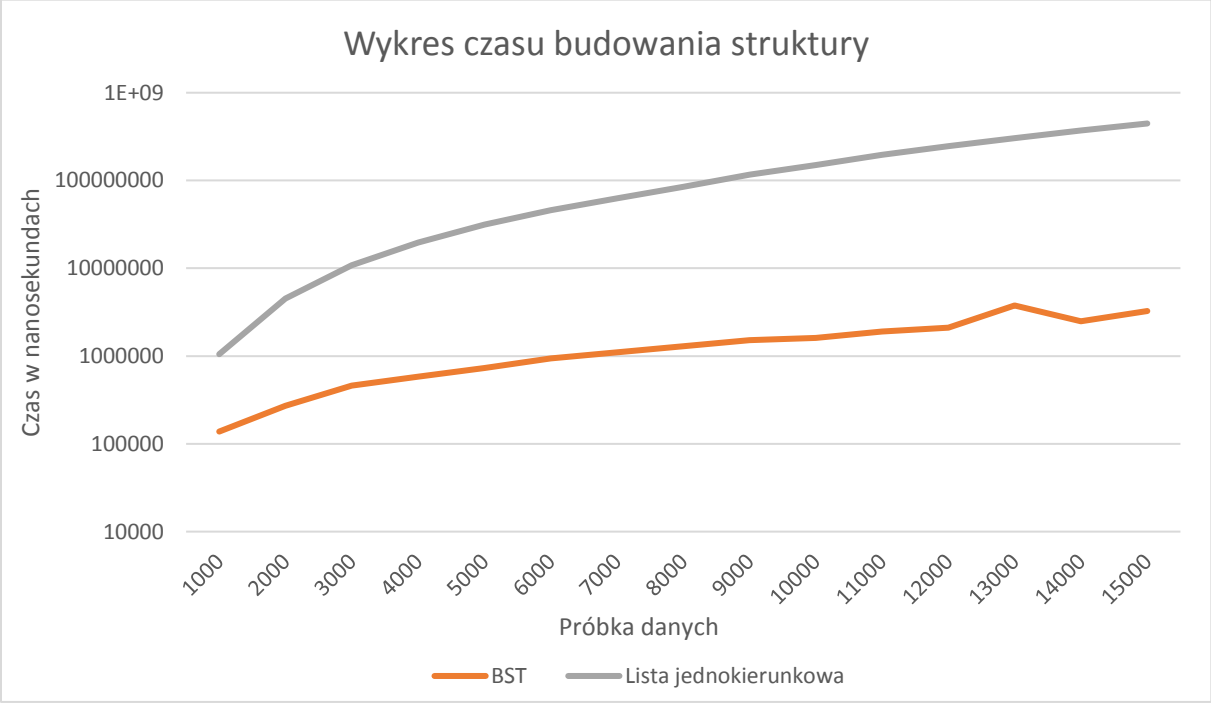
Hubert Cwajda, 141203
Michał Wiśniewski, 141335

