Sprawozdanie projektowe z przedmiotu algorytmy i struktury danych

Wojciech Świder gr. 8
nr albumu 179988

Wstęp

Treścią zadania projektowego jest znalezienie dla zadanego ciągu liczb całkowitych (tablicy), najdłuższego podciągu o zadanej sumie.

Zadanie posiada następujący przykład:

Wejście

A[ ] = [0,6,5,1,-5,5,3,5,3,-2,0]
Suma = 8

Wyjście

Podciągi o sumie 8 to: [-5,5,3,5], [3,5], [5,3].
Najdłuższy podciąg to: [-5,5,3,5] o długości 4.

Biorąc pod uwagę powyższy przykład z zadania, zamierzam uwzględnić w programie wypisywanie każdego znalezionego podciągu, oraz wypisywanie długości najdłuższego podciągu.

Spis treści

1. Opis problemu

Celem zadania będzie napisanie algorytmu który ma za zadanie dla zadanego ciągu liczb całkowitych (tablicy), wypisanie listy wszystkich możliwych podciągów o zadanej sumie. Następnie algorytm ma znaleźć spośród nich najdłuższy podciąg, wypisanie go oraz podanie jego długości.

2. Opis podstaw teoretycznych zagadnienia

Wczytywanie danych będzie się odbywało w formie podania przez użytkownika konsoli długości ciągu którego chce zbadać.(Bądź pobieranie tablicy z pliku.) Następnie program zapyta użytkownika o sumę jaką chce, aby miały znalezione podciąg. Po czym program wygeneruje tablice losowych liczb (zakres dla większości testów ustawiony na przedział liczb <-5, 5>).

Podstawą do rozwiązania zadania będzie zawarcie w algorytmie iteracji podanego wcześniej ciągu aby znaleźć możliwy podciąg o zadanej sumie S. Potrzeba też uwzględnić dużą możliwość występowania więcej niż jednego podciągu, dlatego też funkcja iterowania będzie występować w dwóch pętlach aby zbadać każdą możliwość. Długość wewnętrznej pętli J będzie ściśle zależna od pętli zewnętrznej N programu . Z każda następna iteracją [I] (zaczynając od I = 0) pętli zewnętrznej, pętla wewnętrzna będzie zaczynać swoje działanie od P = I czyli następnego wyrazu ciągu. Z tego wynika ,że algrorytm będzie się wykonywał do momentu aż N będzie mniejsze od I. Poniżej znajduje się wizualizacja dwóch pętli i jak one na siebie oddziaływują. Wyraźnie można zobaczyć ,że w wewnętrznej pętli występuje “przesunięcie”.

Iteracja w dwóch pętlach, z obrazowanie

i = 0	*Długość iteracji J = N - i*								→
Długość ciągu N	*3*	*-4*	*0*	*5*	*-5*	*3*	*5*	*0*	*i++*
		-4	*0*	*5*	*-5*	*3*	*5*	*0*	*i++*
			0	*5*	*-5*	*3*	*5*	*0*	*i++*
				5	*-5*	*3*	*5*	*0*	*i++*
					-5	*3*	*5*	*0*	*i++*
						3	*5*	*0*	*i++*
							5	*0*	*i++*
								0	*i++*
y = i	*y++*	*y++*	*y++*	*y++*	*y++*	*y++*	*y++*	*y++*	*i > N*

W istniejącej wewnętrznej iteracji pętli zostanie zawarta funkcja sprawdzająca czy podciąg jest równy zadanej sumie. Do tego w pętli zewnętrznej N będzie potrzebna zmienna suma_pomocniczna która będzie się zerować co iteracje pętli zewnętrznej. Zmienna suma_pomocnicza będzie przechowywać dodawane do siebie wyrazy ciągu w pętli wewnętrznej, przy każdym dodaniu wyrazu będzie sprawdzać czy suma_pomocnicza równa się zadanej sumie. Podciągi znalezione zostały oznaczone kolorem zielonym.

