Sprawozdanie z projektu numer 2

Sokal Mateusz

Rzeszów 2021
Inżynieria i Analiza Danych
Grupa projektowa 7

1. Opis podstaw teoretycznych

Tematem projektu było porównanie czasu pracy dwóch algorytmów sortujących: sortowania gnoma oraz sortowania przez scalanie

Sortowanie gnoma (ang. gnome sort) – jeden ze starszych, oraz prostszych algorytmów sortowania. Idea tej metody opiera się na przeglądaniu danych z tablicy po kolei, do momentu natrafienia na element nie spełniający warunku (większy/mniejszy lub równy elementowi poprzedniemu) wtedy pobieramy numer danego indeksu i wstawiamy go w miejsce, gdzie będzie spełniał warunek.

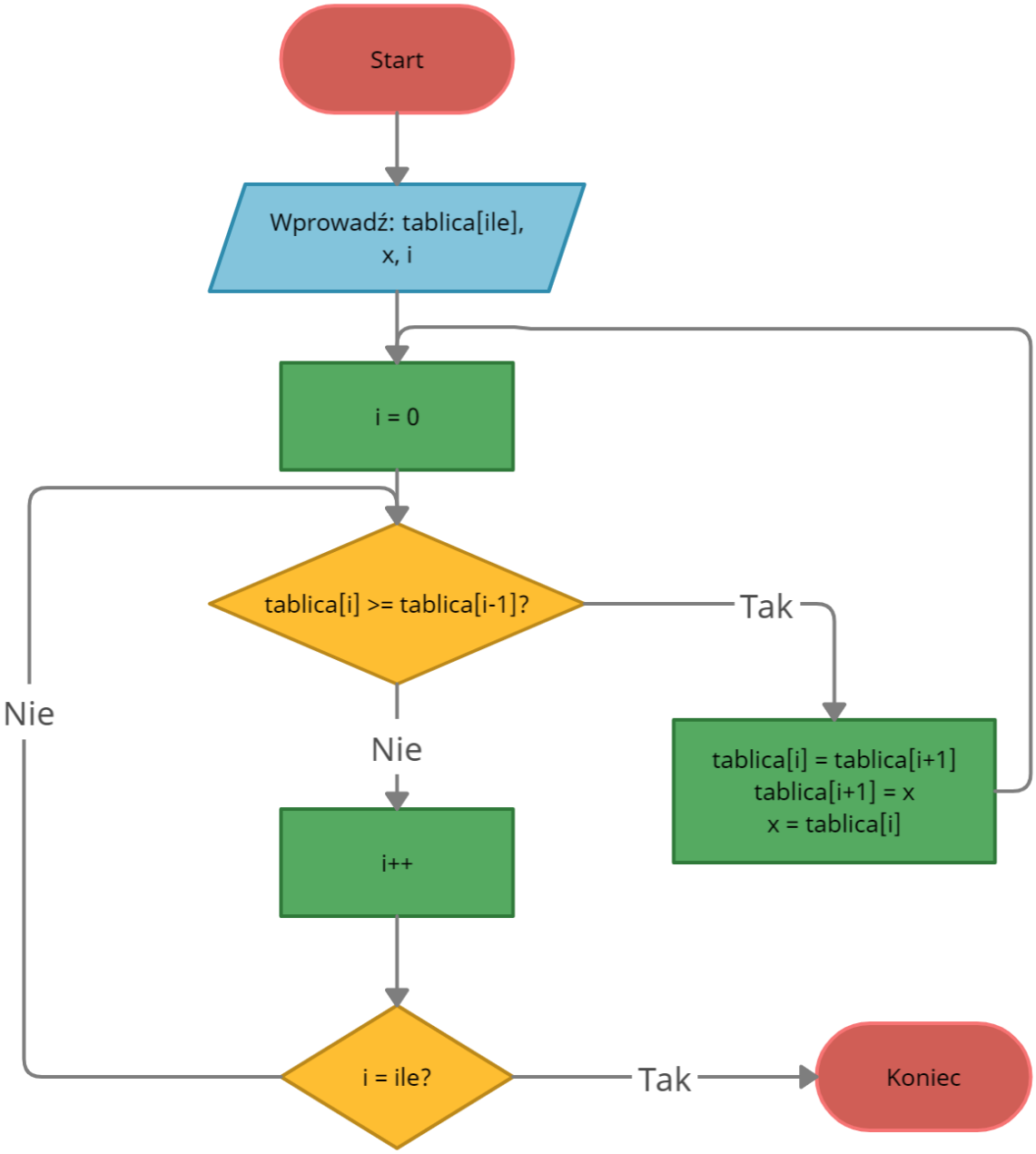
Sortowanie przez scalanie (ang. merge sort) – algorytm stosujący metodę „dziel i zwyciężaj" (ang. „divide and conquer"). Tą metodę sortowania odkrył John von Neumann.

