Notatki z AiSD. Nr 12. 11 kwietnia 2022

Drzewa zbalansowane:
Drzewa AVL

IIUWr. II rok informatyki. Przygotował: Krzysztof Loryś

1 Drzewa zbalansowane - wstęp

Najpoważniejszą wadą binarnych drzew przeszukiwań jest brak zabezpieczenia przed nierównomiernym rozrastaniem się, przez co pesymistyczny czas wykonywania operacji na nich może być liniowy względem liczby wierzchołków. Proste sposoby zaradzenia temu zjawisku mogą być niepraktyczne. Idealnym rozwiązaniem byłoby utrzymywanie drzew w stanie dokładnego zbalansowania, tj. tak, by wszystkie liście leżały na co najwyżej dwóch poziomach. Niestety przywracanie struktury takiego drzewa po zaburzeniu jej operacjami insert czy delete jest niezwykle kosztowne.

Innym, niestety także zbyt kosztownym, rozwiązaniem byłoby pamiętanie sumy długości wszystkich ścieżek od korzenia do wierzchołków drzewa i przeorganizowywanie drzewa dopiero gdy suma ta przekroczy jakąś graniczną wartość (np. $2n \log n$).

Znanych jest wiele różnych odmian drzew zbalansowanych. My poznamy drzewa AVL, drzewa czerwono-czarne, B-drzewa, drzewa samoorganizujące się (drzewa splay) oraz drzewce (ang. treaps).

2 Definicja

Definicja 1 *Binarne drzewo przeszukiwań jest* drzewem AVL*, jeśli dla każdego wierzchołka wysokości jego lewego i prawego poddrzewa różnią się o co najwyżej* 1.

Uwaga: Skrót AVL pochodzi od pierwszych liter nazwisk autorów (Adelson-Velskij i Landis).

3 Zasadnicza cecha

Twierdzenie 1 *Wysokość drzewa AVL o n wierzchołkach jest mniejsza niż* $1.4405 \log(n+2)$.

Fakt 1 *Liczba wierzchołków w dowolnym drzewie binarnym jest o* 1 *mniejsza od liczby pustych wskaźników (tj. równych* `NIL`*).*

Dowód (Twierdzenia 1)
Niech

$\rho(i) =$ " liczba pustych wskaźników w minimalnym (tj. o najmniejszej liczbie wierzchołków) drzewie AVL o wysokości i. "

Indukcyjnie po wysokości h drzewa dowodzimy, że $\rho(h) = (h+2)$-a liczba Fibonacciego:

- Łatwo sprawdzić, że $\rho(1) = 2$ i $\rho(2) = 3$.
- (Dla $h \geq 3$)

  Niech T będzie minimalnym drzewem AVL o wysokości h ($h \geq 3$). Z minimalności T wiemy, że jedno z poddrzew podwieszonych pod jego korzeniem musi być minimalnym drzewem AVL o wysokości $h-1$, a drugie - minimalnym drzewem AVL o wysokości $h-2$. Ponieważ każdy pusty wskaźnik T jest pustym wskaźnikiem w jednym z tych poddrzew, otrzymujemy wzór

$$\rho(h) = \rho(h-1) + \rho(h-2)$$

Teraz niech n będzie liczbą wierzchołków w T. Z Faktu 1 i powyższych rozważań mamy

$$n + 1 > \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^{h+2} - 1,$$

co po prostych przeksztaceniach daje tezę.

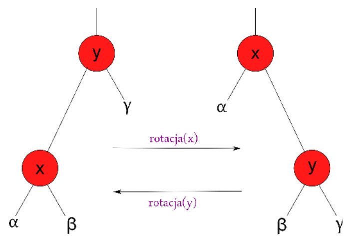
$\square$

4 Operacje słownikowe na drzewach AVL

Wyszukiwanie elementu wykonuje się identycznie jak dla zwykłych binarnych drzew przeszukiwań. Pozostałe dwie operacje mogą zaburzyć strukturę drzewa AVL. Przywracanie tej struktury nazywamy *balansowaniem drzewa.*

4.1 Rotacje

Podczas balansowania korzystać będziemy z wewnętrznych procedur dokonujących rotacji wierzchołków drzewa.



WAŻNE WŁASNOŚCI:

1. Rotacje nie zmieniają porządku infiksowego (czyli porządku BST) elementów zapamiętanych w drzewie.
2. Pojedynczą rotację można wykonać w czasie stałym.

