**Algorytmy i Złożoność Obliczeniowa**

**Badanie efektywności wybranych algorytmów sortowania ze względu na złożoność obliczeniową**

|  |  |
| --- | --- |
| **Imię i Nazwisko** | **Krystyna Nowak** |
| **Numer indeksu** | **272890** |
| **Prowadzący** | **Zbigniew Buchalski** |
| **Termin zajęć** | **Gr 10, Wtorek parzysty 11:15** |

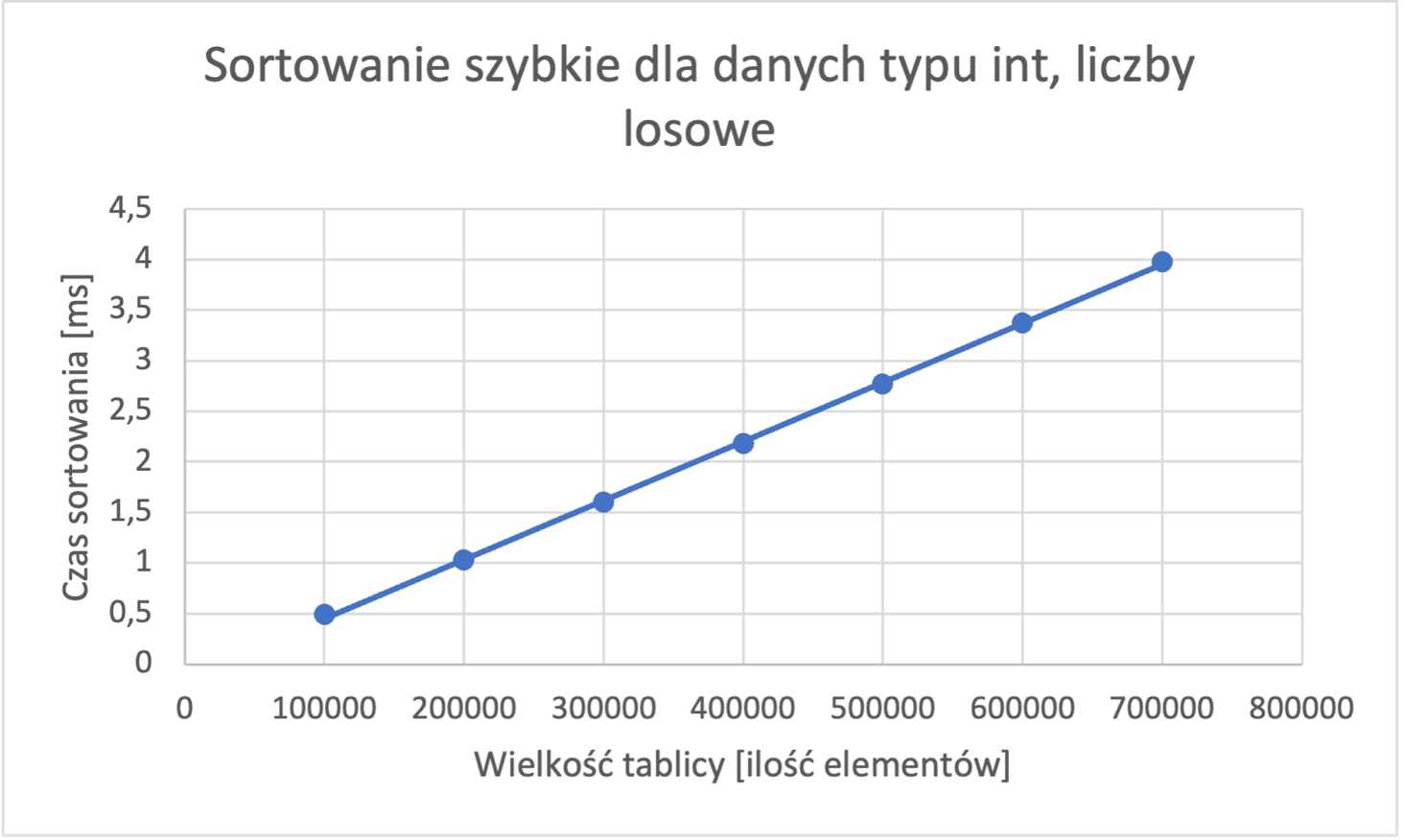
1. **Wstęp teoretyczny.**
2. **Sortowanie szybkie (Quick Sort)** – ten algorytm naley do jednych z najszybszych algorytmów sortujących dane. Złożoność czasowa algorytmu, w najlepszym oraz średnim przypadku, wynosi O(n log n). Ponadto do jego innych istotnych zalet należy również łatwość implementacji algorytmu, łatwość współpracy z różnymi typami danych oraz brak konieczności pracy na dodatkowych tablicach. Niestety algorytm posiada również wady, m.in. niestabilność, wrażliwość na błędy implementacji oraz złożoność w sytuacji pesymistycznej wynoszącą O(n2). Podany algorytm wykorzystuje techinkę „dziel i zwyciężaj”. Wybierany jest jeden element sortowanej tablicy nazywany pivotem, element ten może zostać wybrany za pomocą dowolnego schematu – skrajny lewy element, skrajny prawy, środkowy losowy itd. Następnie elementy większe od pivota ustawiane są po jednej jego stronie, a mniejsze po drugiej (w zależności czy sortujemy rosnąco czy malejąco). W taki sposób powstają dwie tablice (niekoniecznie równej wielkości), w których jedna zawiera elementy nie większe od elementów drugiej. Następnie sortujemy każdą z otrzymanych podtablic według tego samego schematu, aż otrzymamy tablice jedoelementowe. Na koniec jednoelementowe podtablice są łączone.
3. **Sortowanie Shella (Shell Sort)** – algorytm sortowania nazywany uogólnieniem sortowania przez wstawienie, bądź sortowania bąbelkowego. Jest to algorytm wieloprzebiegowy. Każdy z przebiegów polega na sortowaniu przez proste wstawienie elementów oddalonych o ustaloną liczbę miejsc. Pierwszy przebieg polega na podzieleniu danej tablicy na podtablice, dzieląc ją co n elementów. Następnie każda z podtablic sortowana jest pojedynczo odpowiednio zmodyfikowanym algorytmem przez wstawianie. Kolejne przebiegi polegają na zmniejszeniu n odpowiednio