Sprawozdanie – Projektowanie algorytmów i metody sztucznej inteligencji, Projekt 1

Data oddania sprawozdania: 3.04.2019r.
Autor: Tomasz Filip 241630
Prowadzący: dr inż. Łukasz Jeleń
Termin zajęć: środa 11:15

1. Wprowadzenie

Zadanie polegało na zaimplementowaniu trzech algorytmów służących do sortowania oraz przeprowadzeniu testów efektywności dla 100 tablic o elementach typu całkowitoliczbowego o rozmiarach:
10 000, 50 000, 100 000, 500 000, 1 000 000,
w następujących przypadkach:
-wszystkie elementy losowe,
-25%, 50%, 75%, 95%, 99%, 99,7% początkowych elementów posortowanych,
-wszystkie elementy posortowane w odwrotnej kolejności.
Wybrano następujące algorytmy sortowań: przez scalanie, quicksort, introspektywne

2. Zastosowane algorytmy

Sortowanie przez scalanie (mergesort)
Jest to rekurencyjny algorytm bazujący na metodzie „dziel i zwyciężaj", czyli dzieleniu problemu na mniejsze, łatwiejsze do rozwiązania „kawałki". Algorytm dzieli tablicę na dwie mniejsze części, a następnie każdą z nich dzieli, aż do momentu gdy powstaną tablice jednoelementowe. Następnie łączy dwie cząstki je ustawiając elementy we właściwej kolejności, aż powstaje jedna tablica złożona ze wszystkich elementów.
Złożoność obliczeniowa wynikająca z ilości podziałów to log(n), natomiast scalenie dwóch części to koszt O(n). Stąd złożoność obliczeniowa algorytmu to O(n log(n)) i nie zależy ona od liczb w tablicy.

Quicksort (sortowanie szybkie)
Podobnie jak sortowanie przez scalanie, jest to algorytm rekurencyjny oparty o metodę podziału problemu na mniejsze. Spośród elementów tablicy wybierany jest pivot – element do którego porównujemy wszystkie inne. Następnie porównane elementy ustawiane są po lewej lub prawej stronie pivota. W ten sposób powstaje zbiór elementów większych i zbiór elementów mniejszych, a sam pivot ustawiany jest pośrodku. Następnie te zbiory są traktowane w ten sam sposób, aż do powstania już ustawionych w kolejności zbiorów jednoelementowych nie wymagających dalszego sortowania.
Złożoność obliczeniowa zależy od szczęścia przy wyborze pivota i w przypadku przeciętym jest to O(n log(n)), jednak jeżeli zawsze jako pivot wybierany będzie element największy lub najmniejszy, to złożoność obliczeniowa wyniesie $O(n^2)$.

Sortowanie introspektywne (introsort)

Algorytm ten bazuje na algorytmie sortowania szybkiego, jednak została ograniczona maksymalna głębokość rekursji. W przytoczonym wyżej przypadku, gdy pivot wybierany jest na tyle nieszczęśliwie, że złożoność obliczeniowa algorytmu rośnie do $O(n^2)$ aktywowane jest sortowanie przez kopcowanie (heapsort), którego złożoność pesymistyczna to O(n log(n)). Średnio działa ono ok. 3 razy dłużej, jednak lepiej radzi sobie ze „zdegenerowanym" zbiorem liczb powodującym powolne działanie quicksorta. Maksymalna głębokość rekursji, przy której przekroczeniu aktywuje się heapsort wynosi $2*\log_2(n)$ (zaokrąglona do liczby naturalnej), gdzie n to rozmiar tablicy. Złożoność obliczeniowa algorytmu wynosi zawsze O(n log(n)).

3. Testy algorytmów

mergesort								
stopien posortowania	0	0,25	0,5	0,75	0,95	0,99	0,997	1,00 (rev)
rozm tablicy								
10000	0,268275	0,254942	0,238454	0,217023	0,200194	0,198767	0,202521	0,196344
50000	1,523298	1,433865	1,314290	1,224729	1,128329	1,107260	1,100981	1,094180
100000	3,211758	3,035812	2,786146	2,539470	2,351554	2,299843	2,310210	2,280843
500000	17,967896	16,777976	15,469933	14,116413	13,002587	12,805708	12,781554	12,560221
1000000	38,035138	35,878263	32,847008	29,886317	27,340888	26,180036	25,619479	25,510388

