Laboratoria: piątek 9:45

Grupa: L9

Informatyka WIiT

Algorytmy i Struktury Danych

Prowadzący: Dominik Witczak

Sprawozdanie do

Ćwiczenia 1 – Algorytmy sortowania (24.03.2024)

Autorzy:

Maksymilian Norkiewicz 160267

Jędrzej Ogrodowski 160229

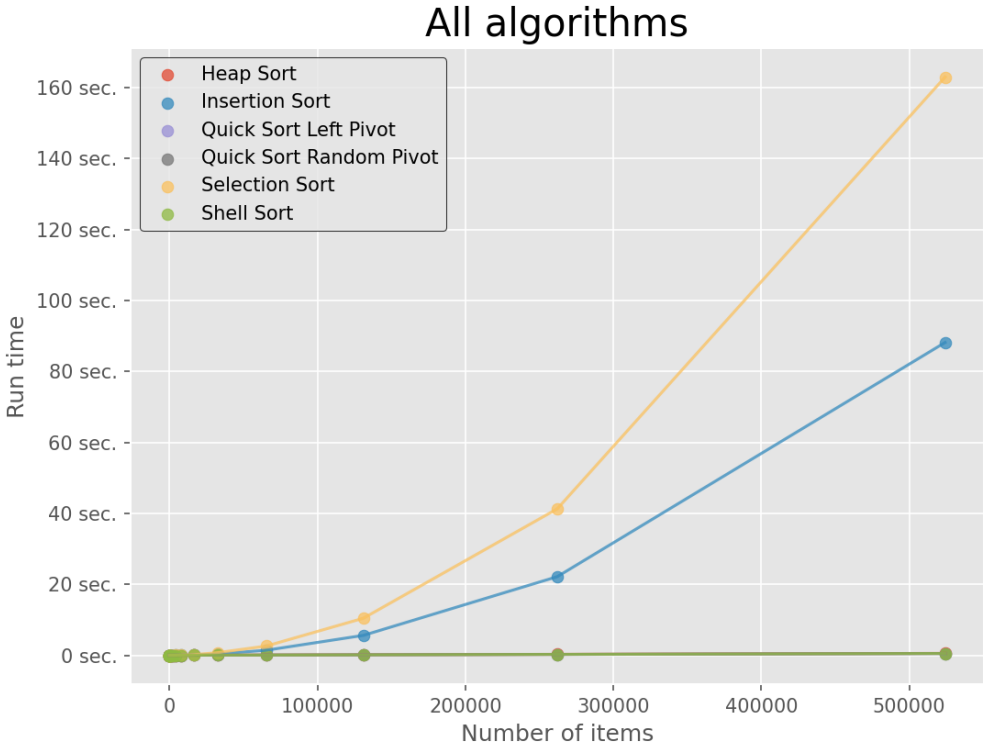
1. Wstęp

Celem projektu jest implementacja oraz badanie efektywności wybranych algo- rytmów sortowania. Wykorzystane zostały algorytmy Insertion Sort, Shell Sort z przyrostami Sedgewicka, Selection Sort, Heap Sort oraz Quick Sort w dwóch wariantach. Algorytmy zostały napisane w języku C++. Testy zostały wykonane na komputerze Apple Macbook Air z procesorem M2.
Projekt dostępny jest na platformie **GitHub.**

