SPRAWOZDANIE

PROJEKT 1

ALGORYTMY SORTOWANIA

Imię i nazwisko	Maciej Pająk
Nr indeksu	241632
Data	10.04.2019 r.
Nazwa kursu	Programowanie algorytmów i metody sztucznej inteligencji
Dane prowadzącego	Dr inż. Łukasz Jeleń
Termin zajęć	ŚR 11:15-13:00

Wprowadzenie

Celem projektu było przeprowadzenie badań nad złożonością obliczeniową zadanych algorytmów sortowania. Badane algorytmy to: quicksort, mergesort oraz introsort. Eksperymenty przeprowadzono dla tablic o różnych rozmiarach i różnym stopniu uporządkowania.

Opis algorytmów

QuickSort

Na początku wybierany jest tzw. element osiowy, w tym przypadku element środkowy. Następnie tablica jest porządkowana w taki sposób by w lewej części były elementy mniejsze, a po prawej większe niż element osiowy. Powtarzamy sortowanie dla obu części. Sortownie kończy się po uzyskaniu części jednoelementowych. Złożoność obliczeniowa tego algorytmu nie jest jednoznaczna, gdyż element osiowy jest losowy. W najgorszym przypadku złożoność obliczeniowa wynosi $O(n^2)$ np. jeśli będziemy ciągle trafiać w najmniejszy lub największy element tablicy, dla średniego przypadku wynosi O(n log n).

W algorytmie w kodzie dla n-elementowej tablicy mamy n porównań, n/2 dekrementacji i n/2 inkrementacji oraz operację zamiany elementów miejscami, która zawiera 3 podstawienia. W najgorszym przypadku dostajemy tylko jedną zamianę miejsc, ale szereg ma n elementów: $2n + 3 + 2(n-1) + 3 + \ldots + 2 + 3 = = n^2 + 2n$, a więc kwadratowa złożoność obliczeniowa. W przypadku posortowanej tablicy zawsze dzielimy tablicę na pół i n/2 zamian co daje szereg: $2^0 \cdot \left(2\frac{n}{2^0} + 3\frac{n}{2^1}\right) + 2^1 \cdot \left(2\frac{n}{2^1} + 3\frac{n}{2^2}\right) + \cdots = 3{,}5n\ logn.$

MergeSort

Na początku dzielimy tablicę w połowie na dwie podtablice. Czynność powtarzamy do uzyskania tablic jednoelementowych. Następnie obie tablice scalamy porównując elementy, wykorzystując to, że „wracając" obie tablice już są posortowane. Złożoność obliczeniowa tego algorytmu wynosi O(n log n).

W algorytmie w kodzie mamy liczbę porównań zależną od przypadku i 2n przepisów do nowej tablicy. Przyjmijmy, że $n = 2^k$. W najgorszym przypadku dla scalenia dwóch n-elementowych tablic trzeba $2n - 1$ porównań oraz $2n$ przepisów. A więc szereg wygląda tak: $\frac{n}{2}(2^1 - 1 + 2^1) + \frac{n}{2^2}(2 \cdot 2^2 - 1) + \cdots + \frac{n}{2^{k-1}}(2 \cdot 2^{k-1} - 1) = 2n - \frac{n}{2} + 2n - \frac{n}{4} + \cdots + 2n - \frac{n}{2^{k-1}} = 2nlogn - 3n$. Podobnie jest dla najlepszego przypadku kiedy mamy tylko n porównań, czyli pierwsza tablica jest w całości mniejsza niż druga, wtedy złaożoność obliczeniowa wynosi $n(logn - 1) + n = nlogn.$

IntroSort

Ten algorytm składa się z dwóch algorytmów: quicksort oraz heapsort. Na początku wyliczana jest stała M=2log(n), gdzie n to liczba elementów do posortowania. Następnie dzielimy tablicę przy pomocy algorytmu quicksort zmniejszając stałą M o 1 dla wszystkich podtablic. Czynność tę powtarzamy dopóki wartość M nie osiągnie zero. Następnie uzyskane podtablice sortujemy algorytmem heapsort. Złożoność obliczeniowa tego algorytmu wynosi O(n log n) ze względu na połączenie obu algorytmów, co zabezpiecza algorytm QuickSort przed najgorszym przypadkiem.

W tym algorytmie złożoność obliczeniowa przyjmuje zawsze $nlogn$ (zwłaszcza dla wielkich tablic) gdyż algorytm HeapSort, który ma taką złożoność stosunkowo szybko działać kiedy QuickSort dostanie gorsze przypadki.

