Bartosz Gulla

Zajęcia: Piątek 7:30

*Projektowanie algorytmów i metody sztucznej inteligencji*

**mgr inż. Marta Emirsajłow**

1. *Wprowadzenie:*

Celem projektu było zapoznanie się z różnymi algorytmami stosowanymi do sortowania elementów oraz przygotowanie programu, którego zadaniem było przygotowanie 10 tablic wieloelementowych (10 tyś, 50 tyś, 100 tyś, 500 tyś, 1 Milion) o różnych warunkach początkowych (Wszystkie elementy losowe, 25% pierwszych elementów posortowane, 50% pierwszych elementów posortowane, 75% pierwszych elementów posortowane, 95% pierwszych elementów posortowane, 99% pierwszych elementów posortowane, 99,7% pierwszych elementów posortowane, wszystkie elementy posortowane odwrotnie), następnie posortowanie 3 różnymi metodami oraz jednoczesne zmierzenie czasu sortowania 100 tablic.

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 to znaczy, że 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:*
   1. Inrosort:
      1. Tabela:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Introsort** | | **Ilość elementów** | | | | |
| **10000** | **50000** | **100000** | **500000** | **1000000** |
| **Wstępne posortowanie talblicy** | **0%** | 0,151 | 0,795 | 1,496 | 7,415 | 33,551 |
| **25%** | 0,21 | 1,245 | 2,897 | 13,778 | 27,684 |
| **50%** | 0,161 | 0,921 | 1,926 | 10,753 | 21,871 |
| **75%** | 0,109 | 0,606 | 1,322 | 7,717 | 15,888 |
| **95%** | 0,074 | 0,41 | 0,867 | 4,996 | 10,538 |
| **99%** | 0,064 | 0,403 | 0,788 | 5,321 | 9,508 |
| **99,7%** | 0,067 | 0,385 | 0,773 | 4,453 | 9,616 |
| **rev** | 0,065 | 0,412 | 0,799 | 4,545 | 9,921 |

* + 1. Wykres:
  1. Mergesort:
     1. Tabela:

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

* + 1. Wykres:
  1. Quicksort:
     1. Tabela:

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

* + 1. Wykres:

1. *Podsumowanie i wnioski:*
2. Wszystkie algorytmy z zadaniem poradziły sobie bardzo dobrze, w żadnym wypadku quicksorta nie wystąpił wypadek pesymistyczny.
3. Ze wszystkich algorytmów najgorzej wypadł mergesort, który był około 4 razy wolniejszy, co najprawdopodobniej wynika z przymusu tworzenia tymczasowych dynamicznych tablic, które są wykorzystywane do scalenia, lepszy od niego był algorytm quicksort, najlepszy okazał się algorytm introsort jednak nie był on dużo szybszy od quicksorta,
4. Najszybciej posortowane zostały tablice o pierwszych 99% posortowanych elementów, co pozwoliło skrócić czas około 2-krotnie względem nieposortowanych elementów.
5. Średnia złożoność obliczeniowa dla wszystkich algorytmów wynosi co przedstawiają wykresy, jednakże nie jest to idealne odwzorowanie funkcji logarytmicznej ze względu na zbyt małą liczbę punktów pomiarowych wszak było ich tylko pięć,
6. Wszelkie odchylenia względem czasów przewidywanych mogą byś spowodowane sposobem implementacji sortowania, w którym korzystałem ze 100 dynamicznych tablic jednowymiarowych tworzonych, usuwanych oraz uzupełnianych w pętli, której czas jest mierzony również mogą one być spowodowane ograniczeniami sprzętowymi komputera, na którym eksperyment był wykonywany.
7. 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/>