Algorytmy i Struktury Danych - Sprawozdanie 2: Złożone struktury danych

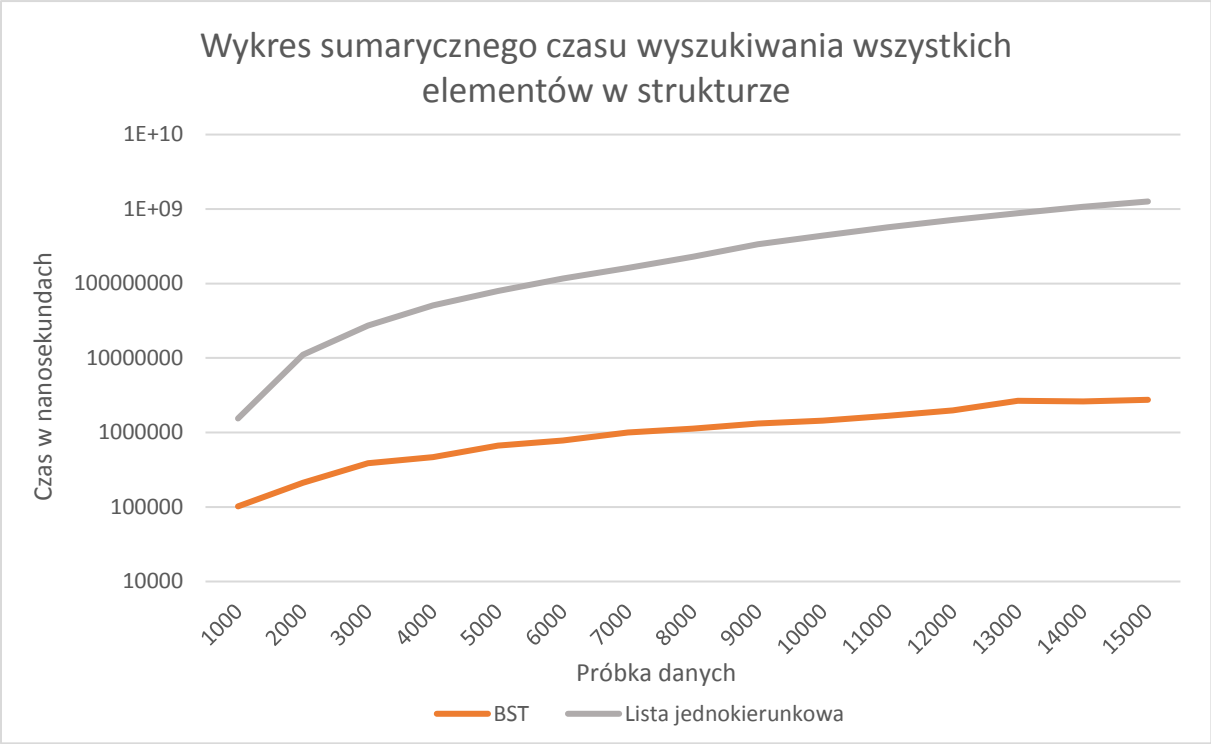
I. Porównanie szybkości operacji: tworzenia, wyszukiwania i usuwania dla posortowanej listy jednokierunkowej oraz drzewa poszukiwań binarnych w oparciu o nieposortowany ciąg nie powtarzających się liczb naturalnych.

- Lista jednokierunkowa to liniowo uporządkowana dynamiczna struktura danych z pojedynczymi dowiązaniami, w której każdy element składa się z dwóch lokalizacji w pamięci. Pierwsza zawiera obiekt sam w sobie, a druga wskaźnik do następnego elementu.
- Drzewo poszukiwań binarnych (BST) to takie drzewo, w którym każdy węzeł ma co najwyżej dwóch potomków i każde lewe poddrzewo zawiera wyłącznie elementy o kluczach mniejszych niż klucz węzła, a prawe wyłącznie elementy o kluczach nie mniejszych.

