Wojciech Dominiak

RAPORT

Algorytmy i struktury danych – laboratorium

Lista 8

ALGORYTM SORTOWANIA SHELLA

1. Opis danych.

Dane wykorzystane w realizacji ćwiczenia to tablice, zawierające kolejno 5 000, 10 000, 50 000 i 100 000 wygenerowanych losowo liczb całkowitych. Zakres liczb całkowitych umieszczonych w tablicach to odpowiednio 0-5 000 dla pierwszego zestawu, 0-10 000 dla drugiego itd. Każda z analizowanych wersji algorytmów sortowania Shella została wywołana dla 5 rodzajów ciągów opisanych w treści zadania. Wszystkie zaimplementowane w ćwiczeniu algorytmy sortowały dane w porządku rosnącym (od najmniejszej liczby do największej).

2. Tabela czasów.

Wykonano 5 pomiarów czasu wykonywania algorytmów dla każdej wielkości tablicy zawierającej dane do posortowania. W tabeli poniżej zestawiono średnie wyniki z tych pomiarów (w ms):

Dla 5 000 elementów:

wersja \ ciąg	**Ciąg a)**	**Ciąg b)**	**Ciąg c)**	**Ciąg d)**	**Ciąg dod.**
Wersja 1	2	1	1	1	1
Wersja 2	4	5	4	4	5
Wersja 3	11	11	13	20	24

Dla 10 000 elementów:

wersja \ ciąg	**Ciąg a)**	**Ciąg b)**	**Ciąg c)**	**Ciąg d)**	**Ciąg dod.**
Wersja 1	3	2	1	1	1
Wersja 2	15	17	15	17	15
Wersja 3	29	30	40	72	92

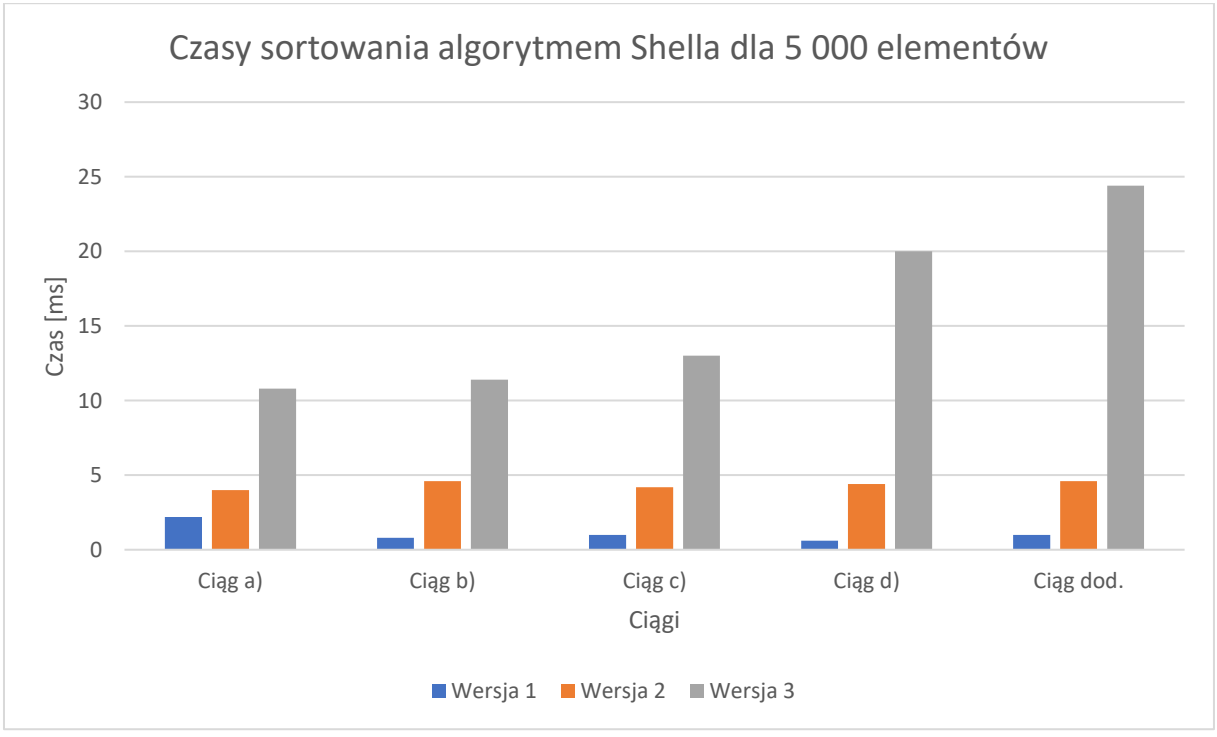
Dla 50 000 elementów:

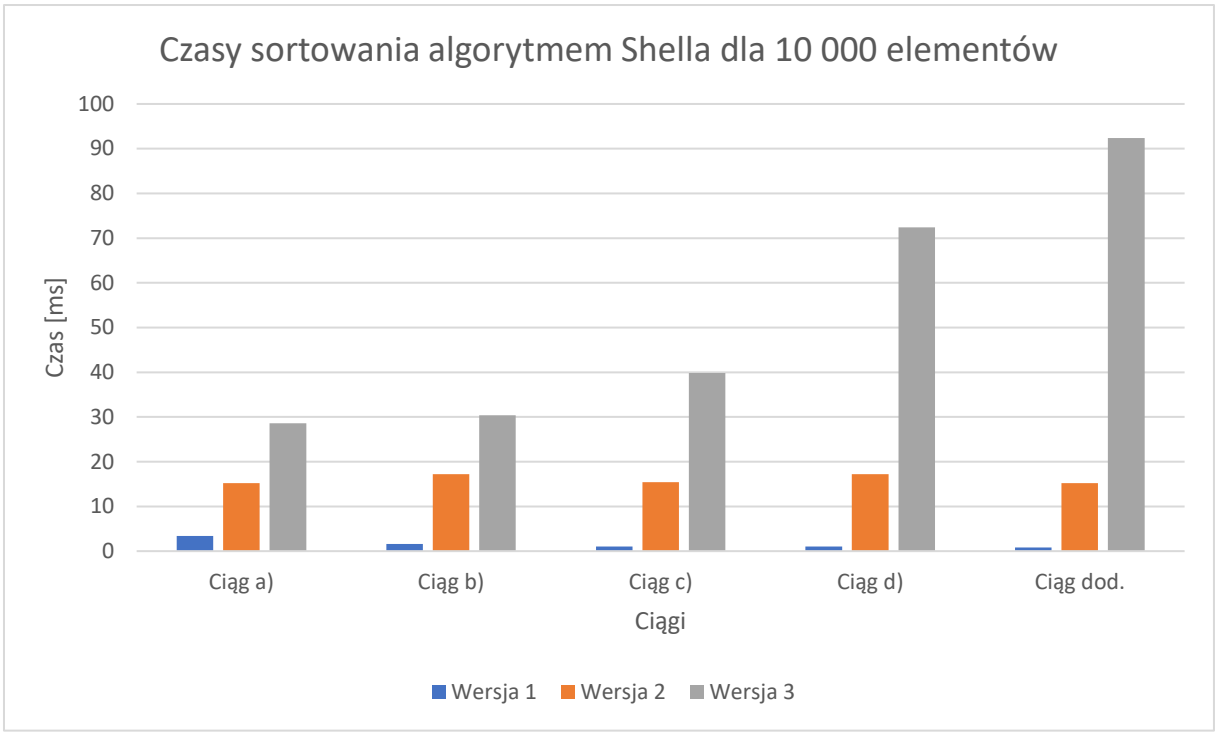
wersja / ciąg	Ciąg a)	Ciąg b)	Ciąg c)	Ciąg d)	Ciąg dod.
Wersja 1	5	6	6	7	5
Wersja 2	377	379	376	379	376
Wersja 3	479	738	963	1671	2048

