Projekt 1

Algorytmy sortujące

PAMSI

Projektowanie algorytmów i metody sztucznej inteligencji

Prowadzący: **mgr. inż. Magdalena Skoczeń**
Termin zajęć: **Środa $18^{\overline{55}}$**

Piotr Niedziółka, 249023

08.04.2020

1 Wprowadzenie

Celem projektu była implementacja trzech algorytmów sortowania oraz analiza ich efektywności. Wykorzystane algorytmy w tym sprawozdaniu to: **QuickSort** - sortowanie szybkie, **MergeSort** - sortowanie przez scalanie, **IntroSort** - sortowanie introspektywne. Algorytmy zostały przetestowane na *100* tablicach o rozmiarach: *10 000, 50 000, 100 000, 500 000* oraz *1 000 000*, przy warunkach:

- wszystkie elementy losowe
- uporządkowana w *25%, 50%, 75%, 95%, 99%, 99,7%*
- tablica posortowana w odwrotnej kolejności

2 Omówienie algorytmów

a) **MergeSort** - Sortowanie przez scalanie

Algorytm sortowania stosujący zasadę **"dziel i zwyciężaj"**, sortowana tablica jest dzielona rekurencyjnie na dwie podtablice, dopóki nie uzyska się tablic jednoelementowych. Następnie tablice te są w odpowiedni sposób scalane, co pozwala na szybkie posortowanie tablicy. Jest to bardzo wydajny algorytm, stosujące dzielenie dużego problemu na mniejsze - łatwiejsze do rozwiązania.

Złożoność obliczeniowa:
Średni przypadek: ***O(n log n)***
Najgorszy przypadek: ***O(n log n)***

b) **QuickSort** - Sortowanie szybkie

Jeden z najpopularniejszych algorytmów, który słynie, jak nazwa mówi, ze swojej szybkości. Wybierany jest jeden element, a następnie tablica jest dzielona na dwie podtablice. Pierwsza z nich to elementy mniejsze od wybranego elementu, a druga z nich to elementy większe lub równe wybranemu elementowi - znanemu jako **pivot**. Proces ten jest wykonywany, aż do uzyskania tablic jednoelementowych. Sortowanie jest w pewien sposób ukryte, jednocześnie najgorszym przypadkiem będzie to, gdy będziemy wybierać za każdym razem skrajne wartości tablicy.

Złożoność obliczeniowa:
Średni przypadek: ***O(n log n)***
Najgorszy przypadek: $\mathbf{O(n^2)}$

c) **IntroSort** - Sortowanie introspektywne

Odmiana sortowania hybrydowego - połączenie dwóch sortowań: sortowanie szybkie, sortowanie przez kopcowanie. W tym algorytmie został wyeliminowany problem złożoności obliczeniowej $O(n^2)$ gdy zostanie wybrany najmniejszy / największy element jako **pivot**. I właśnie to jest głównym założeniem tego algorytmu - logarytmiczna złożoność obliczeniowa.W algorytmie IntroSort wykorzystuje się różne algorytmy sortujące, co sprawia, że program staje się skomplikowany. W trakcie implementacji można popełnić wiele błędów, dlatego trzeba być ostrożnym implementując ten algorytm.

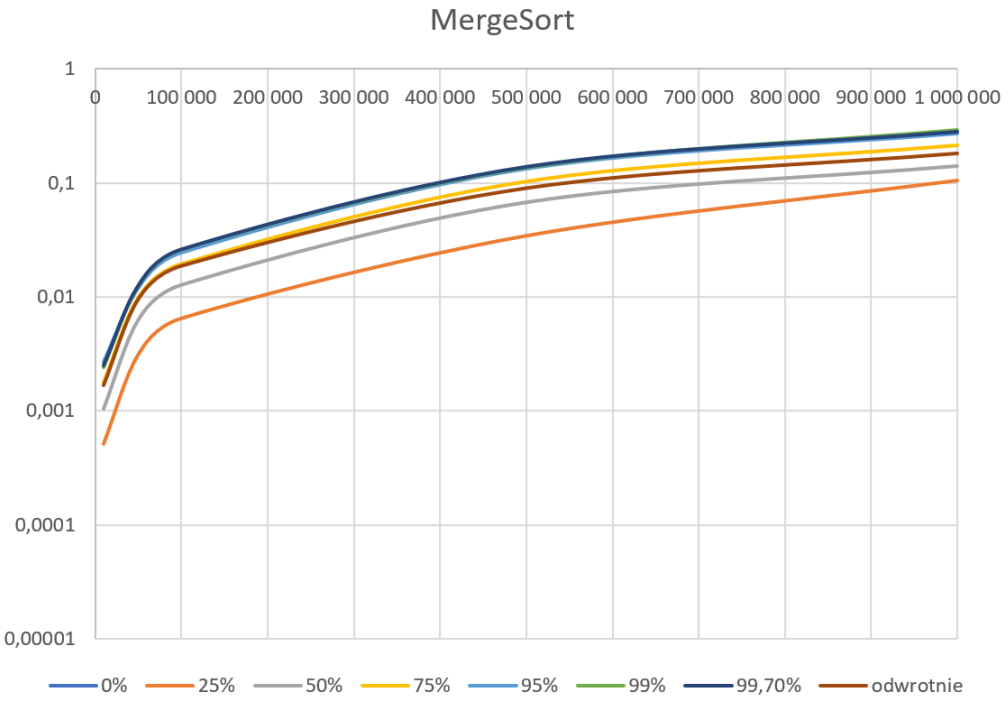
Złożoność obliczeniowa:
Średni przypadek: ***O(n log n)***
Najgorszy przypadek: ***O(n log n)***