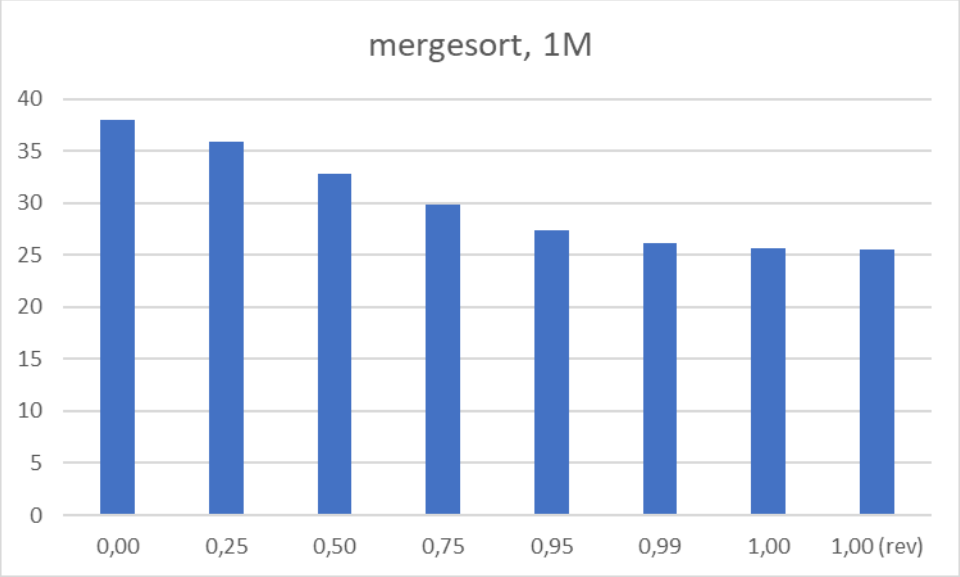
Tabela 1. Testy algorytmu sortowania przez scalanie

quicksort								
stopien posortowania	0,00	0,25	0,50	0,75	0,95	0,99	1,00	1,00 (rev)
rozm tablicy								
10000	0,187384	0,185722	0,4435	0,166234	0,169706	0,145299	0,137134	0,118962
50000	1,086763	1,090073	3,335397	0,981817	1,038281	0,871466	0,813131	0,678479
100000	2,310093	2,315058	8,744102	2,675486	2,517697	2,152995	1,74856	1,431457
500000	13,09112	13,04317	72,80267	12,18941	13,42535	11,24117	10,20425	7,784644
1000000	27,37392	27,49017	212,334	25,78436	28,88784	24,29848	21,92626	16,24506

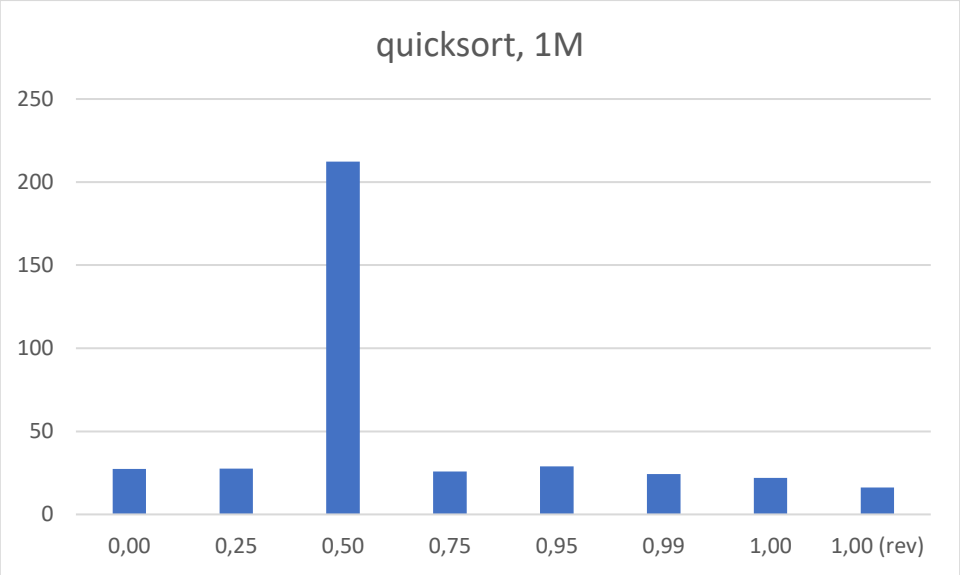
Tabela 2. Testy algorytmu sortowania szybkiego

introsort								
stopien posortowania	0	0,25	0,5	0,75	0,95	0,99	0,997	1,00 (rev)
rozm tablicy								
10000	0,207558	0,208299	0,202413	0,185265	0,181885	0,185485	0,181754	0,124151
50000	1,176769	1,174330	1,159534	1,035321	1,010493	1,002369	1,010497	0,673144
100000	2,524610	2,512681	2,498387	2,208183	2,163814	2,145879	2,148739	1,458933
500000	13,992304	13,833206	13,809645	12,167408	11,971779	11,690101	11,781629	7,877236
1000000	28,958929	28,605645	28,909233	25,035994	25,017174	24,087751	25,546230	17,054708

