Badanie efektywności wybranych algorytmów sortowania ze względu na złożoność obliczeniową

AiZO - zadanie projektowe nr 1

Autor : Arkadiusz Błasiak

Nr albumu: 273022

Data: 18.04.2024 godz. 11.15

# Wprowadzenie

### Pojęcie złożoności obliczeniowej

Złożoność obliczeniowa algorytmu określa, jak wydajny jest algorytm, ile musi on wykonać operacji w zależności ilości danych oraz ile potrzebuje do tego pamięci. Często zdarza się, że dany problem algorytmiczny można rozwiązać kilkoma metodami, czyli algorytmami o różnej złożoności obliczeniowej. Złożoność obliczeniową dzielimy na złożoność pamięciową oraz złożoność czasową.

W tym projekcie pomijamy złożoność pamięciową i skupiamy się w pełnie na złożoności czasowej.

### Kilka ważnych pojęć:

* złożoność czasowa

ilość czasu potrzebnego do wykonania zadania, wyrażona jako funkcja ilości danych.

* złożoność obliczeniowa

ilość zasobów komputerowych potrzebnych do wykonania zadania.

* złożoność oczekiwana

inaczej złożoność średnia; ilość zasobów potrzebna do zrealizowania zadania dla statystycznie oczekiwanych danych; zapisywana za pomocą notacji theta –

* złożoność pesymistyczna

ilość zasobów potrzebna do zrealizowania zadania w przypadku najgorszych danych; zapisywana za pomocą notacji „duże O” –

# Algorytmy sortowania

## Sortowanie przez wstawanie (Insertionsort)

Jeden z najprostszych algorytmów sortowania, którego zasada działania odzwierciedla sposób w jaki ludzie ustawiają karty – kolejne elementy wejściowe są ustawiane na odpowiednie miejsca docelowe. Jego zalety: jest stabilny, jest wydajny dla zbiorów o niewielkiej liczebności, liczba wykonanych porównań jest zależna od liczby inwersji w permutacji, dlatego algorytm jest wydajny dla danych wstępnie posortowanych.

Złożoność oczekiwana:

Złożoność pesymistyczna:

## Sortowanie Shella (Shellsort)

Jeden z algorytmów sortowania działających w miejscu i korzystających z porównań elementów. Można go traktować jako uogólnienie sortowania przez wstawianie lub sortowania bąbelkowego, dopuszczające porównania i zamiany elementów położonych daleko od siebie. Na początku sortuje on elementy tablicy położone daleko od siebie, a następnie stopniowo zmniejsza odstępy między sortowanymi elementami. Dzięki temu może je przenieść w docelowe położenie szybciej niż zwykłe sortowanie przez wstawianie.

Złożoność wersji Shella:

Złożoność oczekiwana:

Złożoność pesymistyczna:

Złożoność wersji Knutha:

Złożoność oczekiwana: brak danych

Złożoność pesymistyczna:

## Sortowanie przez kopcowanie (Heapsort)

Jeden z algorytmów sortowania, choć niestabilny, to jednak szybki i niepochłaniający wiele pamięci. Podstawą algorytmu jest użycie kolejki priorytetowej zaimplementowanej w postaci binarnego kopca zupełnego. Zasadniczą zaletą kopców jest stały czas dostępu do elementu maksymalnego (lub minimalnego) oraz logarytmiczny czas wstawiania i usuwania elementów; ponadto łatwo można go implementować w postaci tablicy.

Złożoność oczekiwana:

Złożoność pesymistyczna:

## Sortowanie szybkie (Quicksort)

jeden z popularnych algorytmów sortowania działających na zasadzie „dziel i zwyciężaj”. Algorytm sortowania szybkiego jest wydajny. Ze względu na szybkość i prostotę implementacji jest powszechnie używany. Jego implementacje znajdują się w bibliotekach standardowych wielu środowisk programowania.

Złożoność oczekiwana:

Złożoność pesymistyczna:

# Plan eksperymentu

# Pomiary

## Sortowanie przez wstawanie (Insertionsort)

# Źródła

* <https://www.algorytm.edu.pl/matura-informatyka/zlozonosc-algorytmu>
* <https://zpe.gov.pl/a/przeczytaj/DNDIkCVVg>
* <https://wikipedia.org/>