Algorytmy i struktury danych

Sprawozdanie – projekt 1.

Zadanie

Dla zadanej tablicy liczb całkowitych o rozmiarze n zawierającej wartości z przedziału [1,n-1]
znajdź element powtarzający się

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

Spis treści

1. Wstęp

1.1. Temat

Treść zadania jest następująca:

Dla zadanej tablicy liczb całkowitych o rozmiarze n zawierającej wartości z przedziału [1,n-1] znajdź element powtarzający się.

Przykład:
Wejście: 1, 2, 3, 4, 4
Wyjście: Powtarzający się element to 4

Wejście: 1, 2, 3, 4, 2
Wyjście: Powtarzający się element to 2

1.2.Opis problemu

Problemem w powyższym zadaniu jest znalezienie elementu, bądź elementów, które występują co najmniej dwa razy w tablicy o rozmiarze n, zawierającej liczby całkowite o wartościach z przedziału [1, n-1]. Tablica ta może być ręcznie wypełniona liczbami całkowitymi lub wygenerowana przy użyciu liczb pseudolosowych. Ja na potrzeby zadania wykorzystałem drugą opcję, generującą tablicę wypełnioną liczbami pseudolosowymi.

2. Analiza i projektowanie

2.1.Opis podstaw teoretycznych zagadnienia

Rozwiązanie zagadnienia powtarzających się elementów w tablicy opierało się przede wszystkim na porównywaniu kolejnych elementów tablicy. Jeśli dany element był równy elementowi o innym indeksie to znaczyło, że były to elementy powtarzające się.

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.
time.h	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
tablica[rozmiar]	zmienna tablicowa typu int, przechowuje elementy tablicy o rozmiarze „rozmiar”
i, k, j	zmienne typu int, są licznikami pętli for
licznik	zmienna typu int, służy do zliczania ile razy powtórzył się dany element tablicy

