Algorytmy i struktury danych

Sprawozdanie – projekt 2.

Zadanie

Sortowanie bąbelkowe, sortowanie przez zliczanie

Adrian Jakubowski

Inżynieria i analiza danych, 1. rok, grupa 3.

Spis treści

1. Wstęp

1.1. Temat

Zaimplementuj sortowanie bąbelkowe oraz sortowanie przez zliczanie.

1) przedstaw schematy blokowe algorytmów oraz pseudokod odpowiadający obu schematom
2) przedstaw teoretyczne podstawy obu metod
3) wykonaj testy porównujące działanie obu metod na różnych próbkach danych i przedstaw ich wyniki sprawozdaniu
4) omów złożoność obliczeniową obu algorytmów
5) przedstaw w postaci wykresów t(N) złożoność czasową obu algorytmów dla przypadków oczekiwanego optymistycznego/ pesymistycznego ("odpowiednio preparując" dane do posortowania dla każdego z algorytmów) otrzymaną eksperymentalnie w wyniku serii testów dla rosnących próbek danych N

1.2.Opis problemu

Problemem w powyższym zadaniu jest posortowanie rosnąco elementów, które występują w tablicy, zarówno za pomocą algorytmu sortowania bąbelkowego, jak i algorytmu sortowania przez zliczanie. Rozmiar tablicy oraz zakres liczb w niej występujących nie został określony, więc zdecydowałem, że rozmiar wprowadza użytkownik z klawiatury, natomiast zakres liczb będzie się zawierał w przedziale [1,rozmiar-1]. Ponadto, stworzyłem program, aby generował on losowe liczby do powyżej wspomnianej tablicy. Nie zabrakło również zliczenia czasów działania tychże algorytmów oraz wyświetlenia wszelkich wyników w pliku tekstowym.

2. Analiza i projektowanie

2.1. Opis podstaw teoretycznych zagadnienia

Rozwiązanie zagadnienia sortowania elementów w tablicy za pomocą sortowania bąbelkowego opierało się przede wszystkim na porównywaniu kolejnych elementów tablicy. Jeśli dany element o i-tym indeksie był większy od elementu o i+1 indeksie to zamieniano te dane elementy. Czynność powtarzano przy użyciu zagnieżdżonych pętli for. W przypadku algorytmu sortowania przez zliczanie, podstawą programu jest przede wszystkim zliczanie ile razy występuje dany element w tablicy. Następnie dodawano do siebie ilość danych liczb, aby ustalić, na którym indeksie tablicy umieścić ostatnią z powtarzających się liczb. Na koniec, za pomocą pomocniczej tablicy wstawiano posortowane elementy taką ilość razy, ile wynosił licznik danej liczby.

2.2. Opis szczegółów implementacji problemu

2.2.1. Biblioteki

iostream	Biblioteka we-wyjścia. Deklaruje obiekty, które kontrolują odczytywanie ze strumieni standardowych i zapisywanie ich w tych strumieniach. Jest to często jedyny nagłówek potrzebny do wprowadzania danych i danych wyjściowych.
ctime	Udostępnia kilka typów danych, dzięki którym możemy odczytywać czas i wykonywać proste operacje na czasie, takie jak dodawanie czy odejmowanie. W przypadku tego zadania przydatna do generowania liczb pseudolosowych.
fstream	Dostarcza funkcji pozwalających nam zarówno zapisywać pliki jak i je odczytywać.
stdio.h	Wysyła sformatowane dane do standardowego strumienia wyjściowego. Tutaj – potrzebna przy liczeniu czasu działania programu.
chrono	Definiuje klasy i funkcje, które reprezentują czasy trwania i czasy natychmiastowe. Tutaj – potrzebna przy liczeniu czasu działania programu.