Zadana suma ciągu = 5							
3	*-4*	*0*	*5*	*-5*	*3*	*5*	*0*
	-4	*0*	*5*	*-5*	*3*	*5*	*0*
		0	*5*	*-5*	*3*	*5*	*0*
			5	*-5*	*3*	*5*	*0*
				-5	*3*	*5*	*0*
					3	*5*	*0*
						5	*0*
							0

W momencie spełniania warunku suma_pomocnicza równa sumie wykonuje się polecenie pętli wywołującej podciąg, nazwę ją pętlą O. Podciąg będzie iterowany liczbą i, czyli liczbą która odpowiada za iterowanie pętli zewnętrznej N. Iteracja wywołująca podciąg będzie trwać do osiągnięcia y czyli zmiennej iterującej pętle wewnętrzną J.

Na końcu spełnionego warunku suma_pomocnicza równa sumie wykona się przypisanie zmiennej długość_podciągu definiowana będzie wzorem długość_podciągu = y + 1 - i. Następnie program sprawdzi czy długość_podciągu jest największa dotychczas znalezioną długością podciągu, jeśli tak to długość_podciągu przypisuje swoja wartość do max_długości oraz do zapamiętania tego ciągu zapisujemy koordynaty następującymi zmiennymi: maxStart = i oraz maxStop = y. Po końcu iteracji pętli zewnętrznej wykona się pętla wywołująca najdłuższy znaleziony podciąg oraz jego długość. Wykona się ona przy pomocy iteracji ciągu od maxStart do maxStop, poza pętla system wypisze max_długość.

3. Opis szczegółów implementacji problemu

Funkcja main wykonuje komplet funkcji zależnie od tego z jakiej wersji podania danych do programu chcemy skorzystać. Wersja 1 korzysta z generowania tablicy na podstawie pliku tekstowego. Wersja druga wymaga podania od użytkownika długości tablicy, następnie program wygeneruje liczby do tablicy z zakresu <5,-5>.

Wersja 1.

```
// int *tablica = fileArrayReader();                    ////////GENEROWANIE TABLICY Z PLIKU
// int dlugosc_tablicy = fileLengthSeeker();            ////////GENEROWANIE TABLICY Z PLIKU
// int suma = localSetData2();                          ////////GENEROWANIE TABLICY Z PLIKU
 // algorytm(tablica, suma, dlugosc_tablicy);           ////////GENEROWANIE TABLICY Z PLIKU
 // delete tablica;                                     ////////GENEROWANIE TABLICY Z PLIKU
```

Wersja 2.

```
int suma = localSetData2();                         ////////GENEROWANIE PSEUDOLOSOWEJ TABLICY W PROGRAMIE
int dlugosc_tablicy = localSetData();               ////////GENEROWANIE PSEUDOLOSOWEJ TABLICY W PROGRAMIE
int *tablica = generateArray(dlugosc_tablicy);      ////////GENEROWANIE PSEUDOLOSOWEJ TABLICY W PROGRAMIE
algorytm(tablica, suma, dlugosc_tablicy);          ////////GENEROWANIE PSEUDOLOSOWEJ TABLICY W PROGRAMIE
delete tablica;                                    ////////GENEROWANIE PSEUDOLOSOWEJ TABLICY W PROGRAMIE
```

W obu wersjach wykona się ta sama funkcja - algorytm w której występuje “serce” programu.

```
int algorytm(int tablica[], int suma, int dlugosc_tablicy) {
    int maxDlugosc = 0;
    int maxStart = 0;
    int maxStop = 0;
    bool pierwszyZnaleziony = false;
    auto start:tme_point<system_clock> = std::chrono::high_resolution_clock::now(); /// kod mierzący czas programu

    for (int i = 0; i < dlugosc_tablicy; i++) {
        if (i == 0) {
            cout << "Podciagi o sumie " << suma << " to: ";
            wynikFile << "Podciagi o sumie " << suma << " to: ";
        }
        int sumaPomocnicza = 0;
        for (int j = i; j < dlugosc_tablicy; j++) {
            sumaPomocnicza += tablica[j];
            if (sumaPomocnicza == suma) {
                if (!pierwszyZnaleziony) {
                    pierwszyZnaleziony = true;
                    cout << "[";
                    wynikFile << "[";
                } else {
                    cout << ", [";
                    wynikFile << ", [";
                }
                for (int x = i; x <= j; x++) {
                    if (x == j) {
                        cout << tablica[x];
                        wynikFile << tablica[x];
                    }
                    else {
                        wynikFile << tablica[x] <<",";
                        cout << tablica[x] << ",";
                    }
                }
```

