POLITECHNIKA
RZESZOWSKA
im. IGNACEGO ŁUKASIEWICZA

Zadanie projektowe 1

Opracowanie: Sabina Lizis

I rok Inżynierii i Analizy Danych, grupa 1 projektowa

Rzeszów 2021

Spis treści

1. Wstęp

Algorytm – skończony ciąg jasno zdefiniowanych czynności koniecznych do wykonania pewnego rodzaju zadań, sposób postępowania prowadzący do rozwiązania problemu[1]. Można go przedstawić na schemacie blokowym. Słowo „algorytm" pochodzi od łacińskiego słowa algorithmus, oznaczającego wykonywanie działań przy pomocy liczb arabskich (w odróżnieniu od abacism – przy pomocy abakusa), które z kolei wzięło się od nazwy „Algoritmi", zlatynizowanej wersji nazwiska „al-Chwarizmi" Abu Abdullaha Muhammada ibn Musy al-Chuwarizmiego, matematyka perskiego z IX wieku. Zadaniem algorytmu jest przeprowadzenie systemu z pewnego stanu początkowego do pożądanego stanu końcowego. Badaniem algorytmów zajmuje się algorytmika. Algorytm może zostać zaimplementowany w postaci programu komputerowego.[1]

2. Opis problemu

Problem dotyczył uporządkowania zadanej tablicy wielu powtarzających się elementów, tak aby zgromadzić te same elementy zachowując kolejność ich pierwszego wystąpienia.

Przykład:

Wejście: [1,2,3,1,2,1]

Wyjście: [1,1,1,2,2,3]

Wejście: [5,4,55,3,1,2,2]

Wyjście: [5,5,5,44,3,1,2,2]

3. Opis podstaw teoretycznych zagadnienia

3.1. Sortowanie

Jeden z podstawowych problemów informatyki polegający na uporządkowaniu zbioru danych względem pewnych cech charakterystycznych każdego elementu tego zbioru. Szczególnym przypadkiem jest sortowanie względem wartości każdego elementu, np. sortowanie liczb. Algorytmy sortowania są stosowane w celu uporządkowania danych, umożliwienia stosowania wydajniejszych algorytmów (np. wyszukiwania) i prezentacji danych w sposób czytelniejszy dla człowieka. Jeśli jest konieczne posortowanie zbioru większego niż wielkość dostępnej pamięci, stosuje się algorytmy sortowania zewnętrznego. Algorytmy, do działania których nie jest potrzebna większa niż stała pamięć dodatkowa (elementy sortowane przechowywane są przez cały czas w tablicy wejściowej), nazywane są algorytmami działającymi **w miejscu**. Algorytmy sortujące, które dla elementów o tej samej wartości zachowują w tablicy końcowej kolejność tablicy wejściowej, nazywamy algorytmami **stabilnymi.**

[1] https://pl.wikipedia.org/wiki/Algorytm

3.2 Formalna definicja problemu sortowania

Dane wejściowe: ciąg n liczb [a_0, a_1, ….., a_n]

Wynik: permutacja (zmiana uporządkowania) [a'_0, a'_1, …, a'_n] ciągu wejściowego taka, że:

$$a'_0 \leq a'_1 \leq \ldots \leq a'_n$$

3.3. Tablice

Tablice są obiektami· Wyróżnia się tablice jednowymiarowe, wielowymiarowe i nieregularne (poszarpane) czyli tablice tablic. Indeksowanie tablic zawsze zaczyna się od zera. Dopuszczalne są tablice bezelementowe Jako tablicę dwuwymiarową możemy sobie wyobrazić planszę prostokątną składającą się z pewnej liczby wierszy i kolumn. Aby przypisać lub pobrać wartość do danej komórki, należy podać jej obie współrzędne.[2]