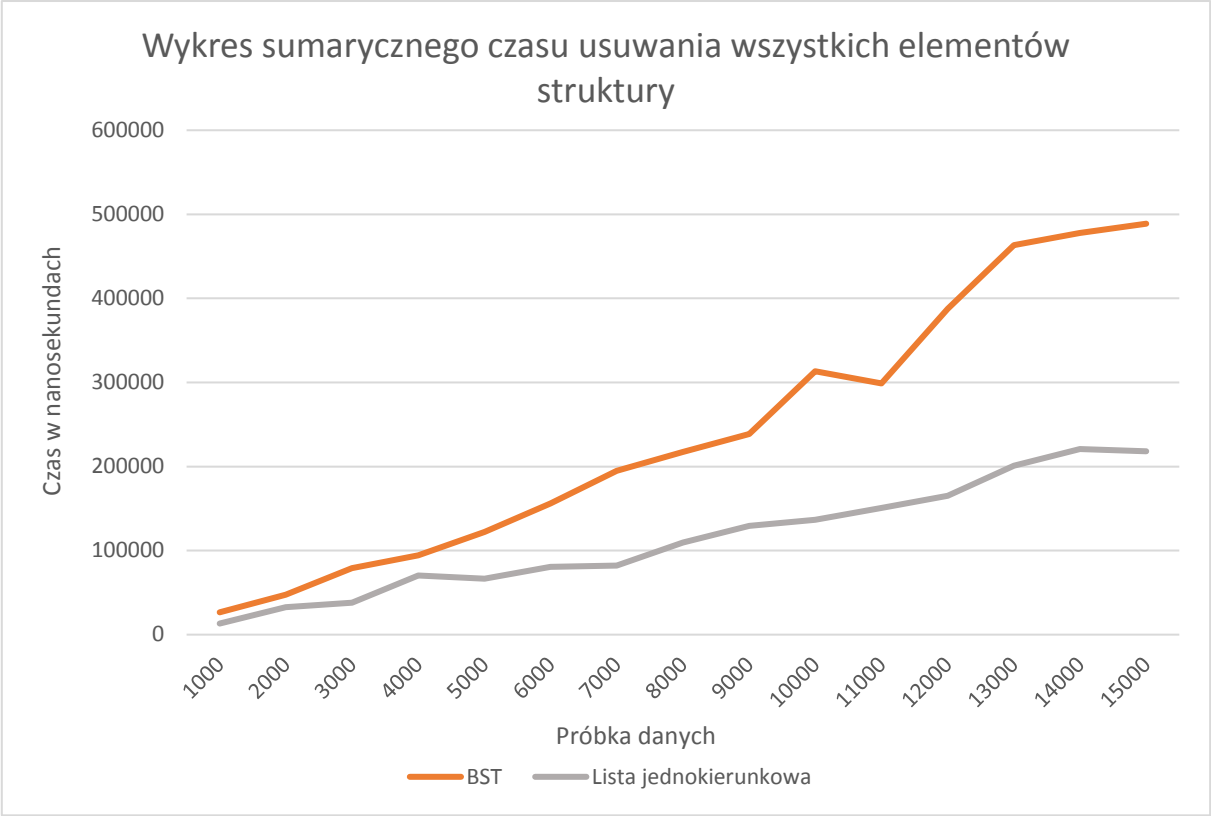


Czas budowania drzewa poszukiwań binarnych jest znacznie mniejszy od niż budowania listy jednokierunkowej. Wynika to z faktu, że operacja poprawnego wstawienia każdego nowego elementu do BST wymaga zwykle mniej porównań i zajmuje czas proporcjonalny do wysokości drzewa, a więc O(h), czyli O($log\ n$) w oczekiwanym przypadku. W najgorszym jest to O(n), kiedy drzewo nie jest zupełnie wyważone i każdy węzeł ma tylko jednego potomka. Wtedy drzewo zachowuje się jak lista jednokierunkowa, dla której taka skrajna sytuacja ma miejsce zawsze, dlatego oczekiwana złożoność obliczeniowa tej operacji wynosi w jej przypadku O(n). Samo w sobie wstawienie elementu nie jest czasochłonne, wymaga jedynie stworzenia nowego

węzła i zmiany wskaźników, co zajmuje O(*1*) czasu. Oczekiwana złożoność obliczeniowa zbudowania n-elementowego drzewa to zatem oczekiwana liczba porównań potrzebnych, by wstawić n losowych elementów do początkowo pustego drzewa, czyli O(*n log n*). Podobnie dla listy jednokierunkowej, dla której złożoność obliczeniowa stworzenia wynosi w związku z tym O(n^2).