3 Przebieg badanych algorytmów

- MergeSort

MergeSort					
	10 000	50 000	100 000	500 000	1 000 000
0%	0	1,00E-05	0	0	0
25%	0,00051	0,00309	0,00646	0,03436	0,10532
50%	0,00105	0,00626	0,01276	0,06762	0,14072
75%	0,00178	0,00985	0,01952	0,10356	0,2153
95%	0,00272	0,01201	0,02457	0,13373	0,27212
99%	0,00241	0,01234	0,02583	0,13573	0,28906
99,70%	0,00249	0,01244	0,02596	0,13799	0,28029
odwrotnie	0,00167	0,00943	0,01855	0,08912	0,17979

Rysunek 1: Tabela czasów MergeSort [s]

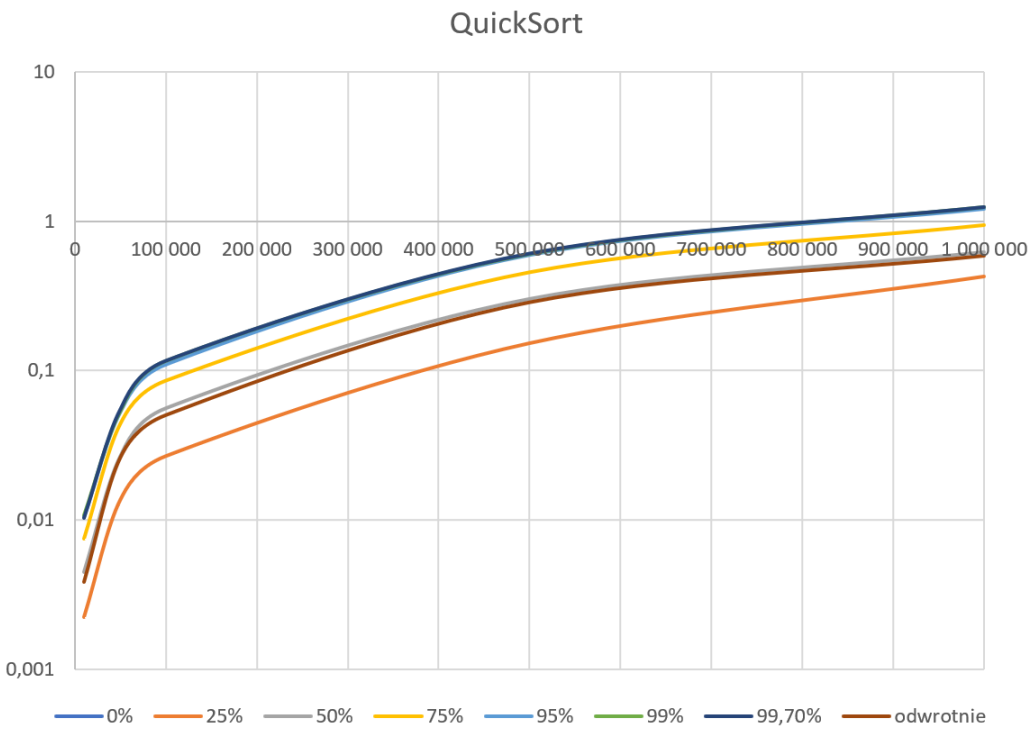


Rysunek 2: Wykres logarytmiczny MergeSort

- QuickSort

QuickSort					
	10 000	50 000	100 000	500 000	1 000 000
0%	0	0	0	0	0
25%	0,00	0,01362	0,02668	0,15197	0,42523
50%	0,0045	0,02678	0,05603	0,3014	0,61974
75%	0,00746	0,04404	0,0856	0,45602	0,94503
