Marek Pierzchała 200533

**Sprawozdanie**

1. Algorytmy sortowania – krótkie omówienie

* Sortowanie bąbelkowe:

Polega na sprawdzaniu dwóch elementów znajdujących się obok siebie, jeżeli porządek nie jest zachowany następuje zamiana miejsc, czynność powtarzana jest aż do posortowania danych

Złożoność obliczeniowa wynosi: O(n2)

* Sortowanie przez kopcowanie:

Oparte jest na dwóch czynnościach, na początku z zadanych danych budujemy kopiec binarny, następnie usuwamy korzeń (wartość minimalna) i w jego miejsce wstawiamy ostatni element kopca, po czym naprawiamy kopiec, czynność powtarzamy aż do opróżnienia kopca

Złożoność obliczeniowa wynosi: O(nlogn)

* Sortowanie Shella

Polega na tym, że sortowany zbiór dzielimy na podzbiory, które są oddalone od siebie o pewien odstęp (w moim przypadku ). Każdy z tych zbiorów sortujemy przez wstawianie a następnie dalej dzielimy, czynność powtarzamy do momentu, aż odstęp będzie wynosił 1

Złożoność obliczeniowa wynosi: O(n2)

1. Wykresy

Dla wszystkich algorytmów testowałem trzy typy danych:

* Losowe liczby
* Prawie posortowane liczby
* Liczby posortowane odwrotnie

Dla sortowania bąbelkowego zgodnie z przewidywaniami pesymistyczny okazał się zbiór liczb posortowanych odwrotnie, typowym zbiór liczb losowych, zaś zbiór liczb prawie posortowanych okazał się optymistycznym.

Złożoności obliczeniowe:

* Liczby posortowane odwrotnie - 51,37
* Liczby losowe - 38,59
* Liczby prawie posortowane - 25,42

Wady:

* Duża złożoność obliczeniowa
* Powolny

Zalety:

* Łatwa implementacja

Dla sortowania shella pesymistyczny okazał się zbiór liczb losowych, typowym zbiór liczb prawie posortowanych, zaś zbiór liczb posortowanych odwrotnie okazał się optymistycznym.

Można zauważyć, że algorytm dobrze radzi sobie z sortowaniem zbiorów w znacznym stopniu uporządkowanych.

Złożoności obliczeniowe:

* Liczby posortowane odwrotnie - 2,31
* Liczby losowe - 6,63
* Liczby prawie posortowane - 3,75

Wady:

* Trudniejsza implementacja
* Gorzej radzi sobie ze zbiorem liczb losowych

Zalety:

* Mała złożoność obliczeniowa
* Świetnie radzi sobie z sortowaniem w znacznym stopniu uporządkowanym
* Zdecydowanie najszybszy z wszystkich badanych przeze mnie algorytmów

Dla sortowania przez kopcowanie pesymistyczny okazał się zbiór liczb losowych, typowym zbiór liczb prawie posortowanych, zaś zbiór liczb posortowanych odwrotnie okazał się optymistycznym.

Można zauważyć, że dla tego algorytmu złożoności obliczeniowe nie wiele się od siebie różnią, co może świadczyć o tym, że jest on mało czuły na postać danych wejściowych

Złożoności obliczeniowe:

* Liczby posortowane odwrotnie - 3,40
* Liczby losowe - 6,60
* Liczby prawie posortowane - 3,72

Wady:

* Trudniejsza implementacja
* Wolniejszy, niż Shella

Zalety:

* Mało czuły na postać danych wejściowych
* Nie wymaga dodatkowej pamięci
* Mała złożoność obliczeniowa

Podsumowanie

Zdecydowanie najszybszym algorytmem okazał się algorytm sortowania Shella, sortowanie przez kopcowanie i Shella mają podobne złożoności obliczeniowe, zdecydowanie najgorszym algorytmem okazał się algorytm sortowania bąbelkowego

Parametry komputera

Procesor: Intel Core i5-2430M

Pamięć ram: 4 GB

Karta graficzna: AMD Radeon HD 6470M 1GB