4.2 Wstawianie elementu

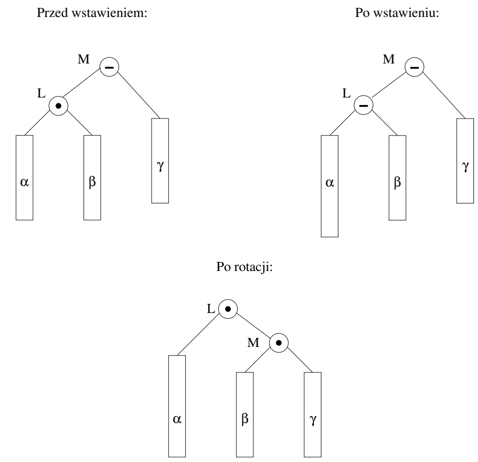
Niech M będzie pierwszym węzłem na drodze od wstawionego elementu do korzenia, w którym nastąpiło naruszenie równowagi drzewa AVL. Oznacza to, że przed operacją wstawienia poddrzewa zakorzenione w M były nierównej wysokości i wstawienie zwiększyło wysokość wyższego poddrzewa. Załóżmy, że tym poddrzewem jest lewe podrzewo i oznaczmy jego korzeń przez L (sytuacja, w której wyższym poddrzewem jest prawe poddrzewo jest symetryczna).

Procedura balansowania musi oddzielnie rozpatrywać dwa przypadki:

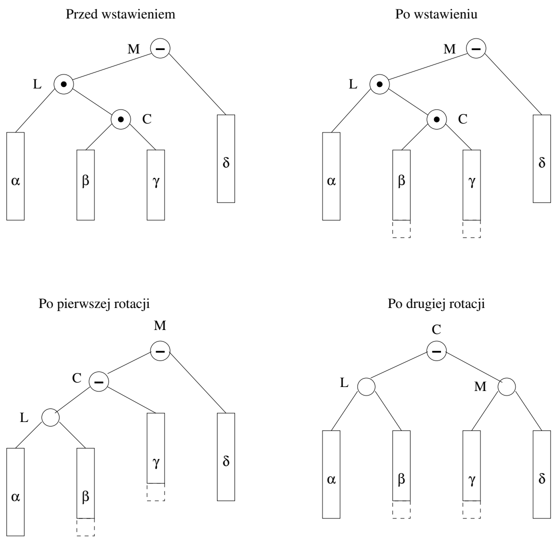
(A) w drzewie o korzeniu L zwiększyła się wysokość lewego poddrzewa (Rysunek 1),

(B) w drzewie o korzeniu L zwiększyła się wysokość prawego poddrzewa (Rysunek 2).

Uwaga: Po zbalansowaniu wysokość drzewa zakorzenionego w C jest równa wysokości drzewa zakorzenionego w M przed operacją wstawienia. Dlatego nie ma potrzeby przywracania zrównoważenia w innych węzłach poza M.



Rysunek 1: Przypadek (A)



Rysunek 2: Przypadek (B)

4.3 Usuwanie elementu

Operacja ta jest znacznie bardziej skomplikowana.

IDEA:

Algorytm *DeleteAVLnode*

1. Znaleźć wierzchołek zawierający element g, który chcemy usunąć.
2. Jeśli jest to wierzchołek wewnętrzny to wstawić do niego element g' z drzewa bezpośrednio następny (bądź bezpośrednio poprzedni) po g.
3. Jeśli g' był w liściu, to przejdź do kroku 5.
4. Do wierzchołka, który zawierał g' wstaw element pamiętany w jego synu (jest on liściem).
5. Usunąć ten liść. Przejść drogę od tego liścia do korzenia przywracając zrównoważenie wierzchołków na tej drodze przy pomocy rotacji.

UWAGI:

1. Tym razem może się zdarzyć, że trzeba będzie dokonywać rotacji dla wszystkich wierzchołków na tej drodze.

2. Szczegółowy opis drzew AVL można znaleźć w ksiażce [1].

4.4 Koszt

Wszystkie operacje słownikowe na drzewach AVL można wykonać w czasie ograniczonym funkcją liniową od wysokości drzewa, a więc w czasie $O(\log n)$.

5 Zastosowanie drzew AVL do implementacji list

Typowymi operacjami na listach są m.in.:

1. wstawianie elementu na wskazaną pozycję,

2. usuwanie elementu ze wskazanej pozycji,

3. konkatenacja list,

4. podział listy na dwie podlisty wg zadanej pozycji.

Przy tradycyjnych implementacjach list (tj. w tablicach lub przy pomocy zmiennych wskaźnikowych) niektóre z tych operacji wymagają czasu liniowego. Drzewa AVL pozwalają na implementację list, która umożliwia wykonanie powyższych operacji w czasie $O(\log n)$. Wystarczy w każdym wierzchołku pamiętać liczbę elementów w jego lewym poddrzewie (liczba ta wyznacza pozycję elementu w liście przechowywanej w drzewie zakorzenionym w tym wierzchołku).

Szczegóły pozostawiamy jako temat do samodzielnych studiów.

Literatura

[1] N.Wirth, *Algorytmy + Struktury Danych = Programy.*