PAMSI - PROJEKT 1
ALGORYTMY SORTOWANIA

Wykonał:
Jakub Pietrus 241174

Prowadzący:
dr Kzysztof Halawa

Termin oddania projektu:
29.03.2019

Termin zajęć:
Wtorek 13.15-15.00

1 Wstęp teoretyczny

Podczas realizacji projektu zaimplementowano trzy rodzaje sortowan - sortowanie przez scalanie, sortowanie szybkie oraz introspektywne, a nastepnie przeprowadzono na nich testy efektywnosci.

1.1 Sortowanie przez scalanie

Sortowanie przez scalanie jest przykładem sortowania rekurencyjnego z grupy algorytmów szybkich, który wykorzystuje metode dziel i zwyciezaj. Tablice do posortowania dzielimy na dwie podtablice, na których rekurencyjnie wywołuje sie te sama funkcje do momentu, gdy podtablice beda zawierały tylko jeden element. Całe własciwe sortowanie przerzucone jest na funkcje scalajaca.

1.2 Sortowanie szybkie

Sortowanie szybkie, podobnie jak sortowanie przez scalanie, wykorzystuje metode dziel i zwyciezaj. W tym rodzaju sortowania praca nad sortowaniem tablicy wykonana jest podczas dzielenia problemu na dwa mniejsze podproblemy. Do wyznaczenia lewego i prawego podproblemu niezbedna jest wartosc, względem której zostana stworzone dwie podtablice - z wartosciami mniejszymi od porównania oraz druga z wiekszymi.

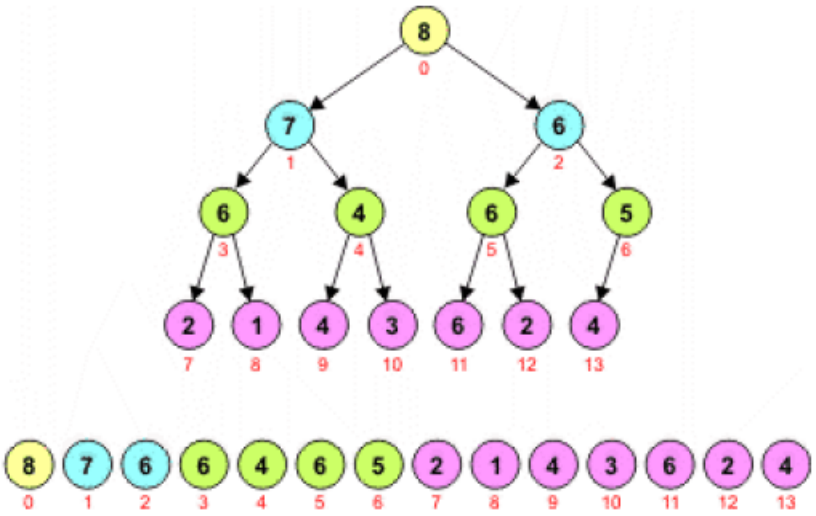
1.3 Sortowanie introspektywne

Sortowanie introspektywne to sortowanie, którego celem jest wyeliminowanie kwadratowej złozonosci obliczeniowej algorytmu sortowania szybkiego. W tym sortowaniu okreslana jest wartosc maksymalnej ilosci rekurencyjnych wywołan ($\max = 2 \cdot \log_2 N$, gdzie N - rozmiar tablicy). Gdy $max = 0$ wywoływane jest

sortowanie przez kopcowanie. W czasie sortowania, gdy $max > 0$, wywoływana jest rekurencyjne funkcja SortujIntro z parametrem max o 1 mniejszym. IntroSort, podobnie jak sortowanie szybkie, również dzieli problem na dwa mniejsze podproblemy.

1.3.1 Sortowanie przez kopcowanie

Sortowanie przez kopcowanie to przykład sortowania z grupy algorytmów szybkich. Sortowanie to wykorzystuje kopiec (binarne drzewo) reprezentowane przez tablice. PRzykład takiej reprezentacji przedstawia poniższy rysunek:



Rysunek 1: Reprezentacja kopca jako tablicy

Synowie k - tego wezła mają indeksy $2 \cdot k + 1$ oraz $2 \cdot k + 2$. Sortowanie rozpoczyna się od ostatniego rodzica i, w razie potrzeby, zamieniane jest dziecko z rodzicem, by zachować strukturę kopca - każdy z rodziców ma wartość wiekszą niz kazde z jego dzieci.

