Dawid Grajoszek, 249021
26.03.2020
Termin zajęć: Piątek, 9:15 - 11:00

Prowadzący:
Mgr inź. Marta Emirsajłow

PROJEKTOWANIE ALGORYTMÓW I METODY SZTUCZNEJ INTELIGENCJI

PROJEKT I
ALGORYTMY SORTOWANIA

1. **Wprowadzenie**

   Tematem tego projektu było zaimplementowanie trzech algorytmów sortowania: szybkiego, przez scalanie oraz introspektywnego. W wyżej wymienionym projekcie mamy do czynienia z algorytmami sortowania, które są stabilne, jak i niestabilne. Do pierwszej grupy należy algorytm sortowania przez scalanie. Oznacza to tyle, że elementy o równej wartości będą, po posortowaniu, występowały w tej samej kolejności, czyli nie są wykonywane dodatkowe, niepotrzebne operacje matematyczne. Pozostałe algorytmy należą już do grupy algorytmów niestabilnych. W powyższym badaniu należało posortować 100 tablic o ilości elementów równej odpowiednio: 10 000, 50 000, 100 000, 500 000, 1 000 000 w następujących przypadkach:

   - wszystkie elementy tablicy losowe
   - 25%, 50%, 75%, 95%, 99%, 99.7% początkowych elementów tablicy jest już posortowanych
   - wszystkie elementy tablicy są już posortowane, ale w odwrotnej kolejności

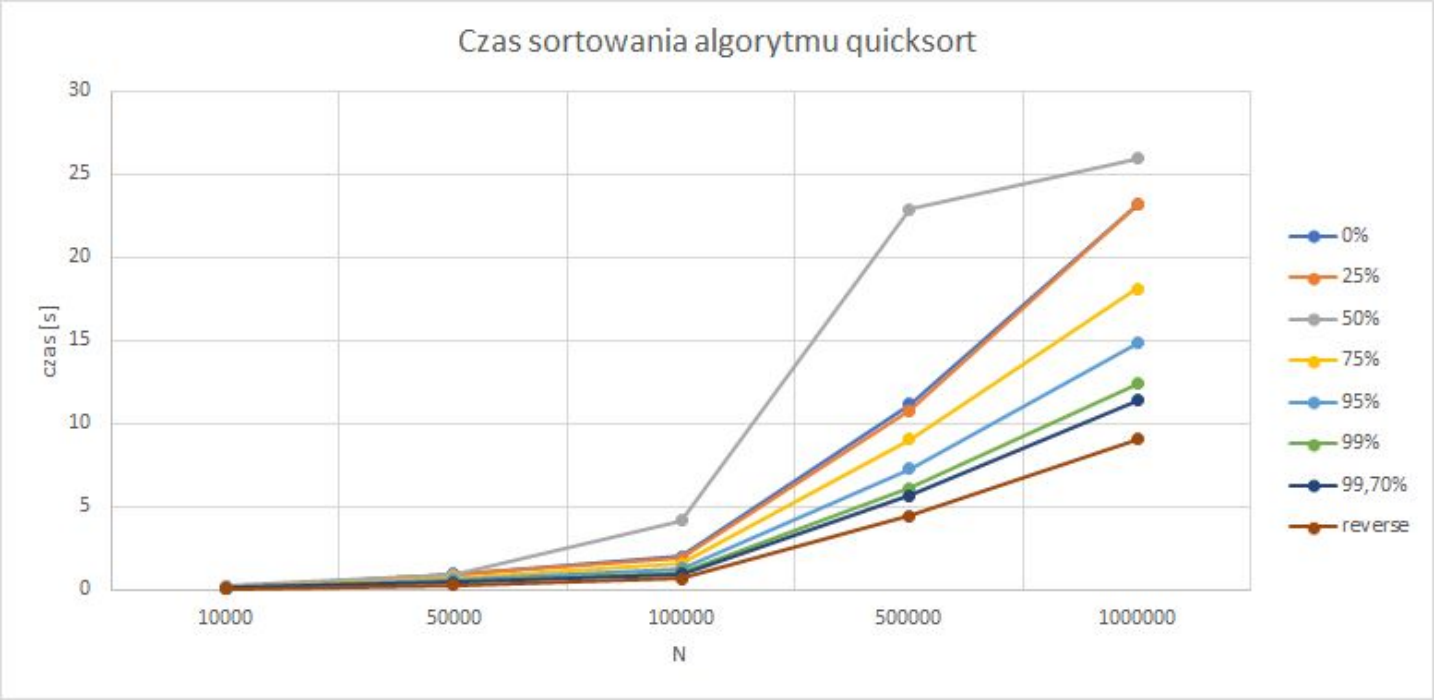
   Należy zmierzyć czas działania każdego z algorytmów i dokonać na tej podstawie analizy i porównania pomiędzy algorytmami.

