Prowadząca zajęcia:
dr hab. inż. Małgorzata Sterna, prof. nadzw.

ALGORYTMY I STRUKTURY DANYCH

Ćwiczenie 2

Wybrane złożone struktury danych

Stanisław Jasiewicz
nr 116753
Wojciech Regulski
nr 132312

Informatyka(WI) I1

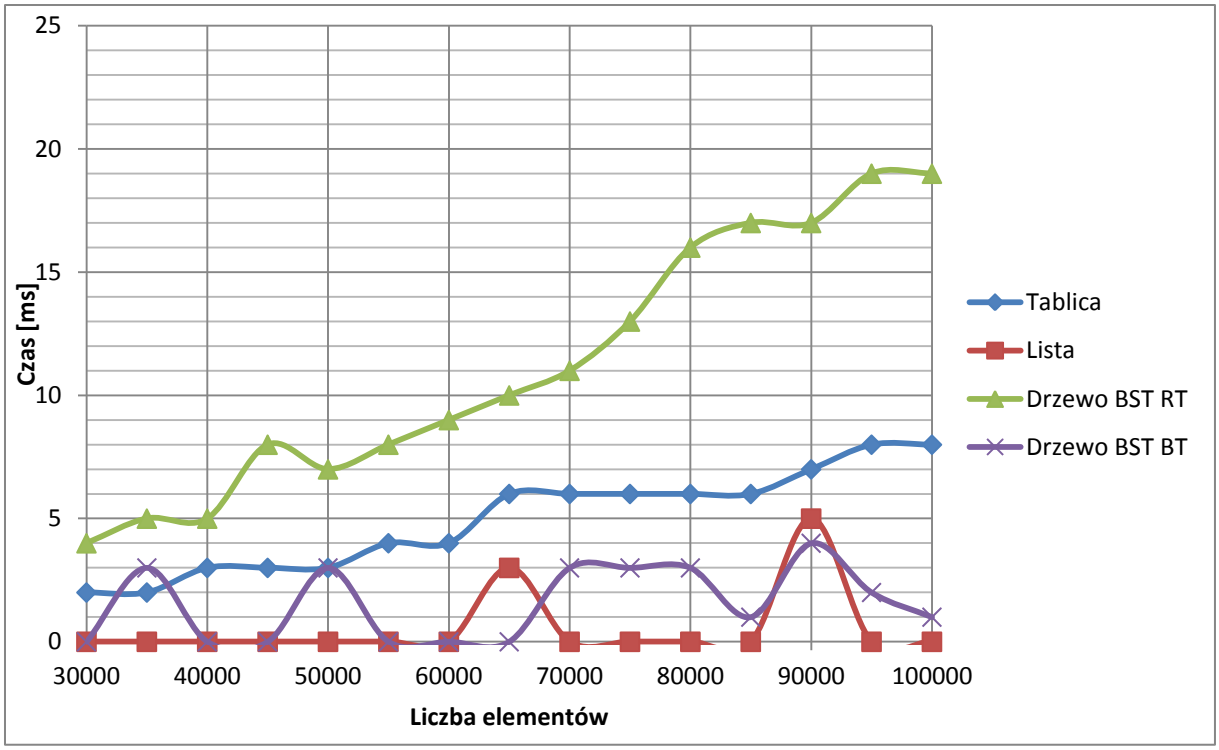
1 Cel

Poznanie różnic czasowych związanych z budową i przeszukiwaniem różnych struktur danych oraz nauka implementacji drzew binarnych BST oraz ich przeszukiwania. Porównanie wybranych struktur danych pod kątem kryteriów takich jak czas tworzenia i przeszukiwania, łatwość implementacji, łatwość modyfikacji, zajętość pamięciowa.

