29.11.2023 Anna Jasielec

Grupa 4

**Porównanie różnych algorytmów sortujących**

Pod uwagę bierzemy takie metody sortowania jak: sortowanie bąbelkowe, sortowanie przez wstawianie, sortowanie przez selekcje, sortowanie grzebieniowe oraz sortowanie metodą shella. Metody te zostały zaimplementowane i były wykonywane dla tablicy losowych elementów. Czas był mierzony z pomocą funkcji clock(). Poniżej przedstawiono porównanie czasu wykonywania się poszczególnych metod oraz liczby operacji w zależności od rozmiaru nieposortowanej tablicy.

1. **Wyniki:**

Liczba operacji i czas poszczególnych sortowań tablicy o rozmiarze 1 000:

Obraz zawierający tekst, zrzut ekranu, Czcionka, czarne

Opis wygenerowany automatycznie

Liczba operacji i czas poszczególnych sortowań tablicy o rozmiarze 5 000:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Liczba operacji i czas poszczególnych sortowań tablicy o rozmiarze 10 000:

Obraz zawierający tekst, zrzut ekranu, Czcionka, czarne

Opis wygenerowany automatycznie

Liczba operacji i czas poszczególnych sortowań tablicy o rozmiarze 50 000:

Obraz zawierający tekst, zrzut ekranu, Czcionka, czarne

Opis wygenerowany automatycznie

Liczba operacji i czas poszczególnych sortowań tablicy o rozmiarze 100 000:

Obraz zawierający tekst, zrzut ekranu, Czcionka, czarne

Opis wygenerowany automatycznie

1. **Analiza wyników.**

Tabelaryczne porównanie szybkości poszczególnych algorytmów w zależności od rozmiaru tablicy:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rozmiar tablicy | **Sortowanie bąbelkowe** | **Sortowanie przez wstawianie** | **Sortowanie przez selekcje** | **Sortowanie grzebieniowe** | **Sortowanie metodą Shella** |
| 1 000 | 0,003 s | 0,001 s | 0,001 s | 0,001 s | 0 s |
| 5 000 | 0,064 s | 0,016 s | 0,029 s | 0,001 s | 0,001 s |
| 10 000 | 0,279 s | 0,065 s | 0,113 s | 0,002 s | 0,002 s |
| 50 000 | 8,177 s | 1,57 s | 2,722 s | 0,009 s | 0,008 s |
| 100 000 | 32,74 s | 6,31 s | 10,898 s | 0,017 s | 0,018 s |

Tabelaryczne porównanie ilości operacji poszczególnych algorytmów w zależności od rozmiaru tablicy:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rozmiar tablicy | **Sortowanie bąbelkowe** | **Sortowanie przez wstawianie** | **Sortowanie przez selekcje** | **Sortowanie grzebieniowe** | **Sortowanie metodą Shella** |
| 1 000 | 736104 | 710808 | 508082 | 32100 | 40957 |
| 5 000 | 18720457 | 18673867 | 12542968 | 186229 | 265461 |
| 10 000 | 74683447 | 74075337 | 50089635 | 414224 | 623712 |
| 50 000 | 1874207604 | 1872747808 | 1250444191 | 2431760 | 3996017 |
| 100 000 | 7500368644 | 7501355928 | 5000898606 | 4856961 | 8764302 |

1. **Wnioski.**

Najlepszą metodą okazało się być sortowanie grzebieniowe, należy ono do metod o złożoności liniowo-logarytmicznej. Jest najbardziej optymalną z wyżej wymienionych metod, ponieważ oprócz bardzo szybkiego czasu wykonywania, cechuje się najmniejszą liczbą operacji i prostotą implementacji. Dodatkowo nie wymaga dodatkowej pamięci, może być stosowane zarówno w przypadku małych jak i dużych zbiorów liczb.

Niewiele gorsza była metoda Shella. Różnica pomiędzy sortowaniem grzebieniowym była dostrzegalna dopiero przy bardzo dużych zbiorach liczb. Sortowanie to daje sobie radę nawet z bardzo dużymi zbiorami liczb, może być stosowane w przypadkach, gdy zrównoważyć prostotę implementacji z umiarkowaną wydajnością sortowania.

Kolejno mamy sortowanie przez wstawianie, które jest prostym i intuicyjnym algorytmem. Może być skuteczny w małych zbiorów, ze względu na swoją prostotę. Jest ono algorytmem "online", co oznacza, że dane mogą być sortowane w miarę ich napływania.

Następne jest sortowanie przez selekcje, którego wydajność była ograniczona w porównaniu do bardziej zaawansowanych algorytmów. Jest prostym algorytmem i znajduje zastosowanie w małych zbiorach danych, zwłaszcza gdy inne bardziej zaawansowane algorytmy są zbyt złożone lub kosztowne.

Najgorzej w całym zestawieniu wypadło sortowanie bąbelkowe. Jest bardzo proste w implementacji, jednak ze względu na swoją małą wydajność dla większych zbiorów danych, rzadko jest stosowane w praktyce. Sprawuje się jedynie przy niewielkich zbiorach danych.