2. **Sortowanie szybkie (ang. quicksort)**

   Algorytm ten jest jednym z najpopularniejszych, który działa na zasadzie “dziel i zwyciężaj”. Sortowanie to polega na wybraniu elementu środkowego tablicy, tzw. pivota, co skutkuje jej podziałem na dwa fragmenty, pomiędzy którymi porównywane są elementy: do lewej części oryginalnej tablicy wędrują te elementy, które są nie większe od elementu środkowego, a do prawej części elementy nie mniejsze od pivota. Algorytm sortowania szybkiego jest algorytmem rekurencyjnym: po odpowiednim podziale całej tablicy, zaczynamy sortować każdą część osobno aż do zatrzymania rekursji. Kończy się ona w momencie, gdy fragment uzyskany z podziału jest już pojedynczym elementem, gdyż taki już nie potrzebuje uporządkowania. Złożoność algorytmu sortowania szybkiego zależy od wyboru elementu środkowego. Jeżeli pivot zostanie wybrany w okolicy środka tablicy, wówczas mamy do czynienia z przypadkiem optymistycznym, gdzie złożoność obliczeniowa to $O(n * logn)$. Przy losowym wyborze pivota złożoność tego algorytmu dorównuje złożoności dla powyższego przypadku. Biorąc pod uwagę przypadek pesymistyczny, czyli wybranie pivota jako element największy lub najmniejszy, wówczas algorytm ten działa stosunkowo wolno, czyli $O(n^2)$. Zdecydowaną zaletą tego algorytmu jest łatwość w implementacji oraz brak potrzeby tworzenia dodatkowej, pomocniczej tablicy.

Czasy sortowania 100 tablic o konkretnym poziomie wstępnego posortowania [s] :

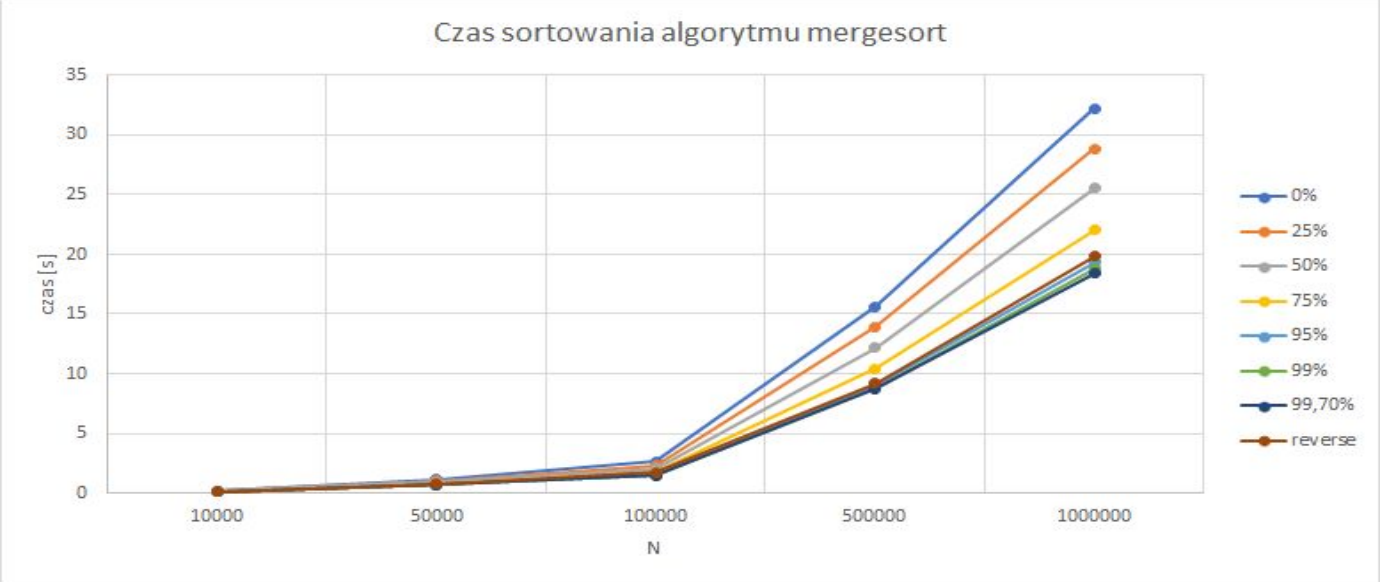
% posort | N	10 000	50 000	100 000	500 000	1 000 000
0 %	0,15168	0,92421	2,001	11,151	23,188
25 %	0,1567	0,90853	1,9123	10,761	23,195
50 %	0,19039	0,86274	4,1822	22,883	40,938
75 %	0,13095	0,73125	1,5603	9,0708	18,128
95 %	0,097301	0,59629	1,2827	7,3015	14,858
99 %	0,079517	0,46414	1,0331	6,13	12,376
99.7 %	0,078531	0,44175	0,96269	5,6511	11,371
odwrotnie	0,045796	0,2892	0,66764	4,4389	9,0482



