

P O L I T E C H N I K A R Z E S Z O W S K A

im. Ignacego Łukasiewicza

WYDZIAŁ MATEMATYKI I FIZYKI STOSOWANEJ

Tomasz Nowak

Grupa P05

**Projekt Algorytmy i struktury danych nr. 1**

Rzeszów 2020

1.Wstęp

Kod jest dostępny na publicznym repozytorium pod linkiem:

https://github.com/Tnovyloo/Projekt-Studia

Celem projektu było napisanie algorytmu obsługującego ciąg liczb całkowitych w postaci tablicy.

Algorytm ten rozwiązałem w środowisku Python 3.9, ponieważ w nim się czuję najlepiej gdy przychodzi mi rozwiązać problemy algorytmiczne. Istnieją do tego może i lepsze oraz szybsze języki programowania lecz ja postawiłem na swoje doświadczenie w środowisku Python’a.

2. Opis problemu

Dla zadanego ciągu liczb całkowitych (w postaci tablicy) znajdź liczbę wszystkich podciągów malejących (Podciąg musi składać się z przynajmniej dwóch wartości).

Przykład.

Wejście: A[] = [5, 4, 2, 2, 1]

Wyjście: Liczba wszystkich podciągów malejących to 4.

[5, 4], [5, 4, 2], [4, 2], [2, 1]

Wejście: A[] = [1, 2, 4, 6, 7]

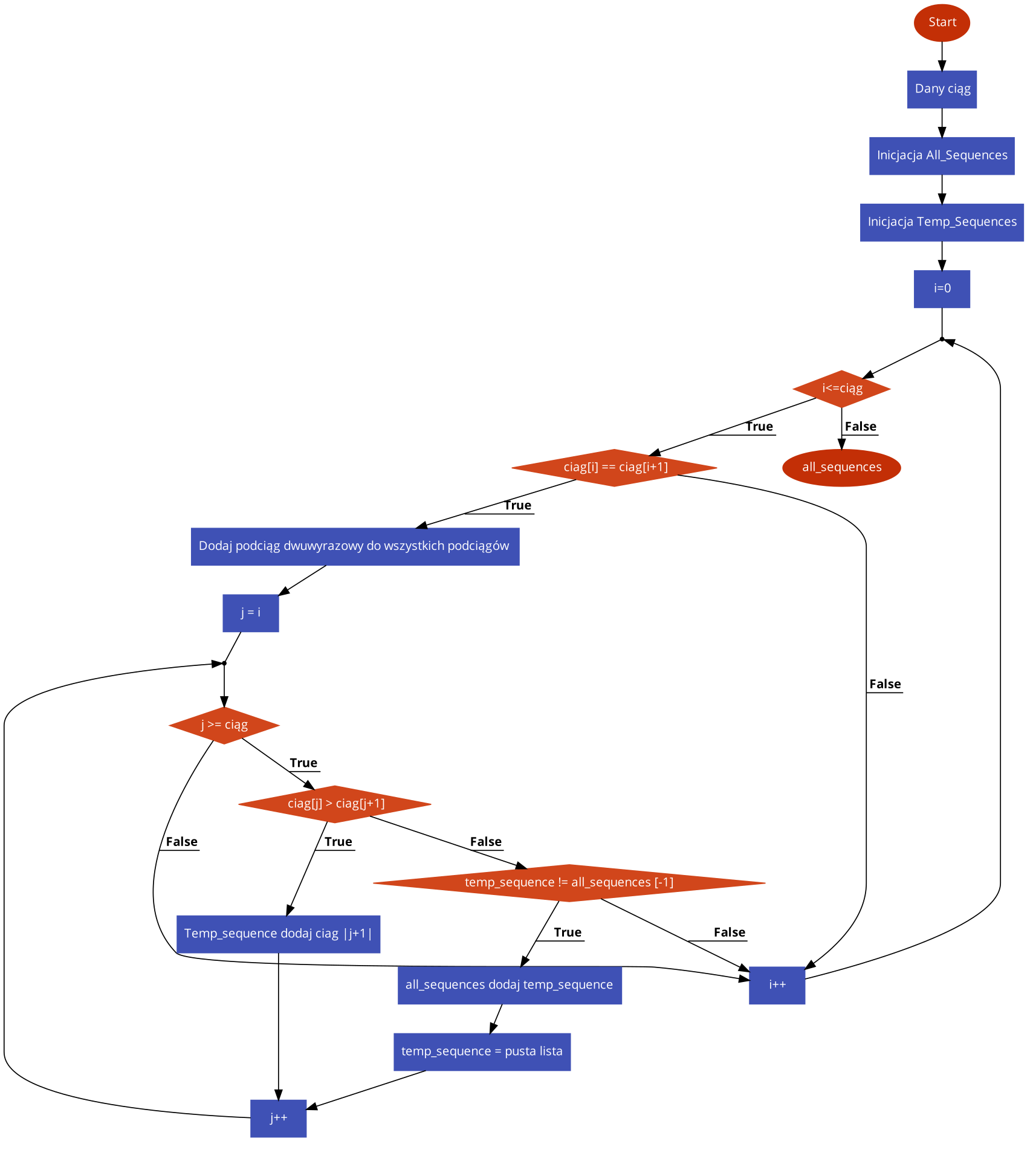
Wyjście: Liczba wszystkich podciągów malejących to 0.