2 Złożoność obliczeniowa

Złozoność obliczeniową każdego z zaimplementowanych sortowań dla przypadku średniego i najgorszego przedstawia tabela poniżej:

sortowanie	czasowa średni | najgorszy	pamięciowa średni | najgorszy
przez scalanie	$O(n \cdot \log_2 n)$	$O(n)$
szybkie	$O(n \cdot \log_2 n) \mid O(n^2)$	$O(1) \mid O(\log_2 n)$
przez kopcowanie	$O(n \cdot \log_2 n)$	$O(1)$
introperspektywne	$O(n \cdot \log_2 n)$	$O(1)$

Tabela 1: Złożoność obliczeniowa zaimplementowanych algorytmów

3 Testy efektywności

Po implementacji sposoby sortowania zostały poddane testow efektywności. Dla każdej z rodzajów tablic:

- elementy losowe
- elementy posortowane w odwrotnej kolejności
- 25%, 50%, 75%, 95%, 99%, 99,7% poczatkowych elementów juz posortowanych

oraz dla tablic posiadających:

- 10 000
- 50 000
- 100 000
- 500 000

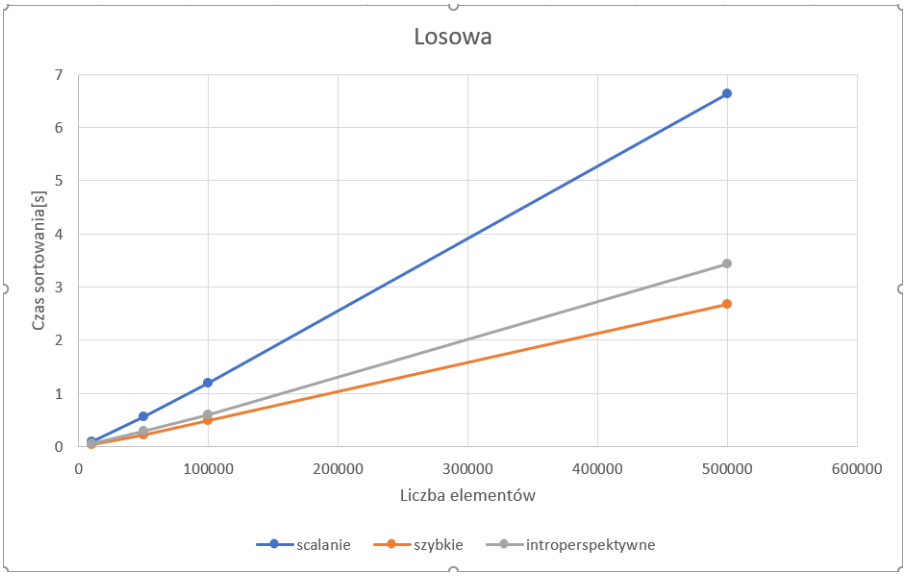
elementów.

3.1 Elementy tablicy losowe

Testom poddano 100 różnych tablic.

	Czas sortowania [s]		
ilość elementów	przez scalanie	szybkie	introperspektywne
10 000	0,098	0,033	0,048
50000	0,561	0,219	0,295
100000	1,199	0,483	0,599
500000	6,641	2,681	3,44