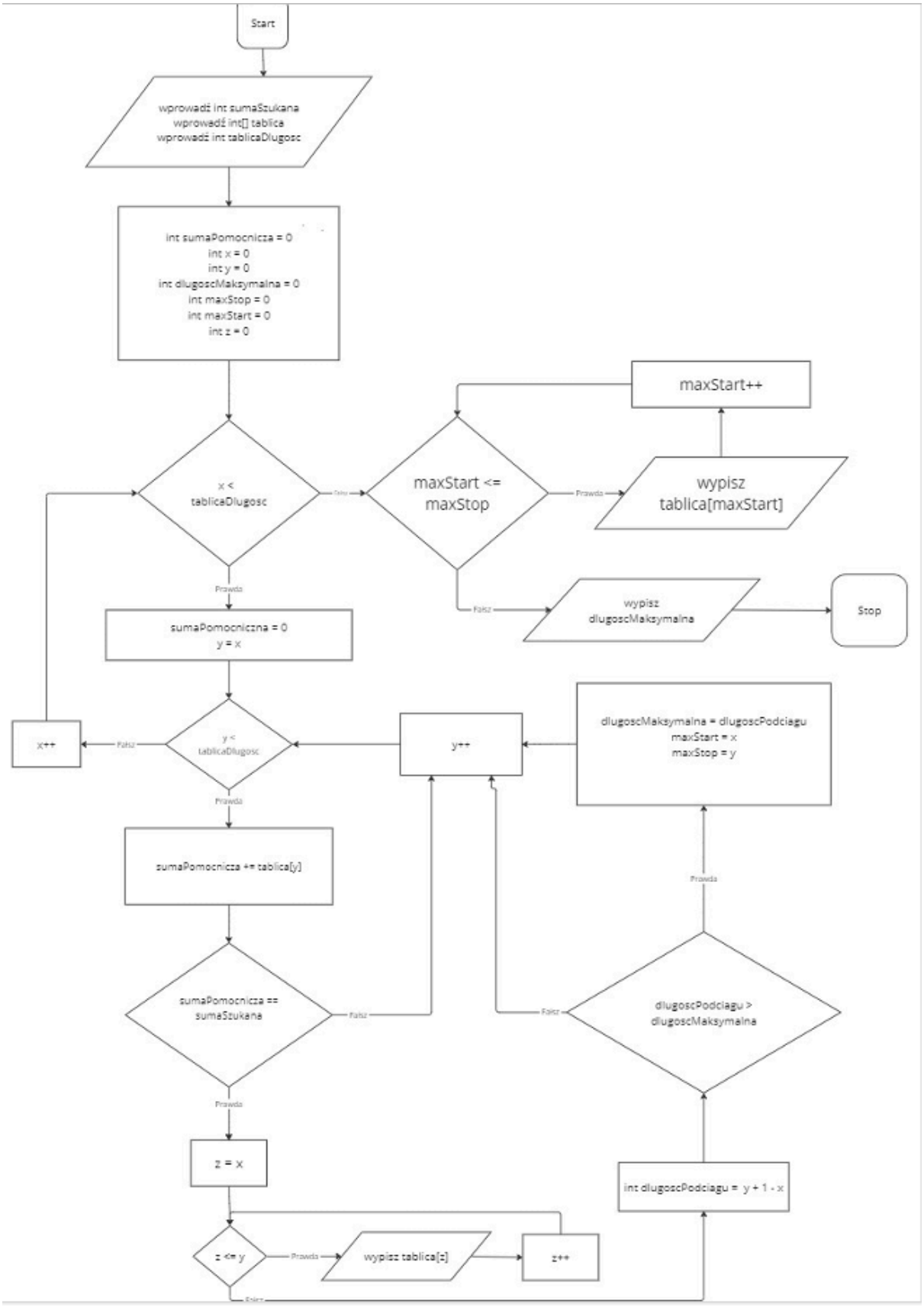
```
            wynikFile <<"]";
            cout << "]";
            int dlugoscPodciagu = j + 1 - i;
            if (dlugoscPodciagu > maxDlugosc) {
                maxStart = i;
                maxStop = j;
                maxDlugosc = dlugoscPodciagu;
            }
        }
    }
}
```

