Projektowanie Algorytmów i Metody Sztucznej Inteligencji

Projekt I

Paweł Prucnal 248937

26.03.2020r.

Piątek 13^{15} mgr inż. Marta Emirsajłow

1. Wprowadzenie

Celem projektu było zapoznanie się z popularnymi algorytmami sortowania stosowanymi w informatyce. Należało zrozumieć, opracować, a następnie zaimplementować trzy wybrane algorytmy sortowania. Następnie, w celu ich głębszego poznania, przeprowadzić szereg testów uwidaczniających ich wady oraz zalety. Pozwoliło to zrozumieć, że żaden ze znanych nam algorytmów nie jest idealny oraz że żaden z nich nie jest najszybszy w każdym przypadku.

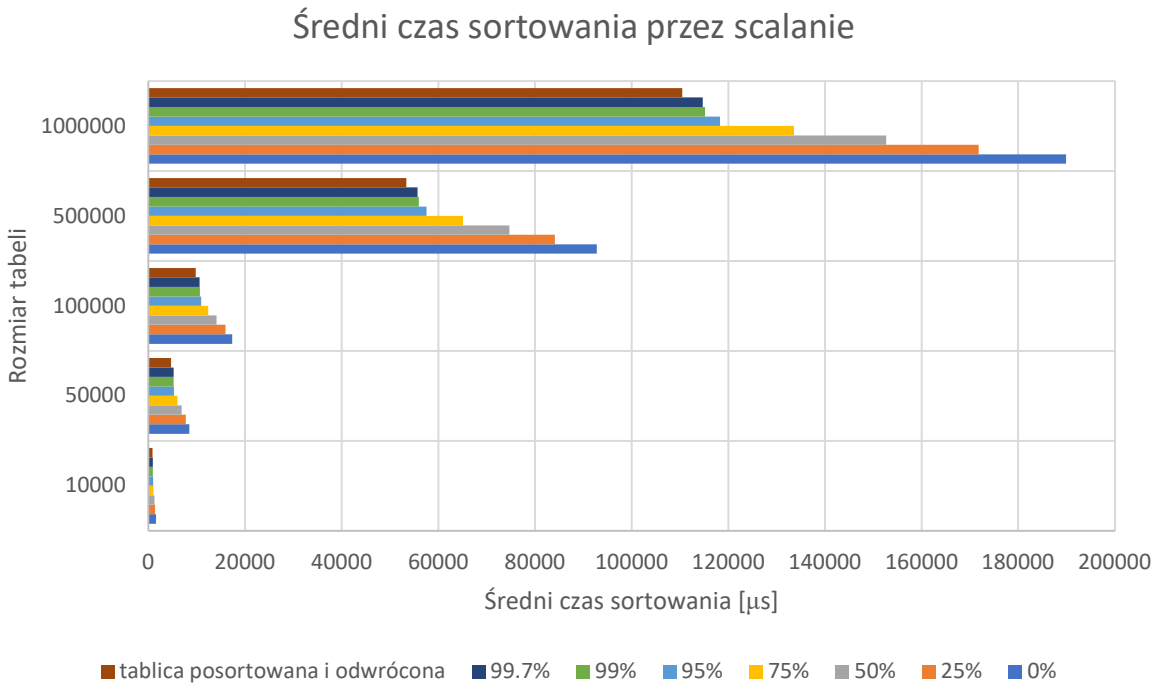
2. Przebieg eksperymentów

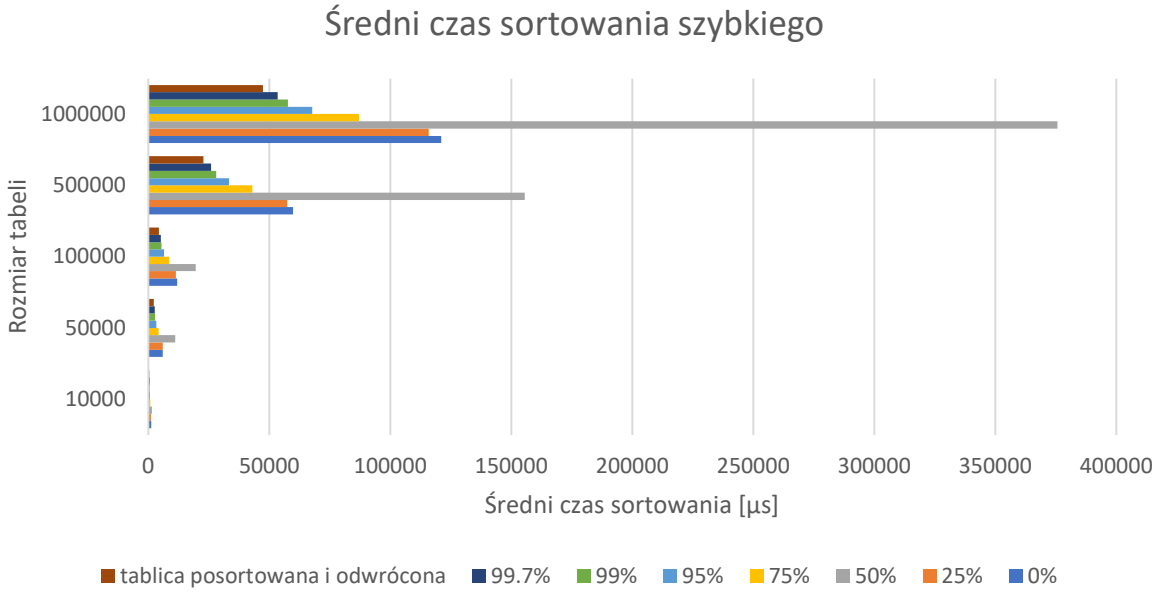
Testy polegały na zmierzeniu czasu sortowania losowo wygenerowanych tablic o różnych rozmiarach dla każdego z badanych algorytmów. By tego dokonać, zaimplementowano je w prostym programie w języku C++, którego zadaniem