Jakub Kryczka

Algorytmy i struktury danych projekt 1

Rzeszów, 2021

1. Wstęp

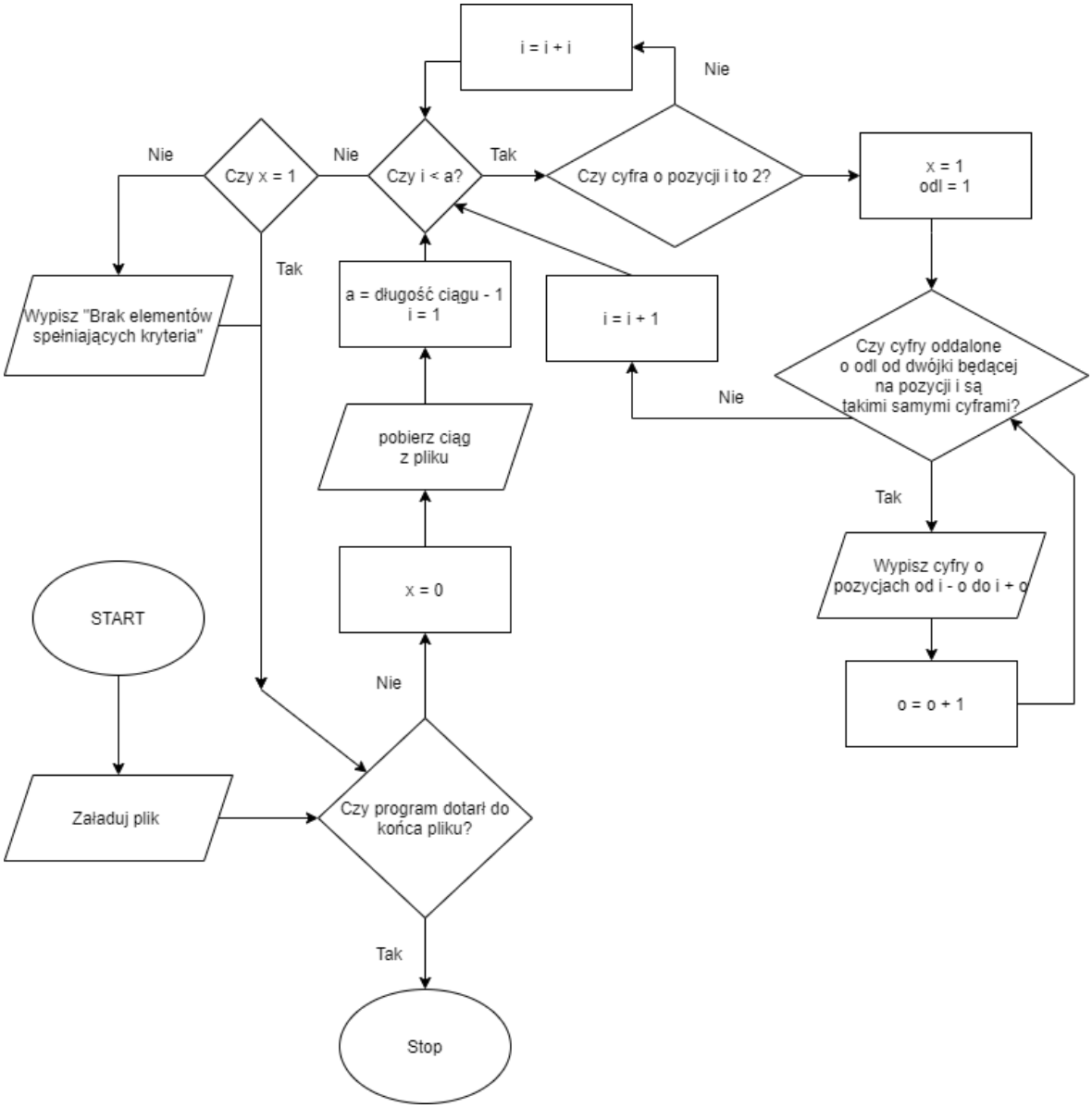
Polecenie jakie otrzymałem wygląda następująco:

Dla zadanego ciągu zer, jedynek i dwójek, znajdź wszystkie podciągi “symetryczne względem dwójek” występujących w ciągu.
Przykład
Wejście: [0, 1, 2, 1, 1, 0, 2, 0, 1, 1, 1, 1, 2, 0]
Wyjście: [1, 2, 1], [0, 2, 0], [1, 0, 2, 0, 1], [1, 1, 0, 2, 0, 1, 1]

Wejście: [0, 1, 2, 0, 0, 0, 2, 1, 0, 0, 2, 2, 1, 0, 1]
Wyjście: brak elementów spełniających zadane kryteria.

Po przeczytaniu polecenia stwierdziłem, że najlepiej będzie napisać program, który będzie w stanie obsługiwać pliki z wieloma ciągami, a nie taki który będzie mógł obsługiwać jedynie pojedyncze ciągi.