Zasada działania tego algorytmu opiera się na trzech krokach:

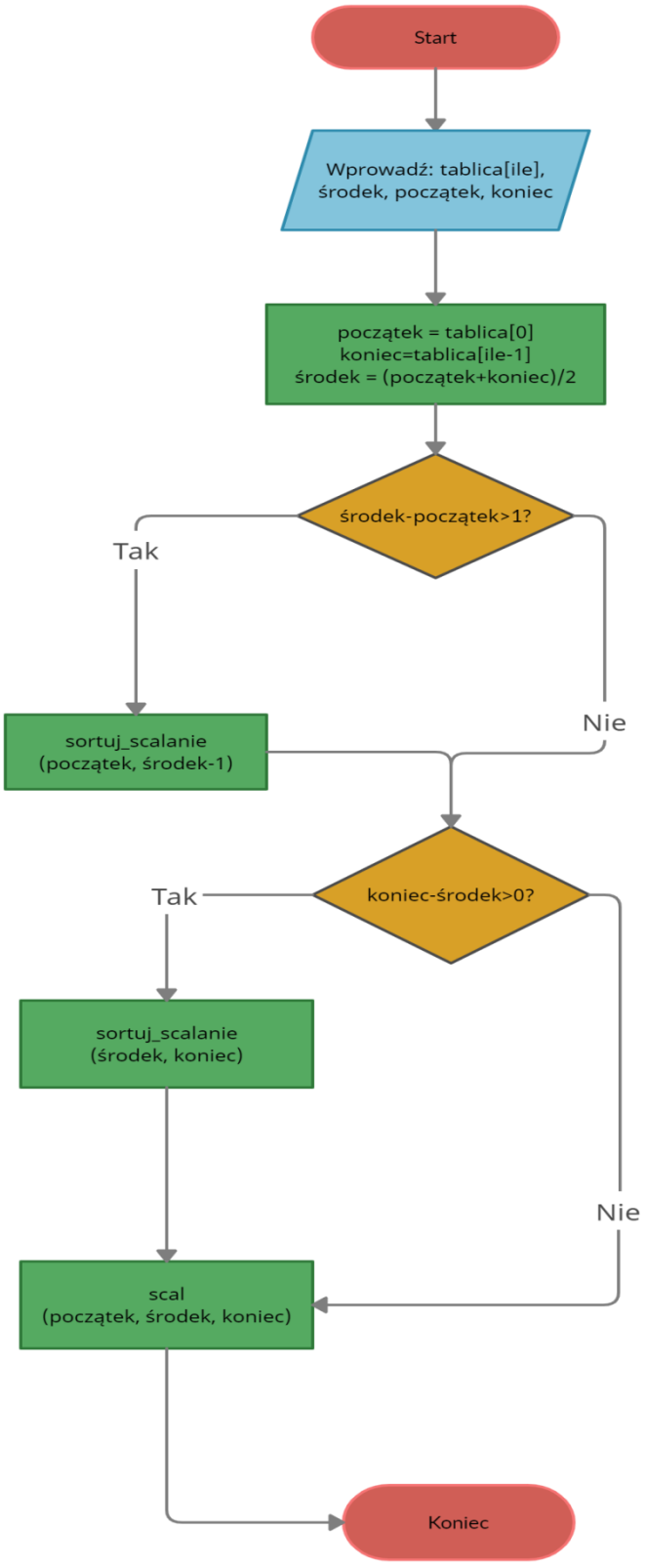
1. Podziel zestaw danych na dwie części
2. Zastosuj sortowanie przez scalanie dla obydwu z nich, chyba, że pozostał tylko jeden element
3. Połącz posortowane ciągi w jeden