2 Pomiary i wykresy

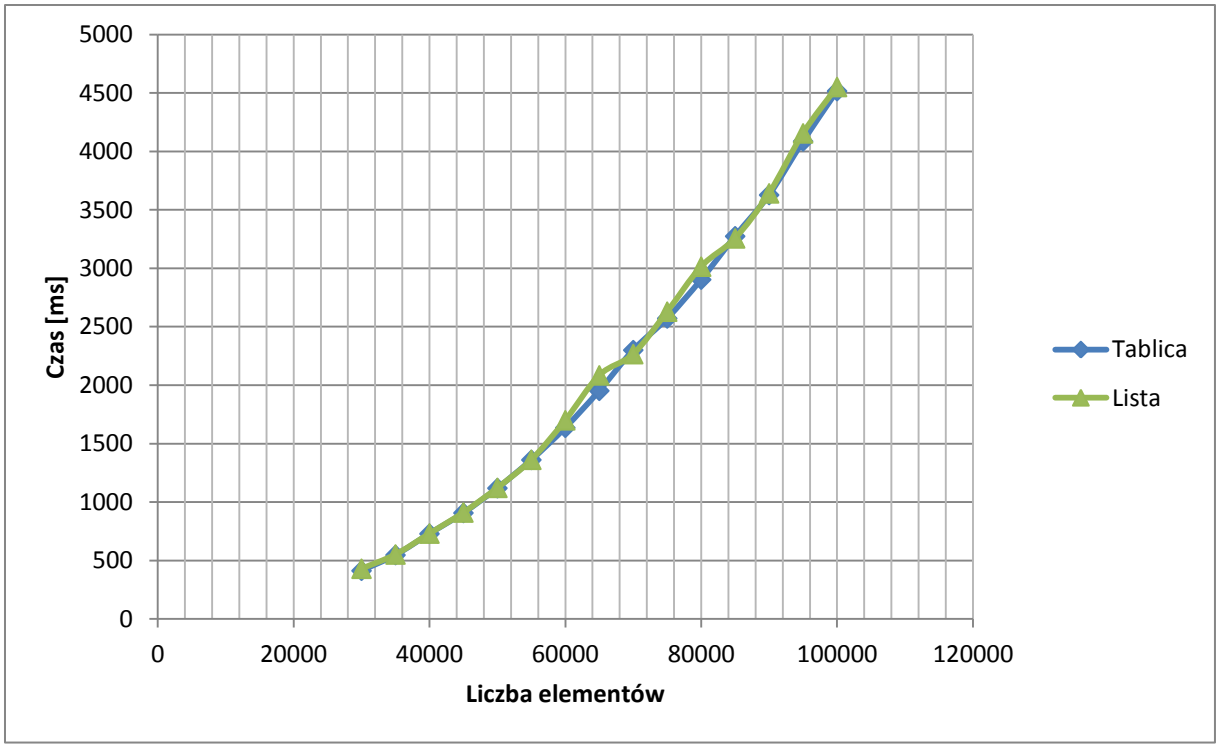
2.1 Czas tworzenia struktur w zależności od liczby elementów

