Algorytmy tekstowe. Laboratorium 2.

Łukasz Kita

April 2020

1 Wstęp

Celem ćwiczenia było zaimplementowanie struktur przechowujących sufiksy danego słowa, co umożliwia m.in. szybkie sprawdzanie, czy dany ciąg znaków jest podsłowem tego słowa. Implementacja odbyła się w trzech wariantach:

1. Struktura trie sufiksów
2. Struktura drzewa sufiksów z tzw. wolną inicjalizacją, to jest niewykorzystującą linków
3. Struktura drzewa sufiksów z szybką inicjalizacją algorytmem McCreigtha.

2 Elementy programu i jego działanie

W programie dostępne są trzy struktury: Trie, SuffixTree oraz SuffixTrieNaive, każda zaimplementowana w osobnym module o nazwach: trie, suffix_tree oraz suffix_tree_naive. Inicjalizacja struktur następuje po uruchomieniu konstruktora z tekstem podanym jako parametr. Podczas konstruowania struktur następuje sprawdzenie, czy dany tekst, na podstawie którego inicjalizowana jest struktura, kończy się znakiem, który nie występował wcześniej (konieczne, aby móc odróżnić liście od węzłów wewnętrznych drzewa). Jeżeli na końcu tekstu nie występuje unikalny znak, to program podejmuje próbę doklejenia na koniec ciągu znaków takiego unikalnego znaku. Każda ze struktur umożliwia sprawdzenie, czy w danym tekście znajduje się wzorzec, za pomocą metody find(wzorzec), która zwraca prawdę, jeżeli dany wzorzec jest w tym tekście oraz fałsz w przeciwnym wypadku.

3 Przebieg ćwiczenia

3.1 Implementacja struktur i przeprowadzenie testów poprawności

Po zaimplementowaniu wyżej wymienionych struktur przystąpiono do sprawdzenia poprawności działania programu. W tym celu przeprowadzono testy, w których jako wejściowe teksty przyjęto:

1. Szereg krótkich tekstów podanych w instrukcji ćwiczenia:
2. Tekst naturalnego pochodzenia - tekst ustawy, podany w instrukcji ćwiczenia
3. Tekst sztuczny, składający się z n=200 000 znaków a i b, rozmieszczonych w losowy sposób, to jest tekst, który może być opisany wyrażeniem regularnym: $[a|b]^n$

Dla każdego z tych tekstów wygenerowano pewne ich podciągi i sprawdzono, czy rzeczywiście znajdują się one w zaimplementowanych strukturach. W przypadku tekstów: naturalnego i sztucznego, z uwagi na fakt, że składają się one z bardzo dużej liczby znaków, zdecydowano się sprawdzić tylko niektóre podciągi. Ponadto, z uwagi na ograniczenia pamięci, musiano zrezygnować ze sprawdzenia poprawności struktury Trie dla całego tekstu naturalnego i sztucznego - wybrano jedynie 2000-znakowych ich fragment. Było to podyktowane faktem, że struktura Trie jest niezwykle pamięciochłonna.

3.2 Sprawdzenie czasów inicjalizacji

Dla każdej ze struktur zmierzono czasy inicjalizacji różnymi tekstami:

1. Tekst naturalnego pochodzenia - tekst ustawy, podany w instrukcji ćwiczenia
2. Tekst sztuczny, składający się z n=300 000 znaków a i b, rozmieszczonych w losowy sposób, to jest tekst, który może być opisany wyrażeniem regularnym: $[a|b]^n$

Ponownie, z uwagi na ograniczenia pamięci, pomiar czasu dla struktury Trie wykonano dla inicjalizacji tekstem długości 2000 znaków.