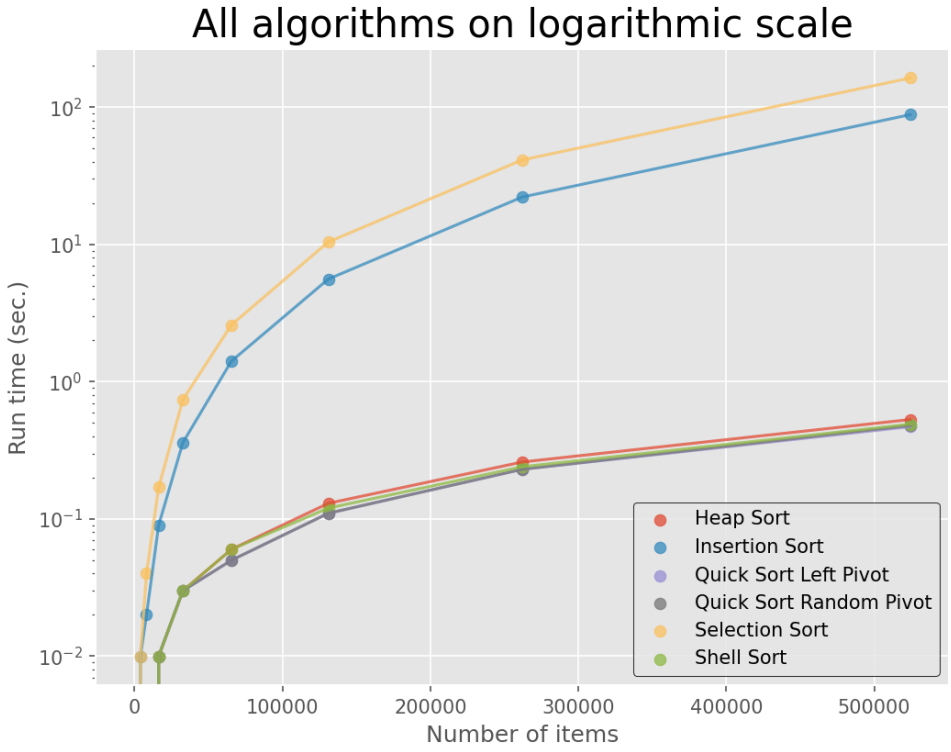
2. Zbiorcze zestawienie czasu wykonania poszczególnych algorytmów

n	HS t[s]	IS t[s]	QSLP t[s]	QSRP t[s]	SLS t[s]	SHS t[s]
4	0.00	0.00	0.00	0.00	0.00	0.00
8	0.00	0.00	0.00	0.00	0.00	0.00
16	0.00	0.00	0.00	0.00	0.00	0.00
32	0.00	0.00	0.00	0.00	0.00	0.00
64	0.00	0.00	0.00	0.00	0.00	0.00
128	0.00	0.00	0.00	0.00	0.00	0.00
256	0.00	0.00	0.00	0.00	0.00	0.00
512	0.00	0.00	0.00	0.00	0.00	0.00
1024	0.00	0.00	0.00	0.00	0.00	0.00
2048	0.00	0.00	0.00	0.00	0.00	0.00
4096	0.00	0.01	0.00	0.00	0.01	0.00
8192	0.00	0.02	0.00	0.00	0.04	0.00
16384	0.01	0.09	0.01	0.01	0.17	0.01
32768	0.03	0.36	0.03	0.03	0.74	0.03
65536	0.06	1.41	0.05	0.05	2.59	0.06
131072	0.13	5.59	0.11	0.11	10.42	0.12
262144	0.26	22.14	0.23	0.23	41.26	0.24
524288	0.53	88.14	0.47	0.48	162.94	0.49

Tabela 1: Zestawienie pomiarów czasu wykonania dla poszczególnych algorytmów. Pomiary są w sekundach. n – długość tablicy, HS – Heap Sort, IS – Insertion Sort, QSLP – Quick Sort Left Pivot, QSRP – Quick Sort Random Pivot, SLS – Selection Sort, SHS – Shell Sort



Wykres 1: Zbiorczy wykres czasu wykonania poszczególnych algorytmów.