3. Obserwacje

Gdy wchodzimy w pierwszą pętle ‘for’ w naszej tablicy musimy sprawdzić czy A[P] == A[P+1], jeśli tak to postępujemy wedle założenia i dodajemy do wszystkich podciągów ten podciąg składający się z dwóch porównywanych wyrazów.

Następnie sprawdzamy kolejne wyrazy i jeśli są one prawidłowe względem założeń to dodajemy je do chwilowego pomocniczego podciągu.

4. Schemat blokowy



5. Pseudokod

Lista = [] -> Wprowadzamy

All\_sequences = []

Temp\_sequence = []

Length -> długość listy

Dla i = 0 do Length

Jeśli Lista[i] == Lista [i+1]

All\_sequences[] Dodaj podciąg Lista[i], Lista[i+1]

Temp\_sequence = Lista[i], Lista[i+1]

Dla j = i do Length

Jeśli Lista[j] > Lista[j + 1]

Temp\_sequence + Lista[j + 1]

Jeśli Nie

Jeśli Temp\_sequence != All\_sequences[-1]

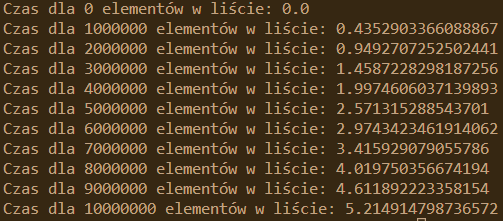
All\_sequences[] += Temp\_sequence[]

Temp\_sequence = 0

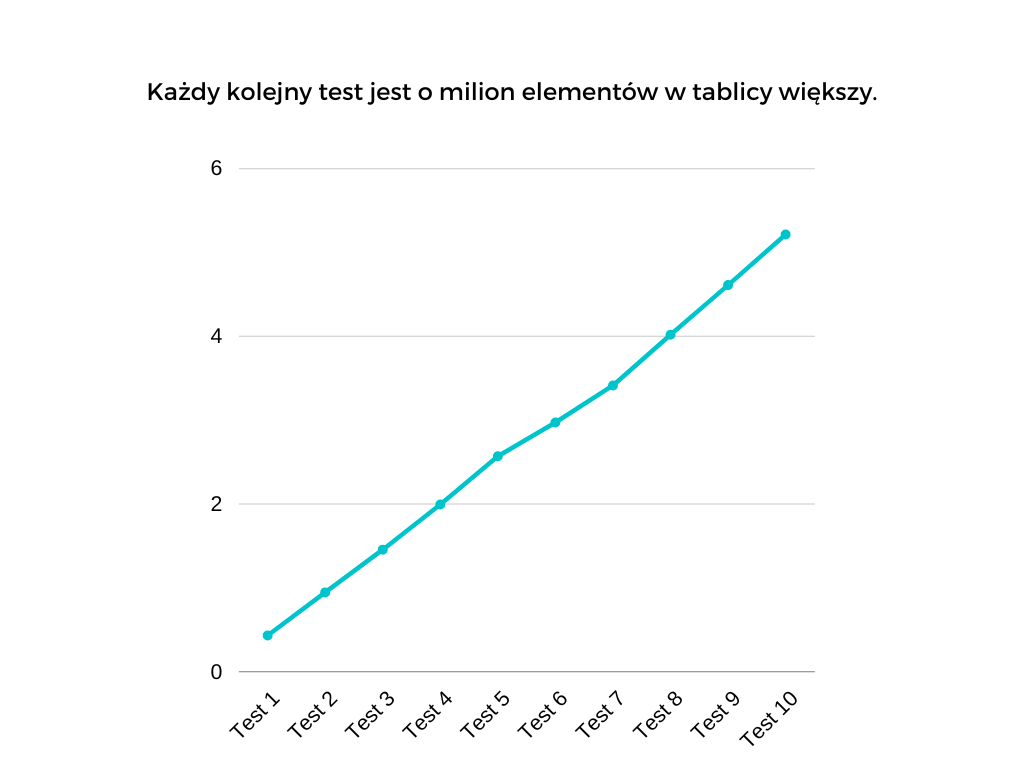
Zakończ

6. Złożoność obliczeniowa

W zawartym kodzie w repozytorium wykonałem funkcję testującą oraz mierzącą czas. Przyjąłem ,że każdy kolejny test będzie się odbywał z milionem więcej elementów w liście a oto rezultat pomiarów:



A teraz przenieśmy nasze wyniki na wykres liniowy aby zilustrować sobie czas działania naszego algorytmu dla każdego osobnego testu.



Co możemy zauważyć to to, że wykres jest liniowy i z każdym kolejnym milionem elementów czas wykonywania algorytmu wydłuża się o pół sekundy. W związku z czym złożoność obliczeniową możemy opisać jako .

7. Podsumowanie

Mam wielką nadzieję, że rozwiązanie oraz dogłębne zbadanie powyższego problemu przypadnie do gustu, każdemu czytającemu moje sprawozdanie. Jeszcze nigdy nie podchodziłem do wykonanych przez siebie algorytmów w tak wnikliwy sposób czyt.: tworzenie kodu blokowego oraz pseudokodu. Największa dla mnie osobiście trudnością była “współpraca” ze środowiskiem Office. A w szczególności na największy aplauz zasługuje Word a zaraz po nim Excel którego finalnie “odłożyłem na bok”.