Sprawozdanie drugie

Złożone struktury danych

Piotr Tylczyński

29.03.2019

1 Opisy

1.1 Lista

Lista jest abstrakcyjnym typem danych. Dostęp do elemntów listy jest sekwencyjny, co oznacza, że odczytanie n. elemntu z listy, będzie wymagało odczytania $n-1$ pierwszych elemntów listy, co niekorzystnie wpływa na złożoność wielu algorytmów operujących na listach.

1.2 Drzewo AVL

Jego nazwa pochodzi od dwóch nazisk Adelsona-Velsky'ego i Landia, którzy byli jego twórcami. Jest to abstrakcyjna struktura danych, która jest samo balansującym się drzewem binarnym. Oznacza to, że jeśli która kolwiek z podgałęzi dowlonego węzła takiej struktury jest dłóższa od drugiej, nastepuje balansowanie drzewa. Taka właściwość pozytywnie wpływa na złożoność obliczeniową operacji przeszukiwania wspomnianej struktury.

2 Złożoność czasowa

2.1 Szukanie elementu

2.1.1 Lista

Algorytm szuukania elementu w liście szukając danego elementu musi przejżeć wszystkie elementu znajdujące się w liście aż do momentu kiedy nie trafi na szukany element. Powoduje to, że złożoność takiego algorytmu opisujemy przez:

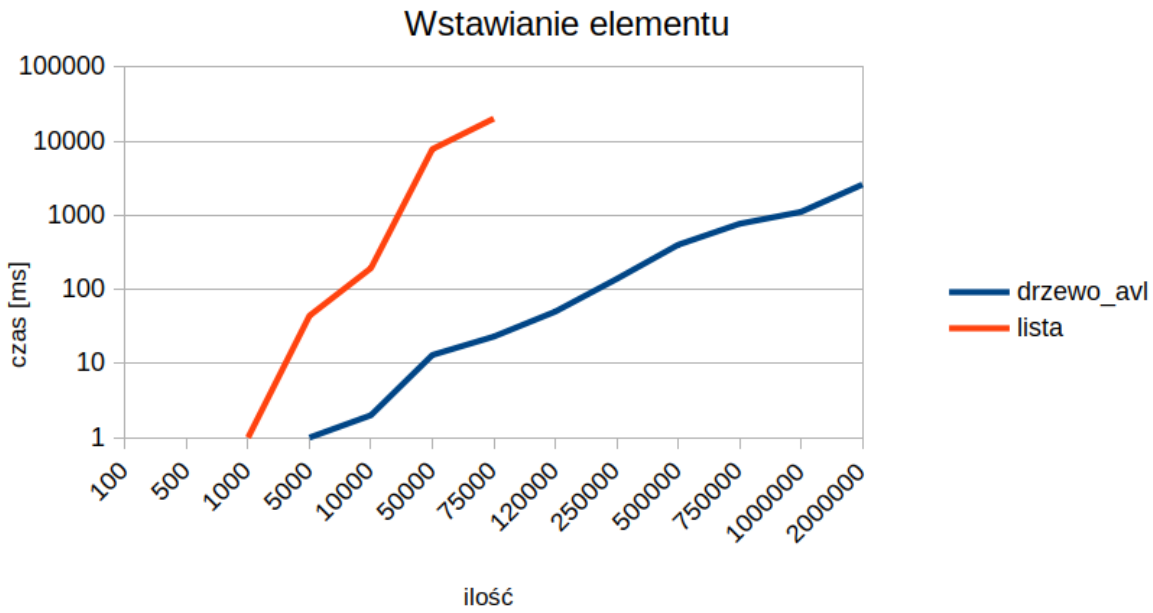
$$O(n) = n$$

2.1.2 Drzewo AVL

Jako, że drzewo AVL jest drzewem binarnym, to lewe dziecko dowolnego węzła w strukturze jset zawsze mnijsze od rodzica, a prawe większe. Taka zależność pozwala na każdorazowe odrzucenie połowy pozostałego do przeszukania zbioru, ponieważ wiadomo, że szukany element tam nie wystąpi. W związku z tym złożoność takiej operacji można określić jako:

$$O(n) = \log_2 n$$

2.2 Dodawanie n elementów w sposób posortowany



2.2.1 Lista

Jak łatwo zauważyć czas wykonania rośnie bardzo szybko co jest spowodowane potrzebą przejżenia $n-1$ elemntów przed możliwością wstawienia nowego elementu n. Dlatego w najgorszym przypadku - kiedy wartości elemntów są ułożone w kolejności odwrotnej do typu posortowania - algorytm wstawiania za każdym razem będzie musiał wyonać $n-1$ sprawdzeń, gdzie n jest aktualną liczbą elemntów. Ogólnie złożonośc można przedstawić jako:

$$\begin{aligned} \text{ilość operacji} &= (n-1)+(n-2)+(n-3)+...+(n-n) \\ &= n^2+... \end{aligned} \quad (1)$$

co daje złożoność:

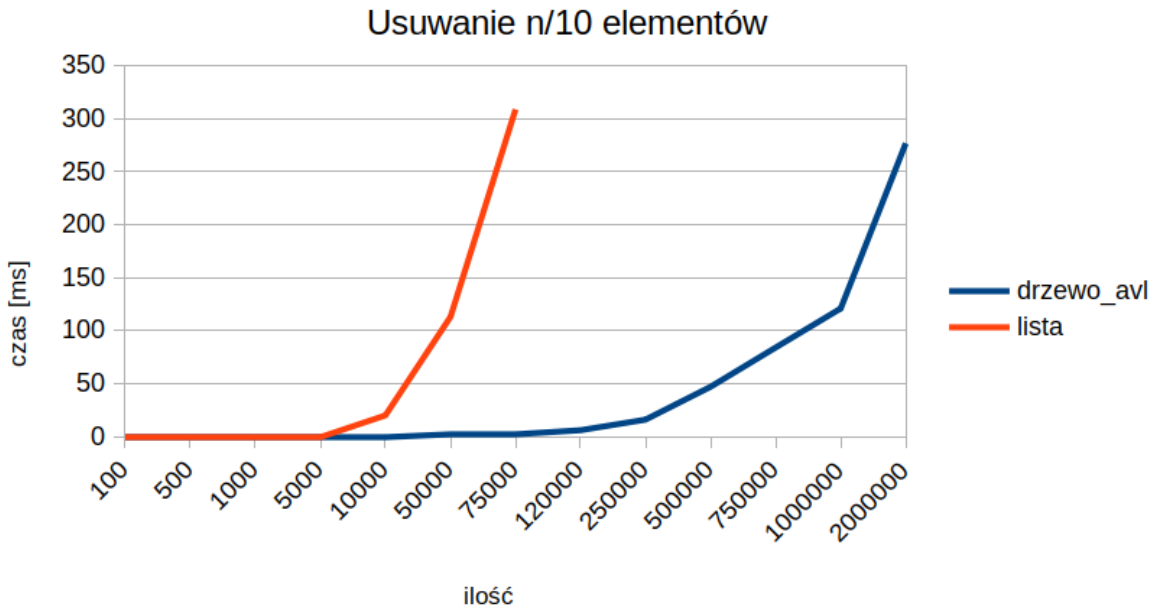
$$O(n) = n^2$$

2.2.2 Drzewo AVl

Ze względu na to, że drzewo AVL jest drzewam binarnym, jego wysokośc można opisać za pomocą logarytmu o podstawie 2. Oznacza to, że dodanie nowego elementu nie zajmie więcej operacji niż wysokość takiego drzewa. Dodatkowo drzewo po dodaniu elemntu należy zbalansować. Balansowanie zostaje zakończone w momencie znalezienia pierwsego nie zbalansowanego węzła. Złożoność wstawiania nowego elemntu opisujemy jako:

$$\begin{aligned} \text{ilość operacji} &= k*(\text{szukanie miejsca}+\text{równoważenie}) \\ \text{ilość operacji} &= \text{k*}(\log_2 n+\log_2 n) \\ \text{O(n)} &= \text{k}\log_2 n \end{aligned} \quad (2)$$

2.3 Usuwanie losowych k elementów



2.3.1 Lista

Dostęp do h elemntu wymaga sprawdzenia $h-1$ elementów. Wynika z tego, że złożoność wynosi