3.4 Wskaźnik

Wskaźnik to zmienna przechowująca adres innej zmiennej. Inaczej mówiąc, jest to zmienna wskazująca na adres innej zmiennej. Skoro jest zmienną, więc posiada także swój adres. Zmienną wskaźnikową tworzymy podobnie jak zwykłe zmienne pamiętając tylko, aby przed nazwą wpisać operator "*". Operator ten służy także do wyłuskania wartości ze zmiennej wskaźnikowej (wyciągnięcia wartości ze zmiennej, na którą wskazuje).[3]

3.5. Budowa pętli for

- nadawanie początkowych wartości zmiennym
- warunek kończący pętlę for (oznaczony jako
- zwiększenie (zmniejszenie) licznika pętli
- powtarzana instrukcja, bądź blok instrukcji.

4. Kod program

```
#include <iostream>

#include<cstdlib>

#include <conio.h>

#include <cstdio>

#include <ctime>

#include<fstream>

#include<string>

#include <chrono>

#include <windows.h>
```

[2] http://www.algorytm.edu.pl/tablice-w-c/tablice-wielowymiarowe.html
[3] http://www.algorytm.edu.pl/wskazniki.html

```
using namespace std;
int rozmiar_pliku;
int a[9] = { 1,2,3,3,3,3,3,3,2 }; //tablica 9 elementowa
int ile(int rozmiar, int* tabl, int szuk)// rozmiar tablicy, tablica, szukany element
{
   int x, wynik = 0;
   for (x = 0; x <= rozmiar - 1; x++)
   {
      if (tabl[x] == szuk)
      {
         wynik++;
      }
   }
   return wynik;
}

int* sortowanie(int tab[], int rozmiar)//sortowanie
{
   int* wynik;
   wynik = new int[rozmiar];
   int licznik_wynik = 0;
   for (int i = 0; i < rozmiar; i++)
   {
      int tmp = tab[i];
      if (tmp != -1)
      {
         int ile_tmp = ile(rozmiar, tab, tmp);
         for (int j = 0; j < ile_tmp; j++)
         {
            wynik[licznik_wynik] = tmp;
            licznik_wynik++;
         }
```

```
            for (int j = 0; j < rozmiar; j++)
            {
                if (tab[j] == tmp)
                {
                    tab[j] = -1;
                }
            }
        }
    }
    return wynik;
}
int* odczyt_z_pliku(string nazwa_pliku)//odczyt z pliku
{
    fstream plik;
    plik.open(nazwa_pliku);
    int rozmiar = 0;
    int tmp;
    while (plik >> tmp)
    {
        rozmiar++;
    }
    rozmiar_pliku = rozmiar;
    plik.close();
    plik.open(nazwa_pliku);
    int* wynik;
    wynik = new int[rozmiar];
    for (int i = 0; i < rozmiar; i++)
    {
        plik >> wynik[i];
    }
    return sortowanie(wynik, rozmiar);
}
```

```
int losowy_test_pomiar_czasu(int rozmiar, int mod)
{
   int* tab;
   tab = new int[rozmiar];
   for (int i = 0; i < rozmiar; i++)
   {
      tab[i] = rand() % mod;
   }
   clock_t start = clock();
   int* wynik = sortowanie(tab, rozmiar);
   clock_t stop = clock();
   return (stop - start);
}

int* losowy_test(int rozmiar, int mod)
