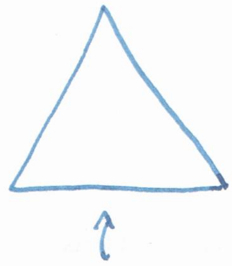
Dijkstra



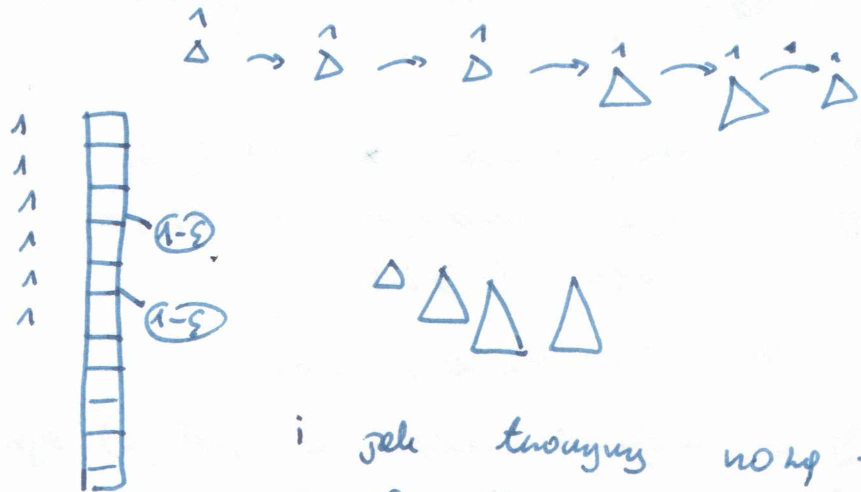
n elem

n - op. delete_min

m - op. decrease_key

koszt Dijkstry $(n \log n + m)$

## Lepsze wytłumaczenie $2\log n$ dla delete_min:



$(\log n) + \log n$

koszt na rozbicie tego co usuwasz

i jak tworzymy nową listę, to musimy przejść całą tablicę i płacimy (1) jeśli w tablicy nie ma drzewa $B_i$, natomiast jeśli mamy $(1-\varepsilon)$ to płacimy $(1-\varepsilon)$ na przepisanie do listy a (1) dajemy jako kredyt na górę. Mamy $2\log(n)$

# DRZEWA SAMOORGANIZUJĄCE SIĘ (drzewo Splay)

drzewo BST;

operacje słownikowe:

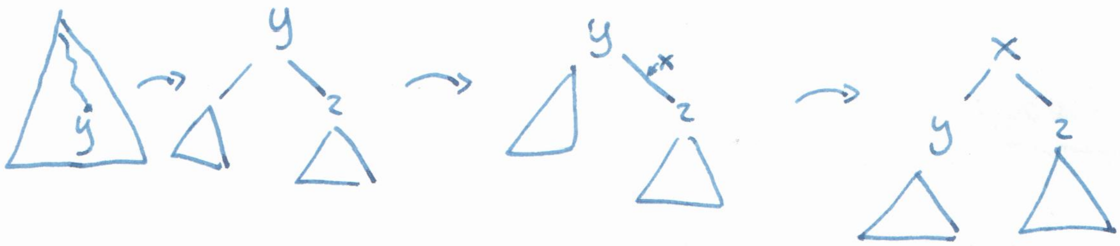
- insert(x)
- find(x)
- delete(x)
- Split(x) - rozbicie drzewa T na $T_1$ & $T_2$, każdy klucz w $T_1 \leq k$, $T_2 \geq k$
- join(x) - zał: klucze $T_1 \leq$ klucze $T_2$
  - wynik: drzewo z kluczami z $T_1$ i $T_2$

Niezmienne operacje Splay(x):

przesuń do korzenia rotacjami (sprytnie)

- x jeśli x jest w drzewie
- w.p.p. poprzednik lub następnik x-a, tj. ostatni v, w którym byliśmy robiąc find

- insert(x): generalnie opcje są dwie: albo wrzucamy x i przepychamy splay'em albo najpierw przepychamy splay:

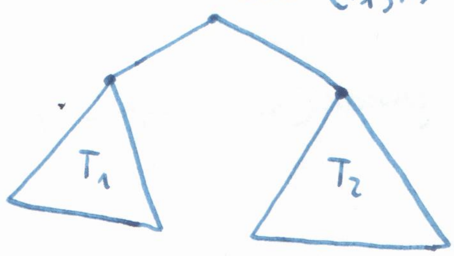



- join($T_1$, $T_2$)

robimy splay($T_1$, x) ← jeśli klucz z $T_2$

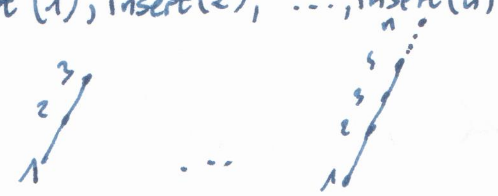
Wtedy w korzeniu $T_1$ mamy max $T_1$. Teraz możemy zrobić takie drzewo (bo x już ofc nie ma w $T_1$)

max klucz ($T_1$, x)

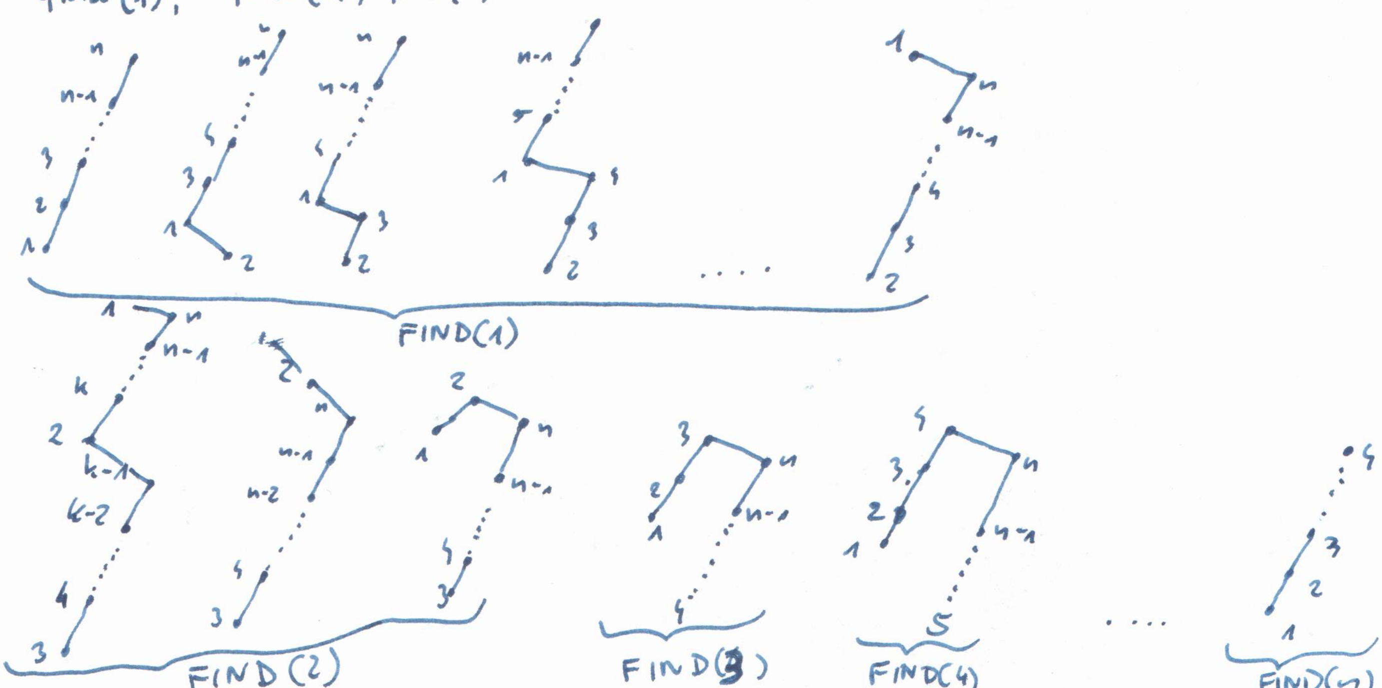



27.04.18

insert(1), insert(2), ..., insert(n)



find(1), find(2), find(3)



Widać, że nas to boli i wcale nam to drzewo się nie równoważy ani nic.