Podobna sytuacja do tworzenia struktur zachodzi dla wyszukiwania w niej wszystkich elementów. Jest to wynikiem tego, iż operacje szukania poszczególnego elementu mają taką samą złożoność obliczeniową co operacje wstawienia, ponieważ aby wstawić poprawnie element do struktury trzeba wpierw wyszukać dla niego odpowiednie miejsce, a samo w sobie wstawienie elementu nie było czasochłonne. W związku z tym oczekiwana złożoność obliczeniowa jest taka sama, czyli O(*n log n*) dla wyszukania n elementów w drzewie poszukiwań binarnych. Czasy wyszukiwania w przypadku list jednokierunkowych są większe niż tworzenia, ponieważ przy każdej operacji działamy już na pełnej strukturze, aczkolwiek złożoność obliczeniowa pozostaje taka sama, czyli O(n^2).

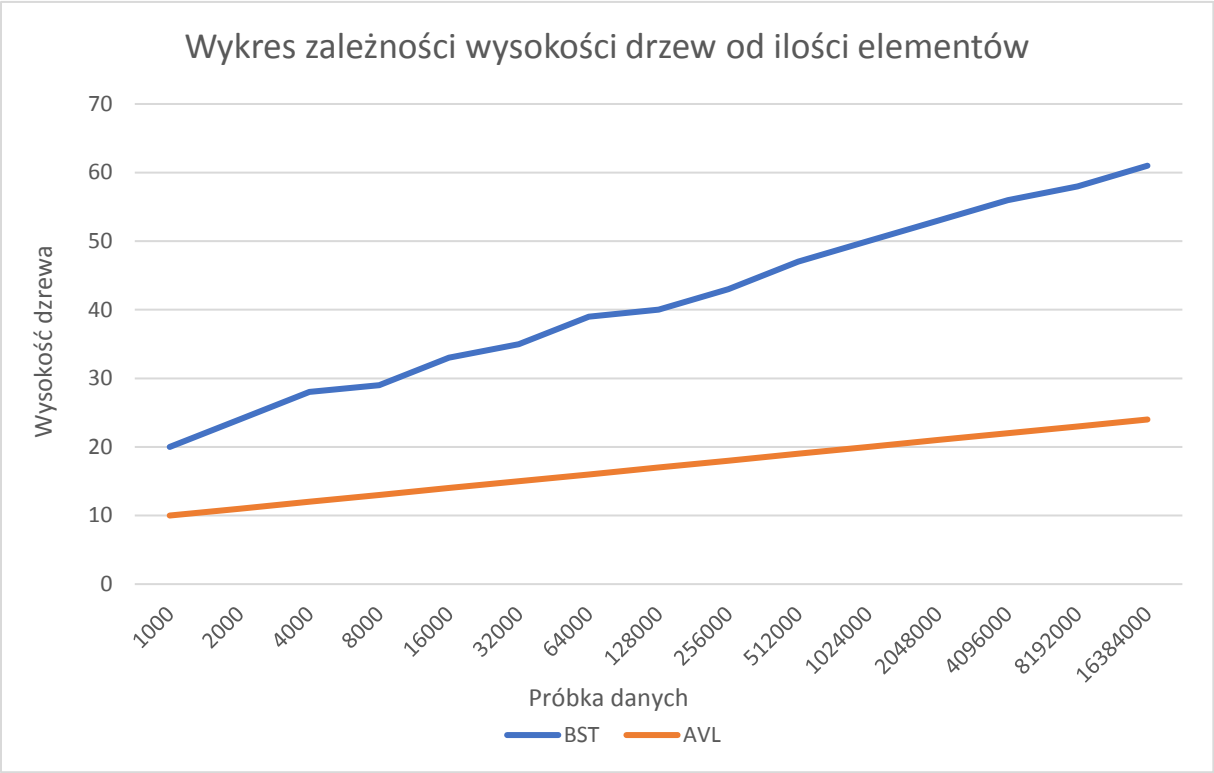


Czas usuwania struktur nieco szybciej wypadał na korzyść list jednokierunkowych. W nich usuwany był po kolei zawsze pierwszy element listy, natomiast drzewo usuwane było w porządku wstecznym. W związku z tym wykonanie tych operacji było szybkie dla obu struktur i złożoność obliczeniowa sumarycznego czasu usunięcia wszystkich elementów struktury wynosi w obu przypadkach O(n). Nieco większy czas osiągało BST, ponieważ operacja ta wymaga dojścia do liści, a dopiero potem usuwania w kolejności: lewy, prawy, korzeń. Z kolei w przypadku listy jednokierunkowej jedyne co wykonujemy, to usuwanie samo w sobie oraz przejście do następnego elementu.