2.2.2.Zmienne

plik	zmienna globalna, plikowa typu ofstream. Służy do zapisywania danych do pliku
rozmiarPobrany	zmienna typu int, jest podawana przez użytkownika na początku działania programu
rozmiar	zmienna stała const typu int. Przechowuje rozmiar tablicy liczb całkowitych
begin	zmienna typu auto, która zapisuje i przechowuje czas na początku działania programu
end	zmienna typu auto, która zapisuje i przechowuje czas na końcu działania programu
elapsed	zmienna typu double, przechowuje czas, który upłynął przez okres działania programu
t[rozmiar]	zmienna tablicowa typu int, przechowuje elementy tablicy o rozmiarze „rozmiar"; na niej odbywa się większość operacji programu
L[rozmiar]	zmienna tablicowa typu int, przechowuje elementy tablicy o rozmiarze „rozmiar"; w programie pełni rolę tablicy liczników danych elementów pomocną w algorytmie sortowania przez zliczanie
S[rozmiar]	zmienna tablicowa typu int, przechowuje elementy tablicy o rozmiarze „rozmiar"; w programie pełni rolę pomocniczej tablicy, do której zapisuje się ostatecznie posortowane elementy w sortowaniu przez zliczanie
i, j	zmienne typu int, są iteratorami pętli for

2.2.3.Funkcje

main	główna funkcja programu, zawiera inicjalizacje i deklaracje zmiennych, funkcje związane z zapisywaniem danych do pliku; w niej wywoływane są pozostałe funkcje
wypelnianieTablicy	zawiera instrukcje służące do generowania liczb pseudolosowych, a także wypełnia nimi tablicę, którą ostatecznie wywołuje do pliku
sortowanieBabelkowe	wdraża algorytm popularnego sortowania bąbelkowego (bubble sort); zlicza czas trwania tego algorytmu, a ostatecznie wywołuje zliczony czas oraz posortowaną tablicę elementów
sortowanieZliczanie	wprowadza algorytm sortowania przez zliczanie; zlicza czas trwania tego algorytmu, a ostatecznie wywołuje zliczony czas oraz posortowaną tablicę elementów

2.3. Pseudokod

2.3.1. Sortowanie bąbelkowe

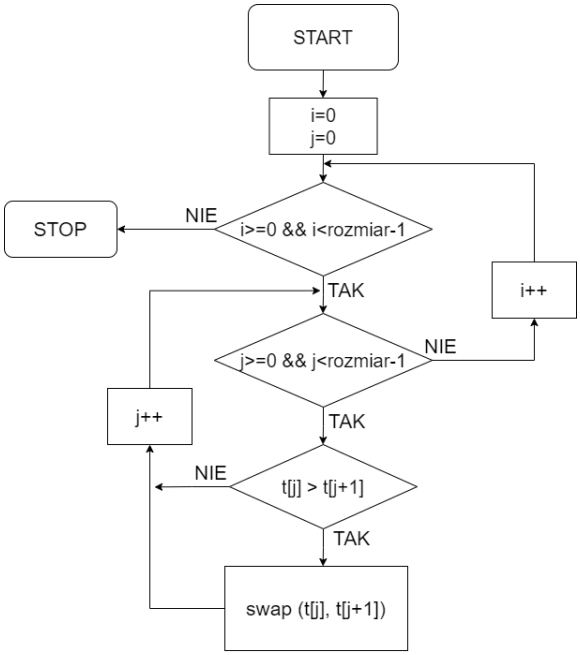
```
i, j <- 0
 dla i<-0 do rozmiar-1
    wykonuj
    dla j<-0 do rozmiar -1
       wykonuj
       jeżeli t[j]>t[j+1]
          zamień (t[j], t[j+1])
```

2.3.2.Sortowanie przez zliczanie

```
L[rozmiar] <- {0}
S[rozmiar] <- {0}

i <- 0
dla i<-0 do rozmiar
   wykonuj
   L[t[i]]++;
dla i<-0 do rozmiar
   wykonuj
   L[i]=L[i]+L[i-1]
dla i<-rozmiar-1 do 0
   wykonuj
   S[--L[t[i]]]=t[i]
```