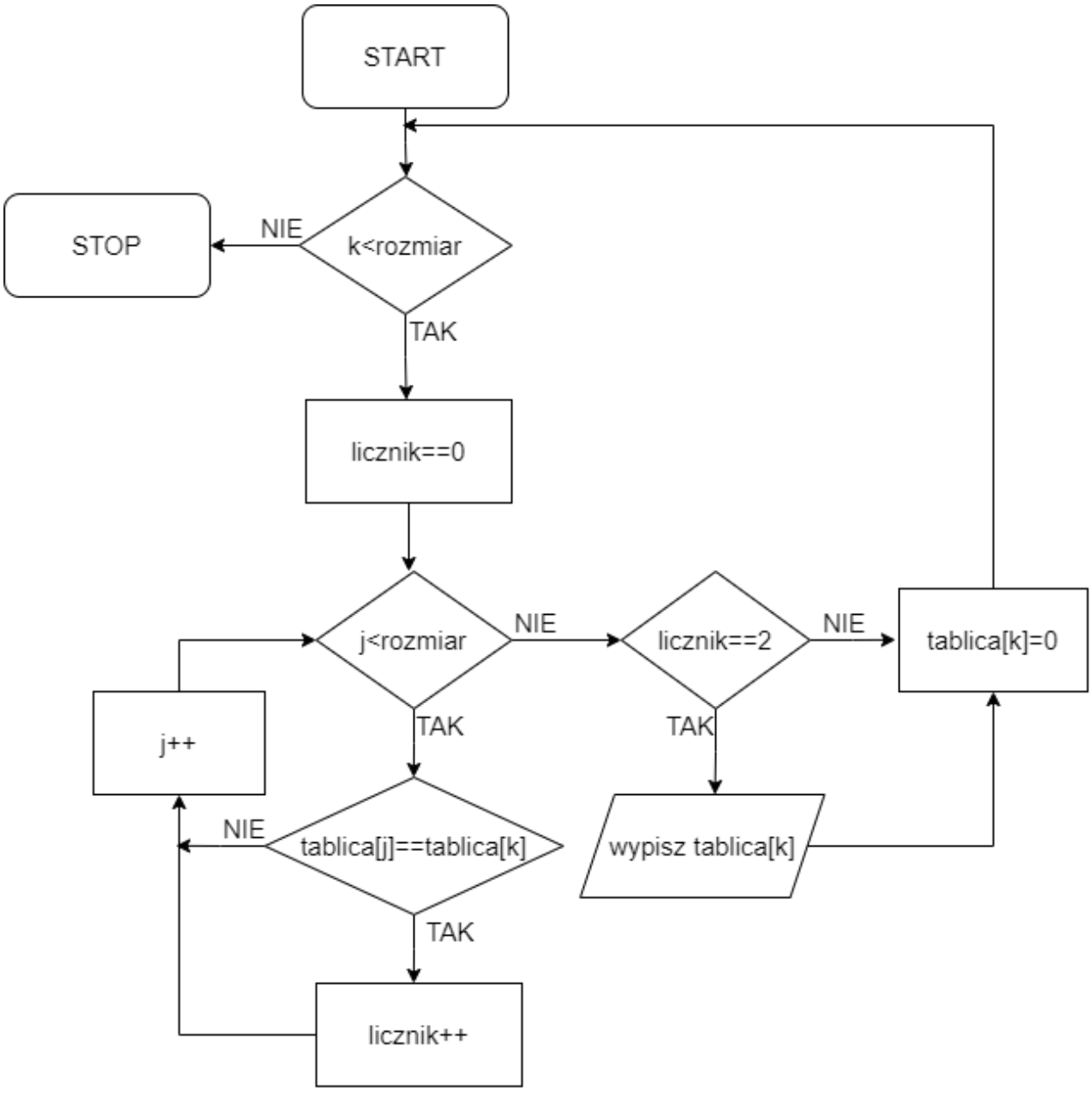
2.2.3.Funkcje

main	główna funkcja programu, zawiera inicjalizacje i deklaracje zmiennych, funkcje związane z obliczaniem czasu obliczeń oraz 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
szukanie	za pomocą zagnieżdżonych pętli for znajduje ona powtarzające się elementy tablicy, a wyniki działań zapisuje do pliku

2.3. Pseudokod

```
main(){

	wczytaj rozmiarPobrany;
		rozmiar <- rozmiarPobrany;

	wypelnianieTablicy(tablica, rozmiar);
	szukanie(tablica, rozmiar);
}


wypelnianieTablicy(){

	dla i <- 0 do rozmiar
		wykonuj
		tablica[i] <- rand()%(rozmiar-1)+1;
		zwróć tablica[i];
}

szukanie(){

	dla k<-0 do rozmiar
		wykonuj
		licznik <- 0;
		dla j<-0 rozmiar
			wykonuj
			tablica[j]==tablica[k];
			licznik++;

		jeżeli licznik==2
		zwróć tablica[k]

	tablica[k] <- 0
}
```

2.4.Schemat blokowy



Rysunek 1

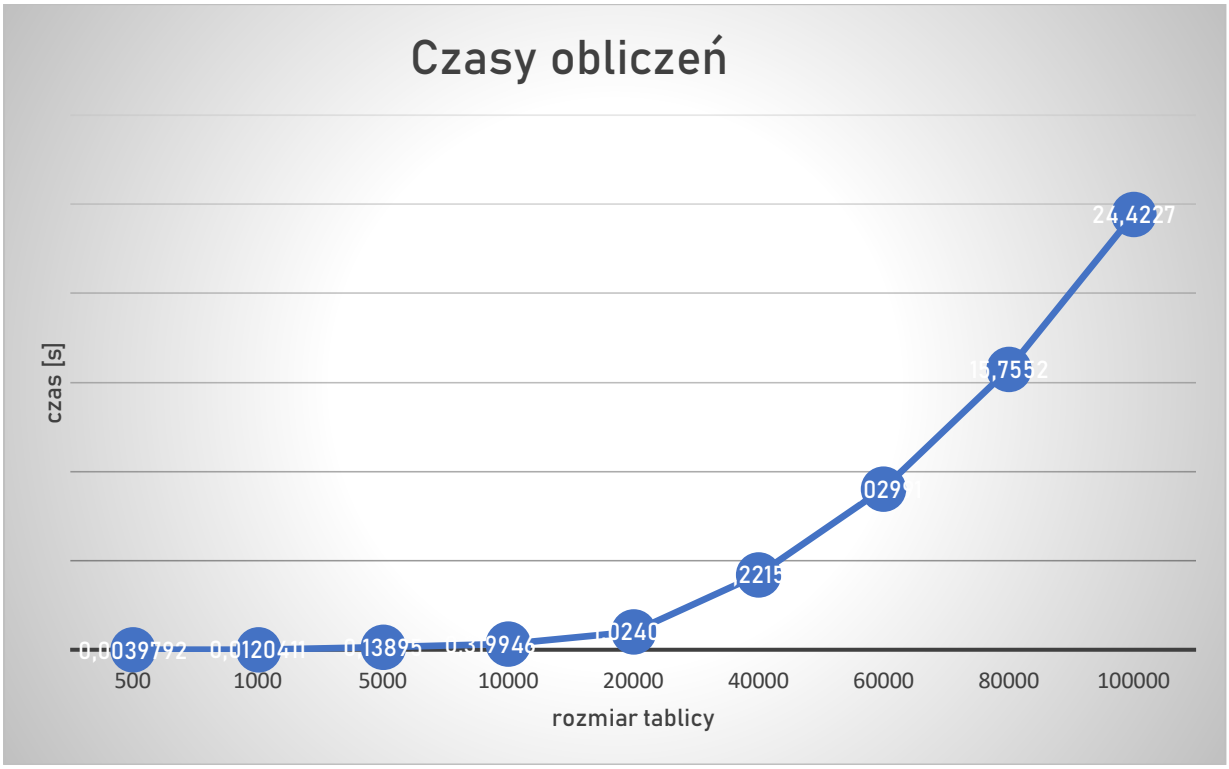
2.5. Kod programu

```
#include <iostream>
#include <time.h>
#include <fstream>
#include <stdio.h>
#include <chrono>

using namespace std;

void wypelnianieTablicy(int tablica[],
int rozmiar);
void szukanie(int tablica[], int
rozmiar);

ofstream plik;

int main()
{

    int rozmiarPobrany;
    cout<<"Podaj z ilu elementow ma
skladac sie tablica: "<<endl;
    cin>>rozmiarPobrany;
    const int rozmiar=rozmiarPobrany;
//constans, bo rozmiar jest staly przez
caly program

    int tablica[rozmiar];

    auto begin =
std::chrono::high_resolution_clock::now(
);

    plik.open("wyniki.txt");

    wypelnianieTablicy(tablica,
rozmiar);

    szukanie(tablica,rozmiar);

    auto end =
std::chrono::high_resolution_clock::now(
);
    double elapsed =
double(std::chrono::duration_cast<std::c
hrono::nanoseconds>(end -
begin).count());

    plik<<endl<<"Zmierzony czas: "<<
elapsed/(1e9)<<endl;


plik.close();

    return 0;}
```

```
// ** WYPEŁNIANIE TABLICY LICZBAMI
PSEUDOLOSOWYMI **
void wypelnianieTablicy(int tablica[],
int rozmiar)
{
    srand(time(NULL));

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

    for(int i = 0; i<rozmiar; i++)
    {
        tablica[i]=rand()%(rozmiar-1)+1;
//rozmiar-1, bo % z liczby musi być
mniejsze od tej liczby, a "+1" żeby było
do "rozmiar"
        plik<<tablica[i]<<", ";
    }
}


// ** SZUKANIE POWTARZAJACYCH SIE
ELEMENTOW TABLICY **
void szukanie(int tablica[], int
rozmiar)
{
    plik<<endl<<endl;

    plik<<"Powtarzajacy sie
element/elementy to:"<<endl;
    for(int k = 0; k<rozmiar; k++)
    {

        int licznik=0;

        for(int j = 0; j<rozmiar; j++)
            if(tablica[j]==tablica[k])
licznik++;


        if(licznik==2)
        {
            plik<<tablica[k]<<" ";
        }

        tablica[k]=0;
    }

    plik<<endl;

}
```