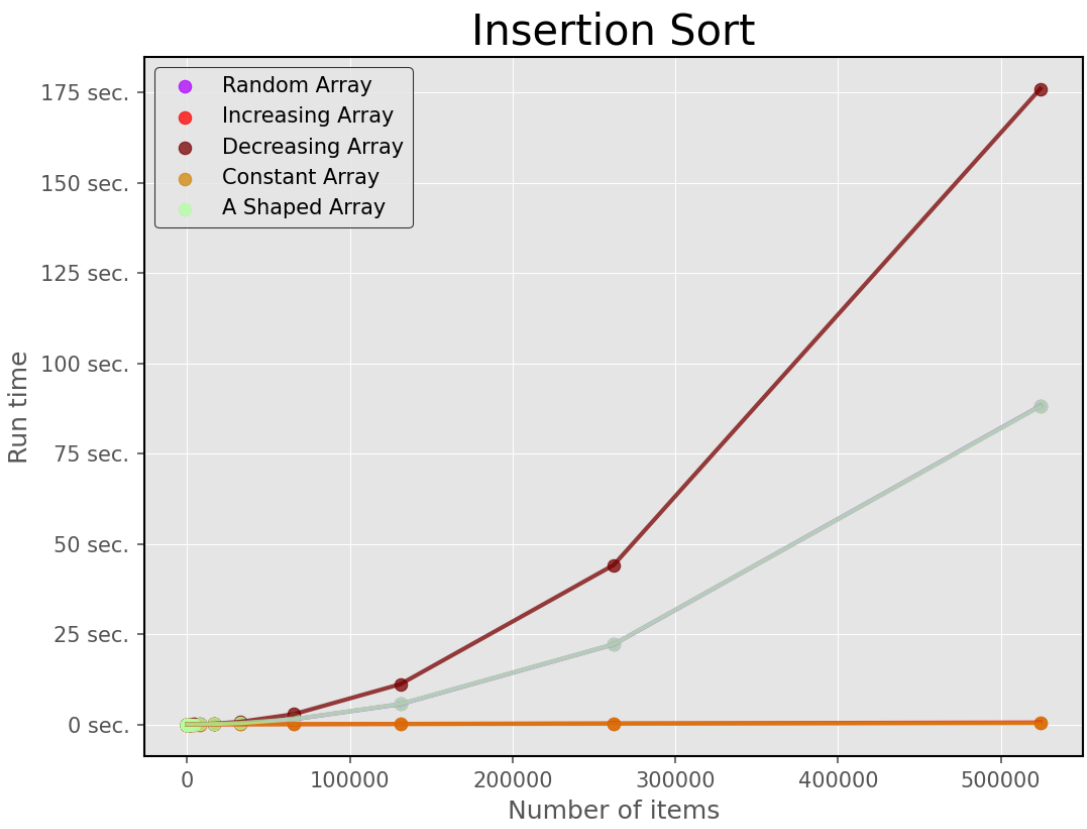
Wykres 2: Zbiorczy wykres czasu wykonania poszczególnych algorytmów na skali logarytmicznej.

3. Złożoność czasowa

Insertion Sort

Złożoność obliczeniowa		
Minimalna	Maksymalna	Średnia
$O(n)$	$O(n^2)$	$O\left(n^2\right)$

Insertion Sort sortuje elementy poprzez iteracyjne "wstawianie" każdego elementu na odpowiednie miejsce w posortowanej już części tablicy.