95%	0,0105	0,05288	0,1098	0,59257	1,20568
99%	0,01079	0,05379	0,1166	0,60732	1,25244
99,70%	0,01031	0,0551	0,1165	0,60902	1,24751
odwrotnie	0,00383	0,02614	0,05029	0,2866	0,5882

Rysunek 3: Tabela czasów QuickSort [s]

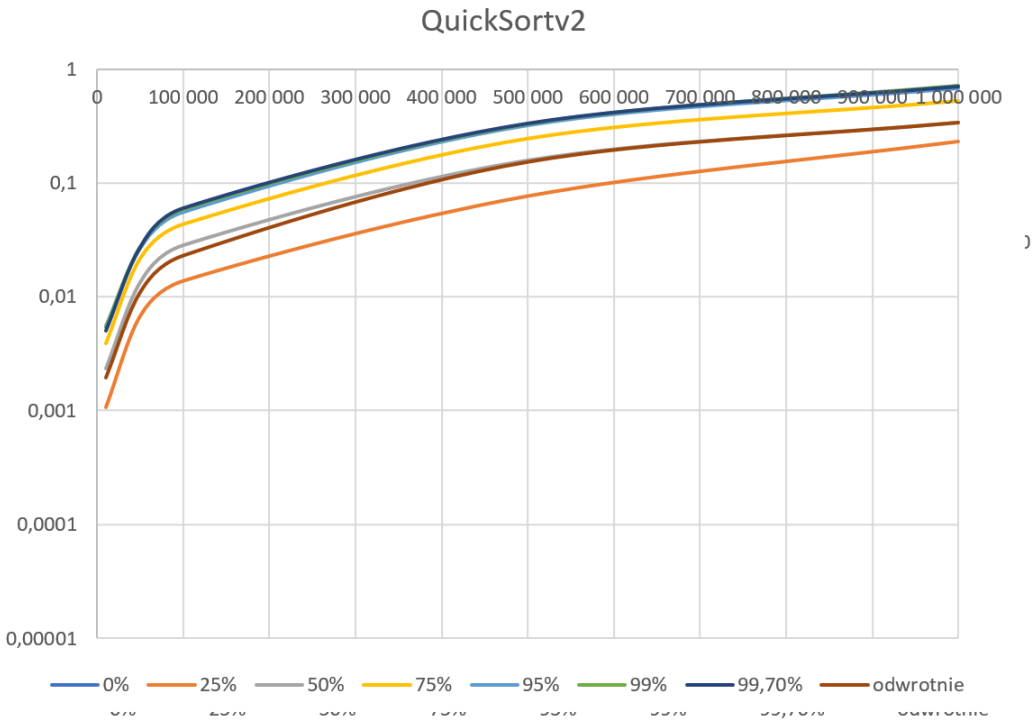


Rysunek 4: Wykres logarytmiczny QuickSort

- QuickSortv2

QuickSortv2					
	10 000	50 000	100 000	500 000	1 000 000
0%	0	1,00E-05	0	0	0
25%	0,00107	0,00683	0,01386	0,07713	0,23285
50%	0,00234	0,01347	0,02845	0,15847	0,34067
75%	0,00393	0,02197	0,04373	0,24647	0,52731
95%	0,00551	0,02673	0,05549	0,32052	0,67748
99%	0,00539	0,02741	0,05816	0,32689	0,70878
99,70%	0,00503	0,02739	0,06011	0,33308	0,7028
odwrotnie	0,00195	0,01102	0,02303	0,15289	0,33897

Rysunek 5: Tabela czasów QuickvSortv2 [s]

