Projektowanie algorytmów i metody sztucznej inteligencji

Bartosz Gulla Piątek 7:30

mgr inż. Marta Emirsajłow

1. Wprowadzenie:

Tematem projektu było wykonanie programu, którego zadaniem było sortowanie tablic o różnych wielkościach (10000, 50000, 100000, 500000, 1000000) o różnej ilości wstępnie posortowanych elementów (0%, 25%, 50%, 75%, 95%, 99%, 99,7% oraz odwrotnie posortowane) używając 3 różnych algorytmów. Do algorytmów należały: Qucksort Mergesort oraz Introsort. Wszystkie te algorytmy zaliczają się do algorytmów szybkich o logarytmicznej złożoności obliczeniowej (wyjątek stanowi pesymistyczny wypadek sortowania Quicksort).

1. Użyte algorytmy:
   1. Quicksort:

Quicksort jest algorytmem rekurencyjnym którego w każdym kroku sortowania szybkiego zostaje wybrany element służący do podziału tablicy. Po czym algorytm porównuje wszystkie elementy tablicy z wybranym i tworzy 2 nowe tablice, jedna zawierająca elementy mniejsze, a druga zawierająca większe. Element wybrany do podziału nie bierze dalej udziału w sortowaniu, ponieważ jest już na swojej pozycji. Algorytm jest wykonywany do uzyskania posortowanej tablicy.

Złożoność obliczeniowa zależy od wyboru elementu służącego do podziału, tak więc w wypadku pesymistycznym, czyli takim w którym wybrany zostaje element największy lub najmniejszy wielkość tablic zmniejsza się zawsze o jeden więc złożoność obliczeniowa wynosi . W wypadku średnim złożoność zależy od wybranego elementu, tak więc w takim wypadku złożoność obliczeniowa wynosi około .

* 1. Mergesort:

Mergesort jest kolejnym przykładem algorytmu rekurencyjnego który w każdym kroku dzieli tablice na połowę aż do uzyskania tablic jednoelementowych które są uznawane za posortowane. Po tym scala je w większe tablice porównując kolejne elementy podtablic.

Można zauważyć, że mergesort zawsze dzieli tablicę na pół, tak więc wypadek średni oraz pesymistyczny różnią się tylko ilością porównań które i tak zazwyczaj zostałyby wykonane dlatego złożoność obliczeniowa dla obu wypadków wynosi

* 1. Introsort:

Introsort jest przykładem algorytmu hybrydowego tz. Składa się z kilku innych algorytmów, w wypadku introsorta są to:Quicksort Insertsort oraz Heapsort.

Algorytm ma za zadanie wyeliminować słabe strony wszystkich tych algorytmów np. eliminując wypadek dla quicksorta ustalając maksymalną głębokość rekurencji jako i wywołując funkcję introsort z głębokością o jeden mniejszą. W wypadku Gdy głębokość będzie równa 0 wykonuje się Heapsort. Aby usprawnić algorytm wykorzystano również algorytm Insertsort który jest wykorzystywany tylko dla tablic mniejszych niż 16 elementów.   
Złożoność obliczeniowa w wypadku średnim jest taka sama jak w wypadku quicksorta czyli . W wypadku pesymistycznym wynosi :

1. Wyniki eksperymentów:

|  |  |  |  |
| --- | --- | --- | --- |
| Mergesort | 10000 | 0% | 0,287 |
| Mergesort | 50000 | 0% | 1,528 |
| Mergesort | 100000 | 0% | 3,088 |
| Mergesort | 500000 | 0% | 14,912 |
| Mergesort | 1000000 | 0% | 56,257 |
| Mergesort | 10000 | 25% | 0,491 |
| Mergesort | 50000 | 25% | 2,664 |
| Mergesort | 100000 | 25% | 5,642 |
| Mergesort | 500000 | 25% | 29,697 |
| Mergesort | 1000000 | 25% | 63,509 |
| Mergesort | 10000 | 50% | 0,452 |
| Mergesort | 50000 | 50% | 2,363 |
| Mergesort | 100000 | 50% | 4,938 |
| Mergesort | 500000 | 50% | 26,811 |
| Mergesort | 1000000 | 50% | 55,554 |
| Mergesort | 10000 | 75% | 0,402 |
| Mergesort | 50000 | 75% | 2,177 |
| Mergesort | 100000 | 75% | 4,392 |
| Mergesort | 500000 | 75% | 23,804 |
| Mergesort | 1000000 | 75% | 49,534 |
| Mergesort | 10000 | 95% | 0,352 |
| Mergesort | 50000 | 95% | 1,921 |
| Mergesort | 100000 | 95% | 3,966 |
| Mergesort | 500000 | 95% | 21,439 |
| Mergesort | 1000000 | 95% | 44,748 |
| Mergesort | 10000 | 99% | 0,347 |
| Mergesort | 50000 | 99% | 1,875 |
| Mergesort | 100000 | 99% | 3,898 |
| Mergesort | 500000 | 99% | 24,934 |
| Mergesort | 1000000 | 99% | 43,433 |
| Mergesort | 10000 | 99,7% | 0,34 |
| Mergesort | 50000 | 99,7% | 1,891 |
| Mergesort | 100000 | 99,7% | 3,873 |
| Mergesort | 500000 | 99,7% | 21,131 |
| Mergesort | 1000000 | 99,7% | 43,913 |
| Mergesort | 10000 | rev | 0,471 |
| Mergesort | 50000 | rev | 1,878 |
| Mergesort | 100000 | rev | 3,889 |
| Mergesort | 500000 | rev | 21,085 |
| Mergesort | 1000000 | rev | 43,931 |