Tabela 2: Czas sortowania dla tablicy losowej

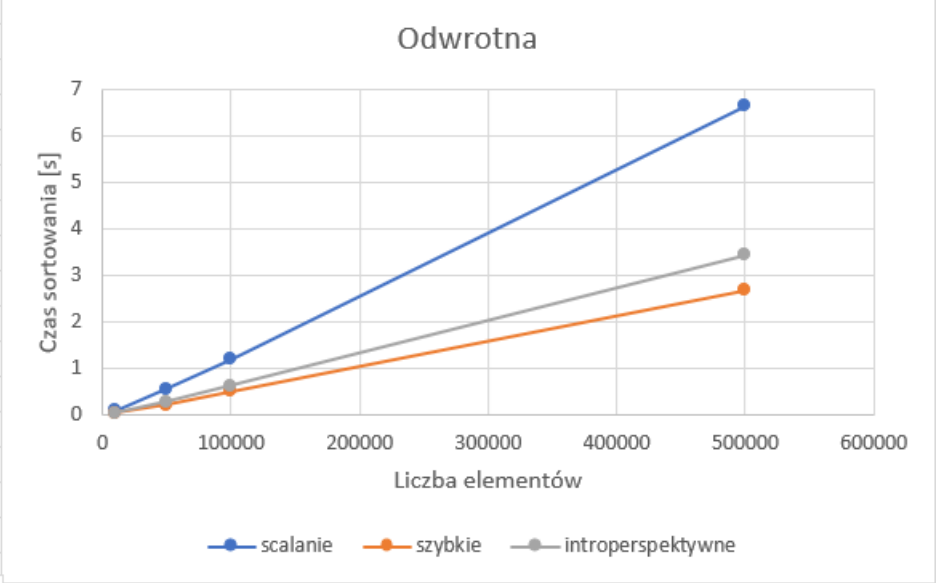


Rysunek 2: Czas sortowania tablicy uporzadkowanej w losowy sposób w funkcji ilosci elementów tablicy

3.2 Elementy tablicy posortowane odwrotnie

	Czas sortowania [s]		
ilość elementów	przez scalanie	szybkie	introperspektywne
10000	0,097	0,042	0,042
50000	0,561	0,219	0,295
100000	1,202	0,51	0,631
500000	6,641	2,681	3,44

Tabela 3: Czas sortowania dla tablicy odwrotnej

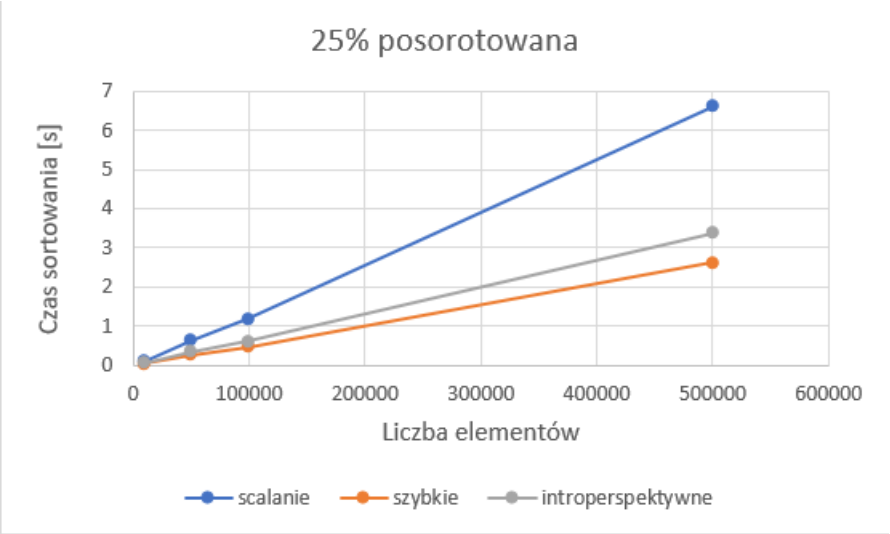


Rysunek 3: Czas sortowania tablicy posortowanej w odwrotny sposób w funkcji ilosci elementów tablicy

3.3 Elementy tablicy posortowane w 25%

	Czas sortowania [s]		
ilość elementów	przez scalanie	szybkie	introperspektywne
10000	0,098	0,037	0,05
50000	0,625	0,267	0,337
100000	1,182	0,48	0,606
500000	6,609	2,617	3,387

Tabela 4: Czas sortowania dla tablicy posortowanej w 25%

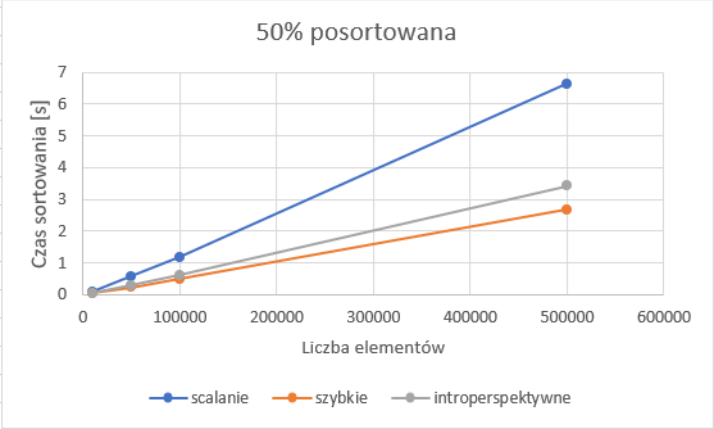


Rysunek 4: Czas sortowania tablicy posortowanej w 25% w funkcji ilosci elementów tablicy