Eksperyment

Do badania algorytmów użyto 100 tablic o rozmiarach 10 000, 50 000, 100 000, 500 000, 1 000 000, w których elementy były:

- losowe,

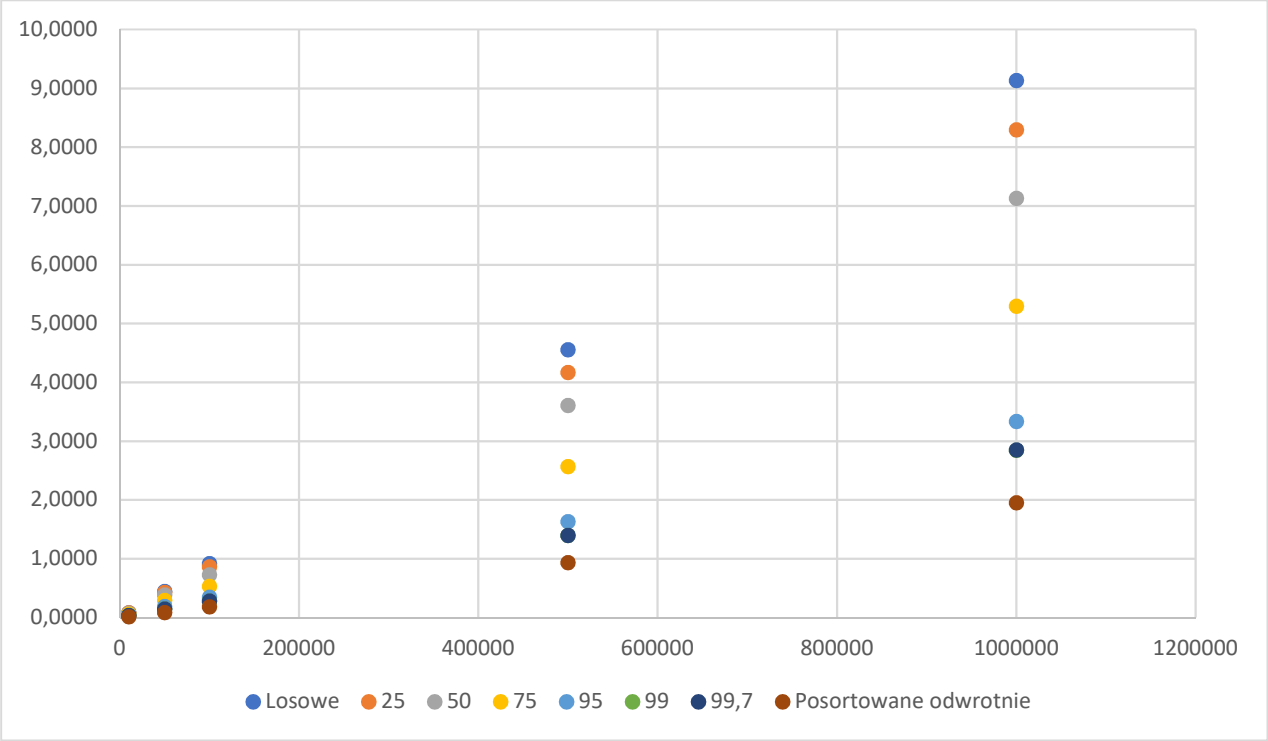
- częściowo posortowane (25%, 50%, 75%, 95%, 99%, 99,7%)

- posortowane odwrotnie

QuickSort

Tab. 1. Wyniki sortownia dla algorytmu QuickSort

QuickSort		10000	50000	100000	500000	1000000
Losowe		0,0816	0,4437	0,9204	4,5545	9,1347
Posortowane częściowo	25	0,0786	0,4270	0,8662	4,1664	8,2949
	50	0,0700	0,3868	0,7263	3,6090	7,1326
	75	0,0577	0,2901	0,5307	2,5664	5,2971
	95	0,0362	0,1913	0,3466	1,6332	3,3355
	99	0,0384	0,1470	0,2822	1,3964	2,8432
	99,7	0,0343	0,1462	0,2816	1,3996	2,8497
Posortowane odwrotnie		0,0147	0,0881	0,1828	0,9325	1,9525

