P O L I T E C H N I K A R Z E S Z O W S K A
im. Ignacego Łukasiewicza
WYDZIAŁ MATEMATYKI I FIZYKI STOSOWANEJ

KRZYSZTOF KOPIEC
173158

ALGORYTMY I STRUKTURY DANYCH

Projekt

kierunek studiów: inżynieria i analiza danych

Opiekun pracy:

Prof. Mariusz Borkowski

Rzeszów 2022

Spis treści

1. Wstęp

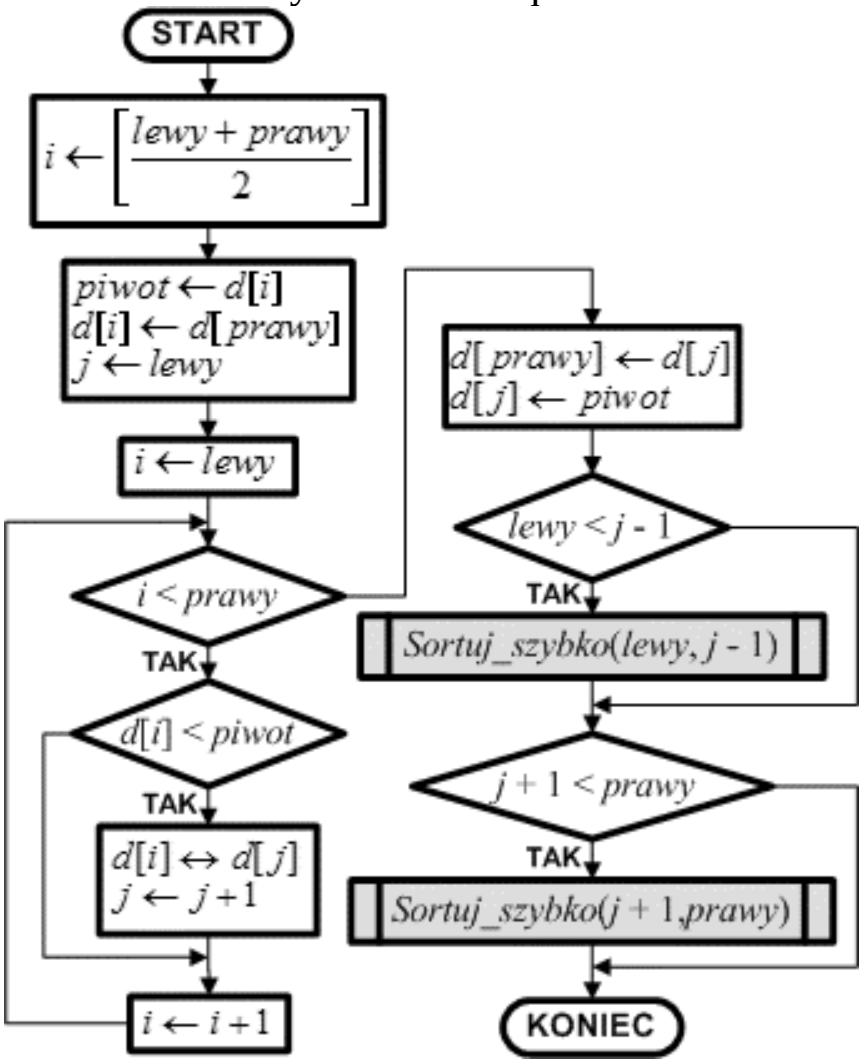
Projekt opiera się na wprowadzeniu oraz porównaniu dwóch alogytmów: sortowania quicksort i sortowania gnoma.

