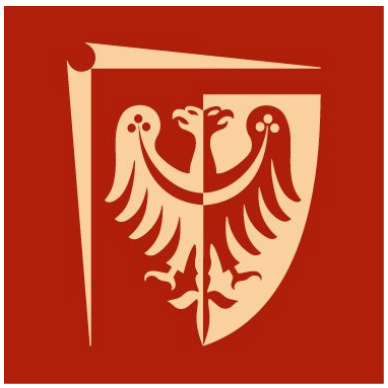


Politechnika Wrocławska

Wydział Informatyki i Telekomunikacji

Informatyczne systemy automatyki

Projekt numer 1 - algorytmy sortowania (merge sort, quick sort i intro sort).

Autor:
Jan Galewicz id. 272587

Przedmiot: Projektowanie i analiza algorytmów - projekt

25 stycznia 2024

Spis treści

Spis rysunków

Spis tabel

1 Wstęp:

W dzisiejszym świecie, w którym ilość danych generowanych i przetwarzanych przez systemy informatyczne stale rośnie, skuteczne sortowanie danych staje się niezwykle istotnym zadaniem. W tej pracy skupię się na badaniu trzech algorytmów sortowania:

- Sortowanie przez scalanie (merge sort).
- Sortowanie szybkie (quick sort).
- Sortowanie introspectywne (intro sort).

Głównym celem było zbadanie wydajności tych algorytmów pod względem czasu działania. Algorytmy były testowane na dołączonej do zadania bazie filmów z serwisu Filmweb.

1.1 Merge sort:

Wynaleziony w 1945 roku przez Johna von Neumanna algorytm sortowania przez scalanie jest algorytmem rekurencyjnym. Wykorzystuje zasadę dziel i zwyciężaj, która polega na podziale zadania głównego na zadania mniejsze dotąd, aż rozwiązanie stanie się oczywiste. Algorytm sortujący dzieli porządkowany zbiór na kolejne połowy dopóki taki podział jest możliwy (tzn. podzbiór zawiera co najmniej dwa elementy). Następnie uzyskane w ten sposób części zbioru rekurencyjnie sortuje tym samym algorytmem. Posortowane części łączy ze sobą za pomocą scalania, tak aby wynikowy zbiór był posortowany.[1]

Dla najgorszego przypadku	O(n log n)
Średnia	Θ(n log n)

Tabela 1: Złożoność obliczeniowa algorytmu sortowania przez scalanie.

1.2 Quick sort:

Sortowanie szybkie zostało wynalezione przez angielskiego informatyka, profesora Tony'ego Hoare'a w latach 60-tych ubiegłego wieku. Algorytm ten opiera się na strategii "dziel i zwyciężaj". W przypadku typowym algorytm ten jest najszybszym algorytmem sortującym z klasy złożoności obliczeniowej O(n log n). W pewnych sytuacjach (zależnych od sposobu wyboru piwotu oraz niekorzystnego ułożenia danych wejściowych) klasa złożoności obliczeniowej tego algorytmu może się degradować do O(n^2), co więcej, poziom wywołań rekurencyjnych może spowodować przepełnienie stosu i zablokowanie komputera.[2]

Dla najgorszego przypadku	O(n^2)
Średnia	Θ(n log n)

Tabela 2: Złożoność obliczeniowa algorytmu sortowania szybkiego.

1.3 Intrasort:

Sortowanie introspektywne jest to odmiana sortowania hybrydowego, w której wyeliminowany został problem złożoności obliczeniowej w najgorszym przypadku sortowania szybkiego. Rozwiązaniem problemu jest badanie głębokości rekurencji. W procedurze głównej sortowania introspektywnego tworzona jest stała M o wartości $2log_2$n, która określa maksymalną dozwoloną głębokość wywołań rekurencyjnych z poziomu, na którym obecnie się znajdujemy. Jeżeli wartość parametru M wynosi 0, wywołania rekurencyjne są kończone i dla podproblemu, którym obecnie się zajmujemy, wywoływana jest procedura Sortowania Przez Kopcowanie. W przypadku gdy M $>$ 0 procedura Intro Sort działa podobnie jak procedura Quick Sort.[3]

Dla najgorszego przypadku	O(n log n)
Średnia	Θ(n log n)

Tabela 3: Złożoność obliczeniowa algorytmu sortowania introspectywnego.