ustalonym sposobem elementów. Działanie to powtarzamy, aż do podzielenia tablicy na podtablice 1-elementowe. Sortowanie to nie jest stabilne, czyli może nie zachowywać wejściowej kolejności elementów o równych kluczach. Ponadto wykazuje ono zachowanie naturalne – krótszy czas sortowania dla częściowo uporządkowanych danych wejściowych. Złożoność obliczeniowa algorytmu Shella to w najlepszym przypadku O(n log n), w średnim O(n^(3/2)), a w najgorszym O(n^2). Wynika z tego, iż algorytm ten jest wydajniejszy od algorytmu sortowania przez wstawianie, jednak mniej efektywny od bardziej zaawansowanych algorytmów (np. sortowania szybkiego lub sortowania przez kopcowanie).
4. **Sortowanie przez kopcowanie** – niestabilny, ale pochłaniający mało pamięci i szybki algorytm sortowania. Jego złożoność czasowa wynosi O(n log n) w najlepszym, najgorszym i średnim wypadku. Jest on nieco wolniejszy od sortowania szybkiego. Algorytm ten tworzy kopiec – strukturę danych, którego korzeń jest największym elementem. Następnie korzeń jest usuwany i ustawiany na końcu tablicy, a kopiec jest naprawiany. Proces jest powtarzany do zdjęcia każdego elementu z kopca.
5. **Sortowanie przez wstawienie** – algorytm sortowania polegający na wstawieniu kolejnych elementów na odpowiednie miejsce. Pierwszy element zbioru jest z góry uznawany za jednoelementową posortowaną tablicę. Dobierany jest następny element i porównywany z posortowaną tablicą, następnie ustawiany jest w odpowiadające mu miejsce. Powtarzamy te działania, aż do końca tablicy. Jest to stabilny algorytm, idealny do małych zbiorów danych, niestety nie nadaje się do dużych zbiorów. Złożoność obliczeniowa tego algorytmu w najlepszym wypadku wynosi O(n), a w średnim i najgorszym O(n^2).
6. **Plan projektu:**
   1. Program daje możliwość wyboru typu danych wejściowych oraz ich początkowego ułożenia (liczby posortowane rosnąco w 33%, 66% czy 100%, malejąco, losowo).
   2. Program generuje tablice zgodnie z podanymi parametrami (wielkość tablicy, ułożenie danych wejściowych).
   3. Program wypisuje zawartość tablicy przed i po sortowaniu.
   4. Program sortuje liczby wybranym algorytmem.
   5. Program zapisuje tablice w pliku.