|  |  |  |  |
| --- | --- | --- | --- |
| Quicksort | 10000 | 0% | 0,179 |
| Quicksort | 50000 | 0% | 0,931 |
| Quicksort | 100000 | 0% | 1,826 |
| Quicksort | 500000 | 0% | 8,64 |
| Quicksort | 1000000 | 0% | 16,574 |
| Quicksort | 10000 | 25% | 0,256 |
| Quicksort | 50000 | 25% | 1,454 |
| Quicksort | 100000 | 25% | 2,785 |
| Quicksort | 500000 | 25% | 13,909 |
| Quicksort | 1000000 | 25% | 31,173 |
| Quicksort | 10000 | 50% | 0,197 |
| Quicksort | 50000 | 50% | 1,086 |
| Quicksort | 100000 | 50% | 2,292 |
| Quicksort | 500000 | 50% | 12,386 |
| Quicksort | 1000000 | 50% | 24,609 |
| Quicksort | 10000 | 75% | 0,137 |
| Quicksort | 50000 | 75% | 0,772 |
| Quicksort | 100000 | 75% | 1,608 |
| Quicksort | 500000 | 75% | 9,264 |
| Quicksort | 1000000 | 75% | 19,05 |
| Quicksort | 10000 | 95% | 0,1 |
| Quicksort | 50000 | 95% | 0,536 |
| Quicksort | 100000 | 95% | 1,099 |
| Quicksort | 500000 | 95% | 6,188 |
| Quicksort | 1000000 | 95% | 12,914 |
| Quicksort | 10000 | 99% | 0,083 |
| Quicksort | 50000 | 99% | 0,485 |
| Quicksort | 100000 | 99% | 1,003 |
| Quicksort | 500000 | 99% | 5,707 |
| Quicksort | 1000000 | 99% | 12,598 |
| Quicksort | 10000 | 99,7% | 0,084 |
| Quicksort | 50000 | 99,7% | 0,512 |
| Quicksort | 100000 | 99,7% | 1,011 |
| Quicksort | 500000 | 99,7% | 5,566 |
| Quicksort | 1000000 | 99,7% | 11,766 |
| Quicksort | 10000 | rev | 0,198 |
| Quicksort | 50000 | rev | 0,489 |
| Quicksort | 100000 | rev | 1,042 |
| Quicksort | 500000 | rev | 5,784 |
| Quicksort | 1000000 | rev | 11,739 |

|  |  |  |  |
| --- | --- | --- | --- |
| Introsort | 10000 | 0% | 0,15 |
| Introsort | 50000 | 0% | 0,80 |
| Introsort | 100000 | 0% | 1,50 |
| Introsort | 500000 | 0% | 7,42 |
| Introsort | 1000000 | 0% | 33,55 |
| Introsort | 10000 | 25% | 0,21 |
| Introsort | 50000 | 25% | 1,25 |
| Introsort | 100000 | 25% | 2,90 |
| Introsort | 500000 | 25% | 13,78 |
| Introsort | 1000000 | 25% | 27,68 |
| Introsort | 10000 | 50% | 0,16 |
| Introsort | 50000 | 50% | 0,92 |
| Introsort | 100000 | 50% | 1,93 |
| Introsort | 500000 | 50% | 10,75 |
| Introsort | 1000000 | 50% | 21,87 |
| Introsort | 10000 | 75% | 0,11 |
| Introsort | 50000 | 75% | 0,61 |
| Introsort | 100000 | 75% | 1,32 |
| Introsort | 500000 | 75% | 7,72 |
| Introsort | 1000000 | 75% | 15,89 |
| Introsort | 10000 | 95% | 0,07 |
| Introsort | 50000 | 95% | 0,41 |
| Introsort | 100000 | 95% | 0,87 |
| Introsort | 500000 | 95% | 5,00 |
| Introsort | 1000000 | 95% | 10,54 |
| Introsort | 10000 | 99% | 0,06 |
| Introsort | 50000 | 99% | 0,40 |
| Introsort | 100000 | 99% | 0,79 |
| Introsort | 500000 | 99% | 5,32 |
| Introsort | 1000000 | 99% | 9,51 |
| Introsort | 10000 | 99,7% | 0,07 |
| Introsort | 50000 | 99,7% | 0,39 |
| Introsort | 100000 | 99,7% | 0,77 |
| Introsort | 500000 | 99,7% | 4,45 |
| Introsort | 1000000 | 99,7% | 9,62 |
| Introsort | 10000 | rev | 0,07 |
| Introsort | 50000 | rev | 0,41 |
| Introsort | 100000 | rev | 0,80 |
| Introsort | 500000 | rev | 4,55 |
| Introsort | 1000000 | rev | 9,92 |

1. Podsumowanie i wnioski:

Wszystkie algorytmy z zadaniem poradziły sobie bardzo dobrze, w żadnym wypadku quicksorta nie wystąpił wypadek pesymistyczny. Ze wszystkich algorytmów najgorzej wypadł Mergesort który był średnio 3 razy wolniejszy co najprawdopodobniej wynika z przymusu tworzenia tymczasowych dynamicznych tablic, które są wykorzystywane do scalenia. Najszybciej posortowane zostały tablice o pierwszych 99% posortowanych elementach co pozwoliło skrócić czas około 2-krotnie względem nieposortowanych elementów.

1. Bibliografia:
   1. <https://pl.wikipedia.org/wiki/Sortowanie_introspektywne>
   2. <https://pl.wikipedia.org/wiki/Sortowanie_szybkie>
   3. <https://pl.wikipedia.org/wiki/Sortowanie_przez_scalanie>
   4. <http://www.algorytm.org/algorytmy-sortowania/sortowanie-przez-scalanie-mergesort/merge-1-c.html>
   5. <https://www.geeksforgeeks.org/heap-sort/>