2. Algorytmy

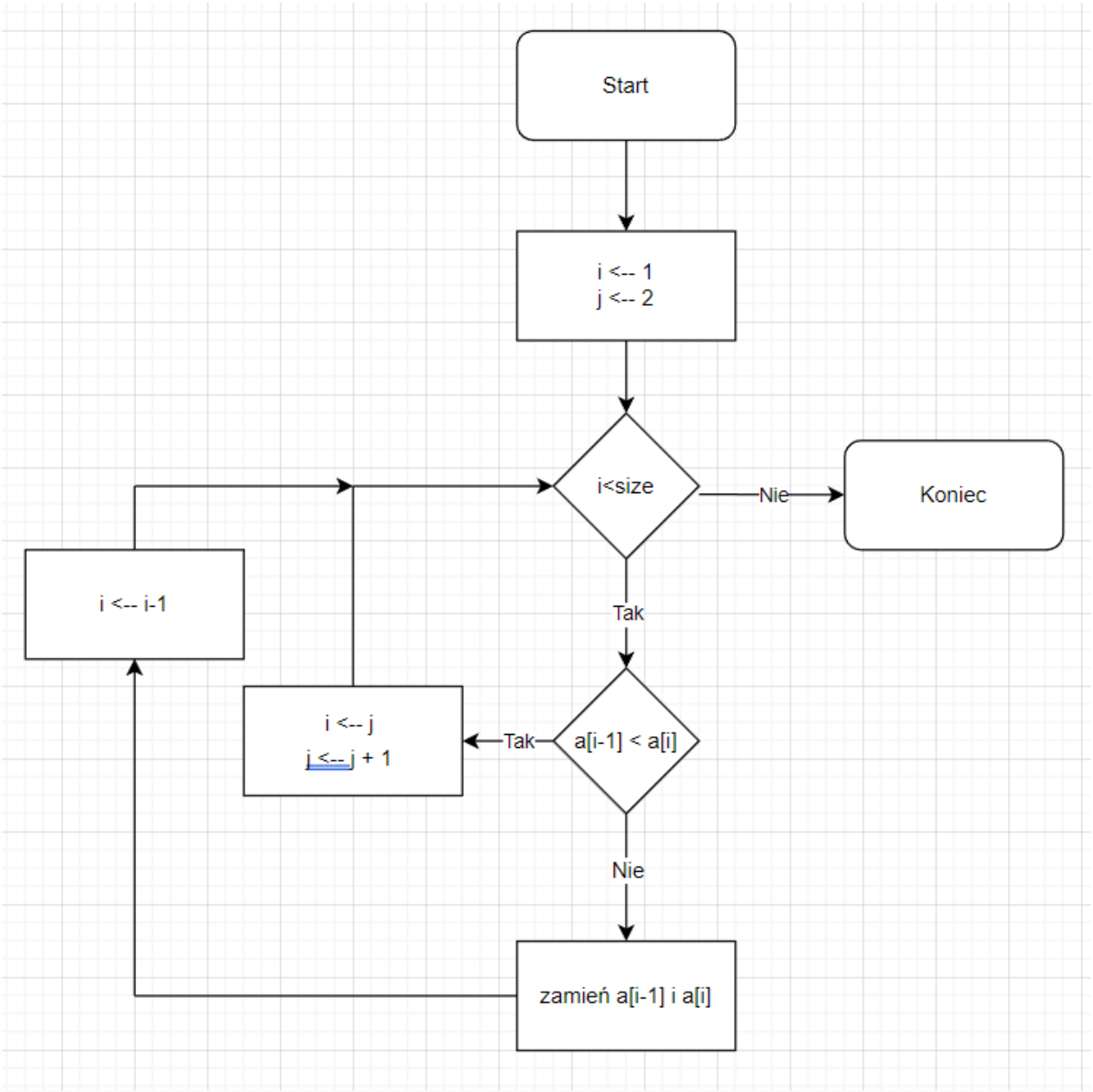
Sortowanie quicksort
Działanie tego algorytmu polega na wyborze jednego z elementów w tablicy oznaczając go jako pivot. Może to być środkowy lub jakikolwiek inny element w zależności od zbioru liczb. Następnie ustawia liczby niewiększe niż pivot po lewej jego stronie, a niemniejsze liczby po jego prawej stronie. Wykonuje takim schematem następne kroki, zajmując się poszczególnymi stronami osobno aż do posortowania całkowitego.
Sortowanie gnoma
Działanie tego algorytmu polega na wybraniu pierwszego elementu z lewej strony i porownanie go z nastepnym z kolei. Gdy element z prawej jest większy przechodzi na kolejny i wykonuje to samo. Gdy jednak jest mniejszy to zamienia miejscami obydwa elementy i startuje od tego miejsca w którym zostal umieszczony przesuniety element.
Udoskonaleniem tego sortowania jest dodanie funkcji, która zapamięta ostatnią pozycje elementu przed zamianą i z niej rozpocznie dalsze sortowanie po zamianie.