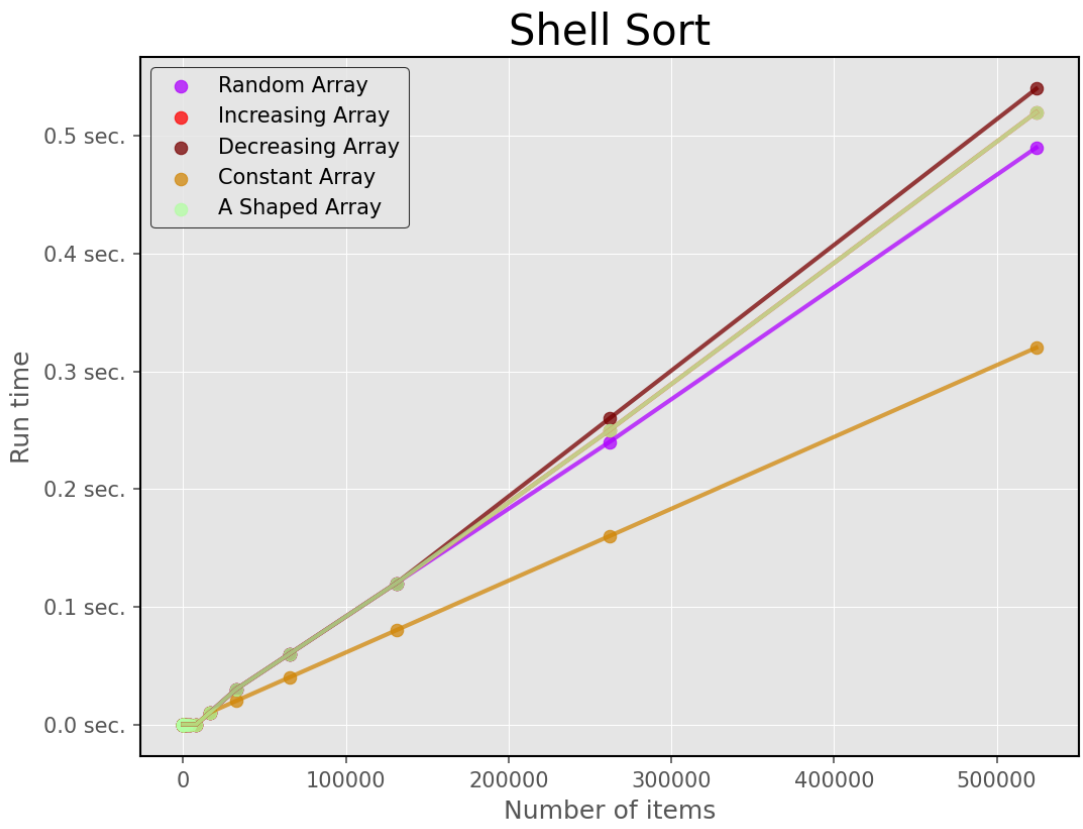


Wykres 3: Złożoność czasowa insertion sort

Shell Sort

Złożoność obliczeniowa		
Minimalna	Maksymalna	Średnia
$O(n\log n)$	$O(n^2)$	$O(n^{\frac{4}{3}})$

Shell Sort jest ulepszoną wersją Insertion Sort, która dzieli listę na mniejsze podlisty zgodnie z przyjętymi wartościami odstępów między sortowanymi elementami, a następnie sortuje je przy użyciu Insertion Sort. Algorytm stopniowo zmniejsza odstępy. W zależności od danego problemu możemy przyjąć różne ciągi odstępów.

