AiSD — zadanie drugie

Gliwiński Jarosław Marek
Kruczyński Konrad Marek
Grupa dziekańska I5

10 kwietnia 2008

1 Wstęp

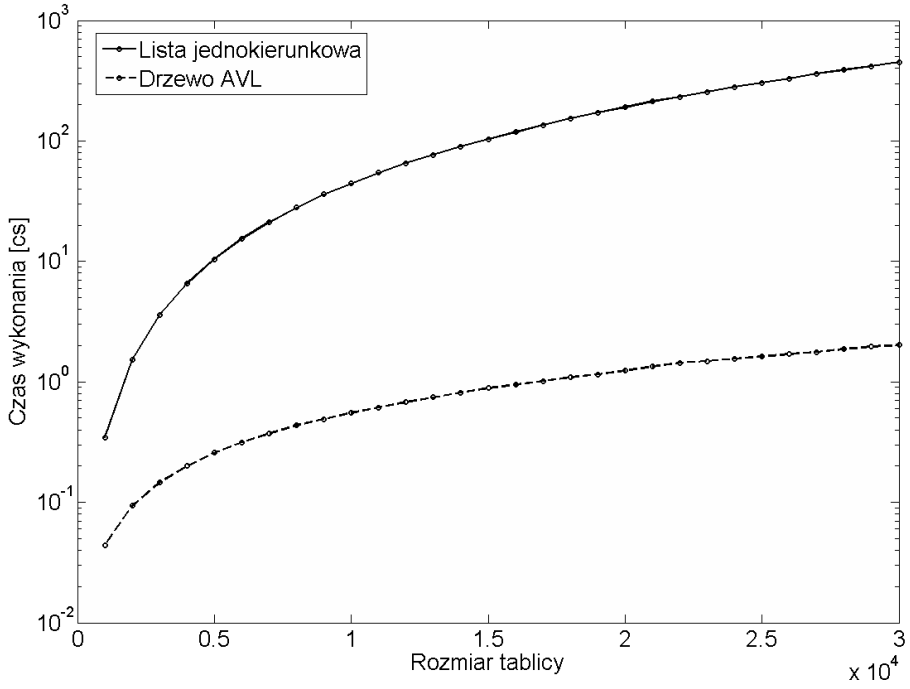
W nowym zadaniu porównywano efektywność kilku operacji na dwóch różnie zorganizowanych strukturach danych, jakimi były drzewo AVL oraz lista uporządkowana. Algorytmy wykonujące operacje oczywiście wykorzystywały właściwości tych struktur danych (np. uporządkowanie) w celu maksymalizacji wspomnianej efektywności.

Porównywanymi operacjami były najczęściej wykonywane na strukturach danych operacje, tj. wstawianie, usuwanie, wyszukiwanie elementu zawierającego żądany klucz oraz dodatkowo operację przetwarzania każdego elementu (symulowaną poprzez zwiększenie klucza o 1). Operacje wykonywane były w tych samych warunkach pomiarowych i w tej samej kolejności dla każdej ze struktur. To ostatnie było konieczne ze względu na fakt, iż niektóre operacje (wstawianie, usuwanie) modyfikowały rozmiar struktury, tak więc gdyby ustawić je w w różnej kolejności dla drzewa i listy, względny stosunek czasów dla poszczególnych operacji mógłby zostać zafałszowany.

2 Pomiary

2.1 Tworzenie struktury

Dla obu struktur zastosowano oczywiście tę samą procedurę. Była ona następująca: na początku tworzony był ciąg liczb o długości n, gdzie n oznacza rozmiar tablicy (odzwierciedlony później na wykresie). Był to ciąg rosnący, na którego pierwszej pozycji stała jedynka, natomiast na k-tej znajdowała się liczba większa o 1 lub 2 (wybór dokonywany był w sposób losowy) od liczby znajdującej się na pozycji $k-1$. Na koniec ciąg permutowano w sposób losowy. Tak przygotowane dane umieszczane były na liście oraz drzewie. Pomiary przeprowadzono dla wartości n od 1000 do 30000 z krokiem 1000. Wyniki prezentują się następująco:



