PROJEKTOWANIE ALGORYTMÓW I METODY SZTUCZNEJ INTELIGENCJI – PROJEKT 1

27.03.2020
Aleksander Górecki, 249003

Prowadzący zajęcia:

Mgr inż. Marta Emirsajłow
Termin zajęć: Piątek 9:15

1. Wstęp teoretyczny

Celem projektu pierwszego było między innymi zapoznanie się z wybranymi algorytmami sortowania oraz pojęciem złożoności obliczeniowej. Główną częścią projektu było przeprowadzenie analizy efektywności trzech wybranych algorytmów sortowania. Analiza ta opierała się na pomiarze czasu działania algorytmów w zależności od ilości elementów tablicy w zestawie tablic, oraz stopniem jej wstępnego posortowania. Wykorzystano do tego celu zestawy stu tablic, o rozmiarach 10 tyś, 50 tyś, 100 tyś, 500 tyś i 1 milion elementów, oraz zmieniano ich wstępne posortowania od 0 do 99,7% oraz rozważono przypadek pełnego posortowania w przeciwnej kolejności.

2. Badane algorytmy

2.1 Quicksort

Quicksort lub sortowanie szybkie, to jeden z najpopularniejszych algorytmów sortowania. Działa na zasadzie ‘dziel i zwyciężaj’ (tak jak również np. mergesort omówiony poniżej). Popularność jak i nazwę zawdzięcza swojej szybkości i łatwości w zastosowaniu.

Algorytm działa rekurencyjnie: wybrany zostaje element tablicy, tzw. pivot, względem którego tablica zostaje podzielona na dwie części, do pierwszej przenosi się elementy od niego mniejsze, a do drugiej większe. Następnie powtarza się tę operację dla powstających coraz to mniejszych fragmentów.

Średnia złożoność obliczeniowa tego algorytmu to $O(n \cdot logn)$ i zależy tego jaki element zostanie wybrany jako pivot. W przypadku pesymistycznym złożoność obliczeniowa wyniesie $O(n^2)$.

2.2 Merge sort

Merge sort, czyli sortowania przez scalanie to algorytm należący do tej samej rodziny do wcześniej opisywany quicksort. Najczęściej implementowany jest rekurencyjnie, natomiast można również go uprościć i zaimplementować poprzez iterację.

Algorytm polega na dzieleniu tablicy na coraz mniejsze podtablice, aż do uzyskania tablic jednoelementowych, możliwych do porównania, oraz scalanie otrzymywanych posortowanych fragmentów aż do uzyskania całości.

Średnia złożoność obliczeniowa tego algorytmu, jak i również złożoność obliczeniowa w przypadku pesymistycznym to $O(n \cdot logn)$.

2.3 Introsort

Introsort, lub sortowanie introspektywne, to algorytm hybrydowy, wykorzystujący działanie innych algorytmów sortowania w zależności od długości sortowanej tablicy lub jej fragmentu.

Introsort korzysta z funkcjonalności takich algorytmów jak insert sort (sortowanie przez wstawianie), heapsort (sortowanie przez kopcowanie) oraz z wcześniej omawianego quicksorta, a dokładnie z jego funkcji wybierającej element dzielący tablicę. Jednym z parametrów algorytmu jest głębokość rekurencji, zależna od długości tablicy. Parametr ten maleje przy każdym wywołaniu rekurencyjnym oraz od jego wartości zależy jaka własność innego algorytmu sortowania zostanie wykorzystana.

