Drzewa AVL

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).

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 przekształceniach daje tezę.

Ważne własności:

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

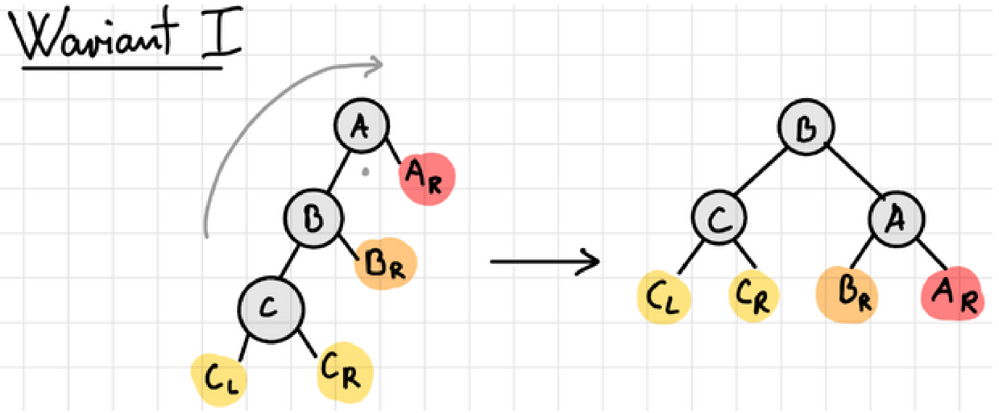
4.2 Wstawianie elementu

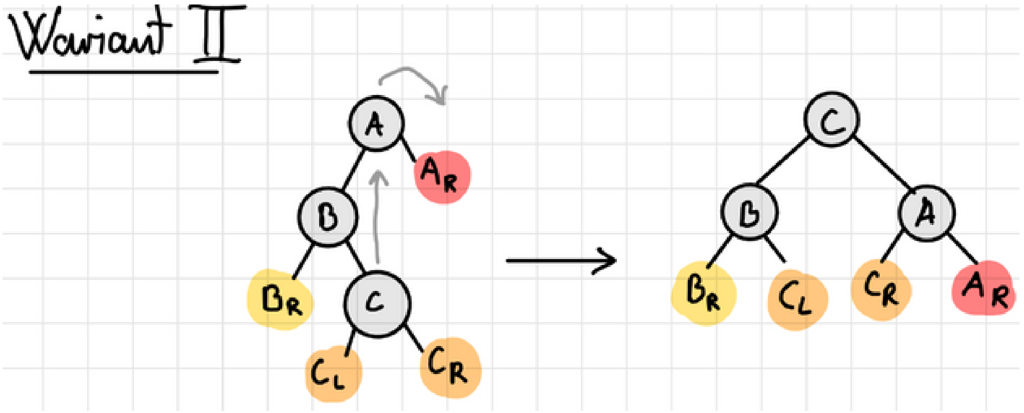
Niech M będzie pierwszym węzłem na drodze od wstawionego elementu do korzenia, w którym nastąpiło naruszenie równowagi drzewa AVL. Oznacza to, że przed operacją wstawienia poddrzewa zakorzenione w M były nierównej wysokości i wstawienie zwiększyło wysokość wyższego poddrzewa. Załóżmy, że tym poddrzewem 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).





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 $O(\log n)$.

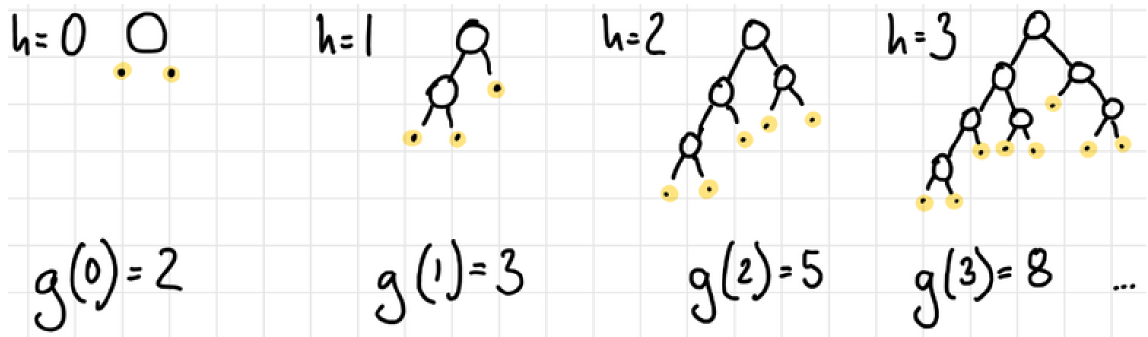
5 Zastosowanie drzew zbalansowanych do implementacji list