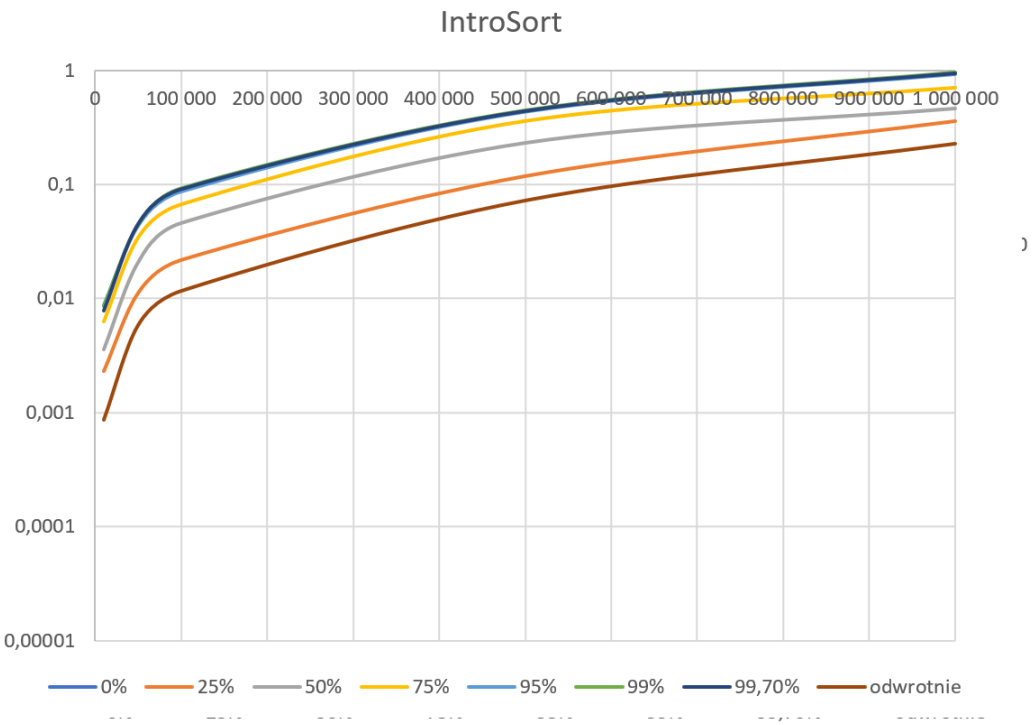


Rysunek 6: Wykres logarytmiczny QuickSortv2

- IntroSort

Introsort					
	10 000	50 000	100 000	500 000	1 000 000
0%	0	1,00E-05	0	0	1,00E-05
25%	0,00231	0,0112	0,02178	0,11829	0,35914
50%	0,0036	0,02083	0,04588	0,23034	0,46047
75%	0,00632	0,03447	0,06734	0,36238	0,70928
95%	0,0087	0,04331	0,08619	0,4308	0,919
99%	0,00854	0,04422	0,09196	0,4429	0,95217
99,70%	0,00777	0,04502	0,09085	0,44195	0,94837
odwrotnie	0,00086	0,00583	0,0116	0,07193	0,22652

Rysunek 7: Tabela czasów IntroSort [s]



Rysunek 8: Wykres logarytmiczny IntrokSort

4 Podsumowanie

MergeSort jest najszybszym algorytmem we wszystkich testach. Drugim pod względem efektywności jest QuickSortv2, następnie IntroSort i najwolniejszy QuickSort. Jest to spowodowane, że MegrgeSort działa bardzo szybko, ale potrzebuje więcej pamięci niż pozostałe algorytmy. IntroSort jest szybszy niż QuickSort - ponieważ został tam wyeliminowany problem kwadratowej złożoności obliczeniowej w najgorszym przypadku, gdy wybierany jest za każdym razem największy lub namniejszy element tablicy. QuickSortv2 jest szybszy niż pozostałe dwa, gdyż został zaimplementowany tak, by zaczynał zarówno z początku i końca tablicy jednocześnie, co pozwala na szybsze posortowanie.

Najszybszy czas sortowania notujemy dla tablic posortowanych malejąco poprzez algorytm IntroSort. Pozostałe algorytmy najszybciej sortują tablicę, która jest uporządkowana w 75%, czyli należy posortować 25%. Najwolniejsze czasy można zaobserwować, gdy tablica nie jest posortowana. Są to np: 95%, 99%, 99,7%

5 Bibliografia

1. www.stackoverflow.com
2. en.wikipedia.org/wiki/Merge_sort
3. en.wikipedia.org/wiki/Quicksort
4. en.wikipedia.org/wiki/Introsort
5. www.geeksforgeeks.org/
6. en.cppreference.com/w/cpp/chrono/c/clock
7. en.cppreference.com/w/cpp/algorithm/reverse
8. en.cppreference.com/w/cpp/algorithm/sort