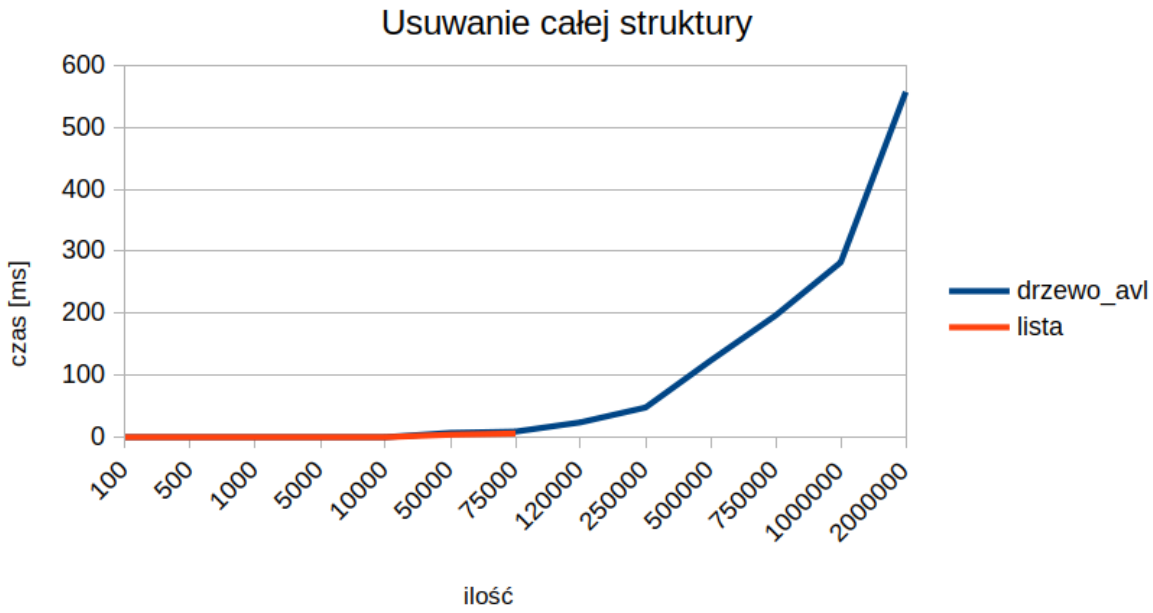
$$\begin{gathered} \text{ilość operacji} = k * \text{znalezienie elementu} \\ \mathrm{O(n)} = \mathrm{kn} \end{gathered} \tag{3}$$

2.3.2 Drzewo AVL

Polega na znalezieniu usuwanego elementu w drzewie i wykonania algorytmu usuwania, w wyniku którego może zostać uruchomiony algorytm balansowania. Algorytm balansowania, nie kończy się w momencie znalezienia pierwszego niezbalansowanego węzła, jak w przypdaku dodawania elementu, lecz trwa do momentu kiedy całe drzewo nie zostanie ponowniezbalansowane.

$$\begin{gathered} \text{ilość operacji} = n * (\text{szukanie elementu} + \text{równoważenie}) \\ \text{ilość operacji} = \mathrm{n} * (\log_2 n + \log_2 n) \\ \mathrm{O(n)} = \mathrm{n}\log_2 n \end{gathered} \tag{4}$$

2.4 Usuwanie całej struktury



2.4.1 Lista

Polega ono na rekurencyjny przejściu przez całą strukturę i usunięciu każdego elemntu. Przejście przez każdy element struktury zajmie dokłanie n operacji, a to daje złożoność:

$$O(n) = n$$

2.4.2 Drzewo AVL

Algorytm usuwania całej struktury polega na przejżeniu wszystkich jej węzłów i ich usunięcie. Złożonośc tej operacj to:

$$O(n) = n$$

3 Zastosowania

3.1 Lista

- Listy jako dynamiczne struktury danych mogą być stosowane do przechowywania danych, których ilość nie jest znana podczas uruchamiania programu.
- Lista zajmuje tylko tyle miejsca ile jest wymagane do przechowania danych, kosztem czasu dostępu do poszczególnych elemtów, dlatego można ją wykorzystać do przechowywania duzych ilośći danych, do których czas dostępu nie jest elemntem kluczowym
- Lista pozwala na przechowanie elementów o takich samych wartościach, co daje możliwośc tworzenia spisów posiadanych elemntów, których rozróżnianie nie jest potrzebne, np. policzenie mediany z ocen, bez wiedzy ile tych ocen faktycznie jest

3.2 Drzewo AVL

- Ze względu na to, że drzewo AVL jest strukturą dynamiczną, nie potrzeba określać jego wielkości/pojemności przy jego deklaracji. Pozwala to na przechowywanie w nim danych, których ilość nie jest znana przy uruchamianiu programu
- Opisywana struktura ze względu na swoją dynamiczność, zajmuje w pamięci tylko tyle miejsca ile jest potzebne na przechowanie danych, dlatego świetnie nadaje się do przechowywania dużych ilości informacji
- Binarna budowa drzewa pozwala na szybkie szukanie oraz stwierdzanie instnienia elementu o danym indeksie
- Możliwość przechowywania zmiennej ilości elementów i krótki czas dostępu, sprawia że drzewo AVL jest idealną strukturą do wykorzystania jako baza danych

Contents