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 zbalansowane pozwalają na implementację list, która umożliwia wykonanie powyższych operacji w czasie $O(\log n)$. Porządek infiksowy w drzewie odpowiada porządkowi elementów w liście a rotacje wykonywane dla utrzymania zbalansowania nie zmieniają tego porządku. Aby znaleźć wskazaną pozycję w liście, wystarczy w każdym wierzchołku pamiętać liczbę elementów w jego poddrzewie.

Szczegóły pozostawiamy jako temat do samodzielnych studiów i jako temat zadań na ćwiczenia.



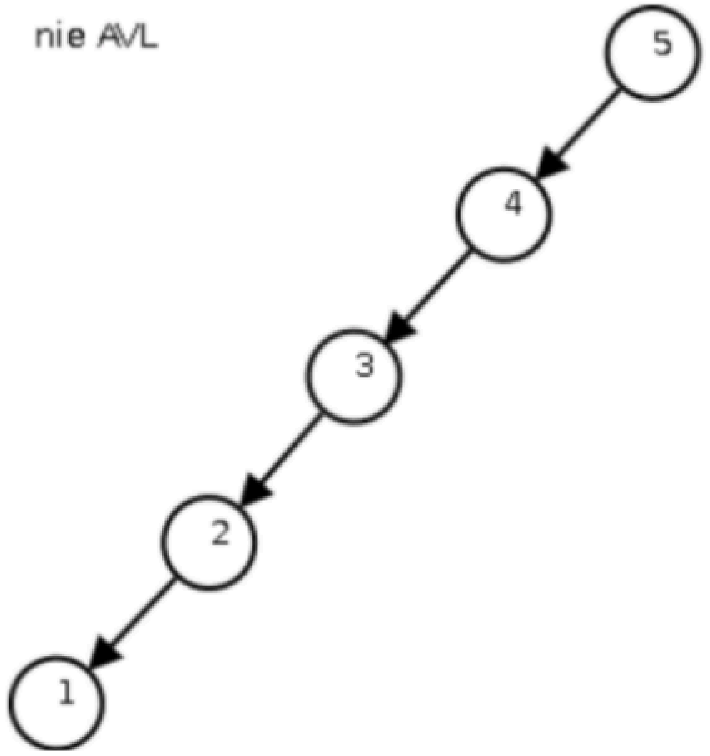
ilość pustych wskaźników w minimalnych drzewa AVL dla każdej wysokości tworzą przesunięty ciąg fibonacciego

Narysuj drzewo binarnych wyszukiwań pamiętające klucze 1,2,3,4,5, które

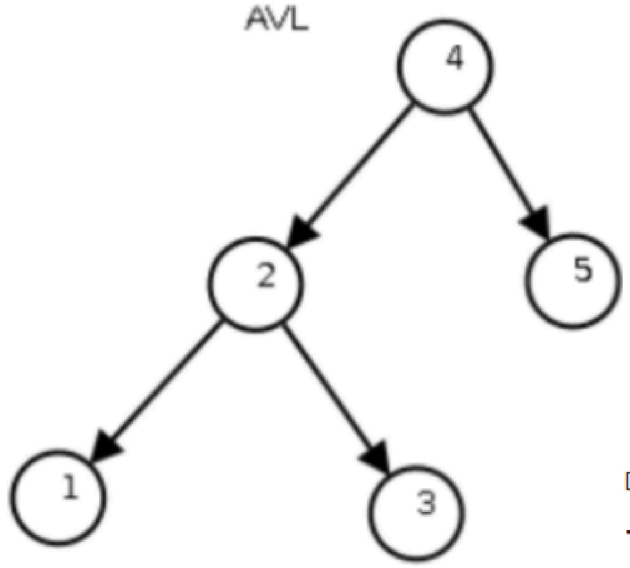
a) jest drzewem AVL

b) nie jest drzewem AVL

nie AVL



AVL



Treść

Podaj maksymalną liczbę rotacji przy usuwaniu klucza z drzewa AVL

▼ Rozwiązanie

W przypadku usuwania klucza z drzewa AVL może się wydarzyć sytuacja, w której wskaźniki balansu będą zepsute aż do korzenia.