**Założenia projektu:**

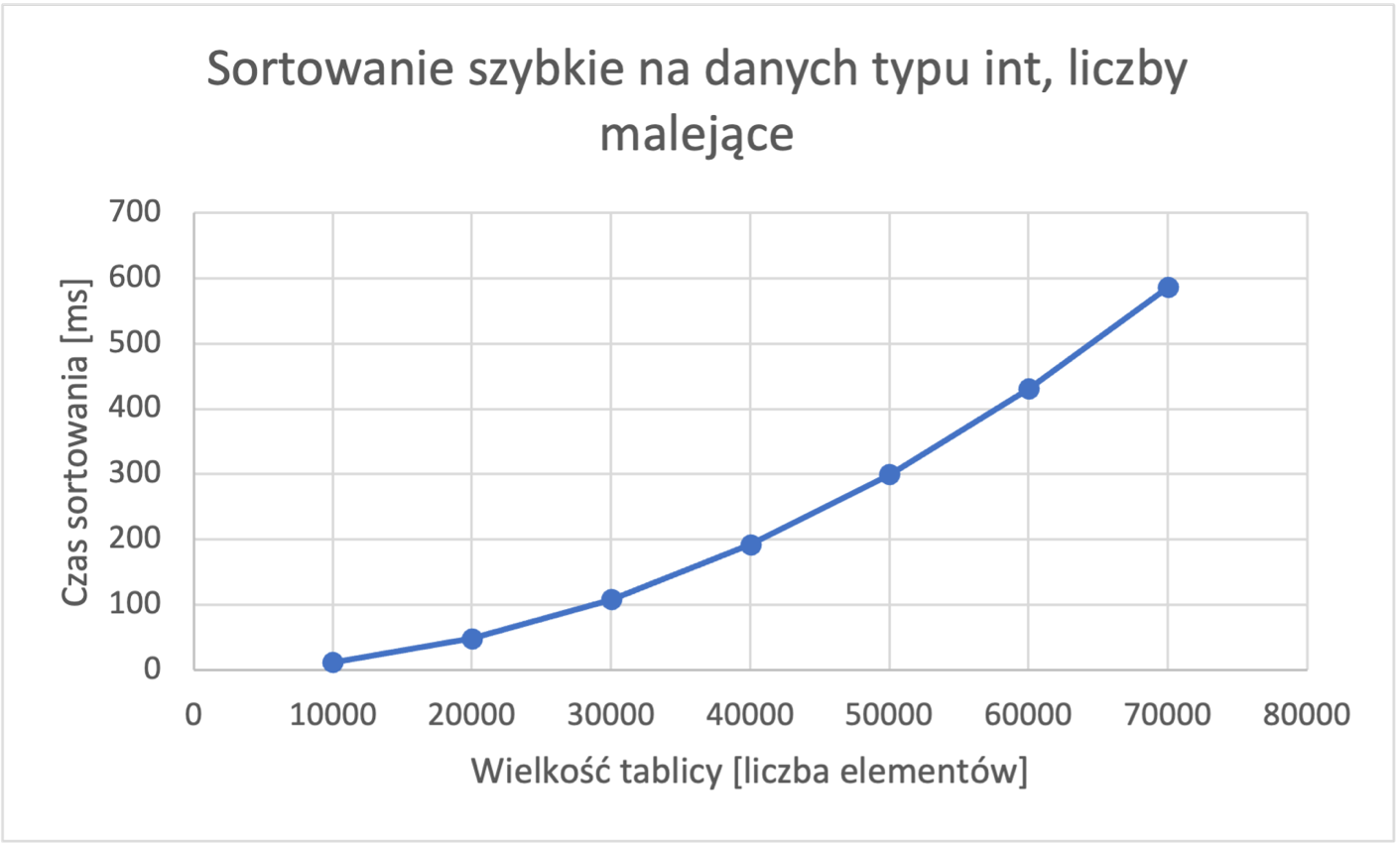
1. Obsługiwane typy danych – int i float.
2. Rozmiary tablic – 100000, 20000, 30000, 40000, 50000, 60000, 70000, 80000, 160000, 320000, 640000 (dobierane w zależności od możliwości komputera).
3. Liczby losowane są do tablic za pomocą funkcji random().
4. Każdy eksperyment został powtórzony 100 razy.
5. Czas zmierzony został za pomocą funkcji clock() z biblioteki <time.h>.
6. **Wyniki eksperytmentów**
7. **Sortowanie szybkie**

**SORTOWANIE SZYBKIE, DANE TYPU INT, LEWY PIVOT**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sortowanie szybkie dla danych typu int, lewy skrajny pivot** | | | | | |
|  | Średni czas sortowania [ms] | | | | |
| Wielkość tablicy [liczba elementów] | Liczby losowe | Liczby malejące | Liczby posortowane w 33% | Liczby posortowane w 66% | Liczby posortowane rosnąco |
| 10000 | 0,49046203 | 12,2352772 | 0,48045545 | 0,46633 | 14,76626 |
| 20000 | 1,03485443 | 48,3160499 | 1,0090198 | 0,97119 | 58,5466699 |
| 30000 | 1,60267722 | 108,210545 | 1,56705941 | 1,51156 | 131,48252 |
| 40000 | 2,17892405 | 192,263721 | 2,14244554 | 2,05445 | 233,530291 |
| 50000 | 2,76597468 | 299,745469 | 2,73161616 | 2,59866 | 364,740419 |
| 60000 | 3,37503165 | 430,705119 | 3,28585149 | 3,17529 | 524,533591 |
| 70000 | 3,98158228 | 585,8802 | 3,90215 | 3,732 | 713,10688 |

**

*Wykres 1 – wykres dla wyników sortowania szybkiego i lewego pivota, dane wejściowe losowe*

