

## DRZEWY SAMOORGANIZUJĄCE SIĘ

## 1 Wprowadzenie

Drzewa samoorganizujące się są kolejnym przykładem struktury danych opartej na binarnych drzewach przeszukiwań. Przymiotnik "samoorganizujące" oznacza, że drzewa te w trakcie wykonywania na nich operacji zmieniają swoją strukturę automatycznie, stosując pewną prostą heurystykę. W przeciwieństwie do drzew zbalansowanych (AVL, czerwono-czarnych) heurystyka ta nie korzysta z żadnych dodatkowych informacji pamiętanych w wierzchołkach. Druga istotna różnica polega na tym, że teraz pojedyncze operacje słownikowe mogą być kosztowne. Jak jednak pokażemy, zamortyzowany koszt ciągu operacji jest niski.

## 2 Operacje na drzewach samoorganizujących się

Oprócz operacji słownikowych ( $find(i, S)$ ,  $insert(i, S)$ ,  $delete(i, S)$ ), odpowiednio wyszukiwania, wstawiania i usuwania klucza  $i$  w (do, z) drzewie  $S$ ) rozważymy realizację następujących operacji:

- $join(S_1, S_2)$  - połącz drzewa  $S_1$  i  $S_2$  w jedno drzewo (przy założeniu, że każdy klucz w drzewie  $S_1$  jest nie większy od każdego klucza z drzewa  $S_2$ ),
- $split(i, S)$  - rozdziel  $S$  na dwa drzewa  $S_1$  i  $S_2$  takie, że każdy klucz w  $S_1$  jest nie większy od  $i$ , a każdy klucz w  $S_2$  jest nie mniejszy od  $i$ .

## 3 Implementacja operacji

Podstawowa idea drzew samoorganizujących się polega na tym, by wierzchołki drzewa zawierające klucz  $i$  (parametr operacji  $insert$ ,  $delete$ ,  $find$ ,  $split$ ) przesuwac serią rotacji do korzenia. Umiejętnie wykonywane rotacje będą powodować "spłaszczenie" drzewa.

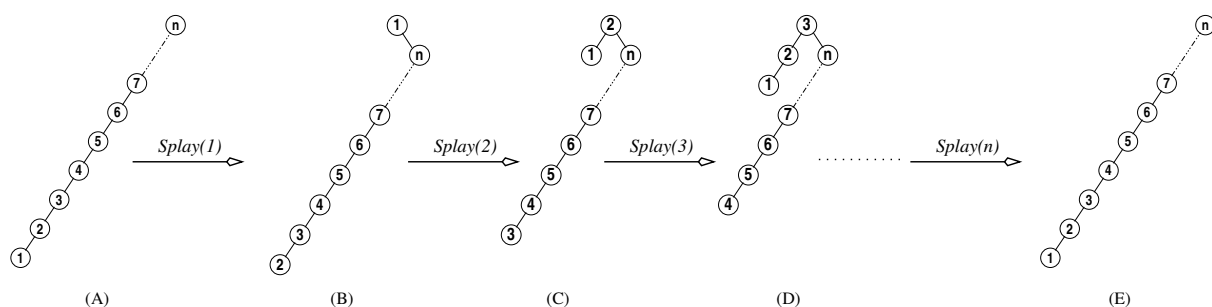
Wygodnie jest nam wprowadzić operację *Splay*, w terminach której wyrazimy wszystkie interesujące nas operacje.

**Definicja 1**  $Splay(j, S)$  - przeorganizuj  $S$  tak, by jego korzeniem stał się wierzchołek zawierający klucz  $k$  taki, że w  $S$  nie ma elementu leżącego między  $k$  i  $j$ .

Tak więc jeśli  $j$  znajduje się w  $S$  to operacja  $Splay(j, S)$  przesunie  $j$  do korzenia. W przeciwnym razie w korzeniu znajdzie się  $k = \min\{x \in S \mid x > j\}$  lub  $k = \max\{x \in S \mid x < j\}$ .

