**PROJEKTOWANIE ALGORYTMÓW i METOD SZTUCZNEJ INTELIGENCJI**

Dawid Sielużycki 241508

1. **Cele zadania**

Zadanie polegało na zbadaniu 3 algorytmów sortowań danych. Wybrałem następujące algorytmy: bąbelkowe, szybkie i przez scalanie. W programie zostały zainicjowane dodatkowe funkcje mające na celu usprawnienie wykonywania testów (zapisy posortowanych danych do plików tekstowych, osobny zapis czasów działania, funkcje usprawniające i została przewidziana metoda czytania danych z pliku).

1. **Algorytmy**
   1. Sortowanie bąbelkowe

Uważany jest za najprostszy w implementacji. Algorytm ten polega na porównaniu dwóch wartości u przesunięcie na wierzch mniejszej z nich. Z powodu bardzo wielu porównań, jedyne uzasadnione użycie ogranicza się do małej ilości danych. Ogólnie nie stosowany. Użyty został w tym zadaniu z powodu sentymentu do niego. Jego złożoność obliczeniowa .

|  |  |
| --- | --- |
| **Dane** | **Czas trwania [s]** |
| 100 | 0,000067 |
| 1k | 0,00675 |
| 10k | 0,7433 |
| 100k | 75,178 |
| 1M | 6969,61 |

* 1. Sortowanie szybkie

Algorytm ten jest często stosowany ze względu na prostotę implementacji i dobrej wydajności. Stosuje metodę „dziel i zwyciężaj”. Polega na podziale tablicy danych na 2 części względem punktu zwanego pivotem (u mnie jest to po prostu środkowy element). Następnie do pierwszej części przenoszone są elementy mniejsze od pivota, a do drugiej większe i następuje sortowanie obu części. W ogólnym przypadku złożoność obliczeniowa wynosi , jednak w przypadku pesymistycznym (zależnym od rodzaju danych jak i pivota) . Aby wyznaczyć złożoność pesymistyczną, użyłem pivota jako skrajnego lewego elementu, a dane były posortowane w odwrotnej kolejności.

|  |  |  |
| --- | --- | --- |
| **Dane** | **Czas trwania [s]**  **ogólny przypadek** | **Czas trwania [s]**  **pesymistyczny przypadek** |
| 100 | 0,000012 | 0,000015 |
| 1k | 0,00014 | 0,001024 |
| 10k | 0,00173 | 0,098 |
| 100k | 0,0207 | 9,76 |
| 1M | 0,24347 | 984,57 |

* 1. Sortowanie przez scalanie

Algorytm również stosujący metodę „dziel i zwyciężaj”. Sortowana tablica dzielona jest rekurencyjnie na dwie podtablice aż do uzyskania tablic jednoelementowych. Następnie podtablice te są scalane w odpowiedni sposób, dający w rezultacie tablicę posortowaną. Jedną z głównych wad jest konieczność tworzenia dodatkowej tablicy. Dodatkowo jest trudniejszy w implementacji niż Quick sort. Jego złożoność obliczeniowa to .

|  |  |
| --- | --- |
| **Dane** | **Czas trwania [s]** |
| 100 | 0,000013 |
| 1k | 0,00015 |
| 10k | 0,001937 |
| 100k | 0,0235 |
| 1M | 0,2158 |

1. **Wnioski**

Na wykresie zbiorczym widać, że zarówno Bubble sort jak i Quick sort w pesymistycznej wersji mają tą samą złożoność co charakteryzuje się jedynie przesunięciem wykresu funkcji.