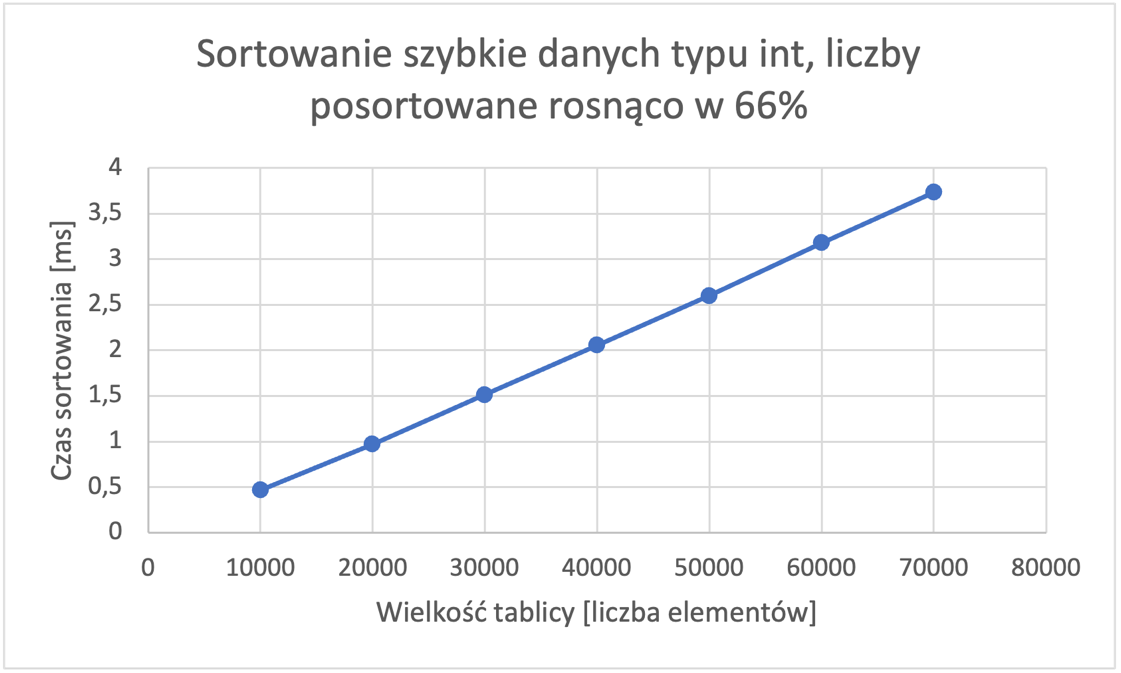
**

*Wykres 2 – wykres dla wyników sortowania szybkiego i lewego pivota, dane wejściowe posortowane malejąco*

*Obraz zawierający tekst, linia, zrzut ekranu, Wykres

Opis wygenerowany automatycznie*

*Wykres 3 – wykres dla wyników sortowania szybkiego i lewego pivota, dane wejściowe posortowane rosnąco w 33%*



*Wykres 4 – wykres dla wyników sortowania szybkiego i lewego pivota, dane wejściowe posortowane rosnąco w 66%*

*Obraz zawierający tekst, zrzut ekranu, linia, Wykres

Opis wygenerowany automatycznie*

*Wykres 5 - wykres dla wyników sortowania szybkiego i lewego pivota, dane wejściowe posortowane rosnąco*

*Obraz zawierający zrzut ekranu, linia, Wykres, tekst

Opis wygenerowany automatycznie*

*Wykres 6 – wykres dla wyników sortowania szybkiego dla lewego pivota, dane wejściowe sortowane na różne sposoby (wykresy dla liczb posortowanych rosnąco w 66%, rosnąco w 33% oraz losowych pokrywają się i zostały zasłonięte)*

**SORTOWANIE SZYBKIE, DANE TYPU INT, PRAWY PIVOT**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sortowanie szybkie dla danych typu int, prawy skrajny pivot** | | | | | |
| Wielkość tablicy  [liczba elementów] | Średni czas sortowania [ms] | | | | |
| Liczby posortowane rosnąco | Liczby posortowane malejąco | Liczby posortowane rosnąco w 33% | Liczby posortowane rosnąco w 66% | Liczby losowe |
| 10000 | 16,04731 | 12,36617 | 0,46338 | 0,42182 | 0,48949 |
| 20000 | 62,12547 | 48,38315 | 0,96665 | 0,87071 | 1,01459 |
| 30000 | 139,22338 | 108,27669 | 1,46493 | 1,35625 | 1,55914 |
| 40000 | 247,24442 | 192,544711 | 1,99207 | 1,83183 | 2,11795 |
| 50000 | 386,16981 | 300,446202 | 2,53201 | 2,31189 | 2,69007 |
| 60000 | 555,35002 | 432,323429 | 3,08362 | 2,7928 | 3,26665 |
| 70000 | 756,23458 | 588,415831 | 3,63041 | 3,31463 | 3,85614 |

Obraz zawierający tekst, zrzut ekranu, linia, Wykres

Opis wygenerowany automatycznie*Wykres 7 -* *wykres dla wyników sortowania szybkiego i prawego pivota, dane wejściowe posortowane rosnąco*

Obraz zawierający tekst, zrzut ekranu, linia, Wykres

Opis wygenerowany automatycznie

*Wykres 8 -* *wykres dla wyników sortowania szybkiego i prawego pivota, dane wejściowe posortowane malejąco*

Obraz zawierający tekst, zrzut ekranu, linia, Wykres

Opis wygenerowany automatycznie

*Wykres 9 -* *wykres dla wyników sortowania szybkiego i prawego pivota, dane wejściowe posortowane rosnąco w 33%*

Obraz zawierający tekst, zrzut ekranu, linia, Wykres

Opis wygenerowany automatycznie

*Wykres 10 -* *wykres dla wyników sortowania szybkiego i prawego pivota, dane wejściowe posortowane rosnąco w 66%*

Obraz zawierający tekst, linia, zrzut ekranu, Wykres

Opis wygenerowany automatycznie*Wykres 11 -* *wykres dla wyników sortowania szybkiego i prawego pivota, dane wejściowe losowe*

Obraz zawierający tekst, zrzut ekranu, Wykres, linia

Opis wygenerowany automatycznie

*Wykres 12 – wykres dla wyników sortowania szybkiego dla prawego pivota, dane wejściowe sortowane na różne sposoby (wykresy dla liczb posortowanych rosnąco w 66%, rosnąco w 33% oraz losowych pokrywają się i zostały zasłonięte)*

**SORTOWANIE SZYBKIE, DANE TYPU INT, ŚRODKOWY PIVOT**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sortowanie szybkie dla danych typu int, środkowy pivot** | | | | | |
| Wielkość tablicy [liczba elementów] | Średni czas sortowania [ms] | | | | |
| Liczby posortowane malejąco | Liczby posortowane rosnąco w 33% | Liczby posortowane rosnąco w 66% | Liczby losowe | Liczby posortowane rosnąco |
| 10000 | 0,12984 | 0,46973 | 0,38112 | 0,51 | 0,14322 |
| 20000 | 0,25377 | 0,98468 | 0,78732 | 1,048 | 0,27837 |
| 30000 | 0,3683 | 1,47869697 | 1,18354 | 1,597 | 0,39681 |
| 40000 | 0,49713 | 1,99351485 | 1,60091 | 2,154 | 0,53296 |
| 50000 | 0,61784 | 2,52044 | 2,02793 | 2,72 | 0,6561 |
| 60000 | 0,746 | 3,06017 | 2,45098 | 3,296 | 0,7681 |
| 70000 | 0,88978 | 3,62162 | 2,89101 | 3,93 | 0,90567 |