Przedstawienie wad i zalet obu struktur:

Lista jednokierunkowa to podstawowa struktura danych, będąca przede wszystkim alternatywą dla tablic, ponieważ obie te struktury służą do przechowywania danych typu liniowego. Zaletą na korzyść listy jest w tym wypadku elastyczność, czyli łatwa możliwość zwiększenia bądź zmniejszenia swojego rozmiaru. Generalną zaletą tej struktury danych jest również łatwa implementacja. Oczywistą wadą jest natomiast konieczność zaczynania od początku i przechodzenia przez wszystkie elementy po kolei w celu dojścia do konkretnego elementu. Same w sobie operacje usuwania oraz dodawania następnego elementu są jednak bardzo szybkie i cechują się złożonością obliczeniową O (*1*). Posortowana lista jednokierunkowa za to nie ma za to wielu zalet nad BST, z racji tego że zachowuje się jak skrajnie źle wyważone drzewo, czyli potrzebuje O(n) czasu na dostęp do danego elementu. Ogólna złożoność obliczeniowa podstawowych operacji wykonywanych na drzewie poszukiwań binarnych to natomiast O(h). Inną zaletą BST jest łatwość w jego odwróceniu względem klucza. Do wad tej struktury należy zaliczyć przede wszystkim przypadki kiepskiego wyważenia, ale także nieco gorszą elastyczność, mniej wygodną wizualizacje danych oraz większe wymagania pamięciowe.

II. Porównanie wysokości binarnego drzewa przeszukiwań w wersji generycznej oraz wyważonej AVL.



Wyważone drzewo AVL dla dużej ilości danych cechuje się znacznie mniejszą wysokością. Każde lewe i prawe poddrzewo każdego węzła w tej strukturze różni się co najwyżej o jeden. Wyważanie drzewa stosuje się aby zwiększyć efektywność wykonywania operacji wyszukiwania, dodawania oraz usuwania elementu i zagwarantować, że nawet w pesymistycznym przypadku złożoność obliczeniowa tych operacji będzie wynosić O(*log n*). Wyważenie drzewa jest sensowne przede wszystkim wtedy gdy zależy nam na szybkim czasie wyszukiwania, a rzadkim dokładaniu i usuwaniu elementów, ponieważ operacja wyważania jest dość czasochłonna.

Wnioski ogólne dotyczące doboru struktur i ich zastosowań:
Listy jednokierunkowe mogą być stosowane do implementacji stosów, kolejek czy niektórych grafów. Przykładami użycia w sytuacjach codziennych mogą być np. podstawowe odtwarzacze muzyki czy przeglądarki grafik, w których podczas odtwarzania bieżącego pliku mamy dostęp do następnego w kolejce. Drzewo poszukiwań binarnych może być wykorzystane do zaimplementowania sortowania (podobnego do algorytmu heapsort), kolejki priorytetowej, słownika czy wielopoziomowego indeksowania w bazach danych. W większości przypadków BST są bardzo przydatne pod warunkiem, że są wyważone Te struktury są generalnie dobrym wyborem wtedy, gdy kluczowy jest przede wszystkim szybki czas wyszukiwania elementu.

Porównanie wad i zalet struktur statycznych i dynamicznych:
Dynamiczne struktury danych dają możliwość zmiany wielkości w dowolnym czasie i określa się je jako elastyczne, podczas gdy statyczne struktury danych cechują się odgórnie określoną zajętością pamięciową. Zaletą tych drugich jest natomiast łatwiejszy dostęp do poszczególnych elementów.