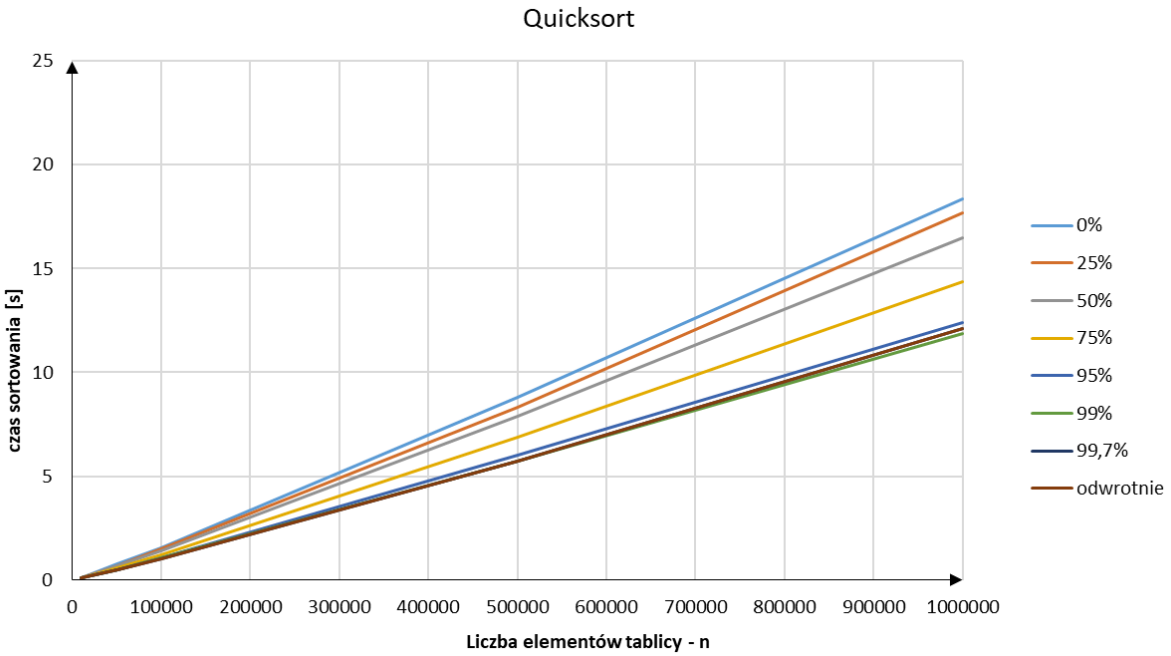
Średnia złożoność obliczeniowa tego algorytmu, jak i również złożoność obliczeniowa w przypadku pesymistycznym to $O(n \cdot logn)$, ponieważ poprzez zabezpieczenie w postaci głębokości rekurencji udaje się uniknąć złożoności $O(n^2)$ quicksorta.

3. Przebieg badań i uzyskane wyniki

Badanie czasu działania algorytmów, umożliwiającego późniejszą analizę ich efektywności, opierało się zliczeniu łącznego czasu jaki dany algorytm poświęcił na posortowanie zestawu 100 tablic, w danym przypadku rozmiaru tablicy oraz jej początkowego posortowania.
Przed pomiarem czasu, każda tablica w zestawie wypełniana była psuedolosowymi liczbami całkowitymi oraz wstępnie sortowana do konkretnego progu przy pomocy std::sort z bliblioteki STL, w celu gwarancji poprawności takiego zestawu. Pomiar czasu rozpoczynał się przed wejściem do pętli wykonującej sortowanie dla każdej tablicy w zestawie, i kończył tuż po jej opuszczeniu, następnie zapisywany był do pliku w milisekundach (dla lepszej czytelności wykresy i tabele wyrażone w sekundach)