Obraz zawierający zrzut ekranu, linia, tekst, Wykres

Opis wygenerowany automatycznie

*Wykres 13 -* *wykres dla wyników sortowania szybkiego i środkowego pivota, dane wejściowe posortowane malejąco*

Obraz zawierający tekst, zrzut ekranu, linia, Wykres

Opis wygenerowany automatycznie*Wykres 14 -* *wykres dla wyników sortowania szybkiego i środkowego pivota, dane wejściowe posortowane rosnąco w 33%*

Obraz zawierający linia, zrzut ekranu, tekst, Wykres

Opis wygenerowany automatycznie

*Wykres 15 -* *wykres dla wyników sortowania szybkiego i środkowego pivota, dane wejściowe posortowane rosnąco w 66%*

Obraz zawierający tekst, zrzut ekranu, linia, Wykres

Opis wygenerowany automatycznie

*Wykres 16 -* *wykres dla wyników sortowania szybkiego i środkowego pivota, dane wejściowe losowe*

Obraz zawierający tekst, zrzut ekranu, linia, Wykres

Opis wygenerowany automatycznie

*Wykres 17 -* *wykres dla wyników sortowania szybkiego i środkowego pivota, dane wejściowe posortowane rosnąco*

Obraz zawierający tekst, zrzut ekranu, linia, Wykres

Opis wygenerowany automatycznie

*Wykres 18 – wykres dla wyników sortowania szybkiego dla środkowego pivota, dane wejściowe sortowane na różne sposoby*

**SORTOWANIE SZYBKIE DANYCH TYPU INT, LOSOWY PIVOT**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sortowanie szybkie danych typu int, losowy pivot | | | | | |
| Wielkość tablicy [liczba elementów] | Średni czas sortowania [ms] | | | | |
| Liczby posortowane malejąco | Liczby posortowane rosnąco w 33% | Liczby posortowane rosnąco w 66% | Liczby posortowane rosnąco | Liczby losowe |
| 10000 | 0,4517 | 0,77728 | 0,70279 | 0,45904 | 0,79563 |
| 20000 | 0,89021 | 1,56297 | 1,40064 | 0,89428 | 1,60481 |
| 30000 | 1,33585 | 2,34952 | 2,1256 | 1,34252 | 2,43903 |
| 40000 | 1,79437 | 3,15917 | 2,86621 | 1,79856 | 3,31551 |
| 50000 | 2,23917 | 3,9965 | 3,62132 | 2,2456 | 4,17431 |
| 60000 | 2,69468 | 4,8223 | 4,3614 | 2,69594 | 5,0478 |
| 70000 | 3,14319 | 5,68875 | 5,12193 | 3,15187 | 5,92248 |

Obraz zawierający tekst, zrzut ekranu, linia, Wykres

Opis wygenerowany automatycznie

*Wykres 19 -* *wykres dla wyników sortowania szybkiego i losowego pivota, dane wejściowe posortowane malejąco*

Obraz zawierający tekst, linia, zrzut ekranu, Wykres

Opis wygenerowany automatycznie

*Wykres 20 -* *wykres dla wyników sortowania szybkiego i losowego pivota, dane wejściowe posortowane rosnącow 33%*

Obraz zawierający tekst, linia, zrzut ekranu, Wykres

Opis wygenerowany automatycznie

*Wykres 21 -* *wykres dla wyników sortowania szybkiego i losowego pivota, dane wejściowe posortowane rosnąco w 66%*

Obraz zawierający tekst, zrzut ekranu, linia, Wykres

Opis wygenerowany automatycznie

*Wykres 22 -* *wykres dla wyników sortowania szybkiego i losowego pivota, dane wejściowe posortowane rosnąco*

Obraz zawierający linia, tekst, zrzut ekranu, Wykres

Opis wygenerowany automatycznie

*Wykres 23 -* *wykres dla wyników sortowania szybkiego i losowego pivota, dane wejściowe ułożone losowo*

Obraz zawierający zrzut ekranu, tekst, linia, Wykres

Opis wygenerowany automatycznie

*Wykres 24 – wykres dla wyników sortowania szybkiego dla losowego pivota, dane wejściowe sortowane na różne sposoby*

1. **Sortowanie Shella**

**SORTOWANIE SHELLA, DANE TYPU INT**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sortowanie Shella dla danych typu int** | | | | | |
| Wielkość tablicy [liczba elementów] | Średni czas sortowania [ms] | | | | |
| Liczby posortowane malejąco | Liczby posortowane rosnąco w 33% | Liczby posortowane rosnąco w 66% | Liczby posortowane rosnąco | Liczby losowe |
| 10000 | 0,15110692 | 0,70771 | 0,64878 | 0,08138 | 101,713535 |
| 20000 | 0,31462264 | 1,54761 | 1,40545 | 0,17468 | 203,54796 |
| 30000 | 0,66225786 | 3,3349 | 3,03853 | 0,37725 | 407,402465 |
| 40000 | 1,36490566 | 7,21704 | 6,55857 | 0,76717 | 815,439636 |
| 50000 | 2,86471698 | 15,64096 | 14,26082 | 1,6309 | 1632,37284 |
| 60000 | 6,03501258 | 34,49104 | 31,40874 | 3,46867 | 3268,3285 |
| 70000 | 12,668434 | 76,5266998 | 69,2603803 | 7,34895 | 6544,81059 |