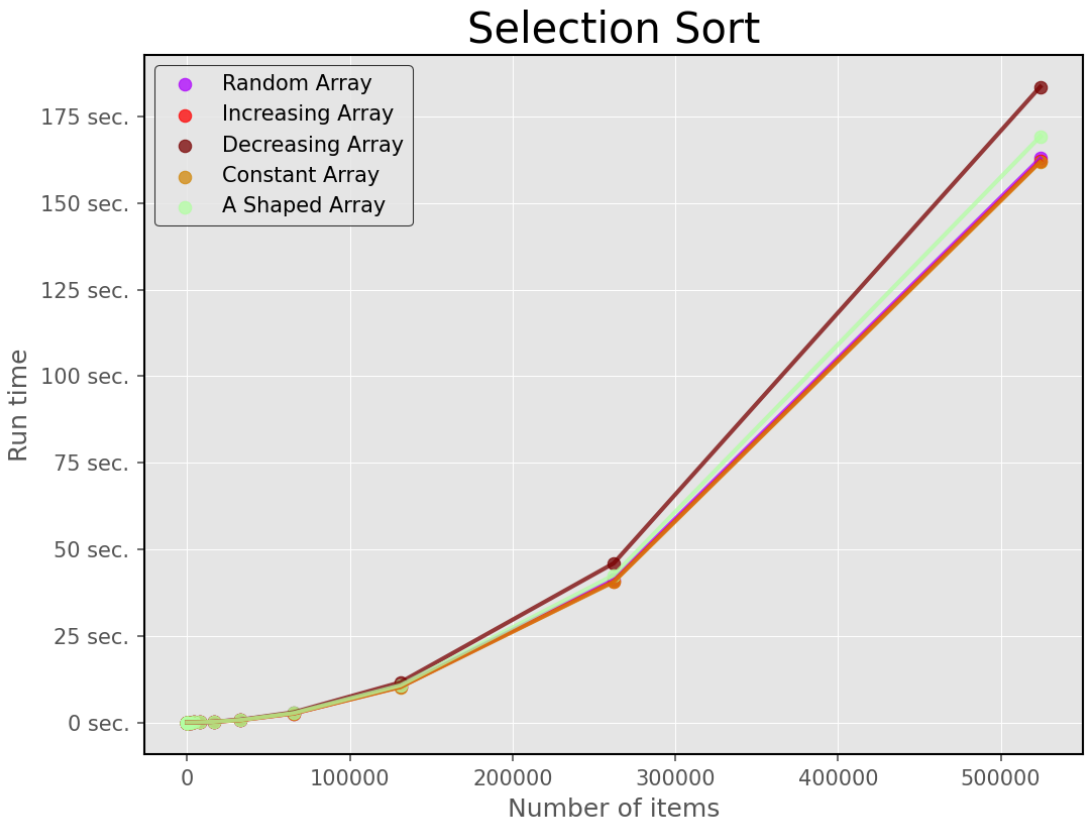


Wykres 4: Złożoność czasowa shell sort

Selection Sort

Złożoność obliczeniowa		
Minimalna	Maksymalna	Średnia
$O(n^2)$	$O(n^2)$	$O\left(n^2\right)$

Selection Sort znajduje najmniejszy element w nieposortowanej części listy i zamienia go z pierwszym nieposortowanym elementem.

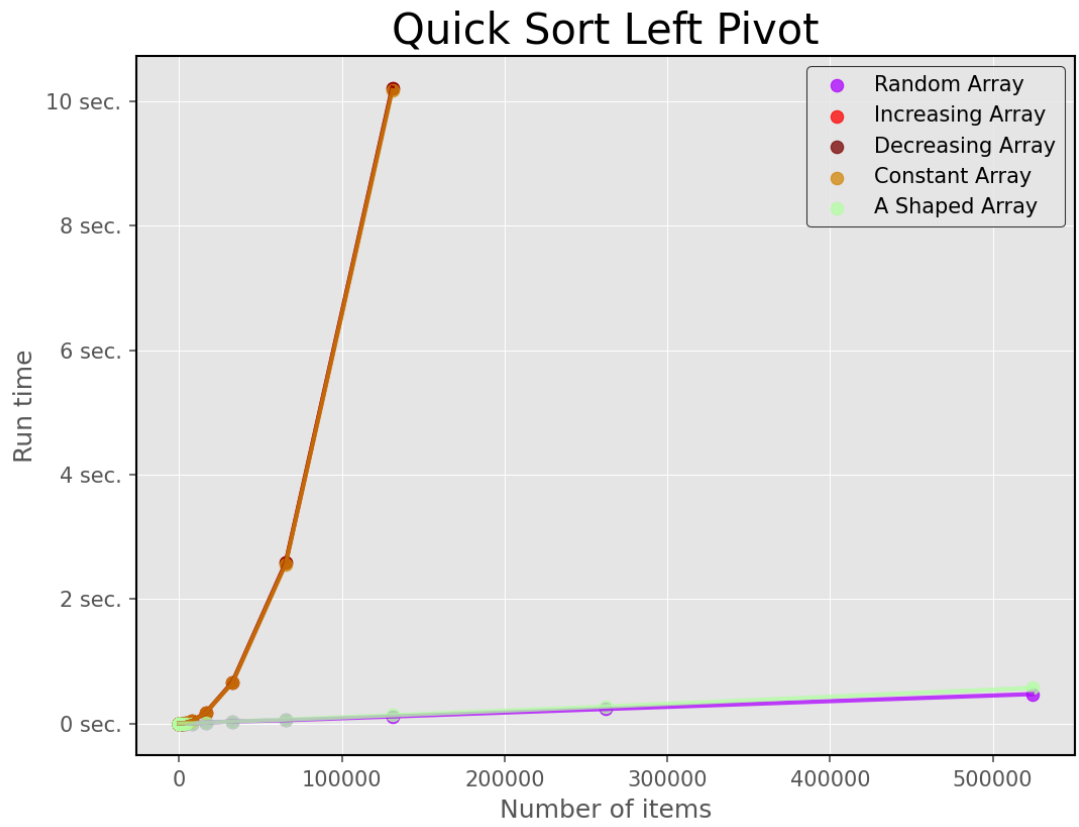


Wykres 5: Złożoność czasowa selection sort

Quick Sort Left Pivot

Złożoność obliczeniowa		
Minimalna	Maksymalna	Średnia
$O(n\log n)$	$O(n^2)$	$O(n\log n)$