Zatem maksymalna liczba rotacji to 1.4405*log(n).

Treść

Rozważ modyfikację drzew AVL, w której wysokość lewego i prawego poddrzewa zakorzenionych w tym samym wierzchołku mogą się różnić o nie więcej niż 2. Udowodnij, że każde takie drzewo z n wierzchołkami ma wysokość Theta(log n)

Treść

Jak mocno można ograniczyć (w pesymistycznym przypadku) liczbę rotacji podczas usuwania wierzchołka drzewa AVL o n wierzchołkach?

Uzasadnij, że nie da się bardziej niż podałeś(aś).

Treść

Niech v będzie wierzchołkiem w drzewie binarnym a Diff(v) niech oznacza wartość bezwzględną róznicy liczby wierzchołków w jego lewym i prawym poddrzewie.

Czy w drzewie AVL o n wierzchołkach największa wartoŚć Diff(v) może być Ω(n)?

Odpowiedź uzasadnij.

Treść

Jaką największą wysokość może mieć drzewo AVL zawierające 67 kluczy? Odpowiedź uzasadnij.

▼ Rozwiązanie

Fakt

W każdym drzewie binarnym z `n` wierzchołkami mamy `n+1` pustych wskaźników.

Twierdzenie

Dla:

`d(h)` - oznacza ilość pustych wskaźników dla minimalnego drzewa AVL o wysokości `h`

`F(i)` - i-ty element ciągu fibonacciego

Zachodzi `d(h) = F(h+2)`

Wtedy możemy zauważyć, że `67 + 1 = 68 >= d(h)`, z powyższego faktu, gdzie `h` to szukana wysokość.

To co musimy znaleźć to największe takie `h` dla którego `F(h+2) <= 68`

Dla pewności powinno się też dodać dowód d(h) = F(h+2), który był na wykładzie

5. (2pkt) Niech $h(v)$ oznacza odległość wierzchołka v do najbliższego pustego wskaźnika w poddrzewie o korzeniu v. Rozważ możliwość wykorzystania drzew binarnych, równoważonych poprzez utrzymywanie następującego warunku:

$$h(\text{lewy syn } v) \geqslant h(\text{prawy syn } v) \text{ dla każdego wierzchołka } v,$$

do implementacji złączalnych kolejek priorytetowych.

6. (0,5pkt) Udowodnij, że każde drzewo BST można przekształcić operacjami rotacji w dowolne inne drzewo BST .

7. (2pkt) Napisz procedurę *Split*(T, k) rozdzielającą drzewo AVL T na dwa drzewa AVL: jedno zawierające klucze mniejsze od k i drugie zawierające pozostałe klucze. Jaka jest złożoność Twojej procedury?

8. (1,5pkt) Zaproponuj strukturę danych do pamiętania zbioru liczbowego i wykonywania na nim operacji: *insert*, *delete*, *mindiff*. Ostatnia z tych operacji zwraca jako wynik najmniejszą różnicę między dwoma elementami zbioru.

9. (2pkt) Zaproponuj strukturę danych do pamiętania ciągu liczbowego i wykonywania na nim operacji:

- *insert*(i, a) - wstaw liczbę a na pozycję i w ciągu;
- *delete*(i) - usuń element znajdujący się na pozycji i;
- *find*(i) - podaj wartość, znajsującą się na pozycji i;
- *sum-of-even*$()$ -podaj sumę elementów znajdujących się na pozycjach parzystych.