było wielokrotne deklarowanie, wypełnianie oraz sortowanie tablic. Przed rozpoczęciem sortowania oraz natychmiast po jego zakończeniu pobierał on dokładny czas z komputera przy pomocy biblioteki ctime. Następnie obliczał on różnicę między tymi dwoma punktami czasu w milisekundach i zapisywał ją do pliku. Proces ten wykonywany był sto razy dla każdej z kombinacji warunków. Należało bowiem sprawdzić jak algorytm poradzi sobie z tablicami o rozmiarach 10000, 50000, 100000, 500000 oraz 1000000 elementów, jak i tablicami nieposortowanymi, posortowanymi w 25%, 50%, 75%, 95%, 99%, 99,7% oraz posortowanymi, a następnie odwróconymi. Przy trzech badanych algorytmach daje to razem 120 kombinacji, dla każdej z których zostało przeprowadzone 100 pomiarów. Wyniki spisane w plikach zaimportowano do programu Microsoft Excel w celu sporządzenia wykresów.

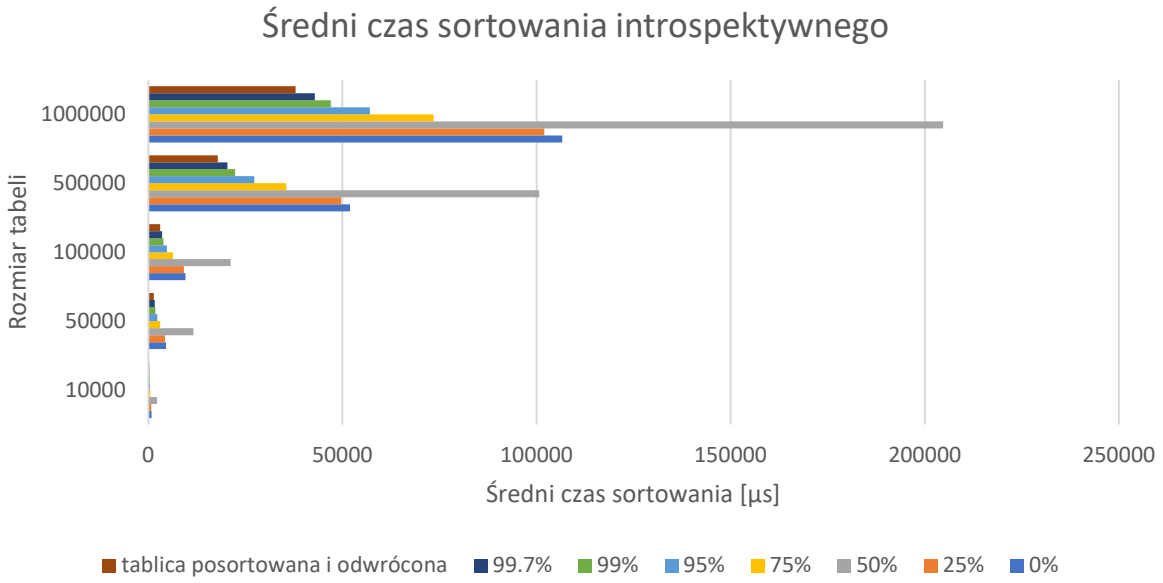
Przeprowadzono testy na algorytmach:

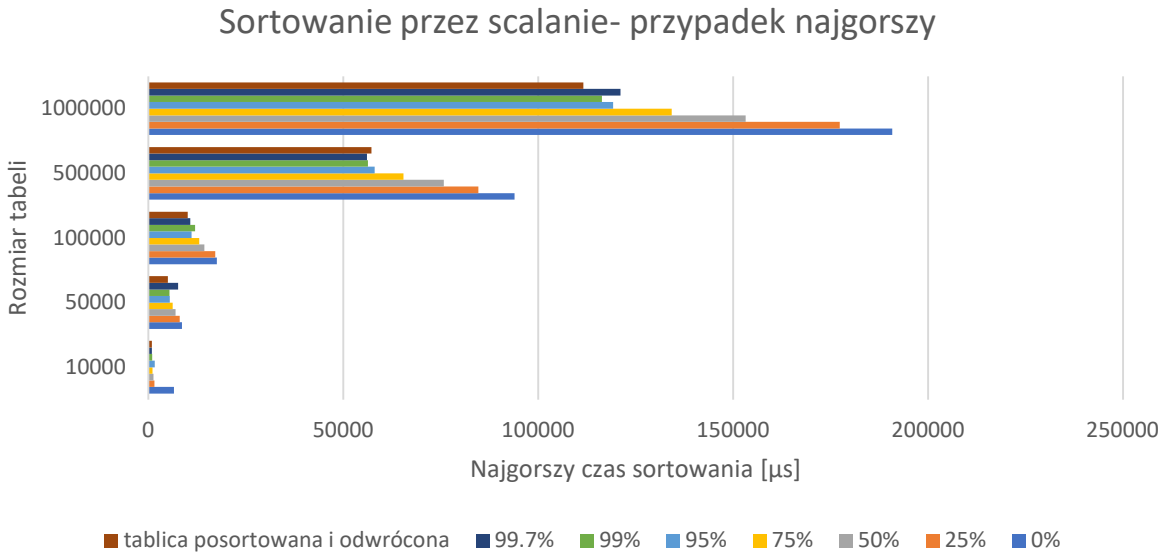
- Sortowania przez scalanie (mergesort),
- Sortowania szybkiego (quicksort),
- Sortowania introspektywnego (introsort).

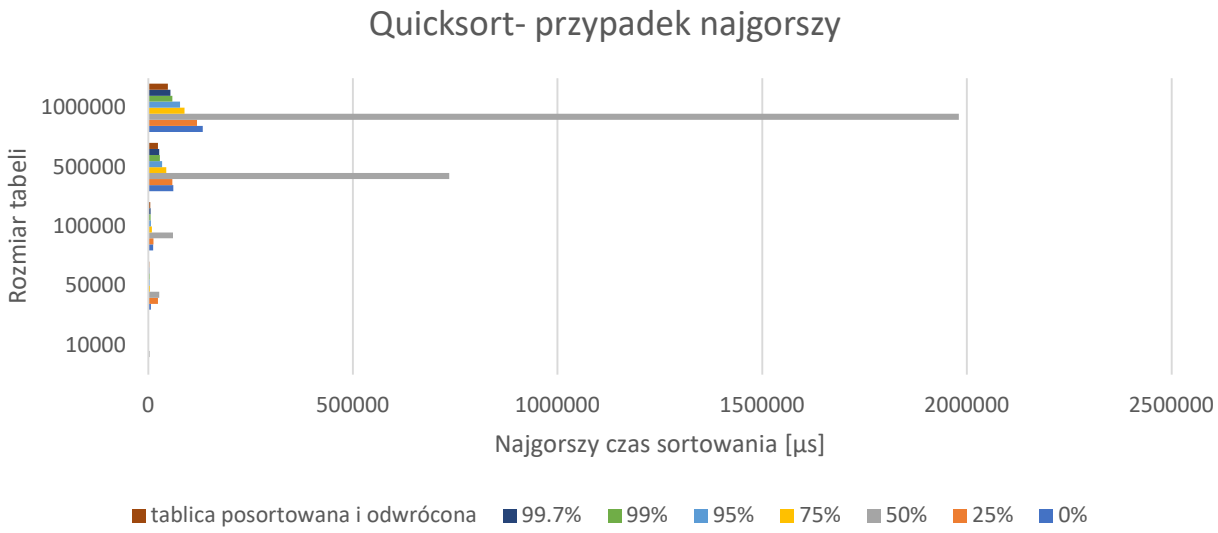
Otrzymano następujące wyniki pomiarów:

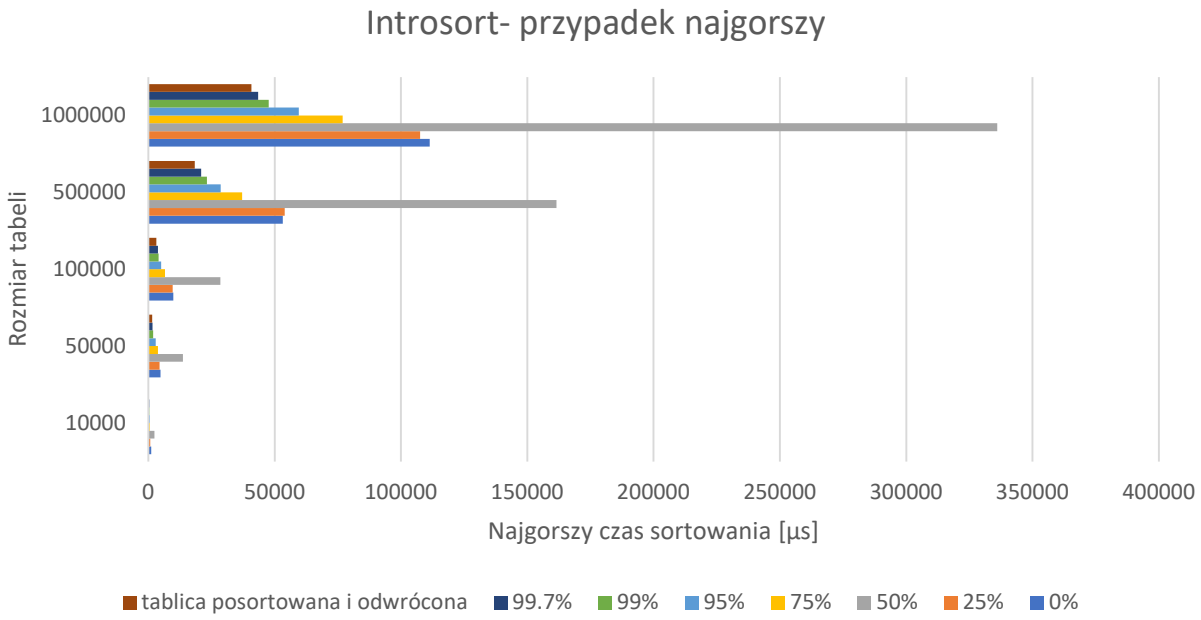





Średni czas sortowania introspektywnego
Rozmiar tabeli
1000000
500000
100000
50000
10000
0
50000
100000
150000
200000
250000
Średni czas sortowania [µs]
tablica posortowana i odwrócona
99.7%
99%
95%
75%
50%
25%
0%


Sortowanie przez scalanie- przypadek najgorszy
Rozmiar tabeli
1000000
500000
100000
50000
10000
0
50000
100000
150000
200000
250000
Najgorszy czas sortowania [µs]
tablica posortowana i odwrócona
99.7%
99%
95%
75%
50%
25%
0%


Quicksort- przypadek najgorszy
Rozmiar tabeli
1000000
500000
100000
50000
10000
0
500000
1000000
1500000
2000000
2500000
Najgorszy czas sortowania [µs]
tablica posortowana i odwrócona
99.7%
99%
95%
75%
50%
25%
0%



	PRZYPADEK ŚREDNI														
	MERGESORT					QUICKSORT					INTROSORT				
	10000	50000	100000	500000	1000000	10000	50000	100000	500000	1000000	10000	50000	100000	500000	1000000
0%	1626	8482	17390	92792	189912	1237	5967	11875	59852	121116	842	4566	9556	51947	106658
25%	1388	7750	15952	84115	171840	1108	5941	11431	57398	115871	727	4315	9144	49629	101968
50%	1235	6890	14133	74691	152677	1516	11089	19575	155528	375746	2246	11616	21166	100722	204749
75%	1076	6037	12396	65142	133615	839	4294	8561	43072	87086	503	3023	6389	35507	73481
95%	1019	5340	10930	57535	118282	653	3342	6511	33292	67710	431	2299	4831	27323	57074
99%	936	5226	10656	55977	115174	558	2847	5502	28021	57693	334	1850	3919	22384	47006
99.7%	929	5254	10588	55742	114729	531	2663	5145	25946	53508	291	1674	3593	20392	42869
tablica posortowana i odwrócona	858	4739	9782	53387	110514	378	2213	4325	22728	47314	252	1413	3074	17899	37951

