(x)$$
$$= 2[r_{T'}(x) - r_T(x)]$$
$$\leqslant 3[r_{T'}(x) - r_T(x)]$$ (ponieważ: $r_{T'}(x) \geqslant r_T(x)$).

□

Twierdzenie 1. *Zamortyzowana* liczba pojedynczych rotacji w operacji `splay` na BST z n wierzchołkami jest $\leqslant 3\lg n + 1$.

D-d. Załóżmy, że `splay` jest wywołana na wierzchołku $x = w_0$ w T.
Niech T' - drzewo po zakończeniu `splay`.

Niech $w_0, \ldots, w_{k-1}, w_k$ - wierzchołki na ścieżce od $x = w_0$ do korzenia w_k drzewa T,
takie że w_i jest dziadkiem w_{i-1} dla $1 \leqslant i \leqslant k-1$,
a w_{k-1} jest synem lub wnukiem w_k.

Niech $T_0 = T$, a T_j będzie drzewem po j-tym kroku `splay`.
(Stąd: $T_k = T'$)
Łatwo zauważyć, że: $r_{T_j}(x) = r_T(w_j)$ dla $j = 0, 1, \ldots, k$.
(Po j-tym kroku x staje się korzeniem poddrzewa z węzłami, które były w poddrzewie o korzeniu w_j.)

Niech t oznacza *faktyczną* liczbę rotacji w `splay`, a t_j - liczbę rotacji w j-tym kroku (tj. 1 lub 2). $t = \sum_{j=1}^{k} t_j$.

Wtedy *zamortyzowana* liczba rotacji wynosi:

$$t + \Phi(T') - \Phi(T) = t + \Phi(T_k) - \Phi(T_0) = \sum_{j=1}^{k} [t_j + \Phi(T_j) - \Phi(T_{j-1})]$$

Z Lematów 2 i 3 (kroki *zig-zig* lub *zig-zag*):

$$t_j + \Phi(T_j) - \Phi(T_{j-1}) \leqslant 3[r_{T_j}(x) - r_{T_{j-1}}(x)] \quad \text{dla } j = 1, \ldots, k-1$$

Z Lematów 1, 2 i 3 (krok *zig-zig* lub *zig-zag* lub *zig*):

$$t_k + \Phi(T_k) - \Phi(T_{k-1}) \leqslant 1 + 3[r_{T_k}(x) - r_{T_{k-1}}(x)]$$

Stąd (oszacowanie sumą *teleskopową*):

$$\begin{aligned}\sum_{j=1}^{k} [t_j + \Phi(T_j) - \Phi(T_{j-1})] &\leqslant 1 + \sum_{j=1}^{k} 3[r_{T_j}(x) - r_{T_{j-1}}(x)] \\ &= 1 + 3[r_{T_k}(x) - r_{T_0}(x)]\end{aligned}$$

Wiemy, że $r_{T_k}(x) = r_T(w_k) = \lg n$ oraz $r_{T_0}(x) = r_T(w_0) \geqslant 0$.

Stąd: $t + \Phi(T') - \Phi(T) \leqslant 1 + 3\lg n$

$\square$

Można wykazać, że wykonanie `Insert` na n-wierzchołkowym BST zwiększa jego potencjał o nie więcej niż $\lg n$ (ćwiczenie: wskazać najgorszy przypadek i oszacować przyrost potencjału przez sumę *teleskopową*) oraz, że `Delete` nie może zwiększyć potencjału (ćwiczenie).

Z tego wynika, że wykonanie ciągu m operacji słownikowych (tj. `Insert, Delete, Search` uzupełnionych o operację `splay`) na początkowo pustym drzewie, wykonuje łącznie $m(4 \lg n+1)$ rotacji, gdzie n jest maksymalnym rozmiarem drzewa w trakcie tych operacji.

(Czyli zamortyzowany koszt jednej operacji jest $\leqslant 4\lg n+1$.)

Dla danego BST A i wartości klucza x operacja $\texttt{split}(x, A, B, C)$ tworzy z elementów drzewa A dwa nowe BST B i C, takie że B zawiera elementy $\leqslant x$ a C - elementy $> x$.

Operacja $\texttt{join}(A, B)$ skleja dwa BST A i B, takie że maksymalny klucz w A jest mniejszy niż minimalny klucz w B, w jedno BST.

Przy pomocy operacji `splay` można efektywnie zaimplementować `split` i `join`. (Ćwiczenie.)

Porównanie ze zrównoważonymi BST

- nie wymagają dodatkowej informacji w węzłach (np. kolor w drzewach czerwono-czarnych)
- struktura adaptuje się dynamicznie do wzorca zapytań (często wyszukiwane klucze przemieszczają się w okolice korzenia)
- faktyczny koszt jednej operacji może być $\Omega(n)$ (wada np. w aplikacjach interaktywnych).