3.4 Elementy tablicy posortowane w 50%

	Czas sortowania [s]		
ilość elementów	przez scalanie	szybkie	introperspektywne
10000	0,098	0,039	0,048
50000	0,571	0,23	0,291
100000	1,181	0,485	0,608
500000	6,637	2,684	3,429

Tabela 5: Czas sortowania dla tablicy posortowanej w 50%

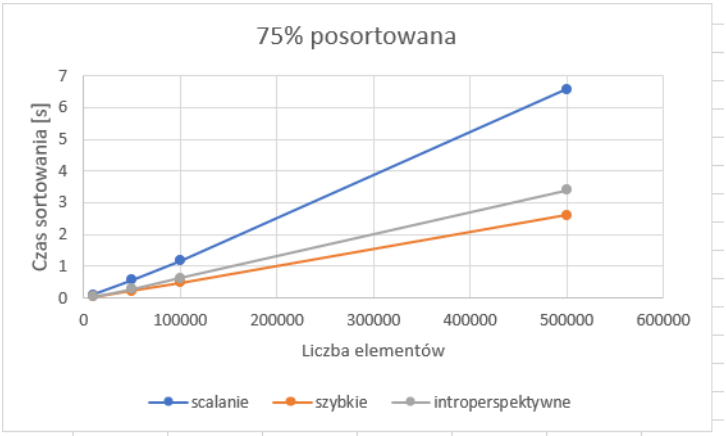


Rysunek 5: Czas sortowania tablicy posortowanej w 50% w funkcji ilosci elementów tablicy

3.5 Elementy tablicy posortowane w 75%

	Czas sortowania [s]		
ilość elementów	przez scalanie	szybkie	introperspektywne
10000	0,095	0,043	0,043
50000	0,56	0,224	0,289
100000	1,18	0,49	0,614
500000	6,577	2,604	3,392

Tabela 6: Czas sortowania dla tablicy posortowanej w 75%

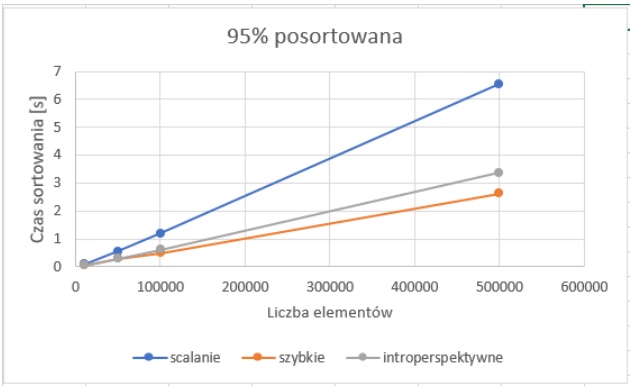


Rysunek 6: Czas sortowania tablicy posortowanej w 75% w funkcji ilosci elementów tablicy

3.6 Elementy tablicy posortowane w 95%

	Czas sortowania [s]		
ilość elementów	przez scalanie	szybkie	introperspektywne
10000	0,094	0,039	0,043
50000	0,547	0,28	0,277
100000	1,191	0,485	0,612
500000	6,548	2,623	3,37