Quick Sort to algorytm sortowania dziel i zwyciężaj. Wybiera się element pivotowy, a następnie dzieli się listę na dwa podzbiory - mniejsze od pivota i większe od pivota. Następnie sortuje się rekurencyjnie oba podzbiory. Dla tablicy o długości 262144 oraz 524288 algorytm zwraca błąd „Command terminated by signal 11”.

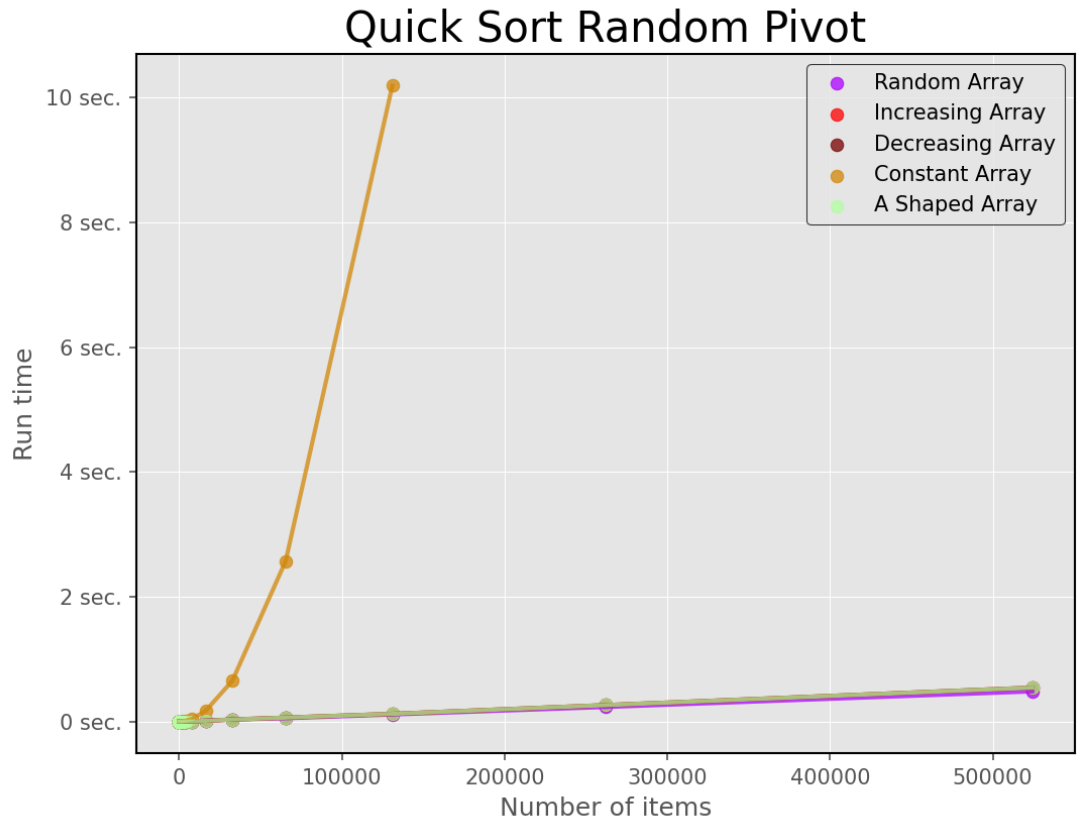


Wykres 6: Złożoność czasowa quick sort left pivot

Quick Sort Random Pivot

Złożoność obliczeniowa		
Minimalna	Maksymalna	Średnia
$O(n\log n)$	$O(n^2)$	$O(n\log n)$

Ten algorytm działa tak samo jak Quick Sort z pivotem na lewo, z wyjątkiem tego, że pivot jest losowo wybierany spośród elementów listy. Dla tablicy o długości 262144 oraz 524288 algorytm zwraca błąd „Command terminated by signal 11”.