2 Badanie algorytmów.

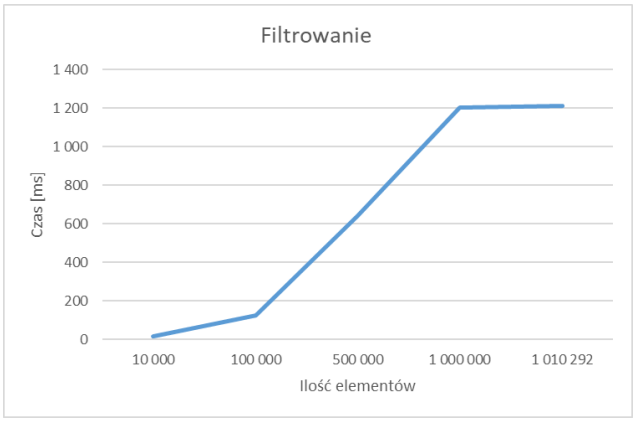
2.1 Metoda testowa:

Testy zostały przeprowadzone na bazie danych filmów ze strony Filmweb. Pomiary zostały wykonane dla następującej ilości elementów: 10 000, 100 000, 500 000, 1 000 000 oraz 1 010 292 (ilosc filmow w bazie). W pierwszym kroku wpisy które nie zawierały oceny były odrzucane. Następnie przeprowadzane były testy danego algorytmu. Zostały one wykonane na komputerze na którym uruchomiony był tylko program badający algorytmy. Każdy pomiar został przeprowadzony pięć razy, a z wyników została obliczona średnia arytmetyczna.

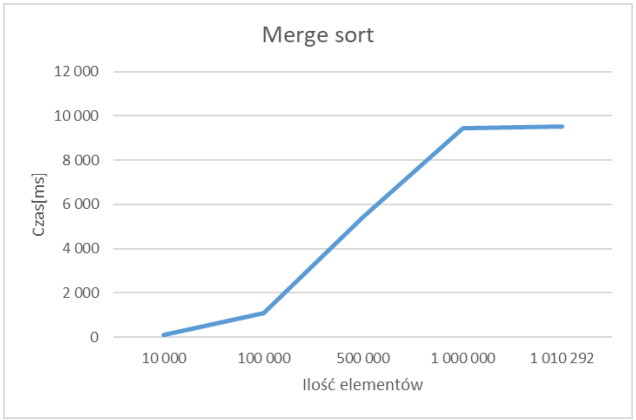
2.2 Tabele pomiarowe i wykresy.

Algorytm	Ilość elementów				
——	10 000	100 000	500 000	1 000 000	1 010 292
Filtrowanie wpisów	15,422 ms	128,129 ms	641,691 ms	1 202,053 ms	1 211,868 ms
Merge sort	84,364 ms	1 068,613 ms	5 438,854 ms	9 453,180 ms	9 530,482 ms
Quick sort	3,161 ms	67,569 ms	381,193 ms	757,248 ms	800,872 ms
Intro sort	7,227 ms	84,628 ms	473,447 ms	936,266 ms	936,604 ms

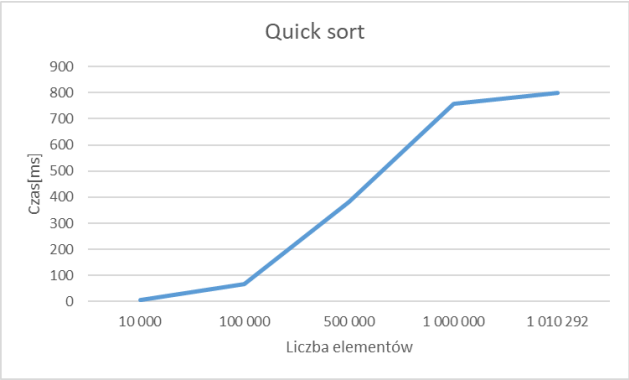
Tabela 4: Porównanie czsów wykonywania operacji w zależności od ilości elementów dla tablicy dynamicznej.



Rysunek 1: Czas filtrowania wyników

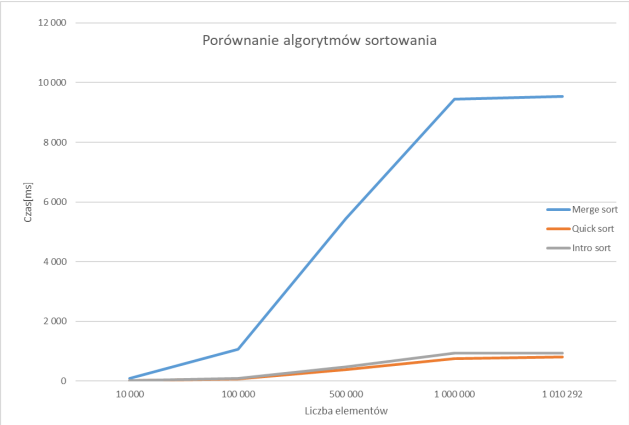


Rysunek 2: Wykres merge sort



Rysunek 3: Wykres quick sort

Rysunek 4: Wykres intro sort



Rysunek 5: Porównanie algorytmów sortowania.