Obraz zawierający tekst, zrzut ekranu, linia, Wykres

Opis wygenerowany automatycznie

*Wykres 25 -* *wykres dla wyników sortowania Shella, dane wejściowe posortowane malejąco*

Obraz zawierający tekst, zrzut ekranu, linia, Wykres

Opis wygenerowany automatycznie

*Wykres 26 -* *wykres dla wyników sortowania Shella, dane wejściowe posortowane rosnąco w 33%*

Obraz zawierający tekst, zrzut ekranu, linia, Wykres

Opis wygenerowany automatycznie*Wykres 27 -* *wykres dla wyników sortowania Shella, dane wejściowe posortowane rosnąco w 66%*

Obraz zawierający tekst, zrzut ekranu, linia, Wykres

Opis wygenerowany automatycznie

*Wykres 28 -* *wykres dla wyników sortowania Shella, dane wejściowe posortowane rosnąco*

Obraz zawierający tekst, zrzut ekranu, linia, Wykres

Opis wygenerowany automatycznie*Wykres 29 -* *wykres dla wyników sortowania Shella, dane wejściowe ułożone losowo*

Obraz zawierający zrzut ekranu, linia, Wykres, tekst

Opis wygenerowany automatycznie

*Wykres 30 – wykres dla wyników sortowania Shell, dane wejściowe sortowane na różne sposoby (na wykresie pominięto przypadek, gdy dane wejściowe są ułożone losowo, ponieważ wyniki te silnie odbiegają od reszty i sprawiają, że wykres staje się nieczytelny – inne wartości nakładają się)*

1. **Sortowanie przez kopcowanie**

**SORTOWANIE PRZEZ KOPCOWANIE, DANE TYPU INT**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sortowanie przez kopcowanie dla danych typu int** | | | | | |
| Wielkość tablicy [liczba elementów] | Średni czas sortowania [ms] | | | | |
| Liczby posortowane malejąco | Liczby posortowane rosnąco w 33% | Liczby posortowane rosnąco w 66% | Liczby posortowane rosnąco | Liczby losowe |
| 10000 | 0,6274908 | 0,68558896 | 0,6768 | 0,68546 | 0,69482209 |
| 20000 | 1,35634356 | 1,46953988 | 1,4555 | 1,477104 | 1,48676687 |
| 40000 | 2,94259509 | 3,16671779 | 3,1377 | 3,170423 | 3,20017791 |
| 80000 | 6,3223681 | 6,89787117 | 6,7491 | 6,762123 | 7,02492638 |
| 160000 | 13,4347975 | 15,0776749 | 14,559 | 14,35113 | 15,5319018 |
| 320000 | 28,5845706 | 33,0099816 | 31,406 | 30,41413 | 34,2728221 |
| 640000 | 60,6810307 | 71,7657058 | 67,689 | 64,11615 | 75,195681 |

Obraz zawierający tekst, zrzut ekranu, linia, Wykres

Opis wygenerowany automatycznie

*Wykres 31 -* *wykres dla wyników sortowania przez kopcowanie, dane wejściowe posortowane malejąco*

Obraz zawierający tekst, zrzut ekranu, linia, Wykres

Opis wygenerowany automatycznie

*Wykres 32 -* *wykres dla wyników sortowania przez kopcowanie, dane wejściowe posortowane rosnąco w 33%*

Obraz zawierający tekst, zrzut ekranu, linia, Wykres

Opis wygenerowany automatycznie

*Wykres 33 -* *wykres dla wyników sortowania przez kopcowanie, dane wejściowe posortowane rosnąco w 66%*

Obraz zawierający tekst, zrzut ekranu, linia, Wykres

Opis wygenerowany automatycznie

*Wykres 34 -* *wykres dla wyników sortowania przez kopcowanie, dane wejściowe posortowane rosnąco*

Obraz zawierający tekst, zrzut ekranu, linia, Wykres

Opis wygenerowany automatycznie

*Wykres 35 -* *wykres dla wyników sortowania przez kopcowanie, dane wejściowe ułożone losowo*

Obraz zawierający zrzut ekranu, linia, Wykres, tekst

Opis wygenerowany automatycznie

*Wykres 36 – wykres dla wyników sortowania przez kopcowanie, dane wejściowe sortowane na różne sposoby*

1. **Sortowanie przez wstawienie**

**SORTOWANIE PRZEZ WSTAWIENIE DANE TYPU FLOAT**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sortowanie przez kopcowanie dla danych typu float** | | | | | |
| Wielkość tablicy [liczba elementów] | Średni czas sortowania [ms] | | | | |
| Liczby malejące | Liczby posortowane rosnąco w 33% | Liczby posortowane rosnąco w 66% | Liczby posortowane rosnąco | Liczby losowe |
| 10000 | 18,07397 | 8,0482 | 0,00491 | 0,00491 | 9,096 |
| 20000 | 72,18535 | 32,0787301 | 19,99582 | 0,00944 | 36,08 |
| 30000 | 161,98912 | 72,1060404 | 45,1701501 | 0,0142 | 81,24 |
| 40000 | 300,48804 | 131,89435 | 83,5822697 | 0,01852 | 147,6 |
| 50000 | 464,80817 | 211,736289 | 135,21152 | 0,02338 | 236,7 |
| 60000 | 665,81038 | 305,46131 | 192,756299 | 0,02712 | 341 |
| 70000 | 898,45367 | 412,30735 | 255,17076 | 0,03206 | 461 |