## Zróbmy, więc inaczej:

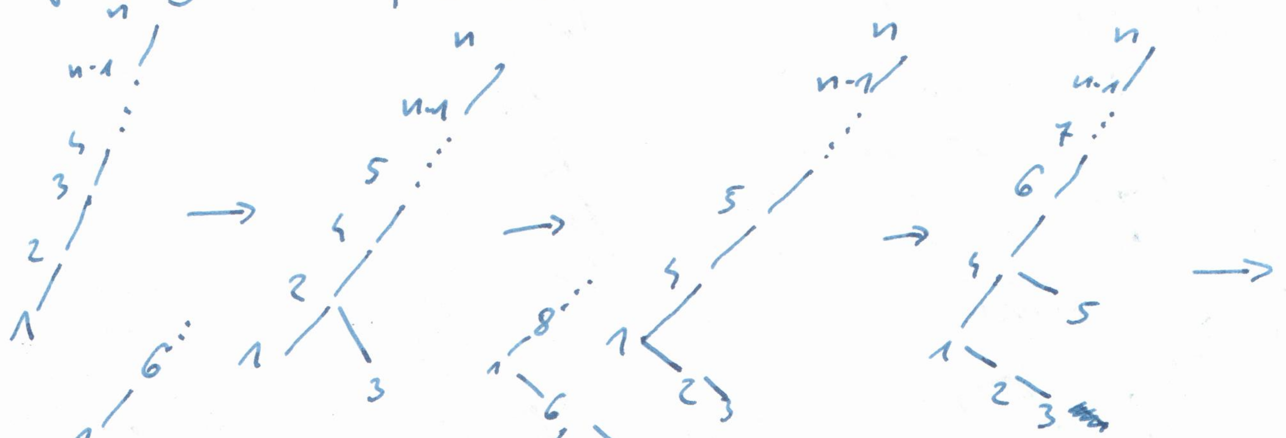
Podczas splay rotując x rozważamy trzy przypadki:

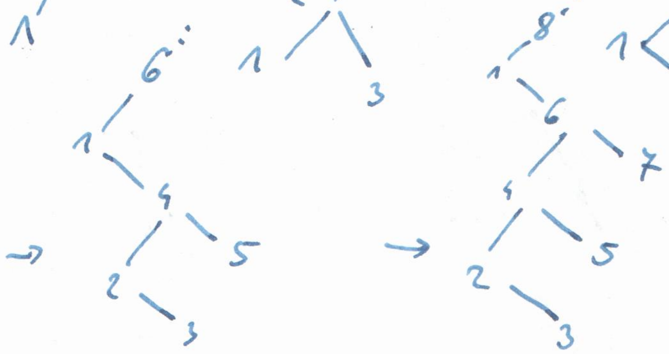
1.  (rotacja(x)) (x jest synem korzenia)

2.  → (rotacja(x), rotacja(x))

3.  → (rotacja(y), rotacja(x))

Popatrzmy na find(1)





Widać, że górny szkielet drzewa skraca się o połowę

Ozn.

Jeśli $x$ - wierzchołek, to $S(x)$ - poddrzewo o korzeniu $x$

$|S(x)|$ - # wierzchołków w $S(x)$

$\mu(S(x)) = \lfloor \log(|S(x)|) \rfloor$

(Niezmiennik): Na koncie każdego wierzchołka $x$ jest co najmniej $\mu(S(x))$ jednostek

FAKT:

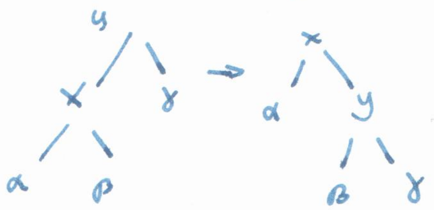
Każda operacja SPLAY($x$) wymaga nie więcej niż $3[\mu(\text{korzeń}) - \mu(x)] + 1$ jednostek do wykonania się i zachowania niezmiennika

D-d

Ozn. $y$ - ojciec $x$-a, $z$ - ojciec $y$-a (o ile istnieje)

$\mu(x) \equiv \mu(S(x))$ - stan konta $x$-a przed rot, $\mu'(x)$ - stan konta $x$-a po rotacji

od. 1 $x$ jest synem korzenia



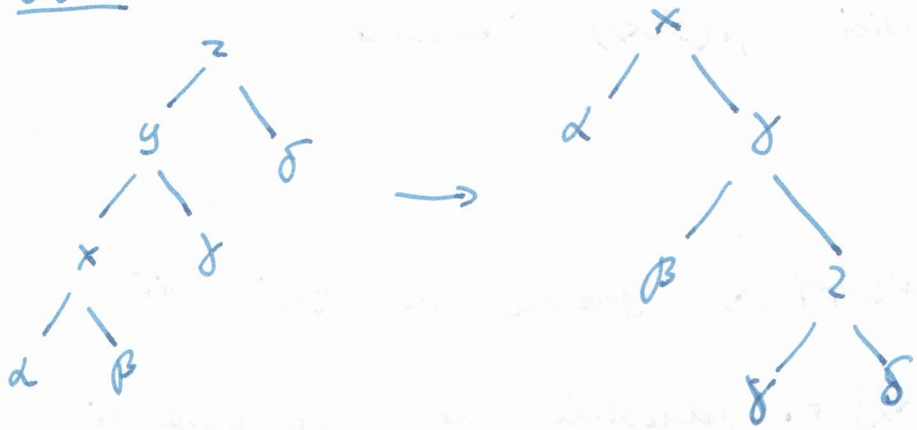
zmiana stanu konta:

$$\mu'(x) + \underset{\mu'(x)}{\underset{\wedge}{\mu'(y)}} - \mu(x) - \underset{\mu(x)}{\underset{\vee}{\mu(y)}} \leq 2\mu'(x) - 2\mu(x) \leq 3[\mu'(x) - \mu(x)]$$

Jeśli odejmiemy $\mu' - \mu$ po wszystkich wierzchołkach to dostaniemy tyle, ile musimy mieć by wykonać operację rotacji. Zauważmy, że poddrzewa inne niż zakorzenione w $x$ i $y$ się nie zmienią, zatem ich $\mu$ nie ulegnie zmianie. Pozostaje policzyć różnicę: $(\mu'(x) + \mu'(y)) - (\mu(x) + \mu(y))$. Szacując rozmiary poddrzew zakorzenionych ~~przez~~ ~~x~~ w $y$ dostaniemy $3[\mu'(x) - \mu(x)]$

Ale bierzemy $3[\mu'(x) - \mu(x)] \underline{+1}$, bo $[\mu'(x) - \mu(x)]$ może się wyzerować, czyli nie mamy co prawda z czego zapłacić za naprawienie kont, bo się będą same, ale za operację fizyczną trzeba zapłacić. Ogólnie to wystarczyłoby nam $2[\mu'-\mu]+1$, ale weźmiemy 3, bo przyda się nam w kolejnych podpunktach.

## ad. 3



Zmiana stanu kont:

(*) $\mu'(x) + \mu'(y) + \mu'(z) - \mu(x) - \mu(y) - \mu(z)$

Obserwacje:

$\mu(x) \leq \mu(y) \leq \mu(z) = \mu'(x)$ oraz $\mu'(z) \leq \mu'(y) \leq \mu'(x)$

oraz $\mu'(x) = \mu(z)$.

Z ost. obserwacji:

(**) $\mu'(y) + \mu'(z) - \mu(x) - \mu(y) \leq$

A dodając jeszcze nierówności z obserwacji mamy

(**) $\leq 2[\mu'(x) - \mu(x)]$. Możemy przemnożyć na ten etap $3[\mu'(x) - \mu(x)] + 1$

Na opłacenie operacji niskiego rzędu mamy $\mu'(x) - \mu(x)$.

Problem: co jeśli $\mu'(x) = \mu(x)$

Pokażemy, że:

$$\begin{cases} \mu'(x) = \mu(x) \\ (**) \geqslant 0 \end{cases}$$ są sprzeczne

Z tego, że $\mu'(x) = \mu(x)$ oraz $(*)$ mamy $\mu(x) = \mu(y) = \mu(z) = \mu'(x)$

$(**) \geqslant 0 \Rightarrow \mu'(x) + \mu'(y) + \mu'(z) \geqslant \mu(x) + \mu(y) + \mu(z) = 3\mu'(x)$,

czyli $\mu'(y) + \mu'(z) \geqslant 2\mu'(x)$. Z tego oraz z $\mu'(z) \leqslant \mu'(y) \leqslant \mu'(x)$

mamy $\mu'(x) = \mu'(y) = \mu'(z)$

Tak więc z zał. mamy:

$$\mu(x) = \mu(y) = \mu(z) = \mu'(x) = \mu'(y) = \mu'(z)$$

Niech $a = |S(x)|$, $b = |$w nowym drzewie $z|$

$a + b + 1 = |S(z)|$

$\lfloor \log(a) \rfloor = \lfloor \log(a+b+1) \rfloor = \lfloor \log(b) \rfloor$

Niech $a \geqslant b$ wówczas

$$\lfloor \log(a+b+1) \rfloor > \lfloor \log(2b) \rfloor \geqslant \lfloor \log(b) \rfloor + 1$$

$\lightning$ sprzeczność

# DRZEWCE (TREAPS)

9.05.2018

$k_1 \leqslant k_2 \leqslant \ldots \leqslant k_n$

$p_1 \quad p_2 \quad \ldots \quad p_n$

- pary: klucz, priorytet w wierzchołkach
- drzewo binarne.
- względem klucza BST
- względem priorytetów kopiec