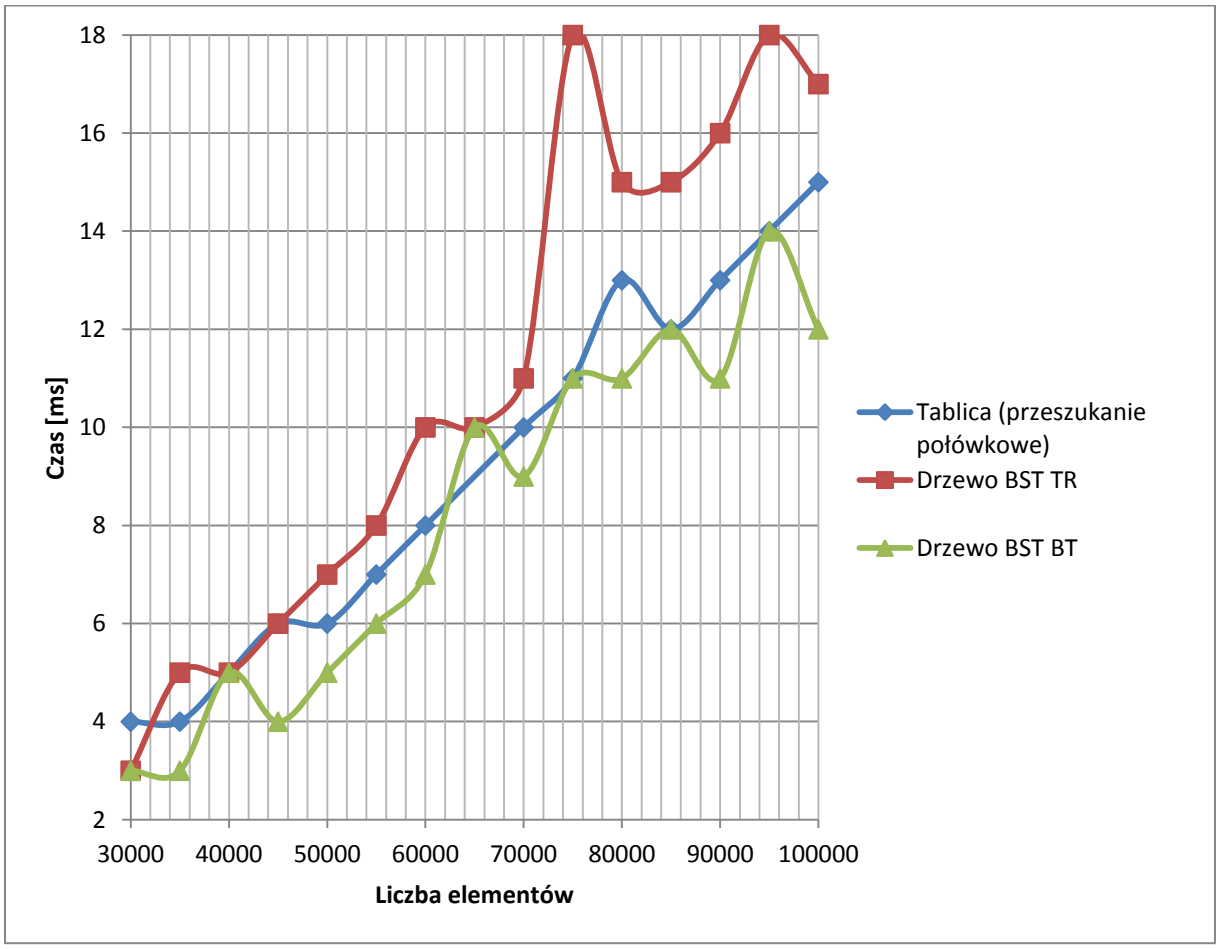
Liczba elementów [-]	Czas tworzenia tablicy c_b [ms]	Czas tworzenia listy c_L [ms]	Czas tworzenia drzewa BST TR c_{TR} [ms]	Czas tworzenia drzewa BST TB C_{TB} [ms]
30000	2	< 1	4	< 1
35000	2	< 1	5	3
40000	3	< 1	5	< 1
45000	3	< 1	8	< 1
50000	3	< 1	7	3
55000	4	< 1	8	< 1
60000	4	< 1	9	< 1
65000	6	3	10	< 1
70000	6	< 1	11	3
75000	6	< 1	13	3
80000	6	< 1	16	3
85000	6	< 1	17	1
90000	7	5	17	4
95000	8	< 1	19	2
100000	8	< 1	19	1



2.2 Czas wyszukiwania w zależności od liczby elementów

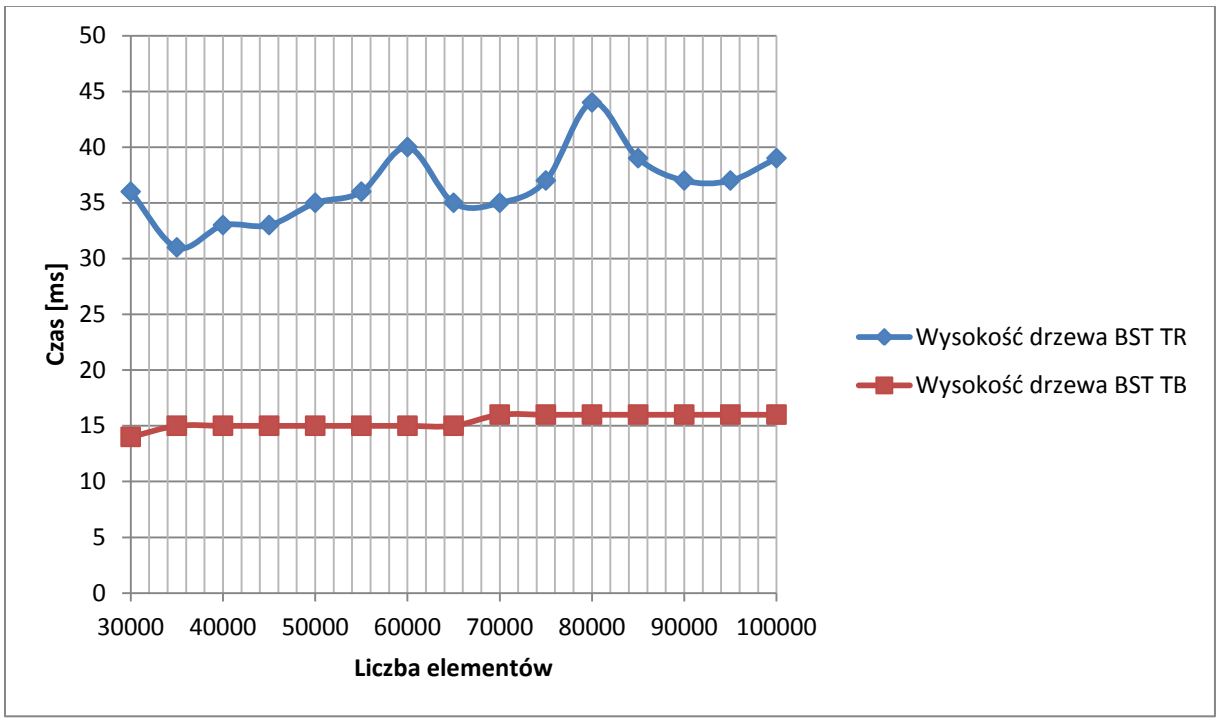
Liczba elementów [-]	Czas przeszukania tablicy s_{sB} [ms]	Czas przeszukania połówkowego s_{bB} [ms]	Czas przeszukania listy s_L [ms]	Czas przeszukania drzewa BST TR s_{TR} [ms]	Czas przeszukania drzewa BST BT s_{TR} [ms]
30000	414	4	431	3	3
35000	548	4	553	5	3
40000	730	5	732	5	5
45000	908	6	912	6	4
50000	1120	6	1123	7	5
55000	1362	7	1363	8	6
60000	1637	8	1701	10	7
65000	1953	33	2085	10	10
70000	2301	10	2267	11	9
75000	2572	11	2630	18	11
80000	2904	13	3015	15	11
85000	3275	12	3257	15	12
90000	3628	13	3642	16	11
95000	4087	14	4155	18	14
100000	4517	15	4552	17	12