3. **Sortowanie przez scalanie (ang. merge sort)**
   Algorytm sortowania przez scalanie jest, podobnie jak quicksort, algorytmem rekurencyjnym, który działa w oparciu o zasadę “dziel i zwyciężaj”. Oryginalna tablica dzielona jest na dwa fragmenty, następnie, rekurencyjnie, lewa i prawa część również dzielona jest na pół aż do uzyskania pojedynczych elementów. Powstałe elementy są scalane poprzez ich wzajemne porównanie i ustawienie na odpowiednich pozycjach w relacji mniejszy-większy. Maksymalna głębokość rekurencji to $log_2 n$. W opisany sposób cała tablica zostaje posortowana ze złożonością czasową $O(n * logn)$ w przypadku optymistycznym. Biorąc pod uwagę przypadek najgorszy złożoność obliczeniowa nie ulega zmianie, ponieważ zwiększa się jedynie liczba porównań elementów w procesie scalania. Nieuniknioną wadą tego algorytmu jest fakt, że potrzebna jest tymczasowa tablica pomocnicza, która będzie przechowywała kopie podtablic do scalenia.

   Czasy sortowania 100 tablic o konkretnym poziomie wstępnego posortowania [s] :

% posort | N	10 000	50 000	100 000	500 000	1 000 000
0 %	0,18621	1,1706	2,619	15,599	32,242
25 %	0,1792	1,0475	2,2934	13,954	28,878
50 %	0,15479	0,92669	2,0327	12,192	25,521
75 %	0,12824	0,78658	1,7403	10,412	22,066
95 %	0,10949	0,67111	1,5181	9,0616	19,325
99 %	0,10832	0,67234	1,4891	8,8018	18,805
99.7 %	0,10722	0,67336	1,4998	8,7166	18,384
odwrotnie	0,11738	0,73396	1,6999	9,1794	19,832