{
   int* tab;
   tab = new int[rozmiar];
   for (int i = 0; i < rozmiar; i++)
   {
      tab[i] = rand() % mod;
   }

   int* wynik = sortowanie(tab, rozmiar);
   return wynik;
}
/*
void generowanie_wykresu()
{
   int liczby[10] = { (int)1e3,(int)1e4,(int)1e5,(int)1e6,(int)(5 * 1e6),(int)1e7,(int)(2 * 1e7),(int)(5 *
1e7),(int)1e8 };
   for (int i = 0; i < 9; i++)
   {
```

```
        cout << liczby[i] << ":" << losowy_test(liczby[i], 20) << "\n";
    }
    return;
}*/
void zapisz_do_pliku(string nazwa_pliku, int* tab)
{
    ofstream plik;
    plik.open(nazwa_pliku);
    int i = 0;
    while (i < rozmiar_pliku)
    {
        plik << tab[i]<<"\n";
        i++;
    }
    plik.close();
    rozmiar_pliku = 0;
    return;
}

int main()
{
    srand(time(NULL));
    while (true)
    {
        int wybor;
        int* wynik_sortowania=NULL;
        int liczba_liczb;
        cout << "Wybierz jedna z opcji: \n"; //opcje do wyboru
        cout << "Jesli chcesz wczytac dane z pliku wcisnij 1.\n";
        cout << "Jesli chcesz podac dane do konsoli wcisnij 2.\n";
        cout << "Jesli chcesz zobaczyc przykladowy zestaw wcisnij 3.\n";
        cout << "Jesli chcesz zakonczyc program wcisnij 4\n";
```

```
string nazwa_pliku;


cin >> wybor;
switch (wybor)//tworzymy opcje do wyboru
{
case 1: {
   cout << "podaj nazwe pliku: \n";


   cin >> nazwa_pliku;
   wynik_sortowania = odczyt_z_pliku(nazwa_pliku);
   liczba_liczb=rozmiar_pliku;
   break;
}
case 2: {
   cout << "podaj ilosc liczb ktore chcesz podac, a nastepnie w nowych liniach podaj kolejne liczby\n";
   int n;
   cin >> n;
   int* tablica;
   tablica = new int[n];
   for (int i = 0; i < n; i++)
   {
      cin >> tablica[i];
   }
   wynik_sortowania = sortowanie(tablica, n);
   liczba_liczb=n;
   break;
}
case 3: {
   cout << "podaj ilosc liczb oraz ich zakres:\n";
   int rozmiar;
   int zakres;
   cin >> rozmiar >> zakres;
```

```
    wynik_sortowania = losowy_test(rozmiar, zakres);
    liczba_liczb=rozmiar;
    break;
}
case 4: {
    return 0;
    break;
}
}
cout << "Wybierz jedna z opcji: \n"; //drugie opcje do wyboru
cout << "Jesli chcesz zapisac dane do pliku wcisnij 1.\n";
cout << "Jesli chcesz wypisac  dane na konsoli wcisnij 2.\n";
cout << "Jesli nic nie chcesz wcisnij 3\n";
cout << "Jesli chcesz zakonczyc program wcisnij 4\n";
cin >> wybor;
switch (wybor) {
case 1: {//tworzymy drugie opcje do wyboru
    cout << "podaj nazwe pliku: \n";
    cin >> nazwa_pliku;
    zapisz_do_pliku(nazwa_pliku, wynik_sortowania);
    break;
}
case 2: {
    int i = 0;
    while (i<liczba_liczb)
    {
        cout << wynik_sortowania[i] << " ";
        i++;
    }
    cout << "\n";
    break;
}
```