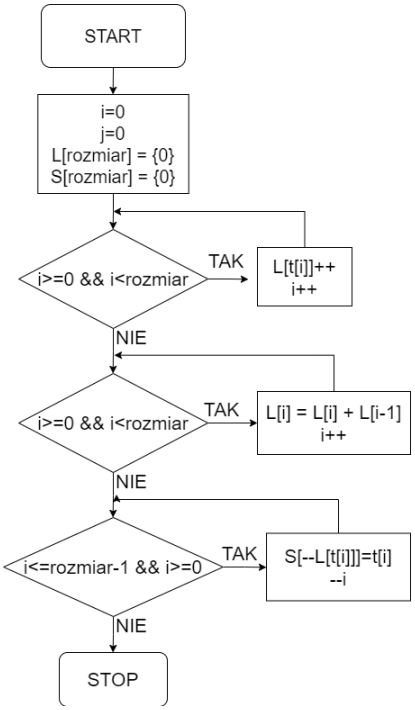
2.4. Schematy blokowe

2.4.1. Sortowanie bąbelkowe



Rysunek 1

2.4.2. Sortowanie przez zliczanie



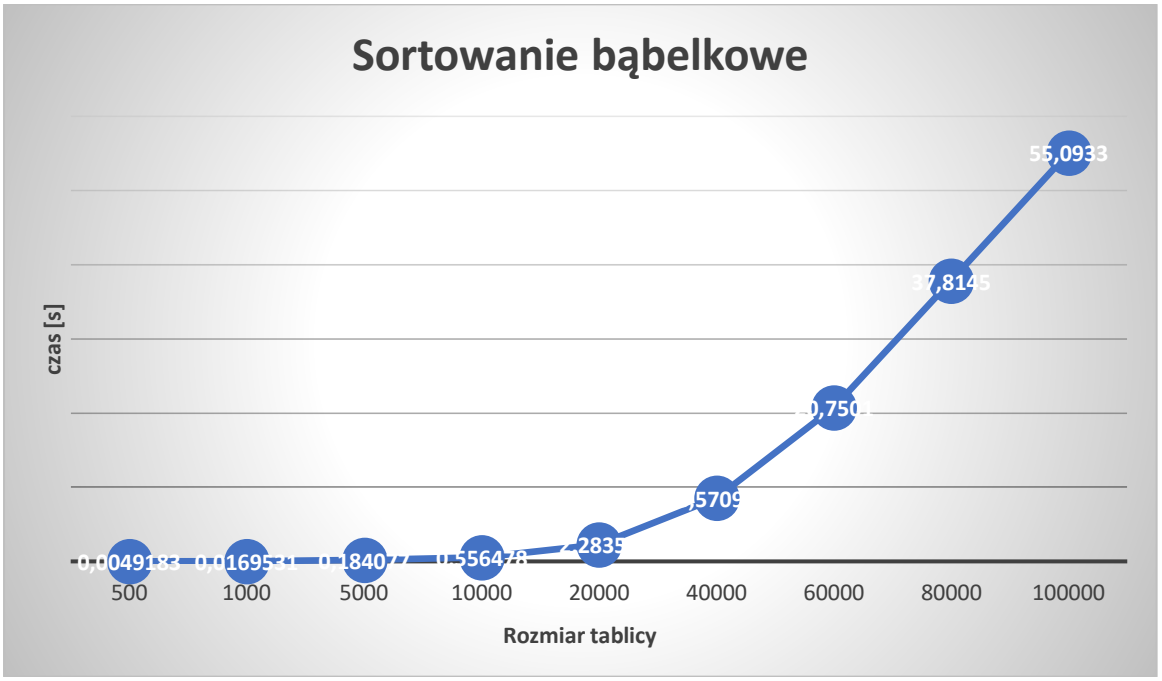
Rysunek 2

2.5. Kod programu

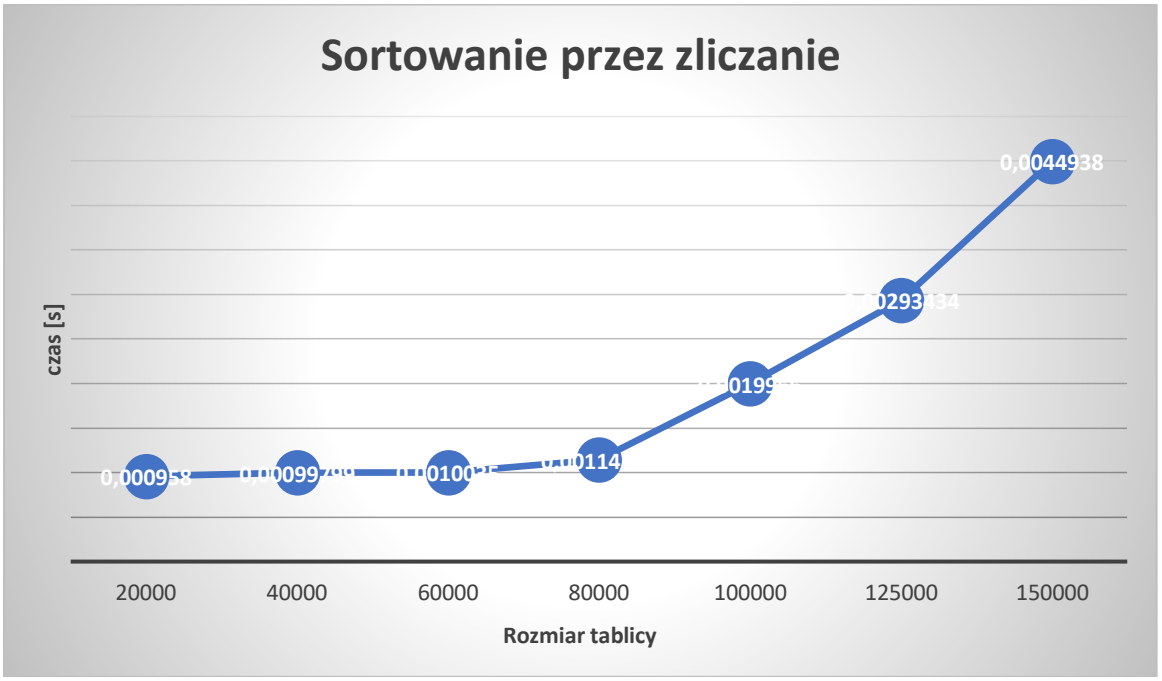
```
#include <iostream>
#include <ctime> //do losowania liczb pseudolosowych
#include <fstream> //do zapisu wyniku do pliku
#include <stdio.h> //do liczenia czasu
#include <chrono> // do liczenia czasu


using namespace std;

void wypelnianieTablicy(int t[], int rozmiar);
void sortowanieBabelkowe (int t[], int rozmiar);
void sortowanieZliczanie (int t[],int rozmiar);

ofstream plik;

int main()
{
    int rozmiarPobrany;
    cout<<"Podaj ilosc elementow do posortowania: "<<endl;
    cin>>rozmiarPobrany;
    const int rozmiar=rozmiarPobrany; //constans, bo rozmiar jest staly przez caly program
    int t[rozmiar];

    plik.open("wyniki.txt");

    wypelnianieTablicy(t,rozmiar);
    plik<<endl<<endl;
    sortowanieBabelkowe(t,rozmiar);
    plik<<endl;
    sortowanieZliczanie (t,rozmiar);


    plik.close();

    return 0;
}

// ** WYPE£NIANIE TABLICY LICZBAMI PSEUDOLOSOWYMI **
void wypelnianieTablicy(int t[], int rozmiar)
{
    srand(time(NULL)); // do losowania liczb pseudolosowych

    plik<<"Wylosowana tablica liczb:"<<endl;