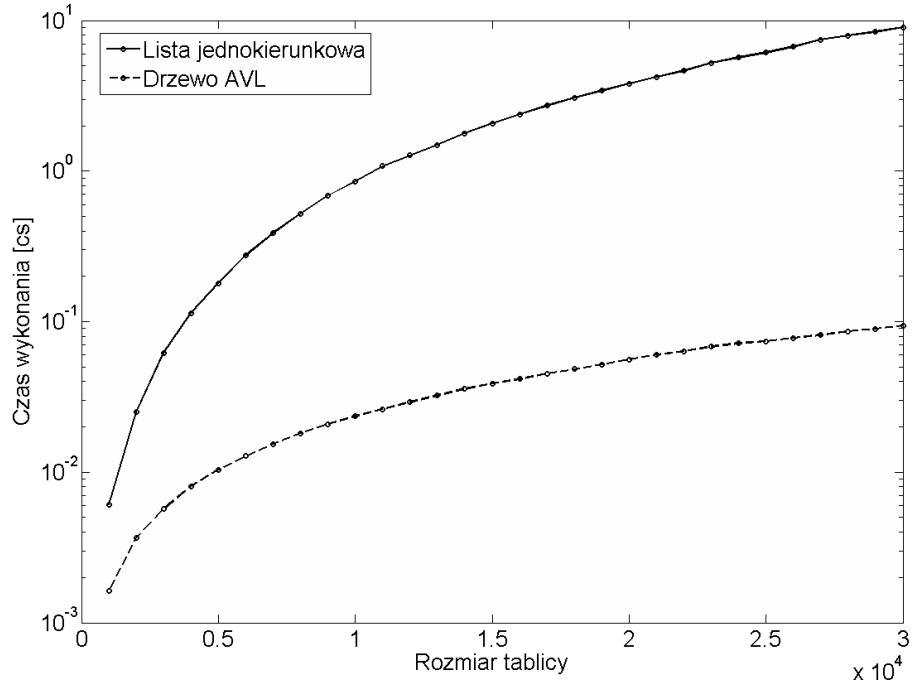
Odchylenia wyników w próbie przedstawiają się:

Dane	Odchylenie względne [%]
Lista jednokierunkowa	$0,9 \pm 0.06$
Drzewo AVL	$2,07 \pm 0,18$

Łatwo zaobserwować znaczną przewagę drzewa AVL nad listą. Ponieważ pesymistyczna złożoność poszukiwania właściwego miejsca do wstawienia na liście jest $O(k)$, gdzie k oznacza rozmiar listy, zaś samych operacji wstawienia mamy n, to zakładając dostawianie zawsze na końcu listy (pesymistyczny przypadek) otrzymamy złożoność $O(n^2)$. Dla drzewa natomiast pesymistyczna złożoność wstawienia elementu wynosi $O(log_2 k)$ i podobnie mamy n operacji wstawienia, stąd ostatecznie pesymistyczna złożoność wynosi $O(nlog_2 n)$. Z kolei złożoności przypadku oczekiwanego są takie same, ponieważ większa ich szybkość zachowywana jest w stałej. Teoria ma swoje odzwierciedlenie na wykresie.

2.2 Wyszukiwanie

Procedura była następująca. Na początku generowano ciąg $\frac{n}{10}$ liczb losowych z zakresu od 0 do $2n$ (mogły się powtarzać). Następnie wykonywano wyszukiwanie. Procedura wyszukiwania dla listy napisana została rekurencyjnie. Jak pokazały testy, wersja ta nie była wolniejsza od iteracyjnej, o ile tylko wszystkie parametry przekazywano przez referencję. Zastosowano optymalny algorytm wyszukiwania, tj. operacja zostaje przerwana, gdy osiągnięte elementy mają wartości większe niż poszukiwany. Oto wykres:



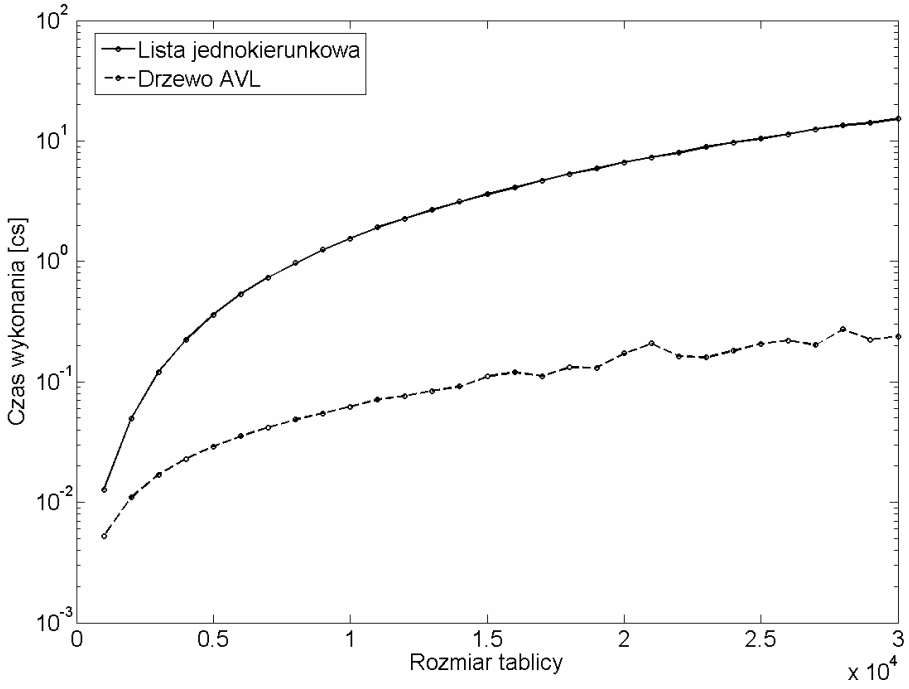
Odchylenia wyników w próbie przedstawiają się:

Dane	Odchylenie względne [%]
Lista jednokierunkowa	$2,6 \pm 0,33$
Drzewo AVL	$2,88 \pm 0,14$

Złożoności obu operacji są takie same jak poprzednio, ponieważ choć pracujemy na większym (w stosunku do ilości elementów wstawianych) zbiorze danych, jednak fakt ten ukrywa się w stałej.

2.3 Wstawianie

Procedura wstawiania była analogiczna do wyszukiwania, tj. umieszczano w strukturze $\frac{n}{10}$ elementów zorganizowanych tak jak poprzednio. Algorytm dla listy był podobny do wyszukiwania za wyjątkiem operacji wykonywanej na końcu. Dla drzewa również postępowano w ten sposób, za wyjątkiem tego, że po wstawieniu właściwym następowało wyważenie. Wykres przedstawia się następująco:



Odchylenia wyników w próbie przedstawiają się:

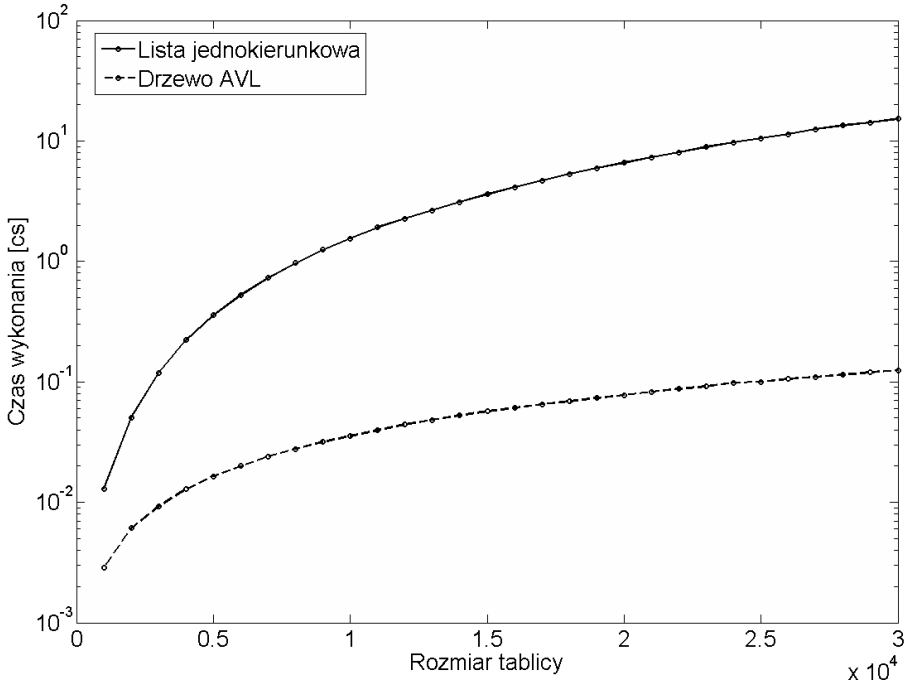
Dane	Odchylenie względne [%]
Lista jednokierunkowa	$1,96 \pm 0,22$
Drzewo AVL	$14,1 \pm 2,5$