Obraz zawierający tekst, linia, zrzut ekranu, Wykres

Opis wygenerowany automatycznie

*Wykres 37 -* *wykres dla wyników sortowania przez wstawienie dla danych typu float, dane wejściowe posortowane malejąco*

Obraz zawierający tekst, zrzut ekranu, linia, Wykres

Opis wygenerowany automatycznie

*Wykres 38 -* *wykres dla wyników sortowania przez wstawienie dla danych typu float, dane wejściowe posortowane rosnąco w 33%*

Obraz zawierający tekst, linia, zrzut ekranu, Wykres

Opis wygenerowany automatycznie

*Wykres 39 -* *wykres dla wyników sortowania przez wstawienie dla danych typu float, dane wejściowe posortowane rosnąco w 66%*

Obraz zawierający tekst, zrzut ekranu, linia, Wykres

Opis wygenerowany automatycznie

*Wykres 40 -* *wykres dla wyników sortowania przez wstawienie dla danych typu float, dane wejściowe posortowane rosnąco*

Obraz zawierający tekst, linia, Wykres, zrzut ekranu

Opis wygenerowany automatycznie

*Wykres 41 -* *wykres dla wyników sortowania przez wstawienie dla danych typu float, dane wejściowe ułożone losowo*

Obraz zawierający zrzut ekranu, linia, Wykres, tekst

Opis wygenerowany automatycznie

*Wykres 42 – wykres dla wyników sortowania przez wstawienie dla danych typu float, dane wejściowe sortowane na różne sposoby*

**SORTOWANIE PRZEZ WSTAWIENIE DANE TYPU INT**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sortowanie przez wstawienie dane typu int** | | | | | |
| Wielkość tablicy [liczba elementów] | Średni czas sortowania [ms] | | | | |
| Liczby posortowane malejąco | Liczby posortowane rosnąco w 33% | Liczby posortowane rosnąco w 33% | Liczby posortowane rosnąco | Liczby losowe |
| 10000 | 18,1136199 | 8,02803 | 5,01095 | 0,00489 | 9,10186 |
| 20000 | 71,9614899 | 32,0295901 | 19,9860101 | 0,00932 | 36,0489401 |
| 30000 | 161,951869 | 71,9321995 | 44,9782801 | 0,01388 | 80,9514999 |
| 40000 | 295,731941 | 130,48133 | 82,4220602 | 0,01838 | 146,525121 |
| 50000 | 459,43448 | 207,16475 | 132,444991 | 0,0227 | 232,609891 |
| 60000 | 659,635102 | 298,61359 | 188,76883 | 0,02711 | 335,60687 |
| 70000 | 892,434103 | 404,807938 | 251,935969 | 0,03141 | 454,930489 |

Obraz zawierający tekst, linia, zrzut ekranu, Wykres

Opis wygenerowany automatycznie

*Wykres 43 -* *wykres dla wyników sortowania przez wstawienie dla danych typu int, dane wejściowe posortowane malejąco*

Obraz zawierający tekst, zrzut ekranu, linia, Wykres

Opis wygenerowany automatycznie

*Wykres 44 -* *wykres dla wyników sortowania przez wstawienie dla danych typu int, dane wejściowe posortowane rosnąco w 33%*

Obraz zawierający tekst, linia, zrzut ekranu, Wykres

Opis wygenerowany automatycznie

*Wykres 45 -* *wykres dla wyników sortowania przez wstawienie dla danych typu int, dane wejściowe posortowane rosnąco w 66%*

Obraz zawierający tekst, linia, zrzut ekranu, Wykres

Opis wygenerowany automatycznie

*Wykres 26 -* *wykres dla wyników sortowania przez wstawienie dla danych typu int, dane wejściowe posortowane rosnąco*

Obraz zawierający tekst, linia, zrzut ekranu, Wykres

Opis wygenerowany automatycznie

*Wykres 47 -* *wykres dla wyników sortowania przez wstawienie dla danych typu int, dane wejściowe ułożone losowo*

Obraz zawierający zrzut ekranu, linia, Wykres, tekst

Opis wygenerowany automatycznie

*Wykres 48 – wykres dla wyników sortowania przez wstawienie dla danych typu int, dane wejściowe sortowane na różne sposoby*