4 Wyniki

Otrzymano następujące wyniki:

```
NATURAL TEXT BENCHMARK:
LENGTH: 100
Trie construction: 0.005160700000000018 sec.
SuffixTree with simple initialization construction: 0.000650700000000004 sec.
SuffixTree with McCreight initialization construction: 0.0002999000000000196 sec.
LENGTH: 1000
Trie construction: 0.9517571 sec.
SuffixTree with simple initialization construction: 0.015003600000000006 sec.
SuffixTree with McCreight initialization construction: 0.00369290000000011 sec.
LENGTH: 2000
Trie construction: 5.0324392 sec.
SuffixTree with simple initialization construction: 0.019510999999999612 sec.
SuffixTree with McCreight initialization construction: 0.008042700000000735 sec.
LENGTH: 3000
Trie construction: 15.0448819 sec.
SuffixTree with simple initialization construction: 0.02763660000000101 sec.
SuffixTree with McCreight initialization construction: 0.012940099999998012 sec.
LENGTH: 10000
SuffixTree with simple initialization construction: 0.07930260000000189 sec.
SuffixTree with McCreight initialization construction: 0.03989570000000242 sec.
LENGTH: 50000
SuffixTree with simple initialization construction: 0.44348659999999995 sec.
SuffixTree with McCreight initialization construction: 0.2183881999999997 sec.
LENGTH: 100000
SuffixTree with simple initialization construction: 1.0520953000000013 sec.
SuffixTree with McCreight initialization construction: 0.4207059000000015 sec.
LENGTH: 150000
SuffixTree with simple initialization construction: 1.9891840000000016 sec.
SuffixTree with McCreight initialization construction: 5.664779899999999 sec.
LENGTH: 200000
SuffixTree with simple initialization construction: 4.306182800000002 sec.
SuffixTree with McCreight initialization construction: 2.401522799999995 sec.
LENGTH: 246472
SuffixTree with simple initialization construction: 5.3774947 sec.
SuffixTree with McCreight initialization construction: 3.093084900000001 sec.
```

Rysunek 1: Wyniki testów dla tekstu naturalnego.

```
ARTIFICIAL TEXT BENCHMARK:
LENGTH: 100
Trie construction: 0.004755899999999258 sec.
SuffixTree with simple initialization construction: 0.0005546999999950231 sec.
SuffixTree with McCreight initialization construction: 0.0003068999999982225 sec
LENGTH: 1000
Trie construction: 2.333219300000003 sec.
SuffixTree with simple initialization construction: 0.012307699999993815 sec.
SuffixTree with McCreight initialization construction: 0.004024399999998707 sec.
LENGTH: 2000
Trie construction: 4.976924199999999 sec.
SuffixTree with simple initialization construction: 0.020289900000001637 sec.
SuffixTree with McCreight initialization construction: 0.0076353999999980715 sec
LENGTH: 3000
Trie construction: 14.567192100000007 sec.
SuffixTree with simple initialization construction: 0.025872200000009116 sec.
SuffixTree with McCreight initialization construction: 0.011680299999994759 sec.
LENGTH: 10000
SuffixTree with simple initialization construction: 0.07982839999999669 sec.
SuffixTree with McCreight initialization construction: 0.040091000000003874 sec.
LENGTH: 50000
SuffixTree with simple initialization construction: 0.45403299999999547 sec.
SuffixTree with McCreight initialization construction: 0.21226900000000626 sec.
LENGTH: 100000
SuffixTree with simple initialization construction: 5.024323800000005 sec.
SuffixTree with McCreight initialization construction: 1.155684100000002 sec.
LENGTH: 150000
SuffixTree with simple initialization construction: 3.0796888000000138 sec.
SuffixTree with McCreight initialization construction: 1.6782706000000047 sec.
LENGTH: 200000
SuffixTree with simple initialization construction: 4.517475599999997 sec.
SuffixTree with McCreight initialization construction: 2.285609699999995 sec.
```

Rysunek 2: Wyniki testów dla tekstu sztucznego.

5 Wnioski

Zaimplementowane struktury umożliwiają szybkie sprawdzanie, czy dany wzorzec należy do tekstu. Struktury drzew sufiksowych są zdecydowanie bardziej wydajne niż struktura Trie. Jest to własność prawdziwa zarówno ze względu na zużycie pamięci, jak i ze względu czas inicjalizacji struktury. Wielkości te dla struktury Trie rosną wraz z kwadratem długości tekstu, natomiast dla drzew sufiksowych zwiększają się w tempie liniowym względem długości tekstu. Algorytm McCreighta inicjalizacji drzew sufiksowych okazał się szybszy od naiwnej ich inicjalizacji zawsze w przypadku tekstu sztucznego, jak i w większości przypadków dla tekstu naturalnego. Dla danych przedstawiony w sprawozdaniu jedynie w jednym przypadku (dla tekstu naturalnego o długości 150 000 znaków) algorytm McCreighta zadziałał wolniej od algorytmu naiwnego, co powtarzało się wielokrotnie przy pomiarach czasu.