    //wypelnienie tablicy liczbami losowymi z zakresu 1 - rozmiar-1
    for(int i = 0; i<rozmiar; i++)
    {
        t[i]=rand()%(rozmiar-1)+1;
        plik<<t[i]<<" ";
    }
}

void sortowanieBabelkowe (int t[], int rozmiar)
{
    auto                    begin                   = std::chrono::high_resolution_clock::now();
//poczatek liczenia czasu

    for(int i = 0; i < rozmiar - 1; i++)
        for(int j = 0; j < rozmiar - 1; j++)
            if(t[j] > t[j + 1])       //za pomoca petli for przegladamy elementy, jesli element o indeksie j-tym jest wiekszy od elementu od indeksie j-tym+1
                swap(t[j], t[j + 1]); // to zamieniamy je miejscami

    auto                    end                     = std::chrono::high_resolution_clock::now();
//koniec liczenia czasu
    double                  elapsed                 = double(std::chrono::duration_cast<std::chrono::nanoseconds>(end - begin).count()); //czas obliczen - roznica konca i poczatku


// Wyświetlamy wynik sortowania
    plik<< "Sortowanie babelkowe:"<<endl;
    for(int i = 0; i < rozmiar; i++) plik<< t[i]<<" ";
    plik<<endl<<"Wykonano w: "<< elapsed/(1e9)<<" dla "<<rozmiar<<" elementow"<<endl;

    plik << endl;
}

void sortowanieZliczanie (int t[],int rozmiar)
{
    auto                    begin                   = std::chrono::high_resolution_clock::now();