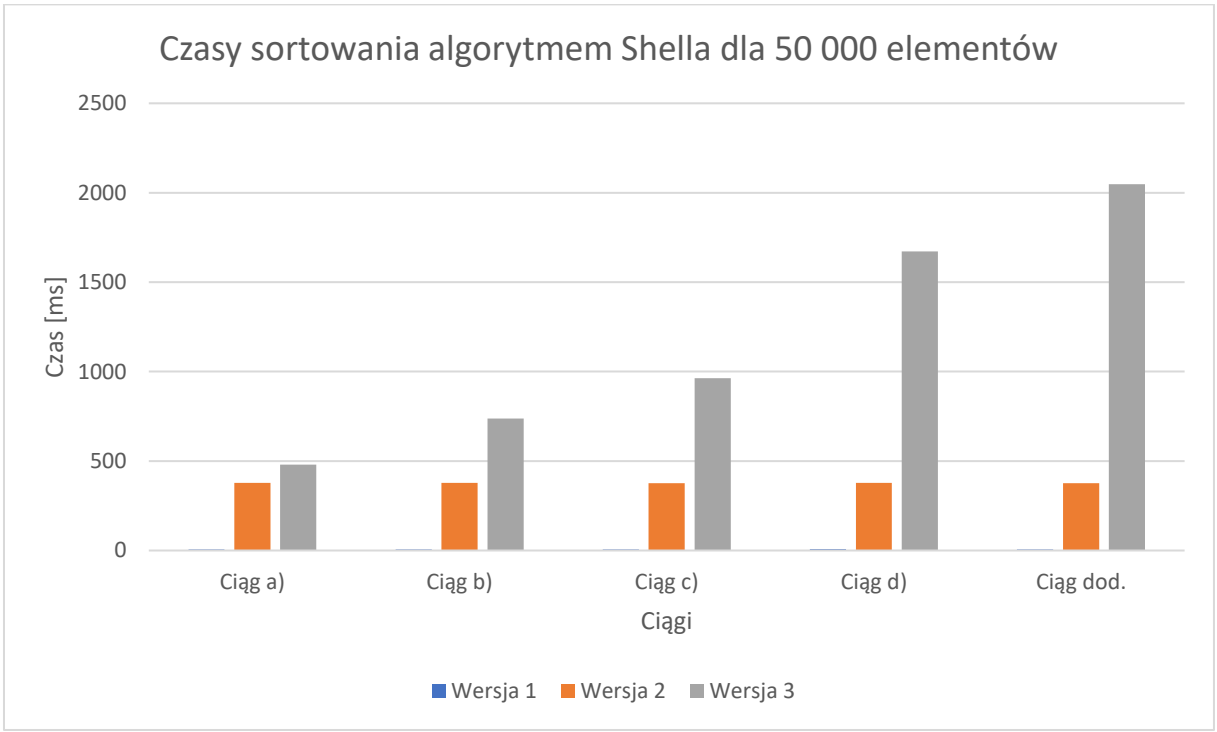
Dla 100 000 elementów:

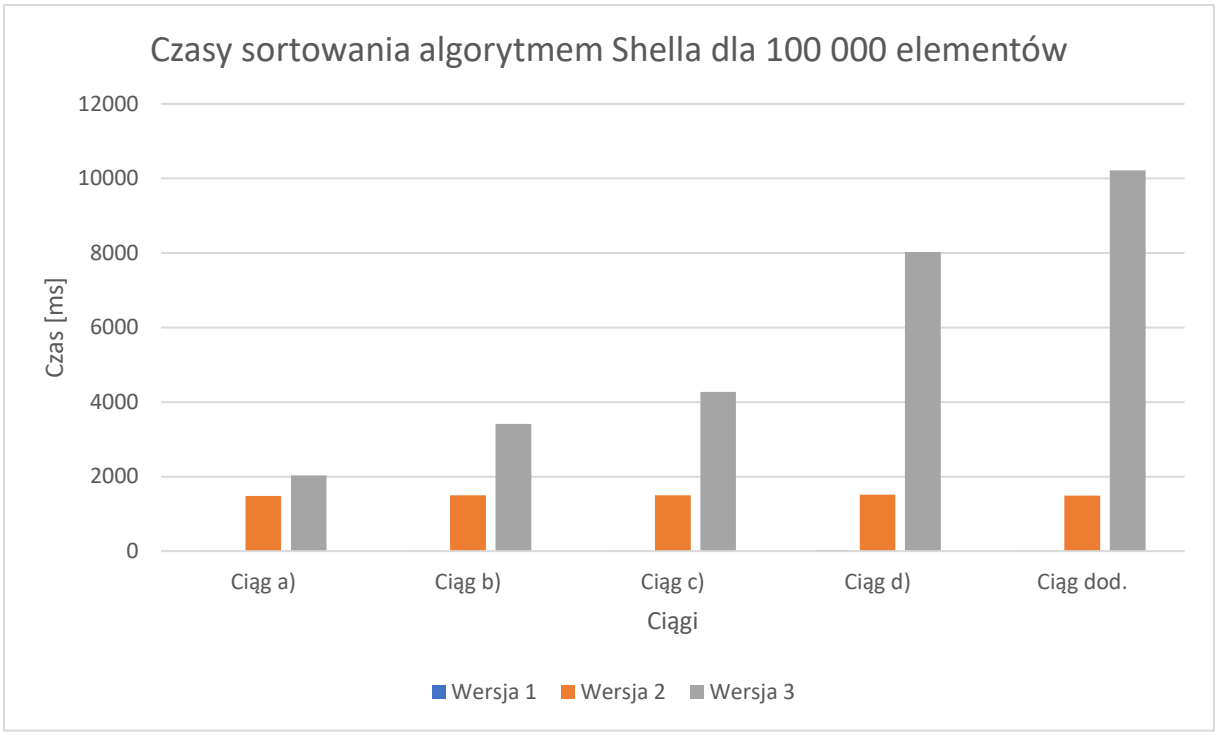
wersja / ciąg	Ciąg a)	Ciąg b)	Ciąg c)	Ciąg d)	Ciąg dod.
Wersja 1	12	15	14	18	12
Wersja 2	1485	1499	1495	1515	1487
Wersja 3	2033	3414	4274	8025	10217

3. Wykresy:




Czasy sortowania algorytmem Shella dla 10 000 elementów
Czas [ms]
100
90
80
70
60
50
40
30
20
10
0
Ciąg a)
Ciąg b)
Ciąg c)
Ciąg d)
Ciąg dod.
Ciągi
Wersja 1
Wersja 2
Wersja 3


Czasy sortowania algorytmem Shella dla 50 000 elementów
Czas [ms]
2500
2000
1500
1000
500
0
Ciąg a)
Ciąg b)
Ciąg c)
Ciąg d)
Ciąg dod.
Ciągi
Wersja 1
Wersja 2
Wersja 3



4. Wnioski.

Najszybszą wersją okazała się wersja 1, używająca algorytmu InsertSort we wszystkich sortowaniach. Wynika to z tego, że InsertSort jest najszybszym algorytmem spośród algorytmów prostych i bardzo szybkim dla danych prawie posortowanych, z którymi mamy do czynienia w tej sytuacji. Najwolniejszą wersją zaś jest wersja 3, używająca BubbleSort w sortowaniach “co h” oraz InsertSort “co 1”, ponieważ co każdą iterację BubbleSort wykonuje wiele niepotrzebnych porównań, co znacznie wydłuża pracę algorytmu. Spośród ciągów analizowanych w zadaniu dla wersji 1. i 2. najefektywniejszymi są ciąg a) i ciąg dodatkowy, jednak różnice są minimalne. Dla wersji 3 najefektywniejszym ciągiem jest ciąg a), który zawiera najmniej kolejnych wartości h, dzięki czemu BubbleSort wykona najmniej niepotrzebnych przejść. Im większa długość tablicy, tym różnice między czasami działania algorytmów stają się wyraźniejsze.