Warto jednak nadmienić, że koszt operacji wyważania w tym przypadku jest stały, tj. dla pojedynczej operacji wstawiania może się wykonać co najwyżej jedna operacja równoważenia węzła.

2.4 Usuwanie

Ponownie mamy do czynienia z operacją podobną do poprzedniej – zarówno pod względem metodologii testu jak i algorytmicznym.

Wyważenie drzewa może być kosztowną operacją, stąd należy ją uwzględnić przy obliczaniu złożoności. Ta dla samej operacji wyważania w przypadku pesymistycznym wynosi $O(log_2 n)$, taka sama jest w przypadku oczekiwanym, stąd złożoność całej operacji n usunięć nie zmienia się, ponieważ do pierwotnej złożoności usuwania bez równoważenia $O(nlog_2 n)$ dodajemy kolejne $O(log_2 n)$ wynikające z operacji równoważenia, a więc w sumie wynosi ona nadal $O(nlog_2 n)$.

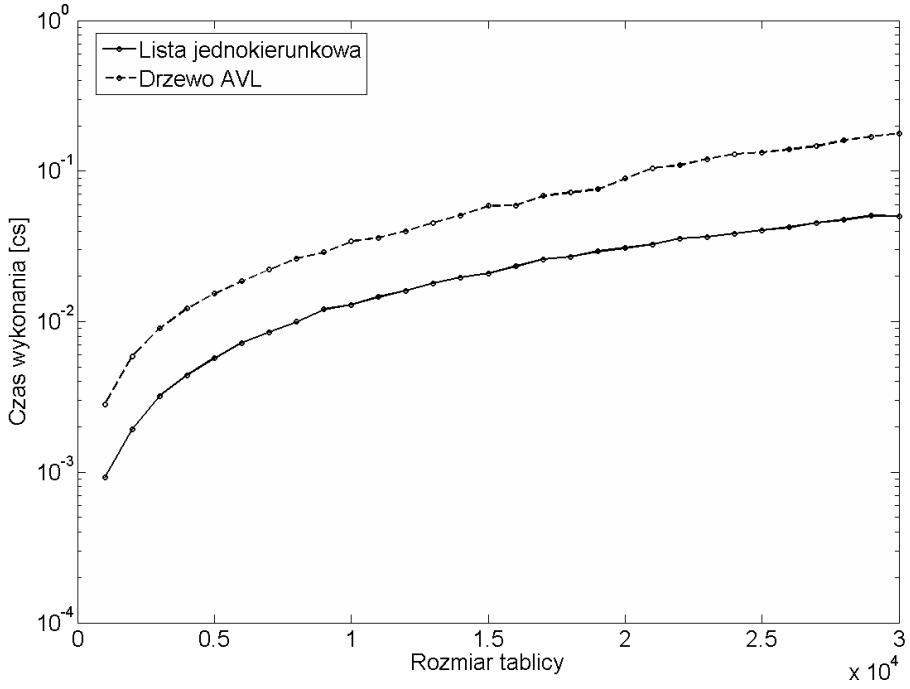


Odchylenia wyników w próbie przedstawiają się:

Dane	Odchylenie względne [%]
Lista jednokierunkowa	$2,34 \pm 0,21$
Drzewo AVL	$2,4 \pm 0,18$

2.5 Przeglądanie

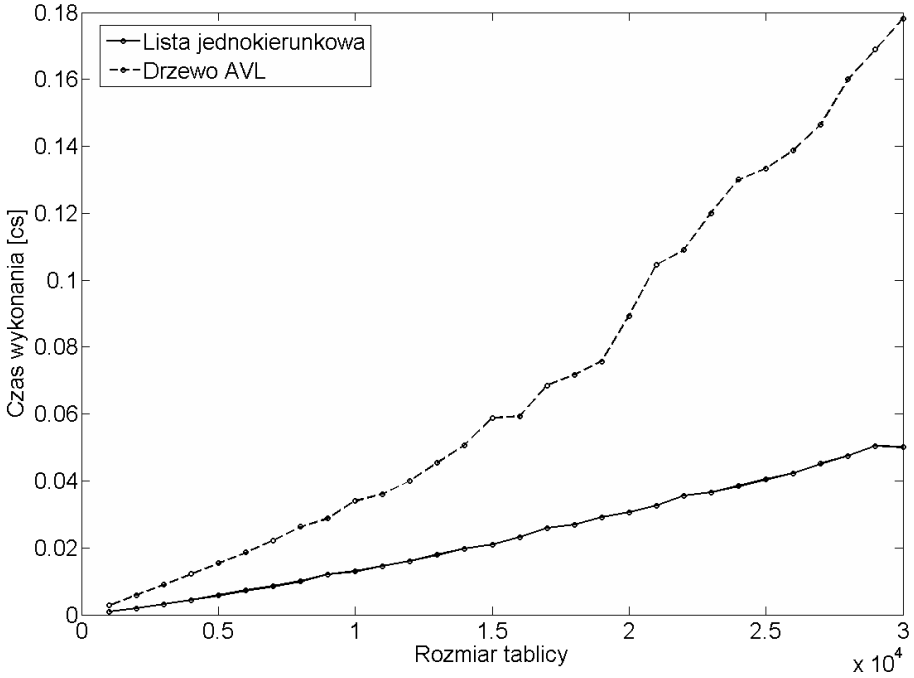
Jest to jedyny przypadek, w którym lista okazuje się szybsza od drzewa AVL, co widać na poniższym wykresie:



Odchylenia wyników w próbie przedstawiają się:

Dane	Odchylenie względne [%]
Lista jednokierunkowa	$5,2 \pm 0,7$
Drzewo AVL	$7,9 \pm 0,9$

Oczywiście złożoność operacji przeglądania jest istotnie odmiennym zagadnieniem niż dotąd analizowane. Na wstępie należy zaznaczyć, że podstawową różnicą jest jednokrotne wykonanie tej operacji (choć oczywiście na wszystkich elementach) w przeciwieństwie do wcześniejszych $\frac{n}{10}$ bądź n powtórzeń. Ponieważ na liście po prostu odwiedzamy każdy element począwszy od głowy aż do ogona i dla każdego wykonujemy tę samą operację, złożoność jest postaci $O(n)$. Doskonale obrazuje to drugi wykres, na którym miast skali logarytmicznej obraliśmy liniową – krzywa z dużą dokładnością przybliża prostą.



Złożoność operacji przeglądania wszystkich elementów w przypadku drzewa pozornie jest identyczna jak w przypadku listy, jednak pomimo wykonywania tej samej operacji na tej samej liczbie elementów czas wykonania jest widocznie dłuższy. Wynika to z różnicy w metodzie poruszania się pomiędzy kolejnymi elementami struktury. W przypadku listy jest to proste liniowe przechodzenie pomiędzy kolejnymi węzłami, podczas gdy w przypadku drzewa dla każdego węzła wywoływana jest dwukrotnie funkcja rekurencyjna przechodząca do lewego i prawego poddrzewa.