3. Schematy blokowe

Schemat blokowy sortowania quicksort



Schemat blokowy sortowania gnoma


Start
i <-- 1
j <-- 2
i<size
Nie
Koniec
Tak
a[i-1] < a[i]
Tak
i <-- j
j <-- j + 1
i <-- i-1
Nie
zamień a[i-1] i a[i]

4. Pseudokod

Pseudokod sortowania quicksort

```
K01:, i ← (lewy+prawy)/2
K02:, piwot ← d[i];   d[i] ← d[prawy];   j ← lewy
K03:, Dla i = lewy, lewy + 1, ..., prawy - 1:
   wykonuj kroki K04...K05
K04:,     Jeśli d[i] ≥ piwot,
   to wykonaj kolejny obieg pętli K03
K05:,     d[i] ↔ d[j];    j ← j + 1
K06:, d[prawy] ← d[j];   d[j] ← piwot
K07:, Jeśli lewy < j - 1,
to Sortuj_szybko(lewy, j - 1)
K08:, Jeśli j + 1 < prawy,
to Sortuj_szybko(j + 1, prawy)
K09:, Zakończ
```

Pseudokod sortowania gnoma

```
K01:  i := 1, j := 2
K02: Dla   i < size
        Jeśli a[i-1] ≤ a[i]
        To    i := j
              j := j + 1
        Inaczej
              zamień a[i-1] oraz a[i]
              i := i – 1
 K03: Jeśli  i = 0
              i := 1
 K04: Zakończ
```

5. Rezultaty testów

Testy sortowania quicksort

Wygenerowane ciągi:

62 12 50 8 58 1 17 22 18 34 83 46 64 22 27 39 70 29 92 70
64 93 82 43 80 95 78 30 0 71 83 6 61 45 90 8 31 9 39 60
30 41 57 33 35 58 32 21 53 96 89 1 81 58 97 1 56 34 94 20
64 72 46 63 34 41 45 60 98 33 33 81 79 93 52 81 41 95 7 94
92 67 88 11 88 15 4 84 57 81 86 40 43 31 94 31 82 40 19 76
46 61 92 39 39 2 96 71 99 6 59 58 16 80 98 42 51 44 19 0
29 97 31 45 86 29 45 44 27 46 19 23 28 33 57 58 89 44 16 74
56 80 85 84 32 9 27 23 72 42 14 24 60 75 45 20 35 24 69 81
96 99 22 29 54 70 63 70 94 59 45 50 52 87 33 21 24 56 55 45
43 19 45 44 48 74 75 73 96 66 26 92 30 3 45 11 3 38 93 81

Ciągi po sortowaniu quicksort:

1 8 12 17 18 22 22 27 29 34 39 46 50 58 62 64 70 70 83 92
0 6 8 9 30 31 39 43 45 60 61 64 71 78 80 82 83 90 93 95
1 1 20 21 30 32 33 34 35 41 53 56 57 58 58 81 89 94 96 97

7 33 33 34 41 41 45 46 52 60 63 64 72 79 81 81 93 94 95 98
4 11 15 19 31 31 40 40 43 57 67 76 81 82 84 86 88 88 92 94
0 2 6 16 19 39 39 42 44 46 51 58 59 61 71 80 92 96 98 99
16 19 23 27 28 29 29 31 33 44 44 45 45 46 57 58 74 86 89 97
9 14 20 23 24 24 27 32 35 42 45 56 60 69 72 75 80 81 84 85
21 22 24 29 33 45 45 50 52 54 55 56 59 63 70 70 87 94 96 99
3 3 11 19 26 30 38 43 44 45 45 48 66 73 74 75 81 92 93 96

Testy sortowania gnoma

Wygenerowane ciągi:

56 23 19 21 88 45 71 88 89 59 61 50 87 68 91 54 75 49 79 31
68 9 94 33 79 58 1 9 72 90 82 74 62 77 42 93 63 89 54 82
34 21 66 42 84 2 82 19 60 26 11 19 12 35 4 38 41 36 13 22
95 95 55 67 5 91 58 34 33 2 55 34 11 41 59 50 82 75 68 82
40 56 64 80 83 31 39 44 60 2 36 71 95 99 88 57 27 70 43 1
19 18 3 25 17 53 62 22 18 29 18 93 94 63 2 51 65 5 72 66
51 22 29 72 60 4 13 11 11 87 12 10 36 84 12 35 63 92 71 1
59 54 55 4 78 31 25 59 3 81 91 89 82 61 57 24 71 21 78 73
89 40 82 64 92 64 46 55 33 86 52 20 69 56 57 48 29 48 68 53
72 94 76 8 93 52 16 23 70 13 50 61 34 99 9 27 91 70 63 17

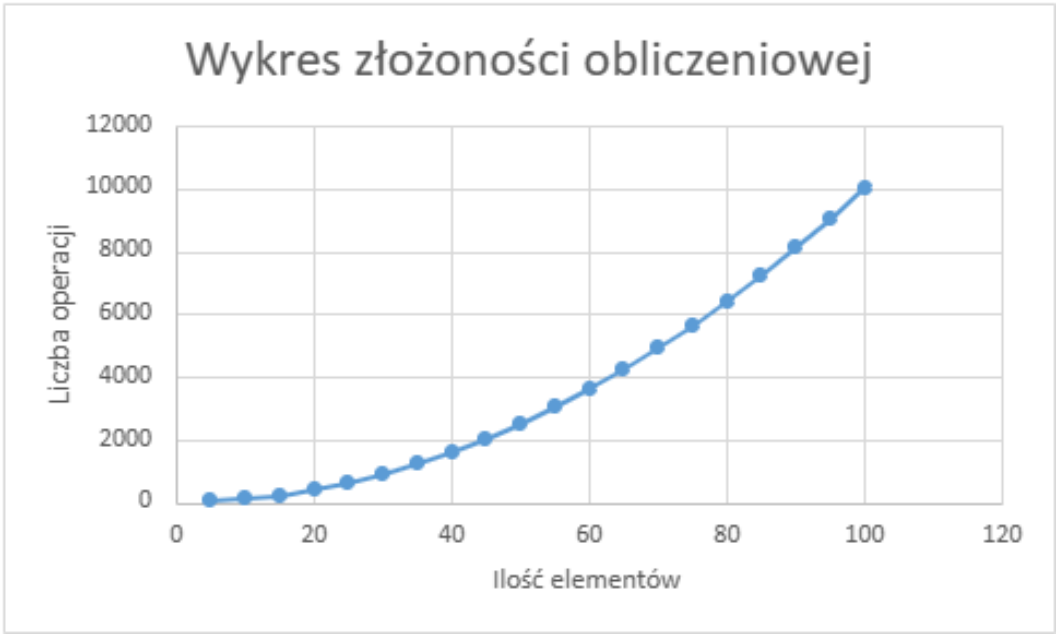
Ciągi po sortowaniu gnom:

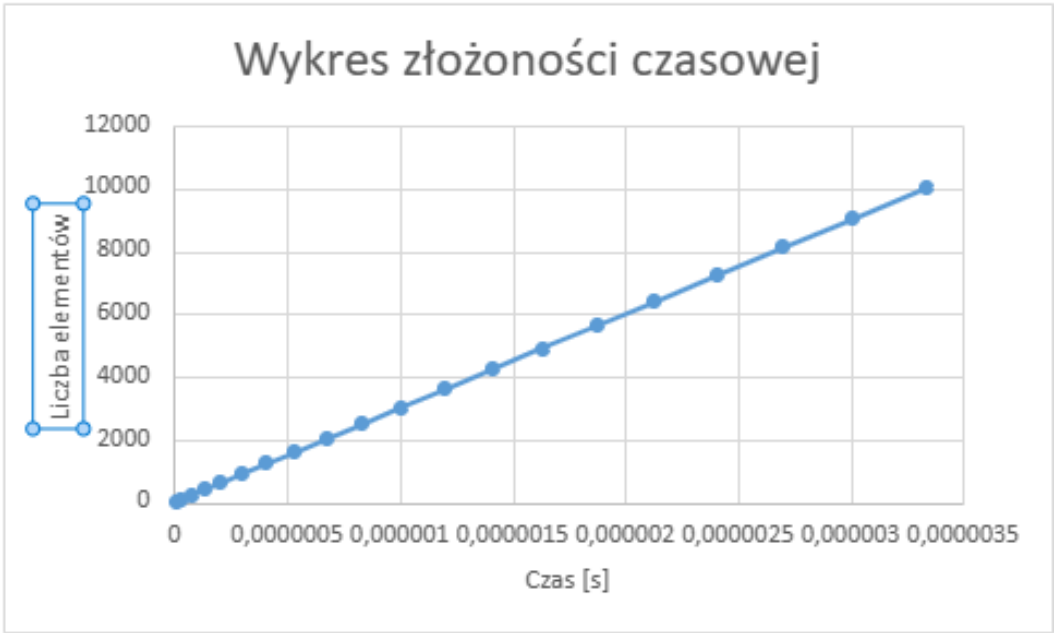
19 21 23 31 45 49 50 54 56 59 61 68 71 75 79 87 88 88 89 91
1 9 9 33 42 54 58 62 63 68 72 74 77 79 82 82 89 90 93 94
2 4 11 12 13 19 19 21 22 26 34 35 36 38 41 42 60 66 82 84
2 5 11 33 34 34 41 50 55 55 58 59 67 68 75 82 82 91 95 95
1 2 27 31 36 39 40 43 44 56 57 60 64 70 71 80 83 88 95 99
2 3 5 17 18 18 18 19 22 25 29 51 53 62 63 65 66 72 93 94
1 4 10 11 11 12 12 13 22 29 35 36 51 60 63 71 72 84 87 92
3 4 21 24 25 31 54 55 57 59 59 61 71 73 78 78 81 82 89 91
20 29 33 40 46 48 48 52 53 55 56 57 64 64 68 69 82 86 89 92
8 9 13 16 17 23 27 34 50 52 61 63 70 70 72 76 91 93 94 99

6. Wykresy złożoności

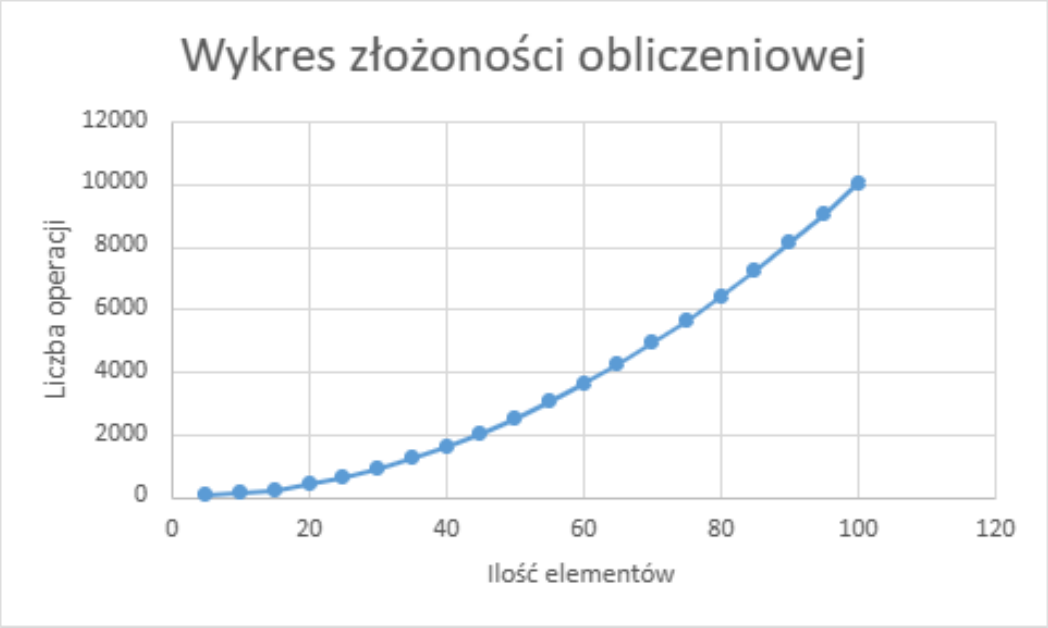
Sortowanie quicksort

W sytuacji pesymistycznej:





Sortowanie gnoma



7. Wnioski

Algorytm sortowania quicksort jest chętnie implementowanym algorytmem ze względu na sporo zalet tego algorytmu jakimi są:

- Złożoność O(n log n)
- Łatwość w implementacji
- Dobra współpraca z różnymi typami danych
- Brak potrzeby tworzenia tablicy do posortowania jak w chociażby w sortowaniu przez scalanie

Jednak występują również wady takie jak:

- Niestabliność i wrażliwość na błędy
- Złożoność w sytuacji pesymistycznej O(n2)

Algorytm sortowania gnoma natomiast jest mniej wydajny w przypadku większej ilości danych niż algorytm quicksort. Co prawda jest również łatwy w implementacji, jednak w przypadku dużej

ilości danych nie działa tak dobrze. W przypadku małej ilości danych sortowanie gnoma sprawdza się bardzo dobrze, więc podsumowując wybór sortowania spomiędzy tych 2 powinien zależeć od ilości danych oraz własnej preferencji.

8. Kod

```
#include <iostream>
#include <iomanip>
#include <cstdlib>
#include <time.h>

using namespace std;

const int N = 20; // Liczebność zbioru.
const int iloscCiagow = 20;
int d[N];

// Procedura sortowania szybkiego
//------------------------------

void Sortuj_szybko(int lewy, int prawy)
{
    int i,j,piwot;

    i = (lewy + prawy) / 2;
    piwot = d[i]; d[i] = d[prawy];
    for(j = i = lewy; i < prawy; i++)
        if(d[i] < piwot)
        {
            swap(d[i], d[j]);
            j++;
        }
    d[prawy] = d[j]; d[j] = piwot;
    if(lewy < j - 1)  Sortuj_szybko(lewy, j - 1);
    if(j + 1 < prawy) Sortuj_szybko(j + 1, prawy);
}

void Sortuj_Nieszybko(int* lista, int n) {
    int pos = 0;
    while (pos < n) {
        if (pos == 0 || lista[pos] >= lista[pos - 1]) {
            pos++;
        } else {
            swap(lista[pos], lista[pos - 1]);
            pos = pos - 1;
        }
    }
}
// Program główny
//---------------

int main()
```

```
{
    int i;
    int iloscciagowdopierwszego = 10;
    int iloscciagowdodrugiego = 10;
    int ciag1[N];
    int ciag[N];
    FILE *glownyplik;
    FILE *sortowaniePierwsze;
    FILE *sortowanieDrugie;

    glownyplik = fopen("generowaneciagi.txt","a+");
    srand((unsigned)time(NULL));

    for(int j = 0; j < iloscCiagow; j++) {
        for (i = 0; i < N; i++) {
            ciag1[i] = rand() % 100;
            fprintf(glownyplik, "%d ", ciag1[i]);
        }
        fprintf(glownyplik, "%s", "\n");
    }

// Sortujemy
    fseek(glownyplik,0,0);
    sortowaniePierwsze = fopen("SortowaniePierwsze.txt","a");
    sortowanieDrugie = fopen("SortowanieDrugie.txt","a");
    for(int j = 0; j < iloscciagowdopierwszego; j++) {
        for (i = 0; i < N; i++) {
            fscanf(glownyplik, "%d", &d[i]);
        }
        Sortuj_szybko(0, N - 1);
        for (i = 0; i < N; i++) {
            fprintf(sortowaniePierwsze, "%d ", d[i]);
        }
        fprintf(sortowaniePierwsze, "%s", "\n");
    }
    for(int j = 0; j < iloscciagowdodrugiego; j++) {
        for (i = 0; i < N; i++) {
            fscanf(glownyplik, "%d", &ciag[i]);
        }
        Sortuj_Nieszybko(ciag, N);
        for (i = 0; i < N; i++) {
            fprintf(sortowanieDrugie, "%d ", ciag[i]);
        }
        fprintf(sortowanieDrugie, "%s", "\n");
    }

    fclose(glownyplik);
    fclose(sortowaniePierwsze);
    fclose(sortowanieDrugie);
    return 0;
}
```