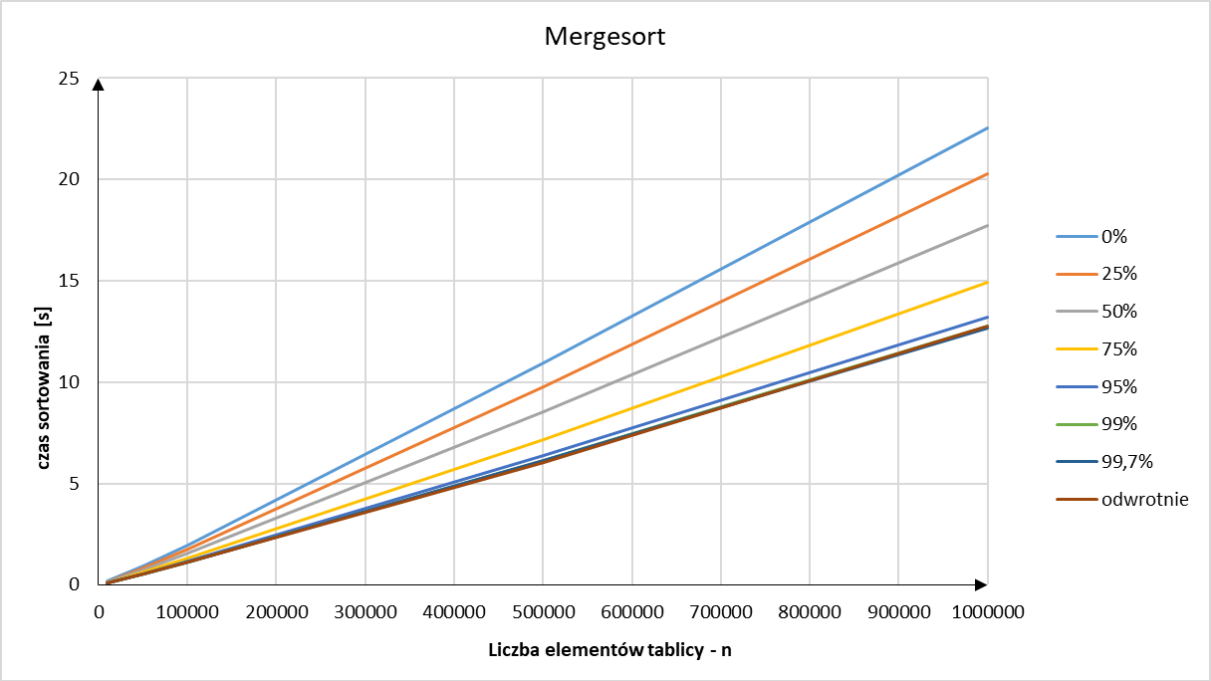
Poniższe wykresy przedstawiają zależność łącznego czasu posortowania zestawu 100 tablic od rozmiaru tablicy w zestawie, a serie odnoszą się do różnych stopni wstępnego posortowania.
Natomiast tabele łączny czas sortowania zestawu 100 tablic w danej konfiguracji rozmiar – wstępne posortowanie.

3.1 Quicksort



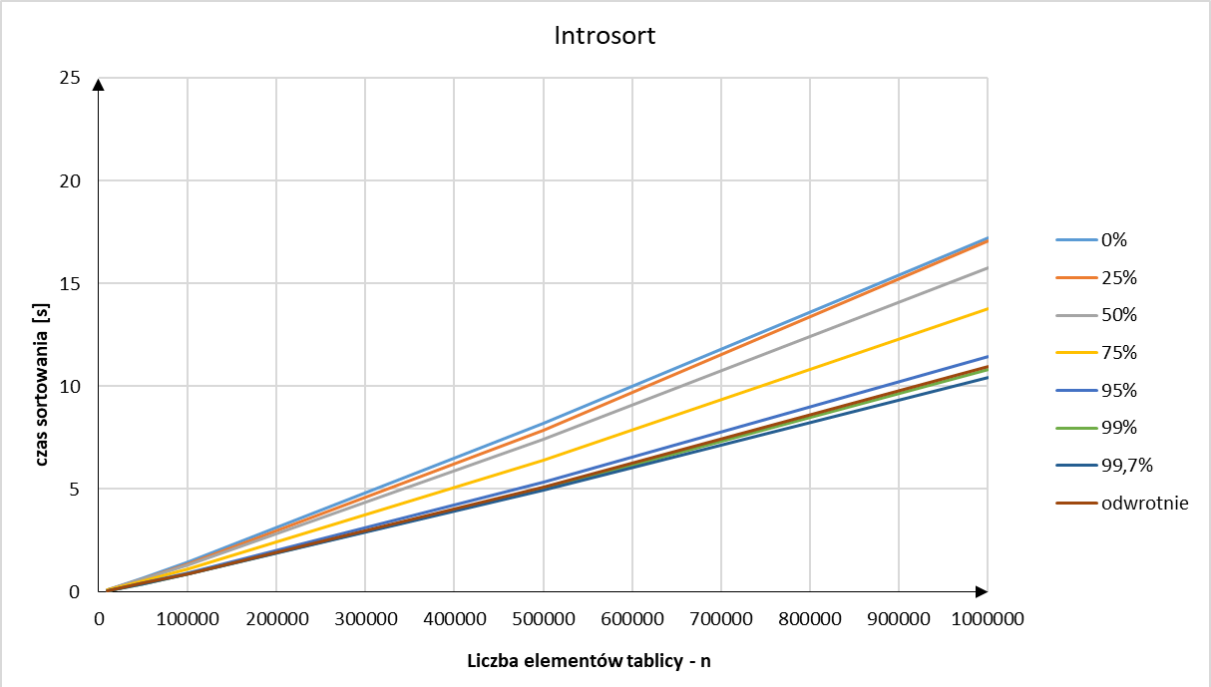
Quicksort		Rozmiar tablicy w zestawie (ilość elementów - n)				
		10000	50000	100000	500000	1000000
Wstępne posortowanie tablicy	0%	0,132	0,760	1,552	8,785	18,338
	25%	0,121	0,695	1,489	8,326	17,682
	50%	0,115	0,683	1,428	7,906	16,472
	75%	0,102	0,587	1,225	6,883	14,384
	95%	0,089	0,516	1,075	5,992	12,388
	99%	0,086	0,490	1,050	5,705	11,861
	99,7%	0,087	0,494	1,034	5,672	11,674
	100% (odwrotnie)	0,085	0,483	1,020	5,738	12,135