Różnica czasu przeglądania byłaby mniej widoczna, gdyby w czasie przetwarzania każdego węzła była wykonywana bardziej skomplikowana operacja. W takim przypadku złożoność całościowa przeglądania z przetwarzaniem w większej części zależałaby od operacji jednostkowej w każdym z węzłów, zaś mniejsze znaczenie miałaby użyta struktura danych.

3 Uwagi ogólne

Warto odpowiedzieć sobie na kilka zasadniczych pytań dotyczących drzew i list. Rozpocznijmy od analizy wysokości drzewa AVL. Miejmy na uwadze zastosowane kryteria równoważenia, tj. dozwoloną różnicę wysokości poddrzew równą 1. Dla ułatwienia rozważań rozpatrzymy na początek przypadek drzewa pełnego o wysokości k. Zawiera ono:

$$n = 1 + 2 + 2^2 + 2^3 + \ldots + 2^{k-1} = 2^k - 1$$

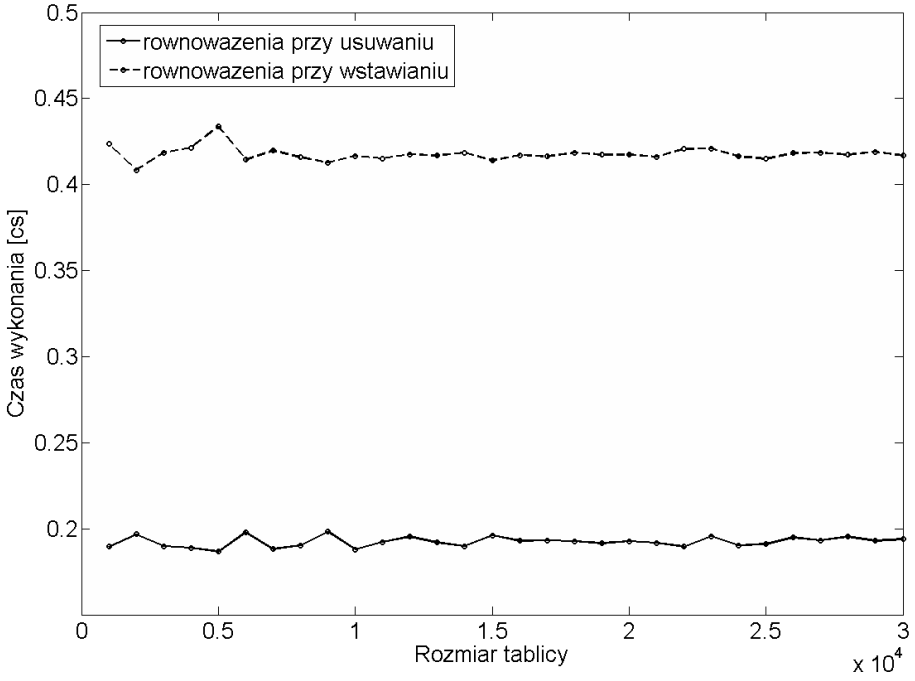
elementów. Daje to $k = \log_2(n+1)$. Teraz jednak należy mieć na uwadze, że jeżeli dołożyć do tego drzewa jeden element, to wysokość wzrośnie o 1. Zatem nasz wzór warto

uzupełnić o sufit. Ponadto bez naruszania kryterium równoważności możemy dokonać przestawienia jednego z liści, czyniąc zeń syna innego liścia. Wówczas rośnie wysokość, co sugeruje aby nasz wzór powiększyć o 1, co da nam górne ograniczenie na wysokość drzewa. Ostatecznie:

$$k = 1 + \lceil \log_2(n+1) \rceil$$

Przyszła pora na kilka słów nt. wyważania drzew. W ogólności samą procedurę stosujemy w celu utrzymania drzewa w dobrej kondycji, tj. zachowania minimalnej, bądź bliskiej temu minimum (w przypadku drzew AVL maksymalnie jeden poziom powyżej tego minimum), wysokości. Powoduje to, iż wszelkie operacje wykonywane na takim drzewie mają niewielką złożoność, ponieważ przy każdym węźle jesteśmy w stanie wyeliminować połowę zbioru danych z naszych poszukiwań, a maksymalna długość ścieżki wyszukiwania nie wykracza poza wyznaczoną wcześniej maksymalną wysokość drzewa.