5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
Czas [ms]
0
20000
40000
60000
80000
100000
120000
Liczba elementów
Tablica
Lista


18
16
14
12
10
8
6
4
2
Czas [ms]
30000
40000
50000
60000
70000
80000
90000
100000
Liczba elementów
Tablica (przeszukanie połówkowe)
Drzewo BST TR
Drzewo BST BT

2.3 Wysokość drzewa w zależności od liczby elementów

Liczba elementów [-]	Czas przeszukania tablicy h_{TR} [ms]	Czas przeszukania połówkowego H_{TB} [ms]
30000	36	14
35000	31	15
40000	33	15
45000	33	15
50000	35	15
55000	36	15
60000	40	15
65000	35	15
70000	35	16
75000	37	16
80000	44	16
85000	39	16
90000	37	16
95000	37	16
100000	39	16



3 Wnioski

Wysokość drzewa zbalansowanego wynosi w przybliżeniu $\log_2 n$, zaś wysokość drzewa niezbalansowanego jest ściśle zależna od uporządkowania tablicy danych wejściowych i w skrajnym przypadku (ciąg w pełni posortowany) może osiągnąć wysokość n. Na wykresie przedstawiającym wysokości drzew w funkcji liczby elementów można stwierdzić, że wysokość drzewa niezbalansowanego ma tendencję do wzrostu wraz ze wzrostem liczby elementów, lecz ponieważ zależy ona od wcześniej wspomnianego uporządkowania danych wejściowych, widoczne są również chwilowe zmniejszenia czasu wraz ze wzrostem liczby elementów. Gdyby wykorzystać tablicę posortowaną do budowy drzewa niezbalansowanego, jego wysokość znacznie by się zwiększyła. Do wykonania pomiarów wykorzystano algorytm do budowy drzewa BST TR, który w przypadku wystąpienia tej samej wartości, co w korzeniu, przypisuje ją do lewego dziecka. Oznacza to, że w przypadku tablicy posortowanej rosnąco, wysokość zbudowanego drzewa prawdopodobnie nie osiągnie n.
W przypadku tablicy posortowanej malejąco, otrzymane drzewo można potraktować jak listę jednokierunkową, w związku z czym jego wysokość będzie wynosić n. Niewielka wysokość drzewa ułatwia jego przeszukiwanie, w związku z czym najkorzystniejszym wariantem jest ten, w którym wysokość jest najniższa. W przypadku posortowanej tablicy wariantem tym jest drzewo zbalansowane. Im bardziej tablica jest nieuporządkowana, tym korzystniejsze staje się wykorzystanie drzewa niezbalansowanego.

Różnica w wysokości obu rodzajów drzew binarnych wynika z faktu, iż w przypadku drzewa niezbalansowanego dane są wprowadzane zgodnie z kolejnością występowania w tablicy, która w założeniu jest losowa. W drzewie zbalansowanym zastosowane jest dzielenie połówkowe, które minimalizuje wysokość drzewa. Wysokość drzewa zbalansowanego jest zawsze najmniejszą możliwą wysokością drzewa binarnego dla danej liczby elementów.