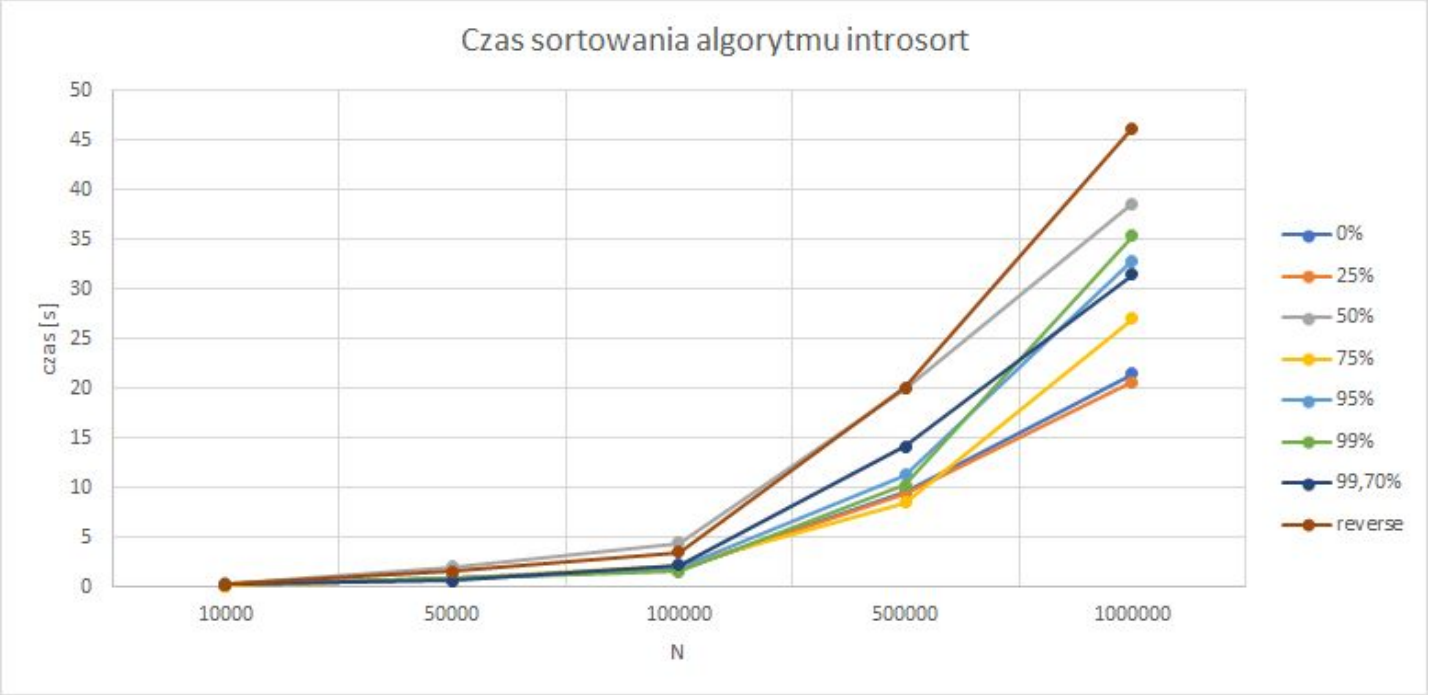


4. **Sortowanie introspektywne (ang. introsort)**

Algorytm sortowania introspektywnego, mimo iż jest hybrydowy, ponieważ zawiera w sobie kilka algorytmów sortujących, jest niestety algorytmem niestabilnym. Bazuje ono na sortowaniu szybkim i przez kopcowanie. Jego głównym zadaniem jest obsługa najgorszego przypadku dla quicksort'u, gdzie złożoność obliczeniowa wynosi $O(n^2)$. Na samym początku określa się maksymalną głębokość wywołań rekurencyjnych zgodnie ze wzorem 2* $log_2 n$. Jeśli wartość obliczonego parametru wynosi 0, wówczas kończy się rekursja i wywoływane jest sortowanie pomocnicze - przez kopcowanie (o złożoności $O(n * logn))$. W przeciwnym przypadku, gdy wartość parametru jest większa od 0, algorytm ten działa podobnie jak quicksort, gdzie następuje podział tablicy na dwa fragmenty. Po tej operacji wywoływana jest rekurencja sortująca z obliczonym wcześniej parametrem pomniejszonym o 1. W przypadku optymistycznym sortowanie introspektywne ma złożoność obliczeniową $O(n * logn)$. W przypadku pesymistycznym algorytm sortowania introspektywnego będzie działał tak jak algorytm sortowania szybkiego, dzięki czemu jego złożoność obliczeniowa nie ulega zmianie, co jest niewątpliwie pożądaną zaletą.

Czasy sortowania 100 tablic o konkretnym poziomie wstępnego posortowania [s] :

% posort | N	10 000	50 000	100 000	500 000	1 000 000
0 %	0,13355	0,79031	1,6937	9,6044	21,458
25 %	0,12671	0,81822	1,6918	9,4288	20,603
50 %	0,31877	2,0346	4,455	19,985	38,633
75 %	0,10557	0,80514	2,0583	8,4763	27,032
95 %	0,21771	0,90338	1,9943	11,269	32,846
99 %	0,2743	0,8735	1,5646	10,249	35,369
99.7 %	0,27881	0,60857	2,1809	14,117	31,439
odwrotnie	0,24922	1,561	3,4731	20,048	46,243



5. Podsumowanie

Wszystkie algorytmy sortujące są dosyć szybkie. Z moich eksperymentów wynika, że to właśnie algorytm sortowania szybkiego poradził sobie najlepiej. Zarówno dla niego, jak i dla sortowania przez scalanie czas potrzebny na posortowanie tablicy już wstępnie uporządkowanej maleje wraz ze wzrostem poziomu uporządkowania. Zupełnie inaczej poradził sobie algorytm sortowania hybrydowego w tym przypadku, którego czas wykonywania rósł w miarę zwiększania się współczynnika posortowania tablicy. Wszystkie wykresy prezentują dobre wyniki w porównaniu do przewidywanej funkcji złożoności obliczeniowej, gdyż przypominają funkcje f(n) = $n * logn$.

6. Literatura

W moich eksperymentach za pomoc posłużyły mi następujące strony:

- sortowanie szybkie Wikipedia
- sortowanie przez scalanie Wikipedia
- sortowanie introspektywne