1. **Wnioski:**
2. **Sortowanie szybkie**

Na podstawie tabeli i wykresów stwierdzić można, że algorytm sortowania szybkiego dla lewego oraz prawego pivota najlepiej sprawdza się dla danych wejściowych posortowanych rosnąco w 33% i 66% oraz losowych, a najgorsze przypadki występują dla danych posortowanych malejąco oraz rosnąco. Wykres dla liczb losowych i posortowanych rosnąco w 33% i 66% zbliżone są do wykresów funkcji n log n, potwierdza to, że złożoność algorytmu sortowania szybkiego dla najlepszego przypadku ma wartość O(n log n). Wykres dla liczb posortowanych rosnąco oraz malejąco przypomina funkcję n^2, więc złożoność dla przypadku najgorszego ma złożoność O(n^2) – co się zgadza. Dla pivota środkowego i losowego najlepszym przypadkiem jest sytuacja, w której dane wejściowe posortowane są rosnąco i malejąco, a najgorszym, gdy są posortowane rosnąco w 66% i 33% oraz losowe. Wykres dla liczb posortowanych rosnąco i malejąco zbliżony jest do wykresu funkcji n log n, potwierdza to, że sortowanie szybkie dla przypadku najlepszego ma złożoność O(n log n). Z kolei wykres dla liczb posortowanych rosnąco w 33% i 66% oraz liczb losowych przypomina wykres funkcji n^2, co potwierdza, że sortowanie szybkie dla przypadku średniego i najgorszego ma złożoność O((n^2). Wybór pivota jako skrajnego elementu (lewego lub prawego) może prowadzić do nierównomiernego podziału (szczególnie, gdy dane są już częściowo posortowane), a wybór pivota jako środkowego, bądź losowego elementu może prowadzić do bardziej równomiernego podziału – ma to wpływ na stabilność i wydajność sortowania. Sortowanie to sprawdza się wyjątkowo dobrze dla danych losowych. Jest on wydajny nawet dla dużych rozmiarów danych. Jednak dla danych już częściowo posortowanych istotny jest wybór pivota, wybór ten może przesądzić nad efektywnością i czasem sortowania, jednak przy dobrej strategii algorytm może okazać się efektywniejszy niż w przypadku losowo ułożonych danych wejściowych.

1. **Sortowanie Shella**

Na podstawie tabeli i wykresów stwierdzić można, że algorytm sortowania Shella najlepiej sprawdza się dla danych wejściowych posortowanych malejąco oraz rosnąco, średnim przypadkiem jest sytuacja, w której dane wejściowe posortowane są rosnąco w 33% i 66%, a najgorszym są liczby ułożone losowo. Wykres dla liczb posortowanych rosnąco przypomina wykres funkcji n log n, więc potwierdza to, że złożoność tego sortowania w przypadku najlepszym wynosi O(n log n). Wykres dla liczb posortowanych rosnąco w 33% i 66% oraz liczb posortowanych malejąco bardziej zbliżony jest do wykresu funkcji n^(3/2), co potwierdza założenie, że złożoność tego sortowania dla przypadku średniego wynosi O(n^(3/2). Dla liczb losowych wykres przypomina wykres funkcji n^2, co potwierdza, że złożoność dla przypadku najgorszego wynosi O(n^2).

1. **Sortowanie przez kopcowanie**

Na podstawie tabeli i wykresów stwierdzić można, że algorytm sortowania przez kopcowanie najlepiej sprawdza się dla danych wejściowych posortowanych malejąco oraz rosnąco, średnim przypadkiem jest sytuacja, w której dane wejściowe posortowane są rosnąco w 33% i 66%, a najgorszym są liczby ułożone losowo. Przy danych posortowanych budowanie kopca wymaga mniej operacji przesiewania, niż przy częściowo posortowanych i dużo mniej niż przy losowych. Wszystkie uzyskane przy tym sortowaniu wykresy przypominają wykresy funkcji n log n, co potwierdza tezę, że złożoność sortowania przez kopcowanie to O(n log n). Wykresy potwierdzają również, że sortowanie przez kopcowanie to stabliny algorytm działający podobnie dla wszystkich przypadków.

1. **Sortowanie przez wstawianie**

Na podstawie tabeli i wykresów stwierdzić można, że algorytm sortowania przez wstawianie dla typu danych int jak i float najlepiej sprawdza się dla liczb początkowych rosnąco, średnim przypadkiem jest sytuacja, w której dane wejściowe posortowane są rosnąco w 33% i 66% oraz ułożone losowo, a najgorszym są liczby posortowane malejąco. Wynika to z tego, że dla liczb posortowanych rosnąco algorytm iteruje po kolejnych elementach nie zmieniając ich miejsc, a dla liczb posortowanych malejąco miejsce każdego elementu musi zostać kolejno zmienione. Wykres dla liczb posortowanych rosnąco przypomina funkcję liniową, potwierdza to założenie o złożoności tego algorytmu w najlepszym wypadku wynoszącej O(n). Dla liczb posortowanych rosnąco w 33% i 66% oraz posortowanych malejąco wykres przypomina funkcję kwadratową, więc zgadza się to z wartością średnią i najgorszą złożoności wynoszącą O(n^2). Dla dużych i nieposortowanych liczb algorytm jest nieoptymalny. Jeśli chodzi o różnice w wynikach dla danych typu int oraz float, to są one niewielkie i praktycznie nieznaczne.

1. **Porównanie sortowań**

Podsumowując – sortowanie szybkie jest uniwersalnym i skutecznym algorytmem, w przypadku odpowiedniego doboru pivota. Sortowanie Shella osiąga dobre wyniki dla posortowanych danych wejściowych. Sortowanie przez kopcowanie sprawdza się przy posortowanych, zarówno malejąco jak i rosnąco, danych wejściowych, jednak nie działa tak wydajnie na danych losowych, mimo to jest wyjątkowo stabilnym algorytmem. Sortowanie przez wstawianie pracuje skutecznie na małych zbiorach danych prawie posortowanych, jednak dla większych zbiorów danych, bądź danych losowych może być nieoptymalny, pomimo to charakteryzuje się stabilnością.

1. **Bibliografia:**

* <http://jaroslaw.mierzwa.staff.iiar.pwr.wroc.pl>
* <https://pl.wikipedia.org/wiki/Sortowanie_Shella>
* <https://www.algorytm.edu.pl/algorytmy-maturalne/quick-sort.html>

.