## 4 Implementacja Splay(x)

Splay łatwo jest zaimplementować przy pomocy rotacji. Jedną z możliwości jest stosowanie rotacji do elementu  $x$  tak długo, aż znajdzie się on w korzeniu. Jak jednak pokazuje poniższy przykład, taka implementacja powoduje, że niektóre ciągi operacji słownikowych byłyby wykonywane w czasie kwadratowym od długości ciągu.

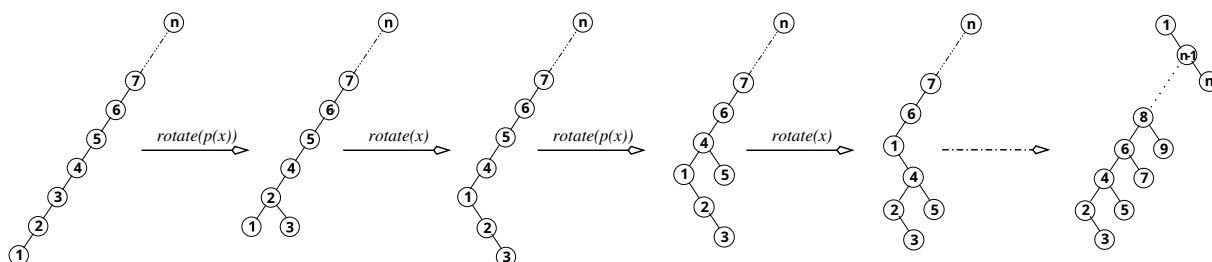


Drzewo (A) może powstać na skutek wykonania ciągu instrukcji:  $insert(1), insert(2), \dots, insert(n)$ . Kolejne operacje:  $Splay(1), Splay(2), \dots, Splay(n-1)$  wykonują odpowiednio:  $n-1, n-1, n-2, n-3, \dots, 1$  rotacji. Po wykonaniu  $Splay(n)$  otrzymujemy z powrotem drzewo (A).  $\square$

Wobec tego musimy zaproponować inny sposób implementacji. Rozważamy 3 przypadki:

- (a)  $x$  ma ojca, ale nie ma dziadka  $\rightarrow rotate(x)$ ,
- (b)  $x$  ma ojca  $p(x)$  i ma dziadka;  $x$  i  $p(x)$  są obydwaj lewymi bądź obydwaj prawymi synami swoich ojców  $\rightarrow rotate(p(x)); rotate(x)$ ,
- (c)  $x$  ma ojca  $p(x)$  i ma dziadka;  $x$  jest lewym a  $p(x)$  prawym synem, bądź na odwrót  $\rightarrow rotate(x); rotate(x)$ .

## PRZYKŁAD 2



To samo zdegenerowane drzewo co w Przykładzie 1 po wykonaniu  $Splay(1)$ , w opisany powyżej sposób, zostaje istotnie spłaszczone - jego wysokość została zredukowana o połowę.  $\square$

## 5 Analiza

Stosujemy analizę zamortyzowaną. Każdy wierzchołek drzewa przechowuje pewien depozyt. Operacja wykonywana na drzewie może zwiększać depozyty, bądź też może być opłacana przez kwoty z depozytów.

### OZNACZENIA

$S(x)$  - poddrzewo o korzeniu w  $x$ ,

$|S|$  - liczba wierzchołków w drzewie  $S$ ,

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

$$\mu(x) = \mu(S(x)).$$

Będziemy utrzymywać następujący niezmiennik:

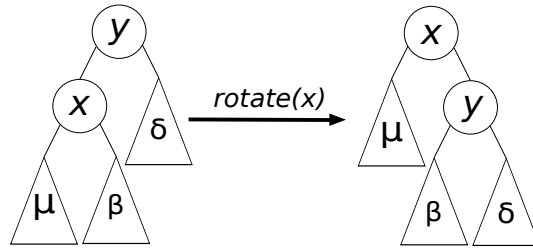
*Wierzchołek  $x$  ma zawsze co najmniej  $\mu(x)$  jednostek na swoim koncie.*

*Insert* daje wierzchołkowi pewien początkowy depozyt.