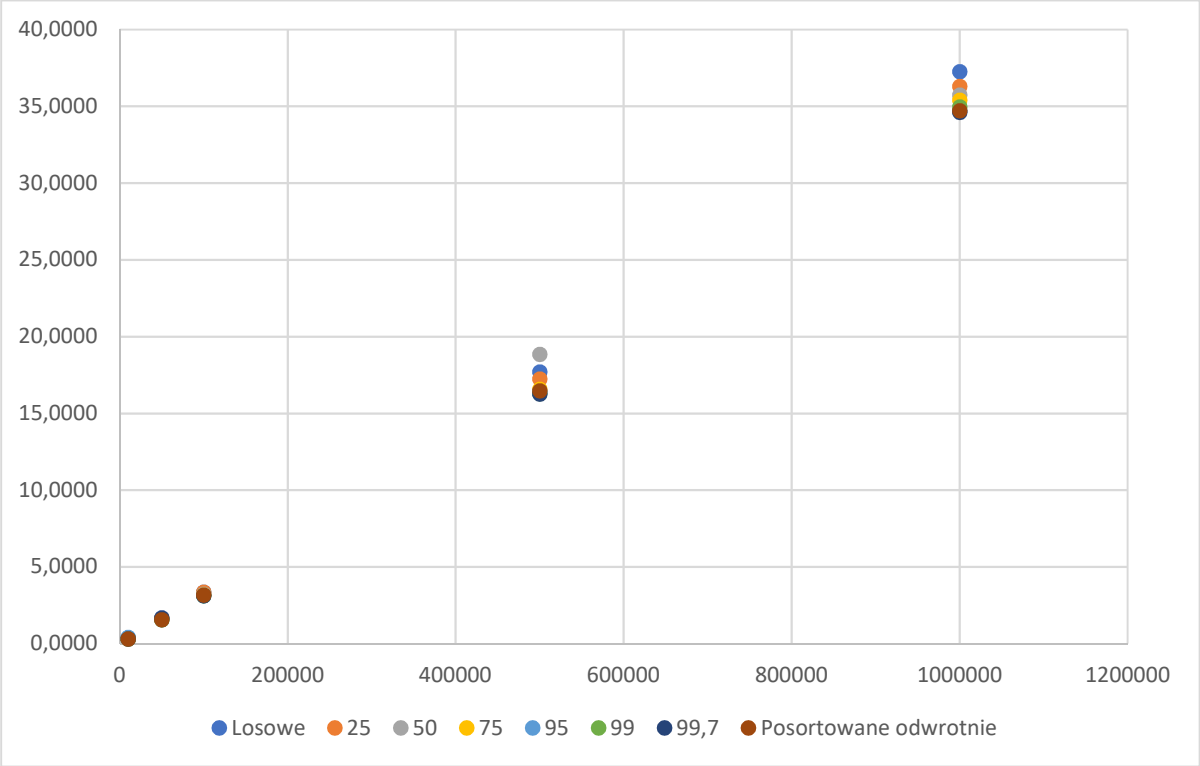


Rys. 1. Prezentacja wyników na wykresie dla algorytmu QuickSort

MergeSort

Tab. 2. Wyniki sortownia dla algorytmu MergeSort

MergeSort		10000	50000	100000	500000	1000000
Losowe		0,3811	1,6783	3,3487	17,6971	37,2645
Posortowane częściowo	25	0,3256	1,6424	3,3561	17,2360	36,3108
	50	0,3305	1,6113	3,2338	18,8397	35,7495
	75	0,3119	1,5752	3,1716	16,5720	35,3970
	95	0,4116	1,5551	3,1317	16,4324	34,7253
	99	0,3150	1,5541	3,1225	16,2916	34,9619
	99,7	0,3061	1,6599	3,1278	16,2726	34,6170
Posortowane odwrotnie		0,3070	1,5676	3,1611	16,4616	34,7239