//poczatek liczenia czasu

    int L[rozmiar]={0};
    int S[rozmiar]={0};

    for(int i = 0 ; i < rozmiar ; i++)
    L[t[i]]++;                              //tablica L-licznikow zlicza ile razy wystapil w tablicy element o indeksie i

    for(int i = 0 ; i < rozmiar ; i++)      //za pomoca tablicy L[i] ustalamy ostatnia pozycje w posortowanej tablicy
    L[i] = L[i] + L[i-1];                //elementu z pod indeksu i - teraz stan licznika L nie jest iloscia wystepowania danej liczby
                                            //tylko ostatnia pozycja w tablicy

    for(int i = rozmiar-1 ; i >= 0 ; --i)    //wstawiamy elementu na odpowiednią pozycję do tablicy S
      S[--L[t[i]]] = t[i];                      //zmniejszajac ilosc Licznikow danej liczby z pod indeksu i


     auto                   end                     = std::chrono::high_resolution_clock::now();
//koniec liczenia czasu
    double                  elapsed                 = double(std::chrono::duration_cast<std::chrono::nanoseconds>(end - begin).count()); //czas obliczen - roznica konca i poczatku


     plik<< "Sortowanie przez zliczanie:"<<endl;
    for(int i = 0; i < rozmiar; i++) plik<< S[i]<<" ";
    plik<<endl<<"Wykonano w: "<< elapsed/(1e9)<<" dla "<<rozmiar<<" elementow"<<endl;

    plik << endl;
}
```