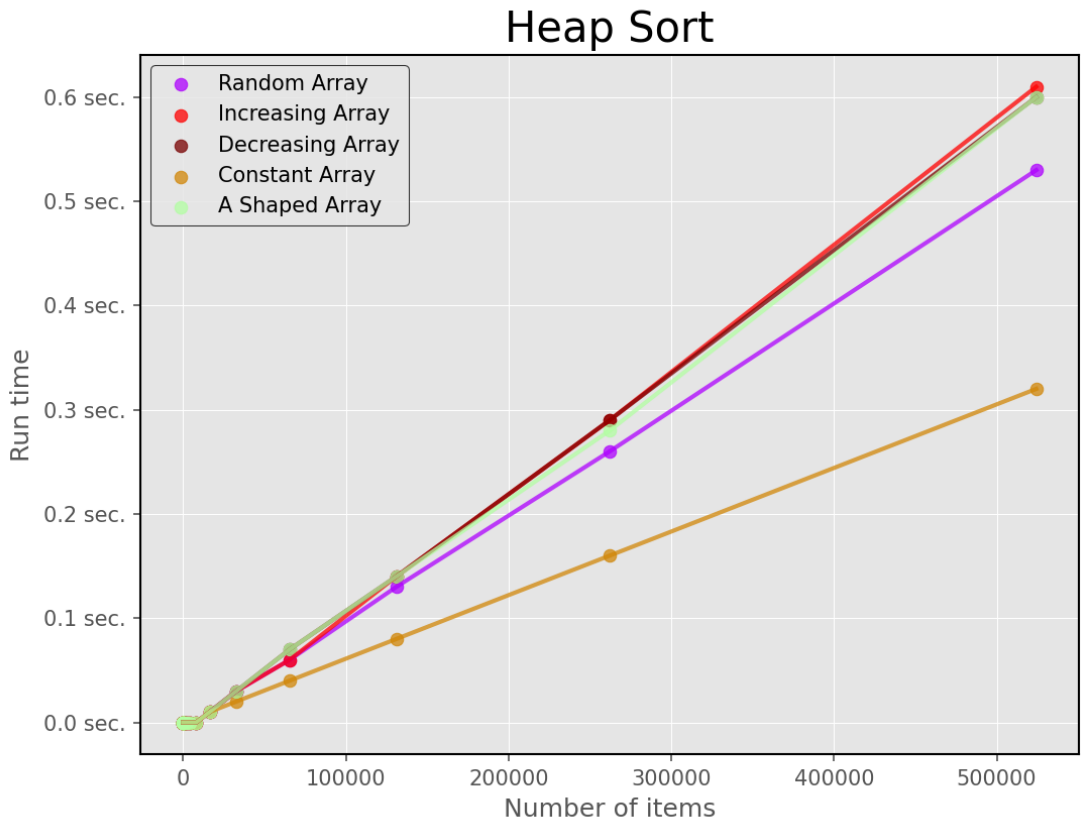


Wykres 7: Złożoność czasowa quick sort random pivot

Heap Sort

Złożoność obliczeniowa		
Minimalna	Maksymalna	Średnia
$O(n\log n)$	$O(n\log n)$	$O(n\log n)$

Heap Sort tworzy maksymalne drzewo kopcowe z danych, a następnie iteracyjnie usuwa największy element z korzenia kopca i przywraca kopiec do jego własności, tworząc posortowaną listę.



Wykres 8: Złożoność czasowa heap sort

4. Podsumowanie
   Wykonując projekt nauczyliśmy się implementacji oraz działania różnych algorytmów sortowania danych. Algorytmy te są szeroko stosowane we wszelkich gałęziach informatyki. Ponadto nauczyliśmy się obsługi Gita oraz działania makefile co z pewnością będziemy mogli wykorzystać w przyszłości.

Analizując wykonane pomiary możemy zauważyć w praktyce przewagę algorytmów szybkich. Wykonane benchmarki zgadzają się z poszczególnymi złożonościami obliczeniowymi algorytmów omawianymi na wykładzie.