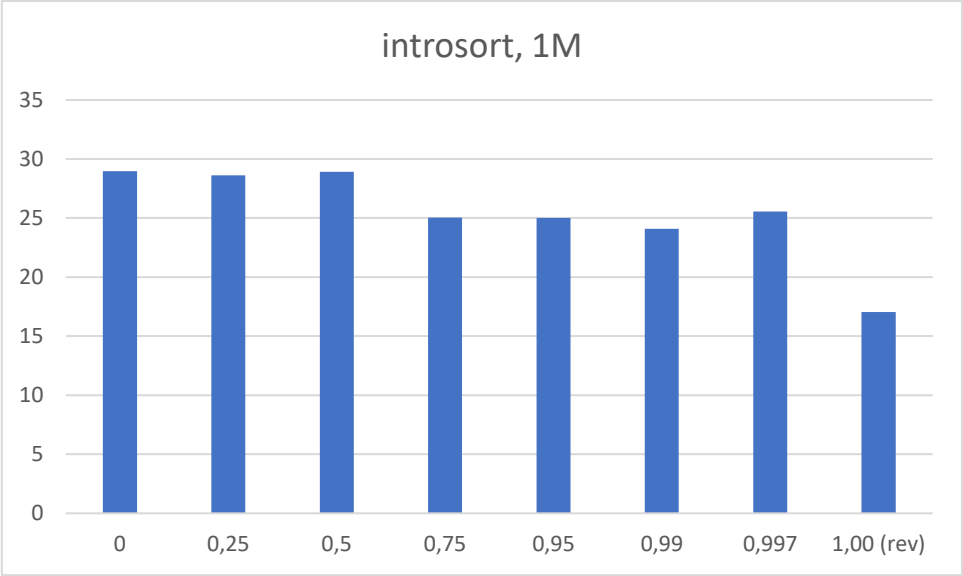
Tabela 3. Testy algorytmu sortowania introspektywnego



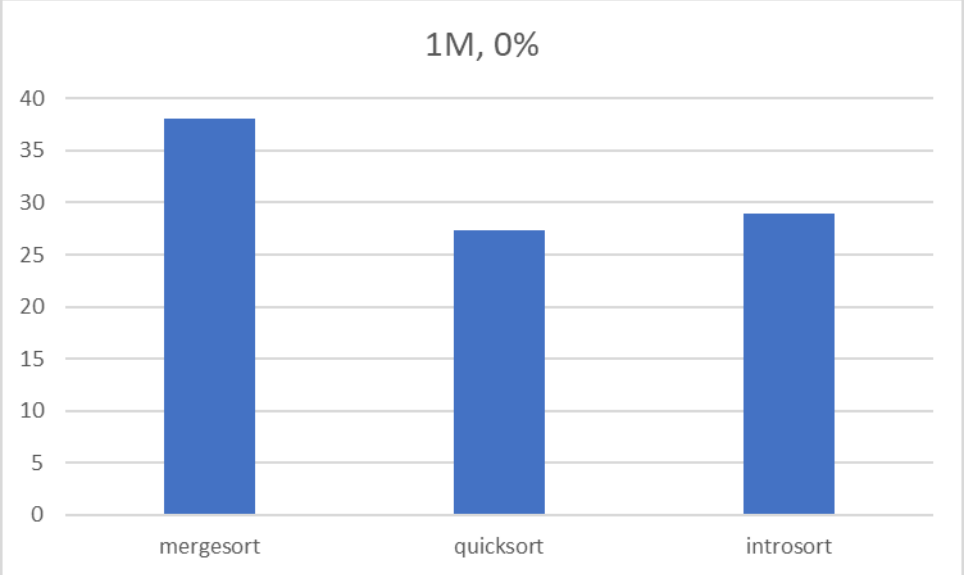
Rys.1 Czas działania sortowania przez scalanie tablic o milionie elementów i różnym stopniu posortowania