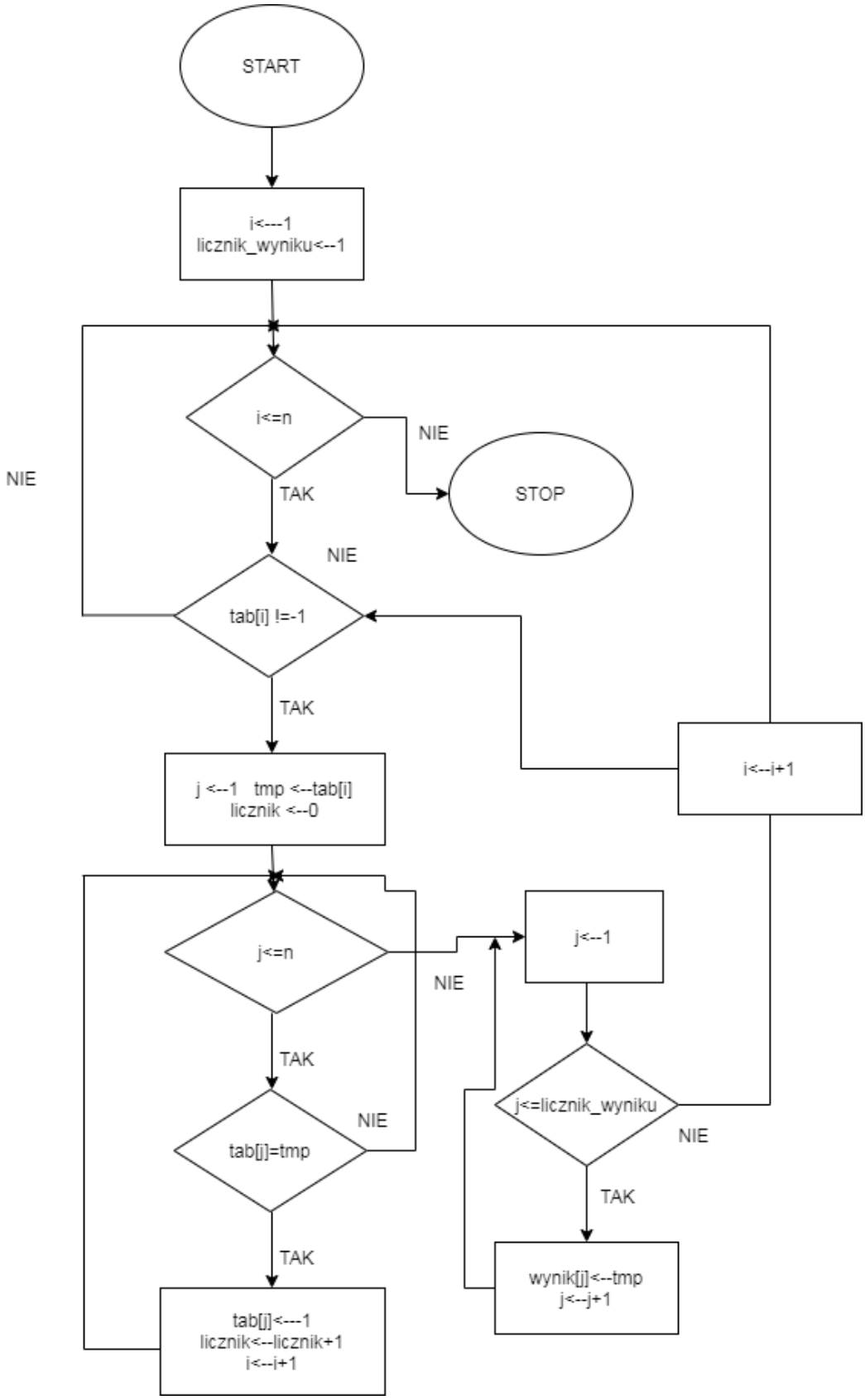
```
        case 3:
        {

            break;
        }
        case 4:
            return 0;
            break;
        }
    }
return 0;
}
```

