**Dane studenta:** Filip Kurasz 249458 27.03.2020

**Nazwa kursu:** Projektowanie algorytmów i metody sztucznej inteligencji

**Dane prowadzącego:** mgr inż. Marta Emirsajłow

**Termin zajęć:** piątek 13:15-15:00

**SPRAWOZDANIE**

**PROJEKT NR 1**

**ALGORYTMY SORTOWANIA**

1. Wprowadzenie

Należało wybrać trzy algorytmy sortowania, zaimplementować i przeanalizować ich efektywność. W poniższym sprawozdaniu wykorzystane zostały następujące algorytmy sortowania: szybkie (quicksort), przez scalanie (mergesort), introspektywne (introsort).

Wszystkie algorytmy testowane były na 100 tablicach o następujących rozmiarach:

10.000, 50.000, 100.000, 500.000, 1.000.000.

W dodatku dla każdego rozmiaru były wymuszane warunki początkowe:

Cała tablica wypełniona losowymi liczbami, tablica posortowana w 25%, w 50%, w 75%, w 95%, w 99%, w 99.7% oraz posortowana malejąco.

1. Opis badanych algorytmów
2. QuickSort

Sortowanie szybkie jest jednym z najpopularniejszych algorytmów sortowań działający na zasadzie „dziel i zwyciężaj”. Jest wydajny, a jego złożoność obliczeniowa jest rzędu , natomiast w przypadku pesymistycznym wynosi ona . Jest powszechnie używany ze względu na szybkość dzięki zastosowaniu rekurencji.

1. MergeSort

Rekurencyjny algorytm sortowania stosujący również zasadę „dziel i zwyciężaj”. Szczególnie przydatny przy danych dostępnych sekwencyjnie, na przykład w postaci listy jednokierunkowej. Jego złożoność obliczeniowa wynosi , w najgorszym przypadku również .

1. IntroSort

Odmiana sortowania hybrydowego, w której wyeliminowany został problem złożoności występującej w najgorszym przypadku algorytmu sortowania szybkiego. Głównym założeniem algorytmu jest obsługa najgorszego przypadku algorytmu sortowania szybkiego tak, aby zapewnić logarytmiczno-liniową złożoność obliczeniową. Jego złożoność obliczeniowa wynosi również dla przypadku najgorszego.

1. Przebieg badanych algorytmów
2. QuickSort – Wyniki w sekundach

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ROZMIAR/PROCENT | 0 | 25 | 50 | 75 | 95 | 99 | 99,7 | ODWROTNIE |
| 10000 | 0,42 | 0,415 | 0,414 | 0,352 | 0,281 | 0,244 | 0,227 | 0,149 |
| 50000 | 2,477 | 2,372 | 2,355 | 2,071 | 1,679 | 1,388 | 1,316 | 0,953 |
| 100000 | 5,377 | 5,328 | 6,356 | 4,871 | 3,729 | 3,271 | 3,116 | 2,197 |
| 500000 | 32,189 | 30,482 | 44,114 | 27,618 | 23,538 | 19,471 | 18,185 | 14,205 |
| 1000000 | 65,391 | 64,311 | 105,89 | 58,107 | 48,363 | 42,693 | 40,628 | 30,951 |

1. MergeSort – Wyniki w sekundach

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ROZMIAR/PROCENT | 0 | 25 | 50 | 75 | 95 | 99 | 99,7 | ODWROTNIE |
| 10000 | 0,865 | 0,869 | 0,89 | 0,825 | 0,838 | 0,821 | 0,811 | 0,312 |
| 50000 | 4,588 | 4,531 | 4,357 | 4,289 | 4,115 | 4,059 | 4,103 | 4,227 |
| 100000 | 9,651 | 9,54 | 9,453 | 9,312 | 9,014 | 9,196 | 9,303 | 9,241 |
| 500000 | 52,142 | 51,761 | 53,658 | 44,33 | 42,146 | 42,634 | 43,762 | 43,709 |
| 1000000 | 99,534 | 97,323 | 91,872 | 89,719 | 89,643 | 89,154 | 88,163 | 89,481 |

1. IntroSort – Wyniki w sekundach

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ROZMIAR/PROCENT | 0 | 25 | 50 | 75 | 95 | 99 | 99,7 | ODWROTNIE |
| 10000 | 0,357 | 0,369 | 0,528 | 0,273 | 0,358 | 0,17 | 0,126 | 0,081 |
| 50000 | 2,046 | 2,038 | 3,78 | 1,622 | 2,031 | 1,182 | 0,799 | 0,464 |
| 100000 | 4,375 | 4,603 | 5,396 | 3,391 | 2,458 | 2,01 | 1,878 | 1,01 |
| 500000 | 25,136 | 24,719 | 31,707 | 19,315 | 14,073 | 11,787 | 6,047 | 3,118 |
| 1000000 | 50,933 | 49,617 | 62,18 | 42,828 | 34,754 | 27,147 | 26,766 | 17,382 |

1. Podsumowanie

Z wykresów odczytać można, że początkowa ilość posortowanych elementów wpływa na czas końcowy. Im więcej elementów posortowanych na początków tym szybszy czas posortowania elementów. Jednak w przypadku sortowania szybkiego występuje złożoność dla przypadku gdzie początkowe elementy są posortowane w 50%. W tym przypadku sortowanie introspektywne używa sortowania przez kopcowanie. Przez co sortowanie introspektywne nigdy nie osiąga złożoności obliczeniowej . Mergesort ma stałą złożoności obliczeniową, jednak ze względu na swoją budowę jego czas jest dłuższy niż Quicksort.

1. Bibliografia

<https://en.wikipedia.org/wiki/Merge_sort>

<https://en.wikipedia.org/wiki/Quicksort>

<https://en.wikipedia.org/wiki/Introsort>

<https://www.geeksforgeeks.org/>

<https://stackoverflow.com/>

<https://www.youtube.com/channel/UC0RhatS1pyxInC00YKjjBqQ>