31.03.2019

Sprawozdanie

Algorytmy sortowania, złożoność obliczeniowa

Kurs: Projektowanie Algorytmów i metody sztucznej inteligencji

Prowadzący:

Autor: Przemysław Malec

Termin zajęć: Wtorki TP 13:15-15

31.03.2019

Wstęp teoretyczny

Sortowanie to jeden z podstawowych problemów informatyki. Dobrze posortowane dane(według właściwej cechy przedmiotu) ułatwiają operowanie nimi i podejmowanie decyzji na ich podstawie. Algorytmy sortowania różnią się między sobą złożonością obliczeniową, złożonością pamięciową oraz sposobem działania. W ostatnim typie podziału możemy wyodrębnić dwie kategorię algorytmy naiwne oraz logarytmiczne.

Omówienie przetestowanych algorytmów

Sortowanie szybkie.

Dzieli ono problem na mniejsze podproblemy tzn. dzieli dane wg „pivota” tj. dowolny element zbioru danych, który możemy dobierać zależnie od problemu. Każde wywołanie funkcji gwarantuje że na lewo od pivota są umieszone dane mniejsze od niego a na prawo większe. Dzielenie problemu odbywa się przez rekurencyjne wywołanie funkcji, które kończą się gdy nastąpi wywołanie zdegenerowane tj. tablica ma jeden element. W tym momencie procedura kończy się a dane są posortowane. Do zalet tego algorytmu trzeba zaliczyć to że do swojego działania nie wykorzystuje dodatkowych pomocniczych struktur danych.

Sortowanie przez scalanie

Również dzieli problem na mniejsze podproblemy. Dzieli tablice danych do momentu gdy tablica ma tylko jeden element. Później następuje scalanie podtablic w jedną(pomocniczą), korzystając przy tym z informacji że każda podtablica jest posortowana, wiec sprawdza tylko kolejny element podtablicy do wpisania a nie wszystkie. Algorytm ten potrzebuje więcej pamięci do swojego działania ponieważ potrzebuje pomocniczej tablicy o wielkości tablicy sortowanej

Sortowanie Introspektywne

Jest ewolucją sortowania szybkiego. Cechuje się lepsza pesymistyczną złożonością obliczeniową. Opiera się na spostrzeżeniu że sortowanie małych tablic wykonuje się szybciej użyciem algorytmów o gorszej złożoności obliczeniowej ponieważ procedury związane z podziałem tablic, sprawdzaniem warunków odbywające się w sortowaniu szybkim zajmują więcej czasu niż samo sortowanie. Dlatego ograniczono głębokość rekurencji, po przekroczeniu której wykorzystuje się inny algorytm zamiast sortowania szybkiego. Daje to korzyść w postaci wyeliminowania najgorszego przypadku jakim była kwadratowa złożoność obliczeniowa.

Wszystkie opisane tutaj algorytmy cechują się średnią złożonością obliczeniową na poziomie logarytmicznej.