Tabela 7: Czas sortowania dla tablicy posortowanej w 95%

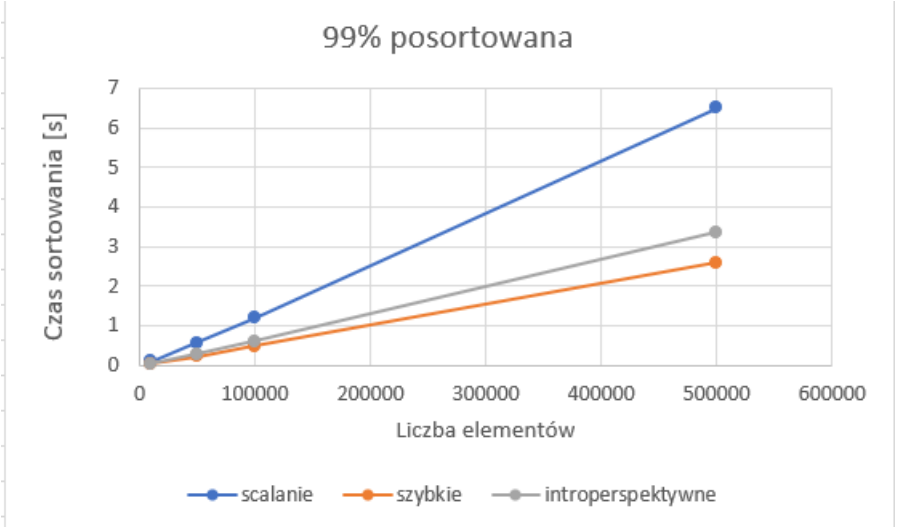


Rysunek 7: Czas sortowania tablicy posortowanej w 95% w funkcji ilosci elementów tablicy

3.7 Elementy tablicy posortowane w 99%

	Czas sortowania [s]		
ilość elementów	przez scalanie	szybkie	introperspektywne
10000	0,097	0,039	0,044
50000	0,561	0,223	0,287
100000	1,188	0,489	0,612
500000	6,51	2,594	3,356

Tabela 8: Czas sortowania dla tablicy posortowanej w 99%

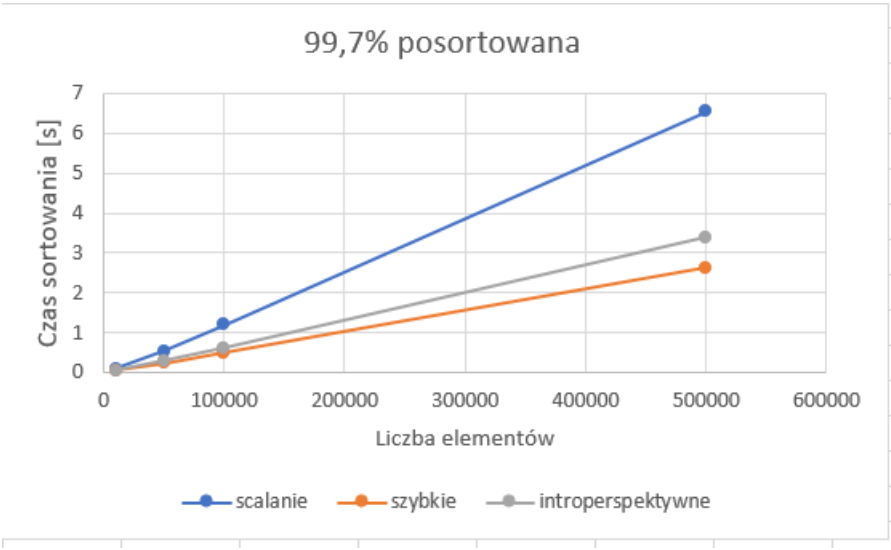


Rysunek 8: Czas sortowania tablicy posortowanej w 99,7% w funkcji ilosci elementów tablicy

3.8 Elementy tablicy posortowane w 99,7%

	Czas sortowania [s]		
ilość elementów	przez scalanie	szybkie	introperspektywne
10000	0,093	0,043	0,044
50000	0,543	0,224	0,281
100000	1,199	0,489	0,612
500000	6,548	2,631	3,402

Tabela 9: Czas sortowania dla tablicy posortowanej w 99,7%



Rysunek 9: Czas sortowania tablicy posortowanej w 99,7% w funkcji ilosci elementów tablicy

4 Podsumowanie

Wszystkie algorytmy są bardzo szybkie, między sortowaniem szybkim a sortowaniem introspektywnym praktycznie nie widać różnic. Najgorzej z całej trójki wypada mergesort, jest jest on ponad 2 razy wolniejszy od 2 pozostałych zaimplementowanych algortymów.

5 Literatura

- https://pl.wikipedia.org/wiki/Sortowanieintrospektywne
- https://www.geeksforgeeks.org/merge-sort/
- http://slideplayer.pl/slide/8941861/
- http://miroslawzelent.pl/kurs - c + +/sortowanie - zlozonosc - algorytmow/

Kod : https://github.com/pieetrus/1_Sortowanie