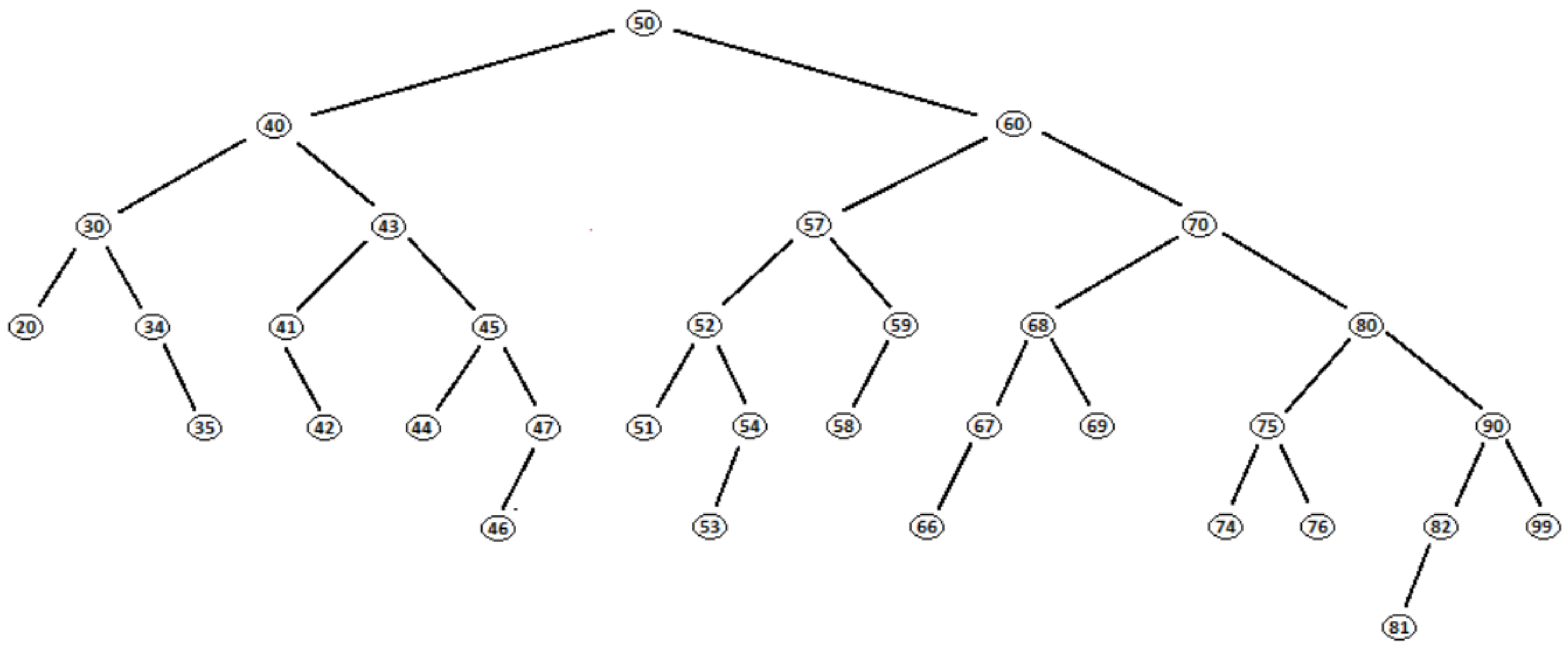
Operacja *insert*(i, a) przeształca ciąg $x_1, x_2, \ldots, x_n$ w ciąg $x_1, x_2, \ldots, x_{i-1}, a, x_i, x_{i+1}, \ldots, x_n$, a operacja *delete*$(i)$ przekształca ten ciąg w ciąg $x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$.

.0. (1.5pkt) Bolesną dolegliwością związaną z drzewami AVL jest konieczność poświęcenia dwóch bitów w każdym węźle na pamiętanie współczynnika zrównoważenia. Zastanów się, czy aby na pewno mamy do czynienia z "koniecznością".

Treść

Czy usunięcie wierzchołka z drzewa AVL może wymagać 3 rotacji dla przywrócenia balansu? Narysuj przykład albo napisz uzasadnienie dlaczego nie.

▼ Rozwiązanie



Po usunięciu 20.