1. Schematy blokowe

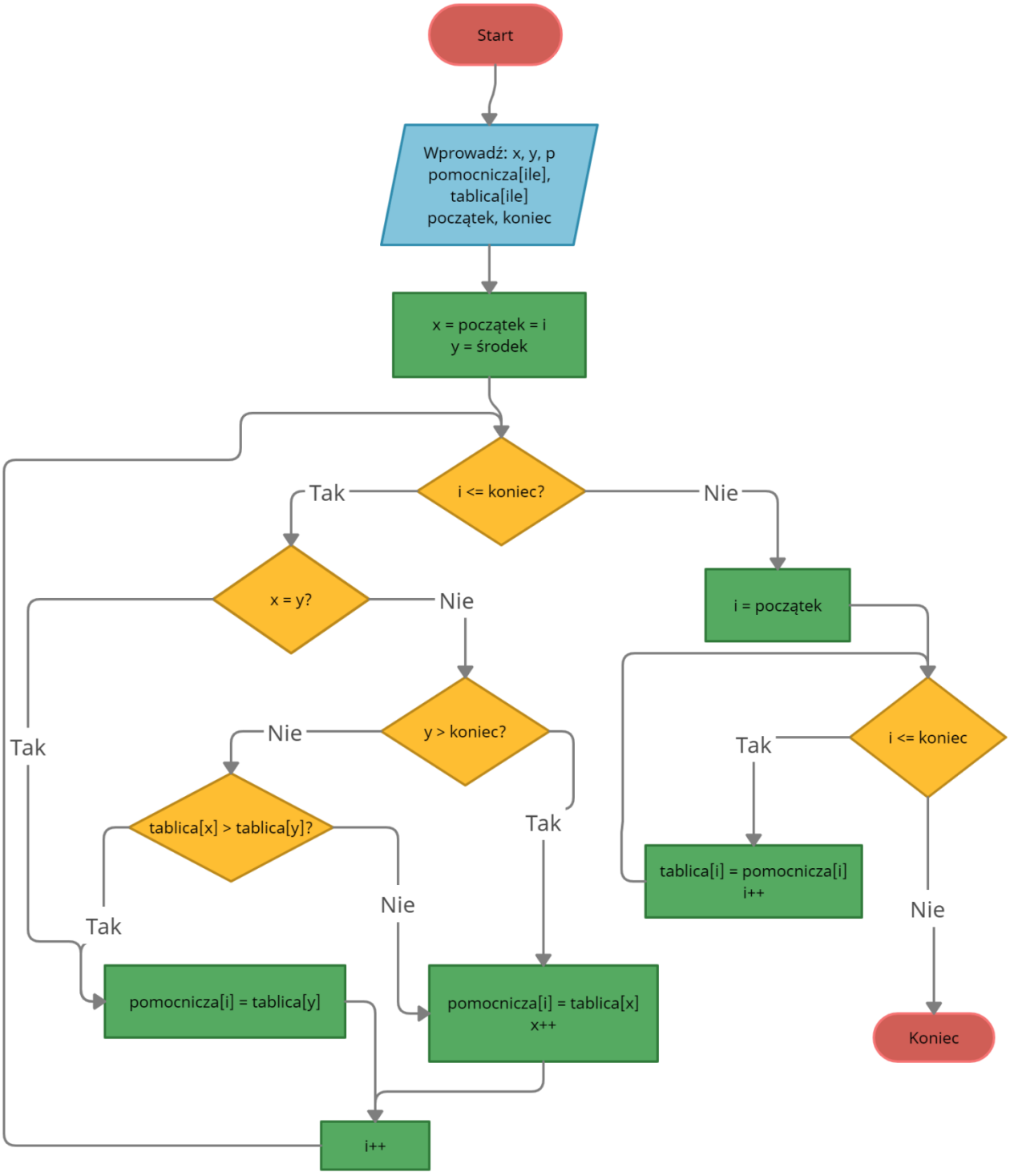
Sortowanie gnoma:



Sortowanie przez scalanie
(algorytm sortujący)



Sortowanie przez scalanie
(algorytm scalający)



2. Algorytmy w pseudokodzie

Sortowanie bąbelkowe:

tab - lista elementów do posortowania

ile - liczba_elementów(d)

Dla j = 1,2,...,ile – 1 **wykonuj**

Dla i = 1,2,...,ile – 1 **wykonuj**

Jeżeli tab[i]> tab[i + 1]
tab[i] tab[i + 1]

Zakończ

Sortowanie przez scalanie:

tab – lista elementów do posortowania,

pom – tablica pomocnicza,

początek1 - pierwszy element w pierwszym podzbiorze,

koniec - indeks ostatniego elementu w drugim podzbiorze,

początek2 - indeks pierwszego elementu w drugim podzbiorze

Pseudokod algorytmu scalania:

$x \leftarrow$ początek1; $y \leftarrow$ początek2; $i \leftarrow$ początek1

Dla i = początek1, początek1 + 1, ..., koniec:
Jeśli (x = początek2) $\vee$ (y $\leq$ koniec i d[x] > d[y]),
to pom [i] $\leftarrow$ tab [y]; y $\leftarrow$ y + 1
W przeciwnym wypadku pom [i] $\leftarrow$ tab [x]; x $\leftarrow$ x + 1