2.3 Tabele pomiarowe i wykresy.

Parametr	Ilość elementów				
——	10 000	100 000	500 000	1 000 000	1 010 292
Wartość średnia	5.46	6.09	6.67	6.64	6.64
Mediana	5	7	7	7	7

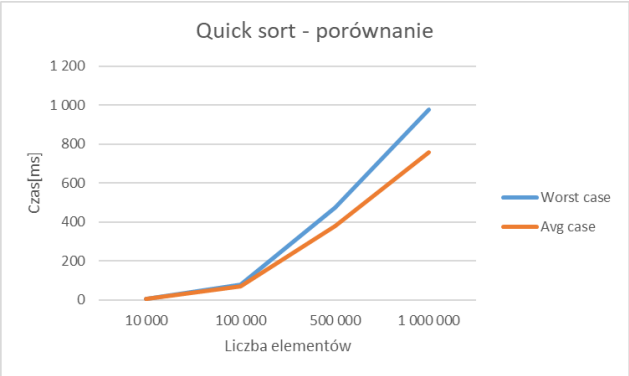
Tabela 5: Wartość średnia oraz mediana ocen rankingu, dla danej ilości elementów.

2.4 Porównanie różnych scenariuszy quick sort

Podczas badań chciałem przeprowadzić porównanie quick sort'a dla różnych danych wejściowych. W pierwszym przypadku algorytm dostaje losowo rozmieszczone filmy z ocenami a pivot wybierany jest jako mediana z pierwszej, środkowej i ostatniej oceny. W drugim przypadku algorytm dostaje posortowane dane i jako pivot wybiera zawsze ocenę pierwszego elementu.

Scenariusz	Ilość elementów			
——	10 000	100 000	500 000	1 000 000
Avg case	3,161 ms	67,569 ms	381,193 ms	757,248 ms
Worst case	4,830 ms	76,214 ms	473,761 ms	976,599 ms

Tabela 6: Porównanie quick sort dla różnych danych wejściowych.



Rysunek 6: Porównanie sortowania szybkiego.

3 Podsumowanie.

3.1 Wnioski

- Z wykresu 6 wynika, że quick sort jest najszybszym algorytmem spośród zaprezentowanych. Jego jedynym problemem jest to, że w niektórych przypadkach jego złożoność obliczeniowa, może się pogorszyć do $O(n^2)$.
- Algorytm intro sort oferuje bardzo szybki czas sortowania (niewiele gorszy od quick sort), a dodatkowo eliminuje on możliwość pogorszenia się złożoności obliczeniowej do $O(n^2)$.
- Merge sort jest najwolniejszym algorytmem spośród 3 zaprezentowanych algorytmów. Jego plusem jest to, że jego złożoność obliczeniowa jest gwarantowana (brak występowania problemu $O(n^2)$ jak w przypadku quick sort) oraz to, że jest to algorytm stabilny.[1]
- W punkcie 2.4, podczas badań nie udało mi się uzyskać złożoności $O(n^2)$. Gdy zbyt mocno modyfikowałem kod, quick sort tworzył zbyt dużo rekurencji i przepełniał stos programowy (stack overflow). Przy lekkiej modyfikacji kodu złożoność obliczeniowa nie wzrastała do poziomu $O(n^2)$. Niemniej na rysunku 6 możemy zauważyć wzrost czasu sortowania przy zmianie danych wejściowych. Przy odpowiedniej modyfikacji danych wejściowych oraz wybranie "odpowiedniego" pivot'u jesteśmy w stanie spowolnić algorytm quick sort.

Literatura

Materiały pomocnicze:
Algorytmy sortujące. Polska. I-LO w Tarnowie. Wrzesień 2015. [dostęp 30.03.2024]:
https://eduinf.waw.pl/inf/alg/003˙sort/0013.php
https://eduinf.waw.pl/inf/alg/003˙sort/0018.php
Sortowanie introspektywne. Polska. Wikipedia. 4 wrzesień 2021. [dostęp 30.03.2024]:
https://pl.wikipedia.org/wiki/Sortowanie˙introspektywne
Quick sort - sortowanie szybkie. Polska. Programowanie i algorytmy. 2023. [dostęp 05.04.2024]
https://www.algorytm.edu.pl/algorytmy-maturalne/quick-sort.html