W pierwszej części funkcji algorytm, program wypisuje wszystkie znalezione podciągi odpowiadające zadanej sumie. Algorytm również zachowuje w zmiennych maxStart, maxStop wartości do wyznaczenia w dalszej części najdłuższego znalezionego podciągu.

```
if (maxDlugosc != 0) {
    cout << "." << endl << "Najdluzszy podciag to: " << "[";
    wynikFile << "." << endl << "Najdluzszy podciag to: " << "[";
    for (int x = maxStart; x <= maxStop; x++) {
        if (x == maxStop) {
            cout << tablica[x];
            wynikFile << tablica[x];
        }
        else {
            cout << tablica[x] << ",";
            wynikFile << tablica[x] << ",";
        }
    }
    cout << "]";
    wynikFile <<"]";
    cout << " o dlugosci " << maxDlugosc << ".";
    wynikFile<<" o dlugosci "<<maxDlugosc<< ".";
} else {
    cout << "brak podciagow.";
    wynikFile<<"brak podciagow.";
}
cout << endl;
auto end :time_point<system_clock> = std::chrono::high_resolution_clock::now(); /// kod mierzący czas programu
chrono::duration<double, milli> elapsed = end - start; /// kod mierzący czas programu
cout << elapsed.count() << "ms" << endl; // czas programu w milisekundach
return maxDlugosc;
```

Dodatkowo program zapisuje znalezione podciągi w pliku tekstowym wynikZadania.txt niezależnie od wybranej wersji programu w funkcji main.

4. Schemat blokowy algorytmu



5. Algorytm zapisany w pseudokodzie

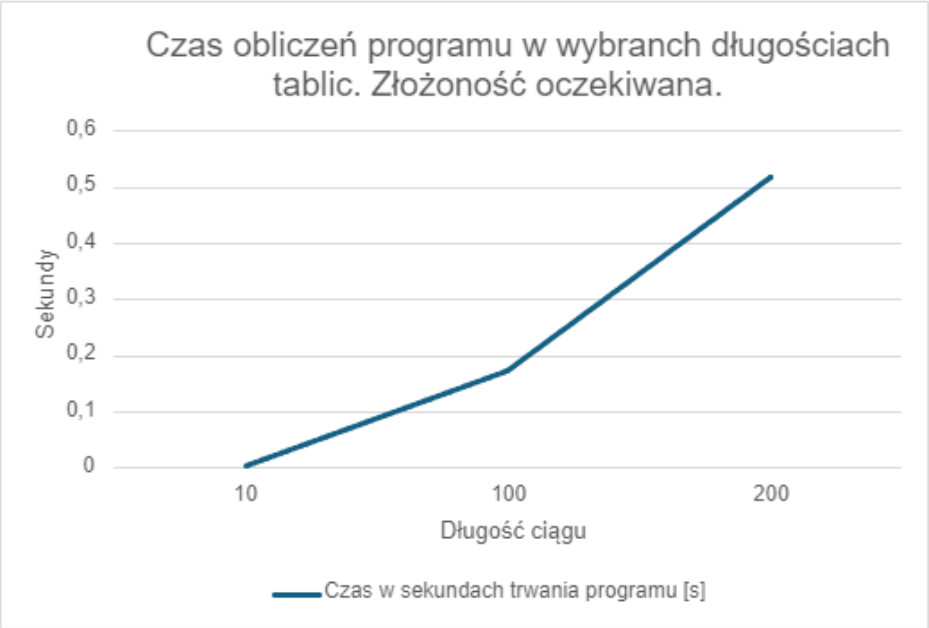
```
wczytaj(sumaSzukana)
wczytaj(tablica[])
wczytaj(tablicaDlugosc)
inicjuj sumaPomocniczna <-- 0
inicjuj dlugoscMaksymalna <-- 0
inicjuj maxStop <-- 0
inicjuj maxStart <-- 0
inicjuj z <-- 0


dla i <-- 0 do dlugoscTablicy - 1 powtarzaj
        sumaPomocniczna <-- 0
        dla j <-- i do dlugoscTablicy - 1 powtarzaj
                sumaPomocnicza <-- sumaPomocnicza + tablica[j]
                jeżeli sumaPomocnicza = tablica[j] to
                        dla z <-- i do j powtarzaj
                                wypisz(tablica[z])
                        inicjuj dlugoscPodciagu <-- j + 1 - i
                        jeżeli dlugoscPodciagu  >  dlugoscMaksymalna to
                                dlugoscMaksymalna <-- dlugoscPodciagu
                                maxStart <-- i
                                maxStop <-- j
dla k <-- maxStart do maxStop powtarzaj
        wypisz(tablica[k])
wypisz(dlugoscMaksymalna)
```