31.03.2019

Opis wykonanych badań

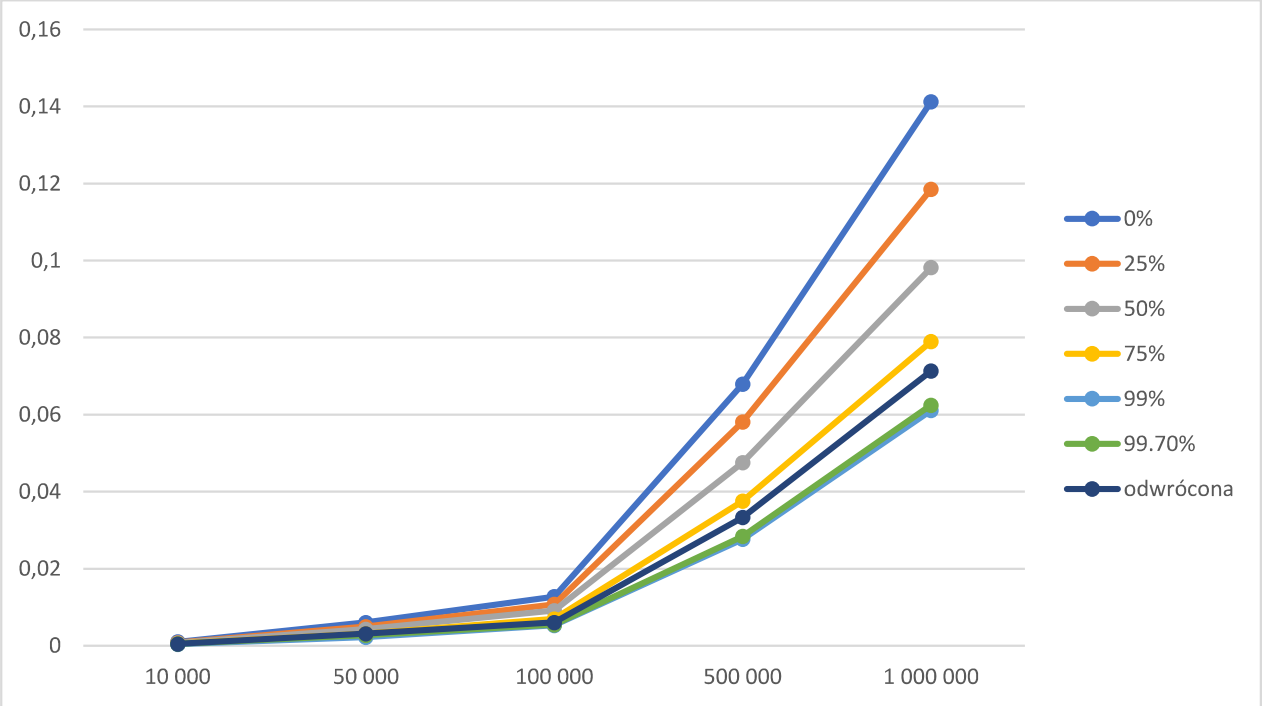
Do przetestowania zaimplementowanych algorytmów użyłem funkcji testującej która wykonuje następujące operacje:

Inicjalizuje i wypełnia zadaną ilość tablic zadanej długości pseudolosowymi liczbami całkowitymi. Wstępnie sortuje początkową część tablicy w stopniu jaki został zadany. Sumuje czasy sortowania każdej tablicy a wyniki zapisuje do pliku.