2. Schemat blokowy



3. Pseudokod

pętla wykonująca się dopóki plik się nie skończy

    x = 1

    pobierz ciąg z pliku

    a = długość ciągu

        pętla wykonująca się a-2 razy, b = 2

            jeśli cyfra o indeksie równym b to 2

                x = 0

                pętla wykonująca się a razy, odl = 1

                    jeśli cyfry odlegle o wartość odl są równe sobie

                    wypisz do pliku cyfry o indeksach od b-odl do b + odl

                    jeśli nie to wyjść z aktualnej pętli

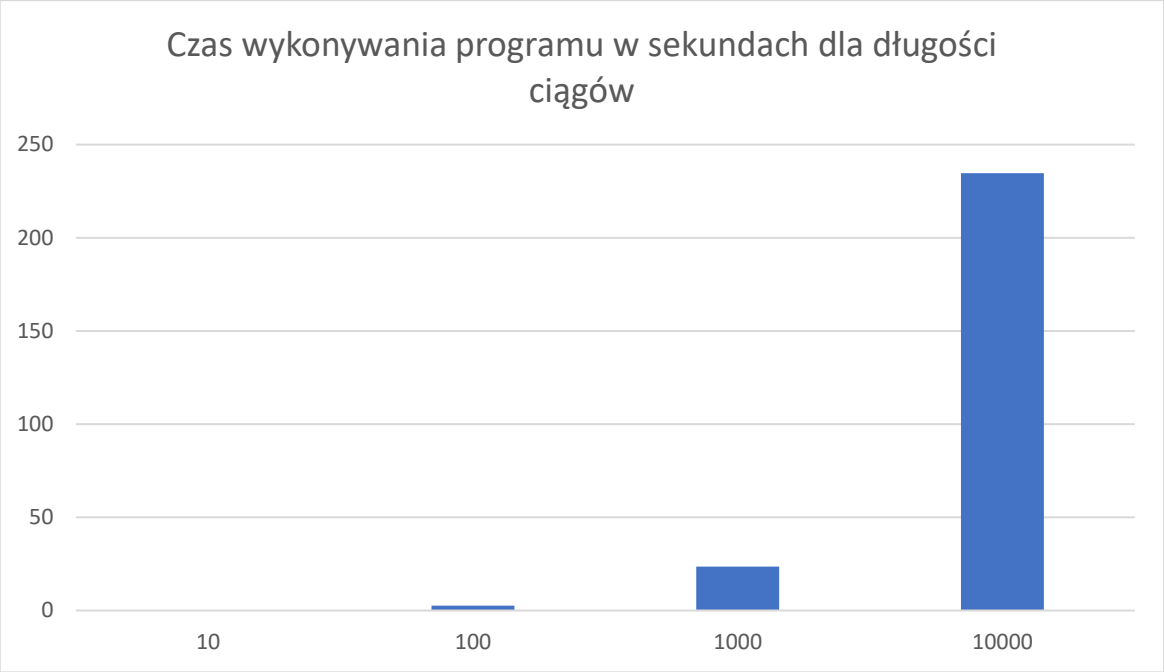
jeśli x = 1

    wypisz “brak elementów spełniających zadane kryteria.”

zakończ program

4. Testy przeprowadzone na programie

Przeprowadziłem test długości wykonywania programu na dla 1000 ciągów o różnych długościach:



5. Kod programu

```
#include <iostream>
#include <string>
#include <fstream>
#include <chrono>

using namespace std::chrono;
using namespace std;

void wypisz(string str,int start,int dlugosc) //funkcja pobiera z jakigo stringa ma wypisac
podciag, od którego elementu zaczac i ile elementów dlugoœci ma miec ten podciag
{
   fstream wyjscie;
   wyjscie.open("wyjscie.txt",ios::app); //funkcja otwieta plik wyjscia, aby dopisac kolejny
podciag

   for(int i=0;i<dlugosc;i++) //petla wykonujaca sie tyle razy, jakiej dlugoœci ma być
podciag
   {
      wyjscie<<str[start+i]; //wypisuje cyfre z ciagu zaczynajac od elementu od którego
mial zaczac i za kazdym razem o indeksie wiekszym o 1
   }
   wyjscie<<" "; //dodaje spacje po wypisaniu podciagu
}


void program()
{
   fstream wejscie; // otwieram plik z którego beda pobierane dane dla programu
   wejscie.open("nieprzetworzone.txt",ios::in);
   fstream wyjscie; // otwieram plik do ktorego beda wypisywane wyniki
   wyjscie.open("wyjscie.txt",ios::app);
   string inp;

   while(!wejscie.eof()) // petla wykonujaca sie dopuki program nie "dotrze" do konca
pliku
   {

      bool x=1; // zmienna pomocnicza
      wejscie>>inp; // ciag zostaje pobrany z pliku

      int a= inp.size()-1; // zostaje okreœlona dlugosc ciagu
      for (int i= 1 ;i<a; i++) // petla wykonujaca sie dla kazdego elementu ciagu
      {
         if (inp[i]=='2') // program sprawdza czy element ciagu jest rowny 2
         {
            x=0; // zmienna zostaje zmieniona poniewaz element spelniajacy zadane kryteria
zostala odnaleziona
            for(int j = 1;j<a;j++)
            {
```

```
                if(inp[i-j]==inp[i+j]) // program sprawdza czy elementy po obu stronach
polozone w równej odleglosci od danej 2 sa rowne
                {
                   wypisz(inp, i - j ,(2 * j)+1); // jezli tak wypisuje ten podciag i ponawia
sprawdzenie, tym razem dla elementów polo¿onych o jedna pozycje dalej od 2
                }else
                {
                   break; // jezeli nie to wychodzi z petli
                }
             }
          }
       }
   if (x) // jesli zmienna spelniajaca zadane kryteria nie zostala znaleziona
   {
      wyjscie<<"brak elementów spełniających zadane kryteria.";
   }
   wyjscie<<endl; // nowa linijka w pliku wyjsciowym
   }
}

int main()
{
   auto start = std::chrono::high_resolution_clock::now(); // rozpoczyna pomiar czasu
   program(); // glowna czesc programu zostaje wykonana
   auto finish = std::chrono::high_resolution_clock::now(); // konczy pomiar czasu
   std::chrono::duration<double> elapsed = finish - start; // oblicza dlugosc pomiaru
   cout<<"Wykonanie operacji zajelo: "<<elapsed.count()<<" sekund."; // wypisuje
dlugosc pomiaru
}
```