3. Wyniki

3.1. Czasy obliczeń



Rysunek 3



Rysunek 4

Rysunek 3 przedstawia wykres czasu obliczeń algorytmu sortowania bąbelkowego w zależności od rozmiaru tablicy, natomiast *Rysunek 4* ukazuje wykres czasu obliczeń w przypadku algorytmu sortowania przez zliczanie. Porównując oba wykresy, nie ma żadnej wątpliwości, że bardziej efektywny jest algorytm sortowania przez zliczania, gdzie dla 100000 elementów tablicy czas działania programu jest równy niespełna 0,002 sekundy, natomiast dla równej tablicy sortowanie bąbelkowe trwa około 55 sekund.

3.2. Złożoność obliczeniowa

Powyżej przedstawiony, popularny algorytm bubble sort posiada szacowaną złożoność rzędu $O(n^2)$. Wskazuje na to zagnieżdżona pętla „for", która przeszukuje elementy tablicy i zamienia je w przypadku, gdy j-ty element jest większy od j+1 elementu.

W przypadku algorytmu sortowania przez zliczanie, już po wykresach czasu obliczeń można łatwo wywnioskować, że algorytm ten posiada korzystniejszą złożoność obliczeniową, aniżeli sortowanie bąbelkowe. Złożoność sortowania przez zliczanie szacuje się jako O(n).

3.3. Przykłady działania programu

Poniżej przedstawione przykłady zostały skopiowane z pliku „wyniki.txt", do którego program zapisuje rezultaty swoich działań.

Dla rozmiaru tablicy równego 10:

Wylosowana tablica liczb:
2 4 9 1 7 9 7 7 3 2

Sortowanie babelkowe:
1 2 2 3 4 7 7 7 9 9

Sortowanie przez zliczanie:
1 2 2 3 4 7 7 7 9 9

Dla rozmiaru tablicy równego 25:

Wylosowana tablica liczb:
5 11 11 20 23 13 11 4 15 6 11 19 3 14 16 13 16 18 1 20 13 1 7 15 24

Sortowanie babelkowe:
1 1 3 4 5 6 7 11 11 11 11 13 13 13 14 15 15 16 16 18 19 20 20 23 24

Sortowanie przez zliczanie:
1 1 3 4 5 6 7 11 11 11 11 13 13 13 14 15 15 16 16 18 19 20 20 23 24

Dla rozmiaru tablicy równego 50:

Wylosowana tablica liczb:
21 41 22 23 25 45 37 43 46 11 46 22 2 49 47 46 3 12 16 18 14 48 38 17 46 3 15 18 15 2 41 29 28 3 15 26 8 24 23 33 11 36 10 14 45 48 46 19 11 47

Sortowanie babelkowe:
2 2 3 3 3 8 10 11 11 11 12 14 14 15 15 15 16 17 18 18 19 21 22 22 23 23 24 25 26 28 29 33 36 37 38 41 41 43 45 45 46 46 46 46 46 47 47 48 48 49

Sortowanie przez zliczanie:
2 2 3 3 3 8 10 11 11 11 12 14 14 15 15 15 16 17 18 18 19 21 22 22 23 23 24 25 26 28 29 33 36 37 38 41 41 43 45 45 46 46 46 46 46 47 47 48 48 49

Jak pokazują powyższe przykłady, oba algorytmy sortowania dobrze spełniają swą rolę. Ponadto, program zapisuje do tablicy liczby z zakresu od 1 do rozmiar-1, co również jest zgodne z moim zamierzonym początkowo planem.

4.Podsumowanie

4.1. Wnioski

Podsumowując, mogę powiedzieć, że zagadnienie sortowania bąbelkowego od początku nie było mi obce, gdyż zapoznałem się z nim już w szkole średniej. Niewiadomym był dla mnie aspekt złożoności obliczeniowej tego algorytmu, więc musiałem nabyć wiedzę na ten temat. Jeśli chodzi o sortowanie przez zliczanie, było to dla mnie całkowicie nowe zagadnienie, które musiałem zrozumieć przy pomocy stron internetowych zawartych w podrozdziale *Źródła i pomoce*. Mój kod programu powstał również na zasadzie tychże stron, nie inaczej z zapoznaniem się w kwestii złożoności obliczeniowej.

4.2.Źródła i pomoce

- informacje na temat złożoności obliczeniowej oraz jej obliczania -
https://www.samouczekprogramisty.pl/podstawy-zlozonosci-obliczeniowej/

- dokładne opisy i definicje bibliotek
https://docs.microsoft.com/pl-pl/cpp/standard-library/cpp-standard-library-header-files?view=msvc-160
https://cpp0x.pl/kursy/Kurs-C++/Dodatkowe-materialy/Obsluga-plikow/305

- informacje na temat sortowania przez zliczanie
https://eduinf.waw.pl/inf/alg/003_sort/0023.php
https://www.classicistranieri.com/pl/articles/s/o/r/Sortowanie_przez_zliczanie.html
- informacje na temat sortowania bąbelkowego
https://eduinf.waw.pl/inf/alg/003_sort/0004.php