Prezentacja wyników badań

Sortowanie przez scalanie

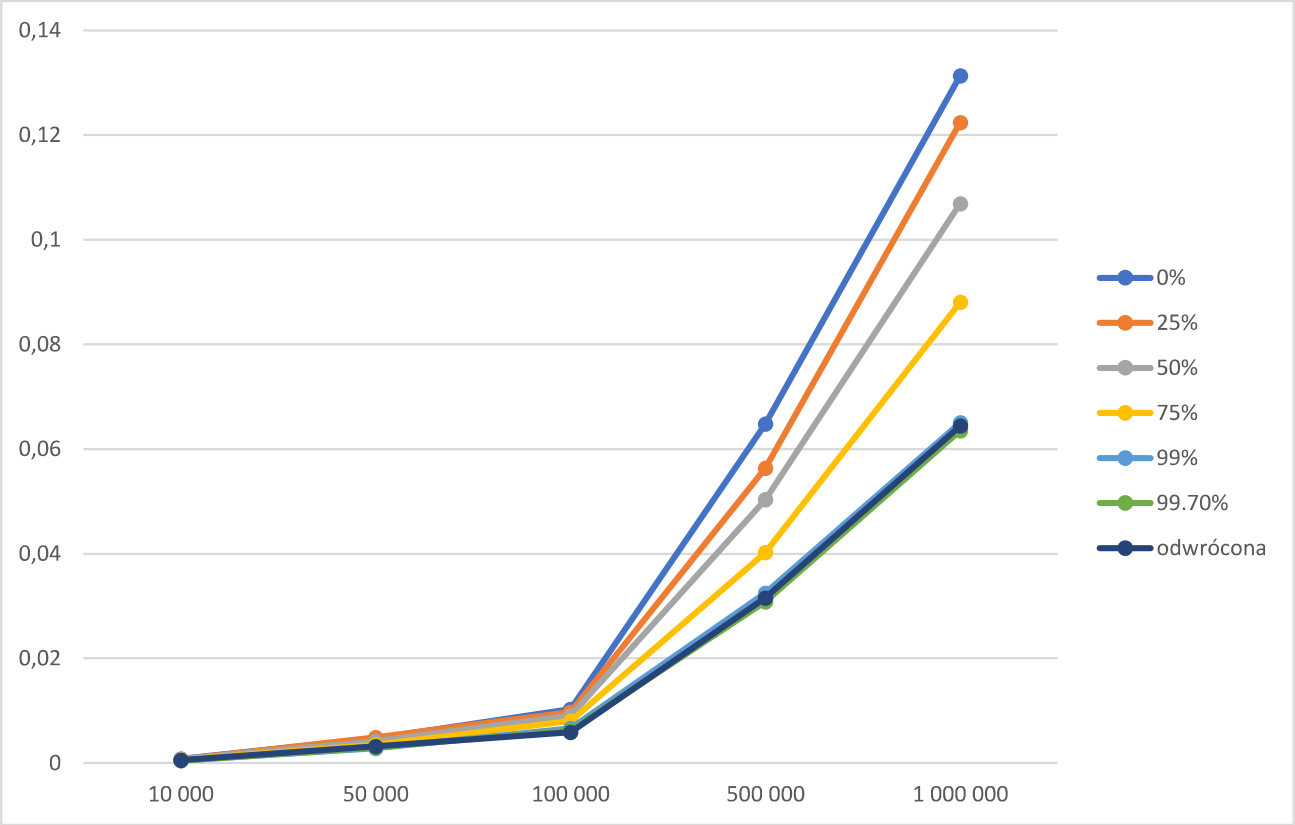
	10 000	50 000	100 000	500 000	1 000 000
0%	0.000997531s	0.00601319s	0.0126957s	0.0679201s	0.141256s
25%	0.000795507s	0.00494815s	0.0107546s	0.0580742s	0.118468s
50%	0.000702026s	0.00422428s	0.00903667s	0.0475205s	0.09813s
75%	0.000548604s	0.00308326s	0.00695184s	0.0375356s	0.0789891s
99%	0.000373409s	0.00218256s	0.0052868s	0.0276623s	0.0610624s
99.7%	0.000434003s	0.00271176s	0.00544403s	0.0283683s	0.0624388s
odwrócona	0.000438426s	0.0031177s	0.00603981s	0.0332718s	0.0713344s



31.03.2019

Sortowanie przez introspektywne

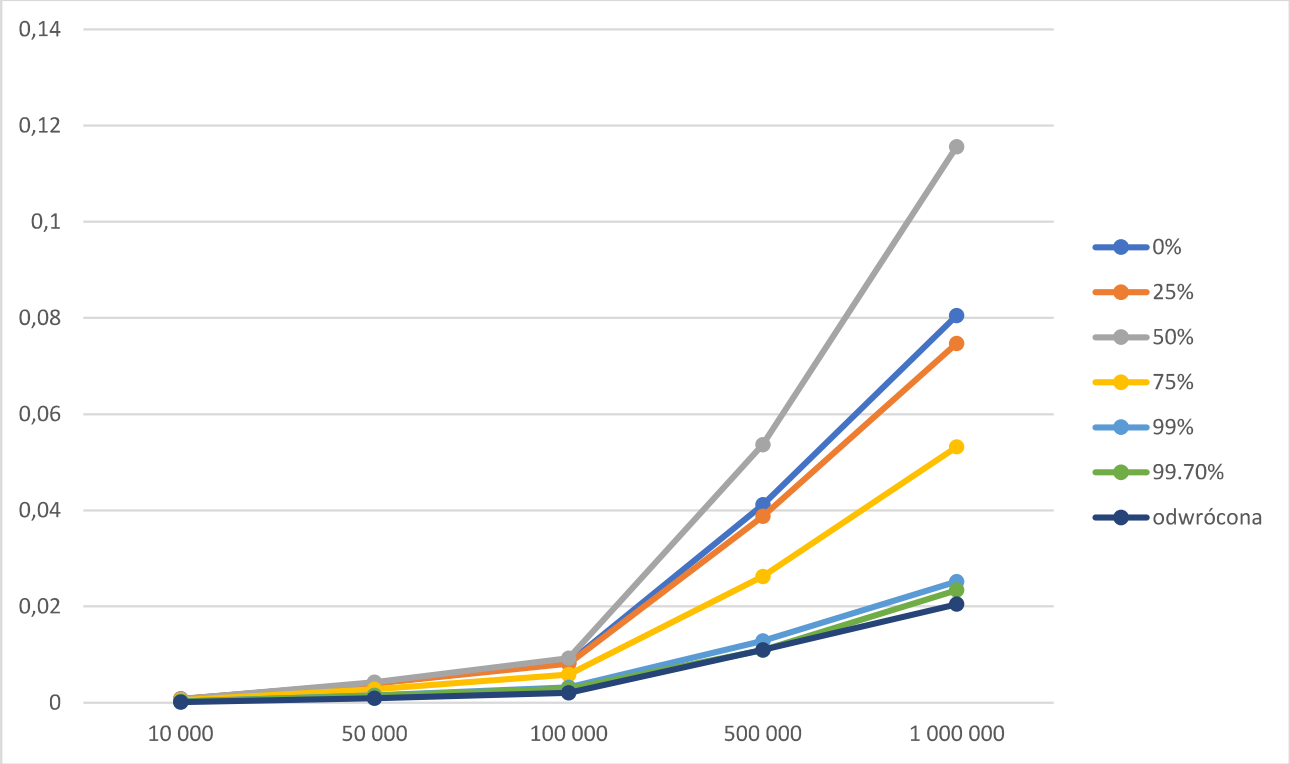
	10 000	50 000	100 000	500 000	1 000 000
0%	0.000801508s	0.00474714s	0.0102897s	0.0647687s	0.131314s
25%	0.000713794s	0.00490595s	0.00973452s	0.0563338s	0.122371s
50%	0.000735312s	0.00417745s	0.00909667s	0.0503275s	0.106905s
75%	0.000632977s	0.00352692s	0.00804346s	0.040232s	0.0880314s
99%	0.00051024s	0.00283406s	0.00662061s	0.0324477s	0.0650038s
99.7%	0.000478628s	0.00295132s	0.006286s	0.0308262s	0.063511s
odwrócona	0.000557327s	0.00321417s	0.00591749s	0.0315368s	0.0644122s