5. Schemat blokowy

Schemat blokowy przedstawia algorytm w postaci symboli graficznych, podając szczegółowo wszystkie operacje arytmetyczne, logiczne, przesyłania, pomocnicze wraz z kolejnością ich wykonywania. Składa się on z wielu elementów, wśród których podstawowym jest blok.[4]

[4] https://informatyka.2ap.pl/ftp/3d/algorytmy/podr%C4%99cznik_algorytmy.pdf



Grafika 1 Schemat blokowy (opracowanie własne)

6. Pseudokod

Funkcja zliczanie ile(int rozmiar, int* tabl, int szuk)

i←x, 0←wynik

dla x=0, x=rozmiar-1 wykonuj i++

jeśli tabl[x], to wpisz x, wynik+++

zwróć wynik;

funkcja sortowanie (int tab[], int rozmiar)

wynik ← new int[rozmiar];

licznik_wynik ← 0;

dla i=0; i<rozmiar; wykonuj i++

tmp ← tab[i];

Jeśli (tmp != -1), to wpisz ile_tmp ← ile(rozmiar, tab, tmp);

Dla j = 0; j < ile_tmp; wykonaj j++, to wynik[licznik_wynik] ← tmp oraz licznik_wynik++;

Dla j = 0; j < rozmiar; wykonuj j++; jeśli (tab[j] ← tmp), to wypisz tab[j] = -1;

Zwróć wynik;

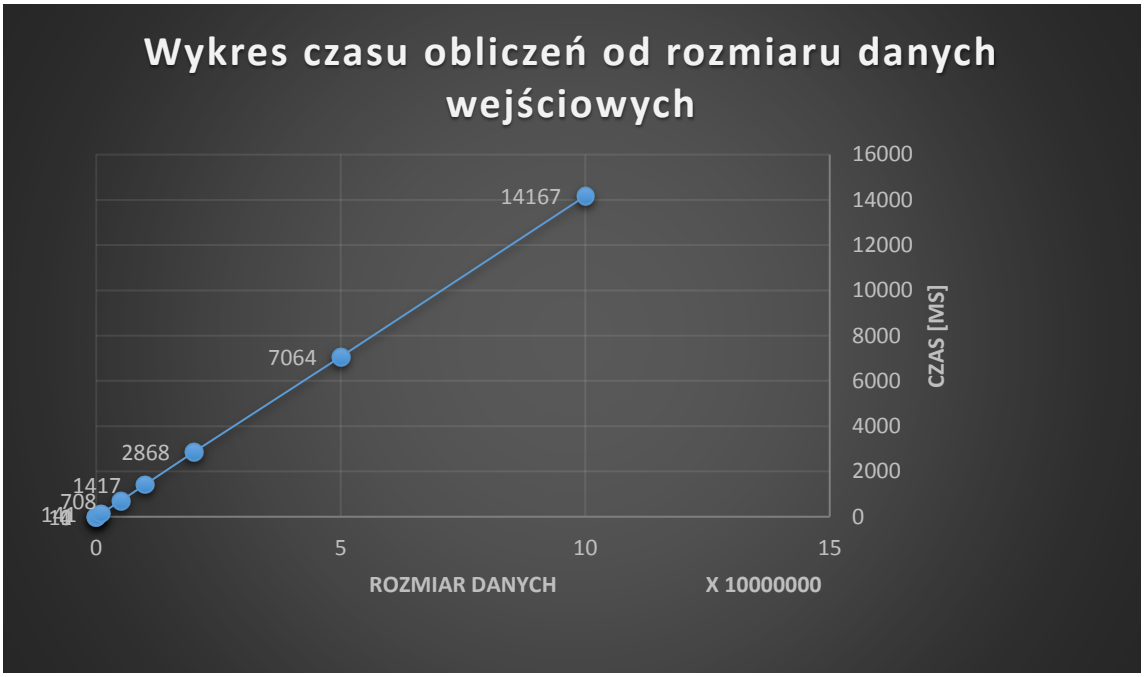
7. Działanie programu

Program posiada konsolę z której możemy wybrać 4 opcje:
Opcja 1: odczyt danych z pliku
Opcja 2: wczytanie danych z klawiatury
Opcja 3: wyświetlenie przykładowego zestawu liczb
Opcja 4: zakończenie działania programu.
Następnie po wybraniu danej opcji możemy wybrać jedną z możliwości:
Opcja 1: zapis danych do pliku
Opcja 2: wypisanie danych na konsoli
Opcja 3: nie chcemy żadnego działania
Opcja 4: zakończenie działania programu