Względną częstotliwość wyważania drzewa obrazuje poniższy wykres:



Odchylenia wyników w próbie przedstawiają się:

Dane	Odchylenie względne [%]
Równoważenia przy wstawianiu	$3,7 \pm 0,4$
Równoważenia przy usuwaniu	$6,4 \pm 0,7$

Jak widać na wykresie, stosunek operacji, w których konieczne było wyważanie do ogólnej liczby operacji jest stacjonarny względem wielkości struktury. W związku z tym

możliwe było określenie ogólnego współczynnika określającego ilość koniecznych równoważeń (przy czym równoważenie rozumiemy jako pojedynczy lub podwójny obrót równoważący pojedynczy węzeł). Uśrednione dla wszystkich operacji wykonanych w toku pomiarów współczynniki wyniosły odpowiednio:

Rodzaj operacji	względna ilość równoważeń [%]
Wstawianie	$41,7 \pm 2,5$
Usuwanie	$19,2 \pm 1,6$

Wyniki te nie odbiegają istotnie od podawanych w literaturze (odpowiednio $\sim 50\%$ oraz $\sim 20\%$).

Na koniec sprawozdania warto wspomnieć o zastosowaniach badanych struktur danych. Jeżeli wziąć pod uwagę wyszukiwanie elementu o zadanym kluczu, usuwanie, czy też wstawianie, już drzewo poszukiwań binarnych dysponuje zazwyczaj znaczną przewagą nad listą, ponieważ o ile na liście operacje te mają złożoność $O(n)$, o tyle na drzewie średnio $O(log_2 n)$. Jeżeli ponadto zastosujemy któryś z algorytmów automatycznego równoważenia drzewa taki jak drzewa czerwono-czarne czy właśnie wykorzystane AVL, również złożoność pesymistyczna będzie tego rzędu. Wynika to z wyeliminowania przypadku drzewa zdegenerowanego do listy (lub przypadków bliskim temu). Jednocześnie należy zauważyć, iż lista jest zdecydowanie prostsza w implementacji, a w związku z tym dla niewielkich struktur danych koszt administracyjny może być porównywalny z kosztem właściwych operacji. Wobec tego listy są bardziej odpowiednie dla niedużych zbiorów danych.

W szczególności drzewa sprawdzają się znakomicie w sytuacjach, kiedy najczęściej wykonywaną nań operacją jest wyszukiwanie, natomiast modyfikacja zdarza się rzadziej. Jak zaobserwowano w ostatnim etapie pomiarów, lista jest znacznie szybsza w przypadku operacji wymagających dostępu do wszystkich elementów. Kluczem do tej przewagi jest fakt, iż przeglądanie w żaden sposób nie wykorzystuje wyszukiwania konkretnego elementu będącego atutem drzewa.

Drzewa (w tym szczególne przypadki jak drzewa AVL) mają różnorakie zastosowania w informatyce w zadaniach wymagających wcześniej wspomnianych własności takich jak bazy danych, kompresja (kodowanie Huffmana), czy też systemy plików.

Listy posortowane wymagają nieco mniej pamięci (nie trzeba przechowywać współczynnika zrównoważenia), ponadto każdy węzeł zawiera tylko jeden wskaźnik. Ich prostota, a zarazem dosyć przyzwoita optymalizacja pozwala na zastosowanie w mniej wymagających wydajnościowo zbiorach danych o umiarkowanych rozmiarach. W szczególności odnosi się to do przypadku, gdy spodziewany jest częsty dostęp do danych w pobliżu głowy listy posortowanej.

Oprócz tego drzewa znajdą zastosowanie w lesie, a listy na wykładach.