Wysokość drzewa binarnego jest istotna, ponieważ, jak zostało powiedziane, wpływa ona bezpośrednio na szybkość przeszukiwania struktury. Drzewo zbalansowane cechuje się niższą złożonością tworzenia oraz przeszukiwania struktury.

Można zauważyć, że czas wyszukiwania elementów w tablicy posortowanej B pokrywa się z czasem wyszukiwania w liście. W obu przypadkach złożoność obliczeniowa wynosi $O(n^2)$, ponieważ wymagany jest obieg po wszystkich elementach tablicy A oraz, dla każdego elementu w niej zawartego, obieg po potencjalnie wszystkich elementach listy bądź tablicy B.

Zarówno wyszukiwanie połówkowe w tablicy, jak i przeszukiwanie drzewa zbalansowanego, wykorzystują metodę dzielenia połówkowego. W obu przypadkach przeszukiwanie następuje dla każdego z n elementów tablicy, zaś jedna iteracja przeszukania ma złożoność obliczeniową $O(\log_2 n)$, co oznacza, że całe przeszukiwanie w obu przypadkach ma złożoność obliczeniową $O(n*\log_2 n)$.

Dane, na których operawano, były typu integer, więc każdy element zajmował 4B pamięci. Tablica zajmowała więc n*4B, a lista n*8B ze względu na dodatkową pamięć potrzebną w każdym elemencie na referencję na następny element. W drzewie, w którym każdy element posiada oprócz danej dwie referencje na następne elementy, pamięć zajmowana przez każdy element to 12B.

Po analizie otrzymanych wyników można stwierdzić, że nakrótszym czasem przeszukiwania cechuje się zbalansowane drzewo BST TB. Czas jego tworzenia również był niewielki - na tyle, że nie udało się stwierdzić jego wzrostu wraz ze wzrostem liczby elementów, ponieważ struktura tworzyła się w czasie mniejszym niż 1ms. Wadą drzewa zbalansowanego jest konieczność wykorzystania posortowanej tablicy, co może nie być możliwe, jeśli dane posiadają określoną kolejność, którą należy zachować. Drzewo jest również strukturą o największej zajętości pamięciowej spośród struktur badanych
w doświadczeniu. Inną istotną wadą jest trudność dodania lub usunięcia elementów z drzewa. Propozycją obejścia tego problemu jest przechowywanie danych dodatkowo w innej strukturze, której modyfikacje są łatwiejsze, i przebudowywanie drzewa na jej podstawie po zastosowaniu zmian. Stwarza to jednak konieczność przechowywania tych samych danych w dwóch miejscach oraz może nie sprawdzić się w przypadku częstych modyfikacji zawartości struktury.

Przeszukiwanie tablicy metodą połówkową cechuje się stosunkowo niewiele większym czasem poszukiwań oraz tworzenia struktury względem drzewa zbalansowanego. Zaletami tablicy względem drzewa jest najmniejsza zajętość pamięciowa spośród analizowanych struktur oraz brak konieczności sortowania danych, obarczona jest jednak podobnymi trudnościami związanymi z dodawaniem oraz usuwaniem danych.

Drzewo niezbalansowane cechuje się największym czasem tworzenia z badanych struktur, zaś przeszukiwanie go jest dłuższe zarówno od przeszukiwania drzewa zbalansowanego, jak i tablicy. Nad drzewem zbalansowanym posiada zaletę braku konieczności sortowania elementów, lecz na jego wysokość silnie wpływa uporządkowanie danych, co może czynić tę strukturę nierzetelną.

Lista cechuje się najdłuższym czasem przeszukiwania, porównywalnym ze zwyczajnym przeszukiwaniem tablicy, posiada za to mniejszą zajętość pamięciową niż drzewo oraz umożliwia w najłatwiejszy sposób dodawanie oraz usuwanie elementów.

Podsumowując, chociaż czasowo najlepiej w doświadczeniu wypadła struktura drzewa zbalansowanego, posiada ona pewne wady, które w zależności od sytuacji mogą uczynić listę lub tablicę lepszym wyborem. Najmniej korzystnie wypada drzewo niezbalansowane ze względu na stosunkowo długi czas tworzenia i przeszukiwania oraz nierzetelność wynikającą z silnego wpływu uporządkowania danych wejściowych na wysokość drzewa.