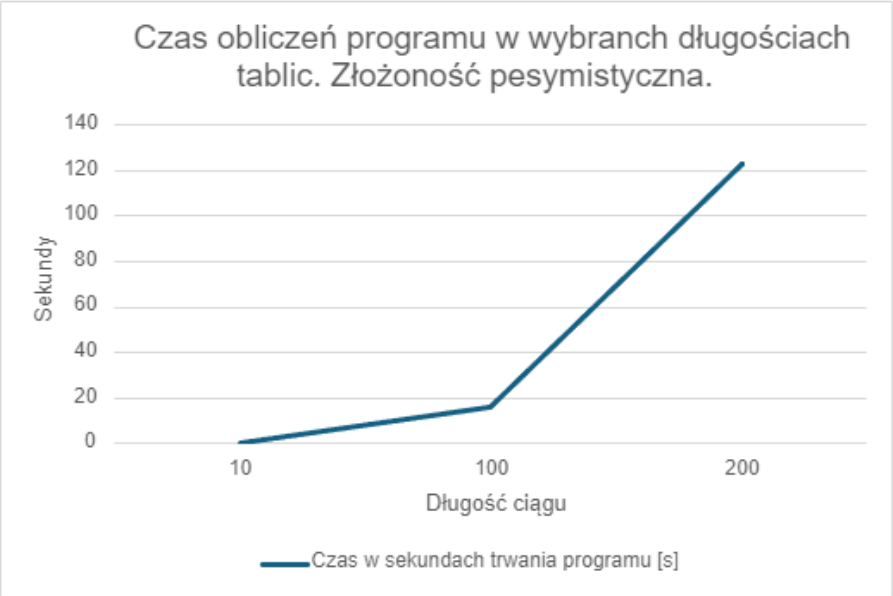
6. Rezultaty testów wykonanych na programie

Badając złożoność obliczeniową zauważyłem ,że może ona się zmieniać w zależności od danych jakie program otrzyma. W złożoności oczekiwanej, czyli “typowych danych” program można opisać O(n^2). Natomiast w momencie nietypowych danych przyjmijmy tablica = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] oraz suma = 0, złożoność obliczeniowa sięga O(n^3) jest to przypadek pesymistyczny. Niżej przedstawiam wykresy dwóch przypadków.

O(n^2) :



O(n^3)



7. Wnioski i podsumowanie

Złożoność obliczeniowa programu wygląda satysfakcjonującą oprócz sytuacji kiedy każdy element ciągu tworzy podciąg, w takim momencie złożoność sięga O(n^3). Program nie jest tak wydajny jak w oczekiwanych złożonościach. Jednak myślę, że zamysł do szukania podciągów o zadanej sumie w ciągu liczb nie powinien analizować takich ciągów w których przykładowo ciąg składa się z samych wyrazów równych 0 a suma ma wynosić 0, ponieważ nie ma to praktycznej wartości.