**Lemat 1** *Każda operacja  $Splay(x, S)$  wymaga nie więcej niż  $3(\mu(S) - \mu(x)) + 1$  jednostek do wykonania operacji i zachowania niezmiennika kredytowego.*

DOWÓD: Rozważamy trzy przypadki jakie możemy napotkać, przesuwając  $x$  w kierunku korzenia, podczas operacji *Splay*. Bez zmniejszenia ogólności zakładamy, że  $x$  jest lewym synem swego ojca.

- (a) W tym przypadku  $x$  nie ma dziadka i wykonujemy pojedynczą rotację  $rotate(x)$ . Niech  $y$  oznacza ojca  $x$ -a.



Niech  $\mu$  oznacza depozyty wierzchołków przed wykonaniem tej rotacji, a  $\mu'$  - depozyty po jej wykonaniu.

Ponieważ jedynie poddrzewa zakorzenione w  $x$  i w  $y$  mogły zmienić swoją wielkość, aby utrzymać niezmiennik musimy zapłacić:

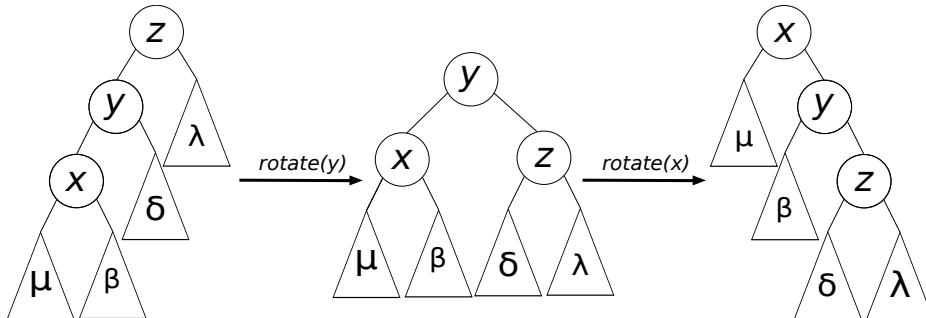
$$\mu'(x) + \mu'(y) - \mu(x) - \mu(y).$$

Łatwo zauważyć, że  $\mu'(x) = \mu(y)$ ,  $\mu'(x) \geq \mu(x)$  oraz  $\mu'(y) \leq \mu'(x)$ . Tak więc

$$\mu'(x) + \mu'(y) - \mu(x) - \mu(y) = \mu'(y) - \mu(x) \leq \mu'(x) - \mu(x) \leq 3(\mu'(x) - \mu(x)).$$

Mając do dyspozycji  $3(\mu'(x) - \mu(x)) + 1$  jednostek jesteśmy w stanie utrzymać niezmiennik i pozostanie nam jeszcze jedna jednostka na opłacenie operacji niskiego poziomu związanych z wykonaniem rotacji (manipulacje wskaźnikami, porównania,...).

- (b) Niech  $y$  i  $z$  oznaczają odpowiednio ojca i dziadka  $x$ -a.



Pokażemy, że  $rotate(y); rotate(x)$  oraz utrzymanie niezmiennika kosztują nie więcej niż  $3(\mu'(x) - \mu(x))$ . Ponieważ jedynie poddrzewa zakorzenione w  $x$ ,  $y$  i  $z$  mogły zmienić swoją wielkość, aby utrzymać niezmiennik potrzebujemy