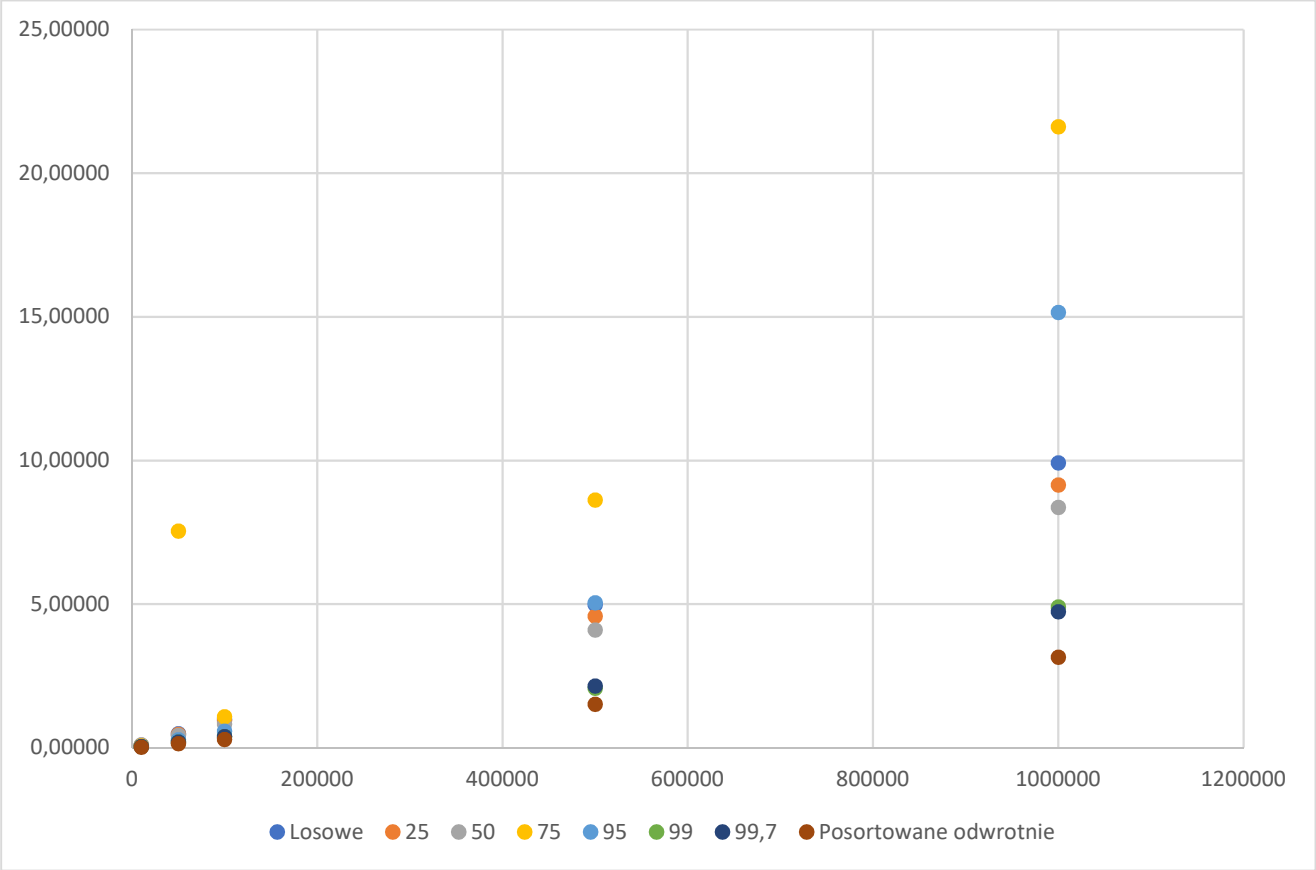


Rys. 2. Prezentacja wyników na wykresie dla algorytmu MergeSort

IntroSort

Tab. 3. Wyniki sortownia dla algorytmu IntroSort

IntroSort		10000	50000	100000	500000	1000000
Losowe		0,08807	0,48347	0,99091	4,98250	9,91587
Posortowane częściowo	25	0,08449	0,46825	0,94143	4,58456	9,14929
	50	0,10267	0,42831	0,82572	4,09876	8,36469
	75	0,06923	7,53835	1,07877	8,62960	21,62350
	95	0,04535	0,27796	0,57649	5,04539	15,16100
	99	0,03974	0,19455	0,38344	2,07554	4,89908
	99,7	0,04287	0,19483	0,38601	2,14806	4,73770
Posortowane odwrotnie		0,02613	0,13905	0,28586	1,51325	3,15667



Rys. 3. Prezentacja wyników na wykresie dla algorytmu IntroSort

Wnioski

Algorytm QuickSort rzeczywiście jest bardzo zależny od podanych danych. Sortowanie tablic już posortowanych lub prawie posortowanych jest kilkukrotnie szybsze niż z elementami losowymi, ze względu na wybór elementu środkowego jako osiowego. Podobnie jest w przypadku algorytmu IntroSort, lecz ze względu na algorytm HeapSort jest on prawie 2-krotnie wolniejszy. Jednakże dla innego wyboru elementu osiowego np. pierwszego elgorytm ten byłby dużo szybszy niż QuickSort. Z nie znanych dla mnie przyczyn algorytm IntroSort tablice posortowane już w 75% sortował dłużej niż pozostałe zwłaszcza dla 50 000 elementów. Algorytm MergeSort jest najwolniejszy z testowanych trzech, jednakże dla różnie posortowanych już danych na wejściu czasy są podobne, jedynie małe odbiegnięcie od tej zależności stanowi pomiar dla tablic posortowanych w 50% dla 500 000 elementów.

Literatura

https://pl.wikipedia.org/wiki/Sortowanie_przez_scalanie

https://pl.wikipedia.org/wiki/Sortowanie_szybkie

https://en.wikipedia.org/wiki/Quicksort

https://pl.wikipedia.org/wiki/Sortowanie_introspektywne

https://en.wikipedia.org/wiki/Introsort