3. Wyniki

3.1. Czasy obliczeń



Rysunek 2

Rysunek 2 przedstawia wykres czasu obliczeń algorytmu w zależności od rozmiaru tablicy. Można z niego odczytać m.in. to, że pierwszą sekundę program przekracza dla około 20000 elementów tablicy. Barierę 10 sekund osiąga przy około 63000 elementach. Dla rozmiaru tablicy równego 100000, czas obliczeń wynosi natomiast około 24,5 sekundy.

3.2. Złożoność obliczeniowa

Powyżej przedstawiony algorytm posiada szacowaną złożoność rzędu $O(n^2)$. Wskazuje na to zagnieżdżona pętla „for" w funkcji „szukanie", która przeszukuje elementy tablicy, porównując je ze sobą w celu wskazania powtarzających się elementów.

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 5:

Wylosowana tablica liczb:
4, 2, 3, 3, 4,

Powtarzajacy sie element/elementy to:
4 3

Dla rozmiaru tablicy równego 10:

Wylosowana tablica liczb:
6, 1, 7, 5, 4, 4, 9, 6, 8, 7,

Powtarzajacy sie element/elementy to:
6 7 4

Dla rozmiaru tablicy równego 30:

Wylosowana tablica liczb:
8, 8, 27, 3, 7, 26, 7, 3, 22, 4, 6, 13, 18, 29, 16, 25, 17, 3, 27, 10, 21, 1, 16, 5, 12, 3, 22, 6, 5, 3,

Powtarzajacy sie element/elementy to:
8 27 7 22 6 16 5 3

Dla rozmiaru tablicy równego 50:

Wylosowana tablica liczb:
26, 38, 3, 6, 29, 34, 26, 20, 12, 12, 19, 19, 9, 19, 48, 48, 34, 18, 18, 7, 47, 15, 46, 26, 35, 5, 35, 44, 26, 26, 36, 10, 10, 38, 2, 49, 2, 8, 32, 48, 46, 8, 32, 17, 3, 19, 33, 12, 1, 24,

Powtarzajacy sie element/elementy to:
38 3 34 12 19 48 18 46 35 26 10 2 8 32

Jak widać na powyższych przykładach program poprawnie spełnia swoje zadanie polegające na znajdywaniu elementów tablicy. Ponadto, elementy tablicy zawierają się w poprawnych przedziałach (1, rozmiar-1).

4.Podsumowanie

4.1. Wnioski

Gwoli podsumowania można stwierdzić, że algorytm poprawnie spełnia swoją funkcję. Nie uniknionym jest także fakt, że z pewnością istnieje łatwiejsza wersja tego algorytmu, jednak brak doświadczenia i wiedzy z zakresu programowania nie pozwoliły mi, abym napisał program w lepszy sposób. Szczególne braki wiedzy zauważam również w kwestii złożoności obliczeniowej.

4.2.Źródła i pomoce

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

- wykres złożoności – https://www.geogebra.org/calculator

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