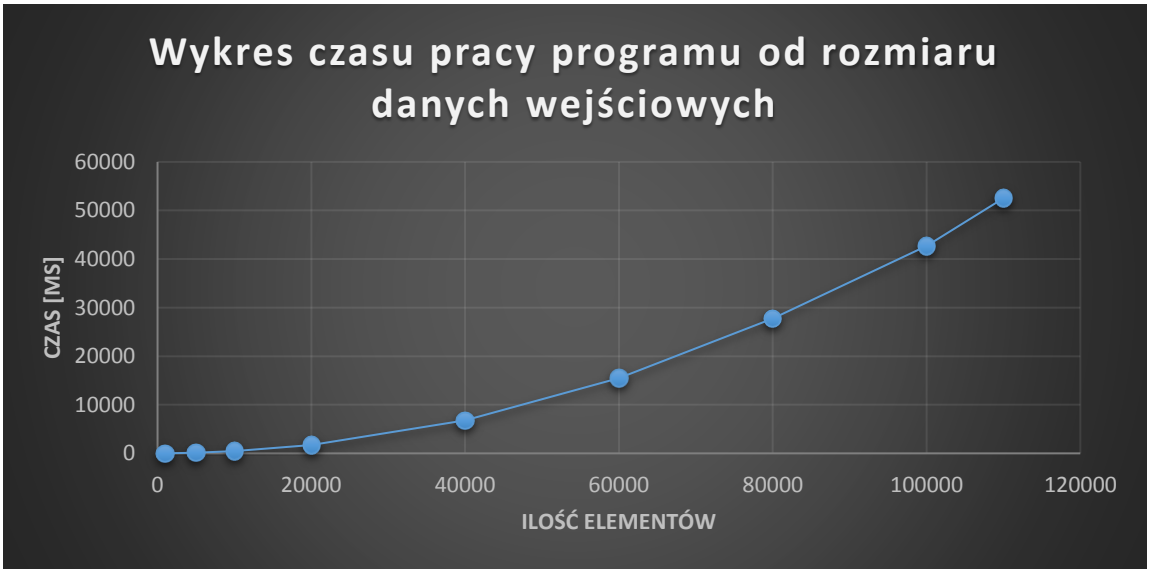
Program liczy liczbę wystąpień elementu x w tej tablicy. Na początku inicjujemy wynik wartością 0. Następnie przejdzie po kolei po wszystkich elementach tablicy i zwiększy wynik o 1 za każdym razem gdy trafimy na element o wartości *x*. Po przejściu po wszystkich elementach otrzymamy liczbę elementów o wartości *x* w tablicy.
Program następnie sortuje liczby, tak, aby zachować daną kolejność, czyli po znalezieniu liczby, która pierwszy raz wystąpiła zapisuje ją w wyjściu jako pierwszą oraz na podstawie ilości wypisanych liczb wypisuje x tych liczb. Następnie kasuje wartość i przechodzi do kolejnej liczby i tak dalej.

8. Wykresy

Na podstawie pracy programu sporządziłam dwa wykresy: Wykres 1 Wykres czasu obliczeń od rozmiaru danych wejściowych - przypadek 1 (opracowanie własne)oraz Wykres 2 Wykres czasu działania programu od rozmiaru danych - przypadek 2 (opracowanie własne)W przypadku Wykres 1 zakres liczb jest stosunkowo mały, natomiast Wykres 2 ma duży zakres liczb, tzn. liczba różnych liczb w tablicy jest porównywalna z rozmiarem tablicy.



Wykres 1 Wykres czasu obliczeń od rozmiaru danych wejściowych - przypadek 1 (opracowanie własne)



Wykres 2 Wykres czasu działania programu od rozmiaru danych - przypadek 2 (opracowanie własne)

9. Podsumowanie

Program posiada konsolę, dzięki której użytkownik może wybrać różne opcje działania programu. Istnieje możliwość odczytu i zapisu do pliku, wpisanie liczb z klawiatury, wyświetlenie przykładowego zestawu liczb oraz zakończenie pracy programu. W programie zostały zawarte m.in. tablice statyczne oraz dynamiczne jedno i dwuwymiarowe, pętle for, while, switch case, wskaźniki.

10. Spis wykresów

11. Spis grafik