3.2 Merge sort



Mergesort		Rozmiar tablicy w zestawie (ilość elementów - n)				
		10000	50000	100000	500000	1000000
Wstępne posortowanie tablicy	0%	0,168	0,937	1,952	10,926	22,539
	25%	0,147	0,845	1,760	9,792	20,268
	50%	0,132	0,739	1,556	8,555	17,739
	75%	0,113	0,633	1,314	7,173	14,939
	95%	0,100	0,554	1,164	6,364	13,196
	99%	0,101	0,542	1,139	6,139	12,749
	99,7%	0,097	0,543	1,130	6,122	12,671
	100% (odwrotnie)	0,098	0,536	1,106	6,049	12,748

3.3 Introsort



Introsort		Rozmiar tablicy w zestawie (ilość elementów - n)				
		10000	50000	100000	500000	1000000
Wstępne posortowanie tablicy	0%	0,112	0,681	1,435	8,197	17,212
	25%	0,108	0,65	1,37	7,866	17,073
	50%	0,105	0,619	1,307	7,415	15,791
	75%	0,089	0,528	1,125	6,42	13,805
	95%	0,073	0,439	0,94	5,35	11,437
	99%	0,069	0,408	0,866	5,019	10,797
	99,7%	0,073	0,41	0,866	4,949	10,455
	100% (odwrotnie)	0,072	0,431	0,89	5,093	10,946

4. Wnioski i uwagi

- Średnia złożoność obliczeniowa każdego z badanych algorytmów to $O(n \cdot logn)$ i otrzymane krzywe na wykresach są tego zapisu zbliżone.
- Duże podobieństwo do funkcji liniowej może być spowodowane korzystaniem z jedynie 5 punktów pomiarowych, przez co niemożliwe jest zaobserwowanie wpływu logarytmu na kształt wykresu.

- Średnio najwolniejszym algorytmem okazał się mege sort, następnie quicksort, a najszybszym introsort. Nie jest on jednak znacząco szybszy od quicksorta, co może być spowodowane niekoniecznie wydajną implementacją algorytmów wchodzących w jego skład (np. tworzenie tablic pomocniczych w rekurencyjnie wywoływanym heapsorcie)

5. *Literatura*

https://pl.wikipedia.org/wiki/Sortowanie_szybkie
https://en.wikipedia.org/wiki/Quicksort
https://pl.wikipedia.org/wiki/Sortowanie_przez_scalanie
https://en.wikipedia.org/wiki/Merge_sort
https://pl.wikipedia.org/wiki/Sortowanie_introspektywne
https://en.wikipedia.org/wiki/Introsort
https://www.geeksforgeeks.org/heap-sort/
https://www.geeksforgeeks.org/merge-sort/