31.03.2019

Sortowanie Quicksort

	10 000	50 000	100 000	500 000	1 000 000
0%	0.00074281s	0.00402586s	0.00829873s	0.0411048s	0.0804976s
25%	0.00080056s	0.00396344s	0.00811094s	0.0387497s	0.0746826s
50%	0.000737145s	0.00423048s	0.00923887s	0.0536716s	0.115589s
75%	0.000483239s	0.00279775s	0.00582194s	0.0262451s	0.0531929s
99%	0.000229232s	0.0015252s	0.00313822s	0.0128374s	0.0251775s
99.7%	0.000256536s	0.0014156s	0.00287769s	0.0109119s	0.0233962s
odwrócona	0.000121138s	0.000930188s	0.00204459s	0.0109695s	0.0204569s



Podsumowanie

- Każdy z trzech przedstawionych algorytmów cechuje się logarytmiczną złożonością obliczeniową
- Dla sortowania introspektywnego i sortowania przez scalanie najgorszym przypadkiem jest tablica nieposortowana co jest intuicyjne.
- Najlepszym jest im więcej elementów w posortowanych tym wyżej wymienione sortowania są szybsze.
- Dla sortowania szybkiego najlepszym przypadkiem jest tablica w pełni posortowana ale odwrócona.

- Najgorszym jest posortowana w 50 %, wpływ na to ma dobór pivota. Gdy dla testów ustawiłem pivota na końcu, całkowicie posortowana tablica była najwolniej sortowana.
- Zaskakujące jest to że sortowanie introspektywne jest minimalne wolniejsze od szybkiego, a jest przecież jego bardziej rozwiniętą wersją. Testowanie algorytmów odbywało się w tych samych warunkach, wiec może to być błąd mojej implementacji jednak nie jestem w stanie go wykryć. Może to być również związane po prostu z tym, że sortowanie introspektywne wykonuje więcej operacji podstawowych(dekrementacja zmiennej informującej głębokości rekurencji, sprawdzanie warunku głębokości rekurencji)

Źródła

https://www.youtube.com/watch?v=82XxdhRCMbI&t=28s

https://www.youtube.com/watch?v=iJyUFvvdfUg&t=485s

http://www.algorytm.edu.pl/

http://www.algorytm.org/

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

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

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