Dla i = początek1, początek1 + 1,..., koniec:
tab [i] $\leftarrow$ pom [i]

Zakończ

Pseudokod algorytmu sortowania:

początek2 ← (początek1 + koniec + 1) **div 2**

Jeśli początek2 - początek1 > 1, **to Sortuj_scalanie**(początek1, początek2 - 1)

Jeśli koniec - początek1 > 0, **to Sortuj_scalanie**(początek2, koniec)

Scalaj(początek1, początek2, koniec)

Zakończ

3. Algorytmy w C++

Sortowanie gnoma:

```
void gnome_sort(int *tablica, int ile)
{
    int x = 0;
    while (x < ile)
    {
        if (x == 0 || tablica[x] >= tablica[x - 1])
        {
            x++;
        }
        else
        {
            swap(tablica[x], tablica[x - 1]);
            x = x - 1;
        }
    }
}
```

Sortowanie przez scalanie:

```
void merge_sort(int *tablica, int lewy, int prawy)
{
    if(prawy<=lewy)
    {
        return;
    }
    int srodek = (prawy+lewy)/2;

    merge_sort(tablica, lewy, srodek);
    merge_sort(tablica, srodek+1, prawy);

    merging(tablica, lewy, srodek, prawy);
}
```

Scalanie:

```
void merging(int *tablica, int lewy, int srodek, int prawy)
{
    int i = lewy, j = srodek + 1;
    int pomocnicza[ile];

    for(int i = lewy; i<=prawy; i++)
        pomocnicza[i] = tablica[i];

    for(int k=lewy; k<=prawy; k++)
        if(i<=srodek)
            if(j <= prawy)
                if(pomocnicza[j]<pomocnicza[i])
                    tablica[k] = pomocnicza[j++];
                else
```

```
                tablica[k] = pomocnicza[i++];
        else
            tablica[k] = pomocnicza[i++];
    else
        tablica[k] = pomocnicza[j++];
```

4. Scalanie vs. Gnom

Liczba danych	Czas G. Sort (s)	Czas S. Sort (s)
10000	0,238	0,001
20000	0,942	0,004
30000	2,121	0,005
40000	4,06	0,008
50000	5,913	0,011
60000	8,485	0,016
70000	11,477	0,021
80000	15,715	0,028
90000	19,142	0,034
100000	24,23	0,042

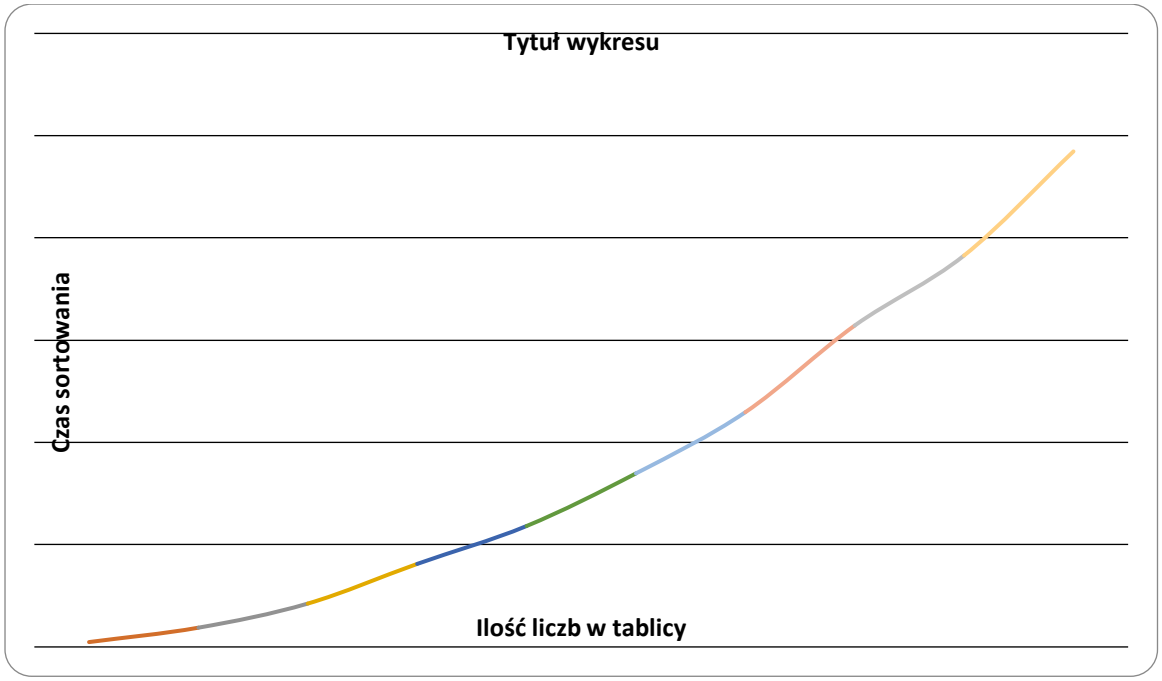
Tabela powyżej przedstawia czas, w którym obydwa algorytmy posortowały tabele o takiej samej ilości, oraz wartościach. Różnica jest szczególnie zauważalna, gdy ilość danych przekroczy 10 tys. liczb.

