Najdłuższe wspólne podciągi

Projekt z przedmiotu analizy algorytmów

Kacper Zając, 293178

Opis problemu

Algorytm ma za zadanie znajdowanie najdłuższego wspólnego podciągu lub podciągów dwóch losowych ciągów znaków składających się z jedynie z małych liter alfabetu łacińskiego. Podciągi mogą różnic się konkretną ilością znaków definiowaną na wejściu.

Przykład:

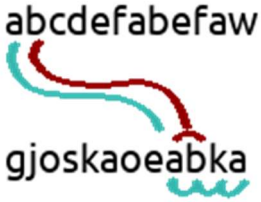
abguis shabpu – ilość różnych znaków – 0, wynik: 2 *ab* *ab*
abguis shabpu – ilość różnych znaków – 1, wynik: 4 *abgu* *abpu*

Metoda rozwiązania

Rysunek 1

Przygotowany przeze mnie projekt czyta dwa dowolne ciągi znaków i zapisuje je w dwóch różnych tablicach. Druga z tablic jest mapowana w celu ułatwienia poruszania się po tablicy (Rysunek 1). Dzięki temu zabiegowi wiemy dokładnie, na której pozycji znajduje się potrzebna nam litera. Aby znaleźć najdłuższy wspólny podciąg musimy sprawdzić niemal każdą możliwość. Należy więc skrupulatnie, idąc po kolei po znaku w ciągu numer dwa liczyć i zapisywać najdłuższy uzyskany ciąg.

Są jednak pewne wyjątki, których ominięcie przyspieszy działanie programu:



Rysunek 2

1. Możemy zrezygnować ze sprawdzania podciągu, kiedy zaczynam sprawdzać podciąg i poprzedni znak ciągu1 zgadza się z poprzednim znakiem ciągu2. Tamten ciąg z pewnością będzie dłuższy. (Rysunek 2) Wynika z tego, że każdy podciąg sprawdzany jest jednokrotnie.
2. Jeśli iterator ciągu jest większy od jego długości, pomniejszonej o długość maksymalnego znalezionego podciągu. Musimy jednak pamiętać o literach, które mogą różnić się w ciągach i ich liczbę należy dodać do długości ciągu.

Uwaga!

W celu usprawnienia działania algorytmu skupiam się jedynie na punktach wspólnych dwóch ciągów. Może się jednak zdarzyć przypadek, w którym algorytm dojdzie do końca ciągu i nie wykorzysta przyznanych mu możliwych różnic w ciągach. W tym wypadku algorytm zawróci i, jeśli to możliwe, doda na początek podciągów dodatkowe znaki.

Złożoność

Optymistyczna jest liniowa. Pesymistycznych przypadków nie udało mi się znaleźć. Najgorszym przypadkiem dla mojego algorytmu będzie najbardziej przeciętny. Jego złożoność wyniesie około:

$$n + n * \frac{n}{26 * 2} * (2 + m)$$

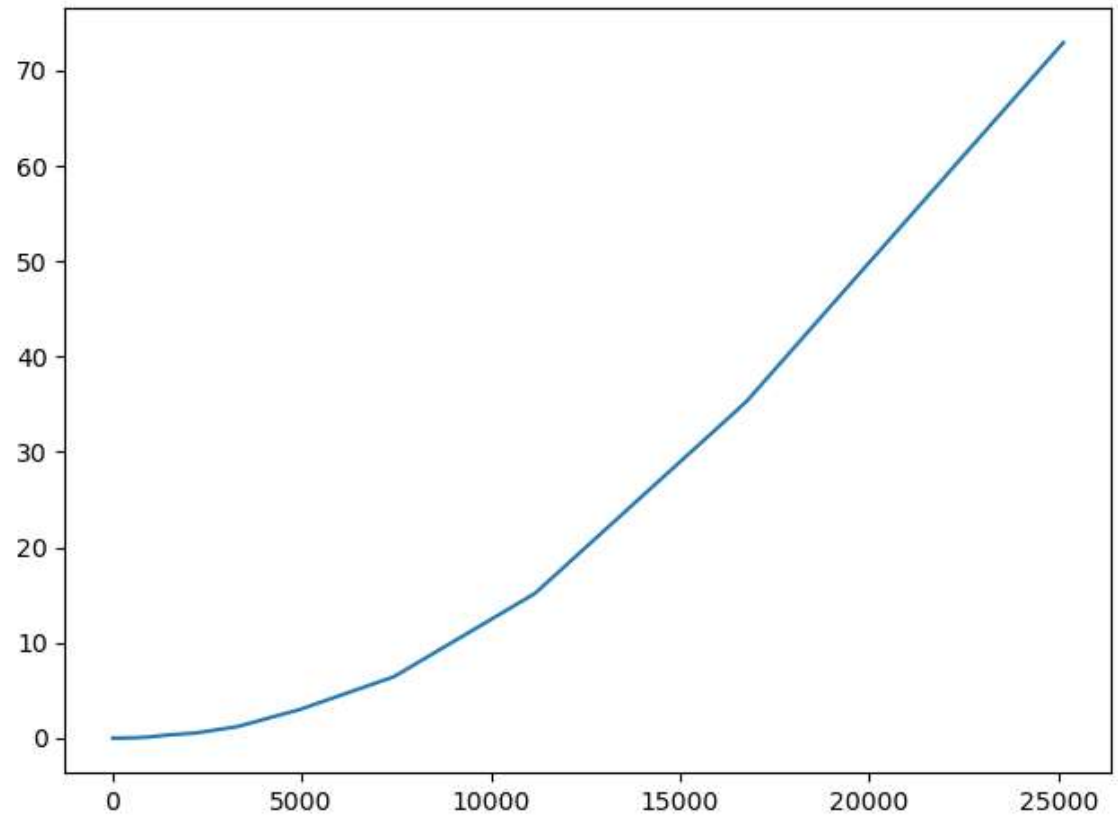
Gdzie: n – długość ciągu, m – liczba różnych znaków, 26 * 2 – liczba znaków w alfabecie pomnożona przez średnią długość bloku znakowego, 2+m – średnia długość bloku znakowego + m

Czasy dla złożoności:

$$complexity = n + \frac{(n)^2}{26 * 2} * (2 + m)$$

n	m	czas	q(n)
4600	0	0:00:02.577128	0.712
4600	3	0:00:06.793139	1.128
4600	15	0:00:20.072970	0.982
5520	0	0:00:03.754097	0.721
5520	3	0:00:08.937408	1.031
5520	15	0:00:29.466242	1.001
6624	0	0:00:05.342991	0.713
6624	3	0:00:12.478820	1
6624	15	0:00:43.061595	1.016
7948	0	0:00:08.342736	0.773
7948	3	0:00:19.385305	1.079
7948	15	0:01:04.754083	1.061
9538	0	0:00:12.002191	0.773
9538	3	0:00:27.793925	1.075
9538	15	0:01:23.303789	0.948

Wykres zależności czasu (oś y) od długości ciągu (oś x)



Wykres 1