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

jednostek. Ponieważ po wykonaniu tych rotacji w poddrzewie o korzeniu  $x$  znajdują się dokładnie te same wierzchołki, które przed wykonaniem rotacji znajdowały się w poddrzewie o korzeniu  $z$ , mamy  $\mu'(x) = \mu(z)$ . Stąd

$$\begin{aligned} (*) &= \mu'(y) + \mu'(z) - \mu(x) - \mu(y) = [\mu'(y) - \mu(x)] + [\mu'(z) - \mu(y)] \leq \\ &\leq [\mu'(x) - \mu(x)] + [\mu'(x) - \mu(y)] \leq 2[\mu'(x) - \mu(x)]. \end{aligned}$$

Mając  $3[\mu'(x) - \mu(x)]$  jednostek do dyspozycji, na opłacenie operacji niskiego poziomu wykonywanych przy tych dwóch rotacjach pozostaje nam  $\mu'(x) - \mu(x)$  jednostek. Może się jednak okazać, że  $\mu'(x) = \mu(x)$ . Wówczas operacje niskiego rzędu będziemy mogli opłacić uszczuplając depozyty, ponieważ (jak pokażemy w poniższym fakcie), równość  $\mu'(x) = \mu(x)$  implikuje, że suma nowych depozytów w  $x$ ,  $y$  i  $z$  jest mniejsza niż suma depozytów w tych wierzchołkach przed wykonaniem rotacji.

**Fakt 1** *Jeśli  $\mu'(x) = \mu(x)$ , to  $\mu'(x) + \mu'(y) + \mu'(z) - \mu(x) - \mu(y) - \mu(z) < 0$ .*

Dowód:

Założmy nie wprost, że

$$\mu'(x) = \mu(x) \text{ oraz } \mu'(x) + \mu'(y) + \mu'(z) \geq \mu(x) + \mu(y) + \mu(z).$$

Ponieważ  $\mu(x) \leq \mu(y) \leq \mu(z) = \mu'(x) = \mu(x)$ , więc

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

Stąd

$$\mu'(x) + \mu'(y) + \mu'(z) \geq 3\mu(z).$$

Z tego, że zarówno  $\mu'(y)$  jak i  $\mu(z)$  są nieujemne i nie większe od  $\mu'(x)$  wnioskujemy, że

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

i ostatecznie

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

Zauważmy teraz, że zbiór wierzchołków znajdujących się przed rotacjami w drzewie o korzeniu  $x$  jest rozłączny ze zbiorem wierzchołków znajdujących się po rotacjach w drzewie o korzeniu w  $z$ . Oznaczmy moce tych zbiorów przez  $a$  i  $b$ . Ponadto wszystkie te wierzchołki i jeszcze wierzchołek  $y$  znajdowały się przed rotacjami w drzewie o korzeniu w  $z$ . Stąd, wobec powyższej równości, mamy

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

Ale  $\lfloor \log(a + b + 1) \rfloor \geq \lfloor \log(2 \min\{a, b\}) \rfloor > \lfloor \log(\min\{a, b\}) \rfloor$ , więc otrzymujemy sprzeczność.

□ (Fakt 1)

(c) Analiza tego przypadku jest analogiczna jak analiza przypadku (b).

W trakcie operacji  $Splay(x, S)$  wierzchołek  $x$  zajmuje coraz wyższe pozycje. Niech  $S_1, S_2, \dots, S_k$  będą drzewami zakorzenionymi w  $x$  w momencie gdy  $x$  zajmuje te pozycje. Wówczas całkowity koszt  $Splay(x, S)$  wynosi

$$\begin{aligned} &3(\mu(S_1) - \mu(x)) + 3(\mu(S_2) - \mu(S_1)) + \dots + 3(\mu(S_k) - \mu(S_{k-1})) + 1 = \\ &3(\mu(S_k) - \mu(x)) + 1 = 3(\mu(S) - \mu(x)) + 1 \end{aligned}$$

□ (Lemat 1)

## Literatura

- [1] D.Sleator, R.E.Tarjan, *Self-adjusting binary trees*, JACM, 32(1985), s. 652-686.
- [2] R.E.Tarjan, *Data Structures and Network Algorithms*, SIAM, 1983.