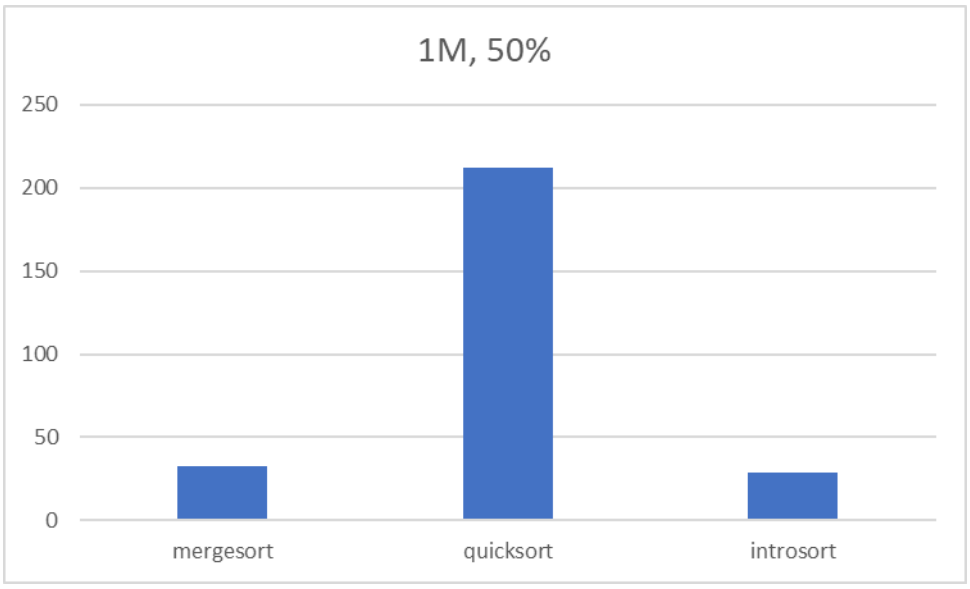
Rys. 2 Czas działania sortowania szybkiego tablic o milionie elementów i różnym stopniu posortowania



Rys. 3 Czas działania sortowania introspektywnego tablic o milionie elementów i różnym stopniu posortowania



Rys.4 Czas działania algorytmów dla tablicy o milionie elementów, całkowicie losowych



Rys. 5 Czas działania algorytmów dla tablicy o milionie elementów, o pierwszych 50% elementów posortowanych

4. Podsumowanie i wnioski

Uzyskane czasy działania algorytmów są zgodne z przewidywaniami, quicksort bardzo dobrze radzi sobie z typowymi tablicami o dużym rozmiarze i losowych liczbach. W przypadku tablicy o 50% posortowanych elementów, jego czas działania był bardzo duży. Wiąże się to z nieszczęśliwym doborem pivota, który prawdopodobnie był wciąż wybierany jako największy element posortowanej części tablicy. Ten przypadek pokazuje użyteczność stosowania sortowania introspektywnego, które w tym przypadku stosuje sortowanie przez kopcowanie, które ma znacznie mniejszą złożoność obliczeniową dla tego typu „zdegenerowanej" tablicy. Dla pozostałych tablic czas działania sortowania introspektywnego jest porównywalny do sortowania szybkiego, co oznacza że nie zaszła konieczność użycia heapsorta. Niewielkie różnice w czasie mogą wynikać ze zmiany warunków testowania algorytmu – sortowanie introspektywne było badane następnego dnia na świeżo włączonym urządzeniu. Sortowanie przez scalanie działa nieco dłużej niż quicksort, jednak jego działanie jest bardzo przewidywalne, a jego algorytm łatwiejszy do zrozumienia niż na przykład sortowania introspektywnego.

Źródła:

https://www,samouczekprogramisty.pl/podstawy-zlozonosci-obliczeniowej/
https://pl.wikipedia,org/wiki/Sortowanie_przez_scalanie
https://pl.wikipedia,org/wiki/Sortowanie_szybkie
https://pl.wikipedia.org/wiki/Sortowanie_introspektywne
https://pl.wikipedia.org/wiki/Sortowanie_przez_kopcowanie
https://www.geeksforgeeks.org/know-your-sorting-algorithm-set-2-introsort-cs-sorting-weapon/
https://www.geeksforgeeks.org/quick-sort/
https://www.mpcforum.pl/topic/1007722-tutc-mierzenie-czasu-wykonywania-algorytmu/