5. Wykresy złożoności obliczeniowej

Sortowanie gnoma

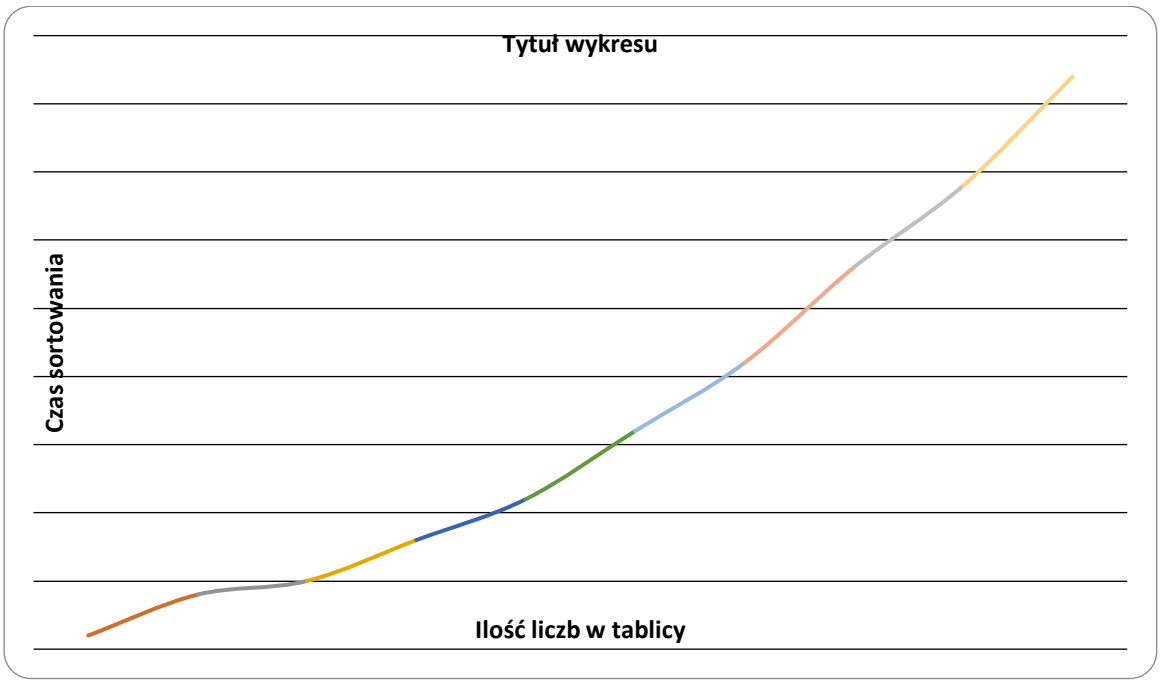
Złożoność obliczeniowa algorytmu gnoma w „średnim” przypadku wynosi O(n^2), które jednak zbliża się do O(n), jeśli zbiór liczb jest bliski posortowania.

Wykres tego sposobu sortowania wygląda następująco.



Sortowanie przez scalanie

Sortowanie przez scalanie należy do algorytmów szybkich. Złożoność obliczeniowa takich algorytmów wynosi O(n*log n). Warto też dodać, że złożoność obliczeniowa takich metod sortowania nie zmienia się w żadnym przypadku.



6. Wnioski

Rozważając wszystkie podane wyżej informacje można dojść do konkluzji, że sortowanie przez scalanie, oraz inne algorytmy typu szybkich są zdecydowanie efektywniejsze, pomimo dłuższych i bardziej skomplikowanych instrukcji. Algorytmy sortowania ze złożonością obliczeniową $O(n^2)$ tracą swoją wydajność, gdy ilość danych do posortowania przekracza ok. 10 tys. liczb. Warto też zauważyć, że przypadki optymistyczne, bądź pesymistyczne mają znikomy wpływ na czas pracy algorytmów „szybkich”