	PRZYPADEK NAJGORSZY														
	MERGESORT					QUICKSORT					INTROSORT				
	10000	50000	100000	500000	1000000	10000	50000	100000	500000	1000000	10000	50000	100000	500000	1000000
0%	6558	8658	17574	93959	190846	1536	6305	12204	60945	133409	1164	4797	9926	53182	111410
25%	1587	8048	17193	84676	177367	1146	23576	12293	58754	118810	753	4505	9672	54004	107578
50%	1298	6984	14368	75800	153255	3180	26750	60586	735658	1980399	2364	13754	28538	161559	335997
75%	1089	6240	13027	65486	134235	860	4378	8821	43862	88811	522	3758	6596	37179	76954
95%	1637	5517	11137	58052	119219	674	3442	6651	33930	77310	579	2983	5083	28616	59556
99%	1019	5419	12023	56315	116348	569	3029	5821	28477	58643	355	1969	4041	23164	47593
99.7%	943	7628	10800	56088	121130	543	2736	5310	26739	53998	351	1724	3853	20895	43432
tablica posortowana i odwrócona	874	5035	10096	57220	111620	386	2306	4407	23786	47907	315	1478	3151	18339	40849

3. Omówienie badanych algorytmów

Badane algorytmy znane są ze swojej wysokiej wydajności, zwłaszcza sortowanie szybkie oraz introspektywne.

Sortowanie przez scalanie polega na rozbijaniu tablicy na dwie części do momentu uzyskania zbioru jednoelementowego, który jest zawsze posortowany. Następnie rozbity zbiór jest scalany już z posortowanymi elementami. Dzięki temu problem posortowania tablicy rozbijany jest na szereg prostszych problemów. Złożoność tego algorytmu zarówno dla średniego jak i najgorszego przypadku powinna wynosić O(n log n).

Sortowanie szybkie (quicksort) polega na wybraniu jednego elementu z tablicy (punktu odniesienia) a następnie porównaniu go z każdym elementem z jego

lewej oraz prawej strony. Gdy znaleziona zostanie para elementów, w której element z lewej strony jest większy od punktu odniesienia oraz element z prawej strony jest mniejszy od punktu odniesienia, zamieniane są one miejscami. Gdy analizowane indeksy przekroczą punkt odniesienia, utworzą one 3 tablice, z których środkowa jest już posortowana. Na pozostałych wykonywane są ponownie powyższe operacje. Quicksort charakteryzuje się złożonością obliczeniową rzędu O(*n* log *n*) dla przypadku średniego oraz O(n^2) dla przypadku najgorszego.

Sortowanie introspektywne jest hybrydą sortowania szybkiego, sortowania przez kopcowanie (heapsort) oraz sortowania przez wstawianie (insertionsort). Poprzez „przełączanie" na algorytm sortowania przez kopcowanie gdy głębokość rekursji przekroczy maksymalną wartość, pozwala on ominąć najgorszy przypadek sortowania szybkiego O(n^2). Stosuje on również sortowanie przez wstawianie dla niedużych zbiorów danych (np. 16 elementów), gdzie quicksort poradziłby sobie znacznie gorzej. Sortowanie introspektywne charakteryzuje się złożonością obliczeniową rzędu O(*n* log *n*) zarówno dla przypadku najgorszego, jak i średniego.

4. Podsumowanie wyników i wnioski

Powyższe wykresy świetnie obrazują najgorszy przypadek sortowania szybkiego oraz introspektywnego. Podanie na wejściu tych algorytmów tablicy posortowanej do połowy znacznie wydłuża czas sortowania. Porównując te pomiary do reszty (np. 25%) można założyć, iż w przypadku quicksorta ten pomiar przypomina bardziej funkcję wykładniczą, gdy reszta- funkcję logarytmiczną. Zastosowanie sortowania introspektywnego zmniejszyło średni czas w tym przypadku niemal o połowę. Najgorszy czas okazał się natomiast około dziesięć razy krótszy w porównaniu do najgorszego czasu zmierzonego dla sortowania szybkiego.

5. Literatura

- www.geeksforgeeks.org,
- www.wikipedia.org,
- www.youtube.com, w szczególności kanały kakaboc oraz Coding Blocks,
- www.algorytm.edu.pl.