Algorytmy Tekstowe

Laboratorium 2 – raport

Mateusz Kocot

1. Wstęp

Porównane zostały szybkości działania algorytmów konstruujących następujące struktury danych:

1. *Trie* (z wykorzystaniem procedury *up_link_down*),
2. *slow suffix tree* – drzewo sufiksów bez wykorzystywania procedury *fast_find* oraz elementów związanych z linkowaniem,
3. *fast suffix tree* – drzewu sufiksów wykorzystujące powyższe elementy, zaimplementowane według algorytmu McCrieghta.

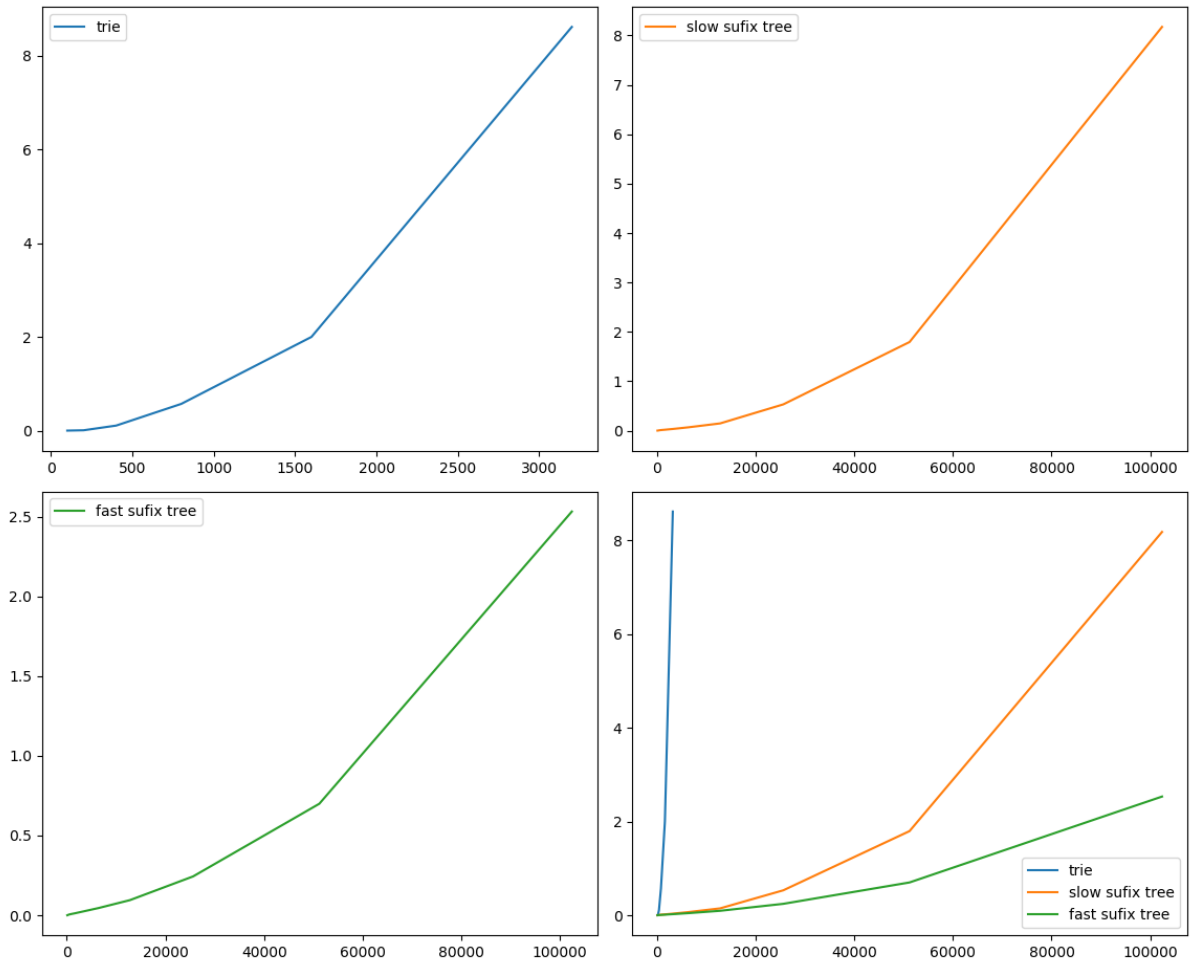
Do porównania wykorzystano następujące dane:

1. tekst ustawy ograniczony do pewnej liczby znaków,
2. tekst składający się z litery *a* powtórzonej określoną liczbę razy,
3. tekst składający się z liter *abcde* powtórzonych określoną liczbę razy.

W związku z wymaganiami, każdy z tekstów zakończony jest znakiem o wartości 0: *0*.

2. Tekst ustawy

Wyniki pomiarów przedstawiono na rys. 1.

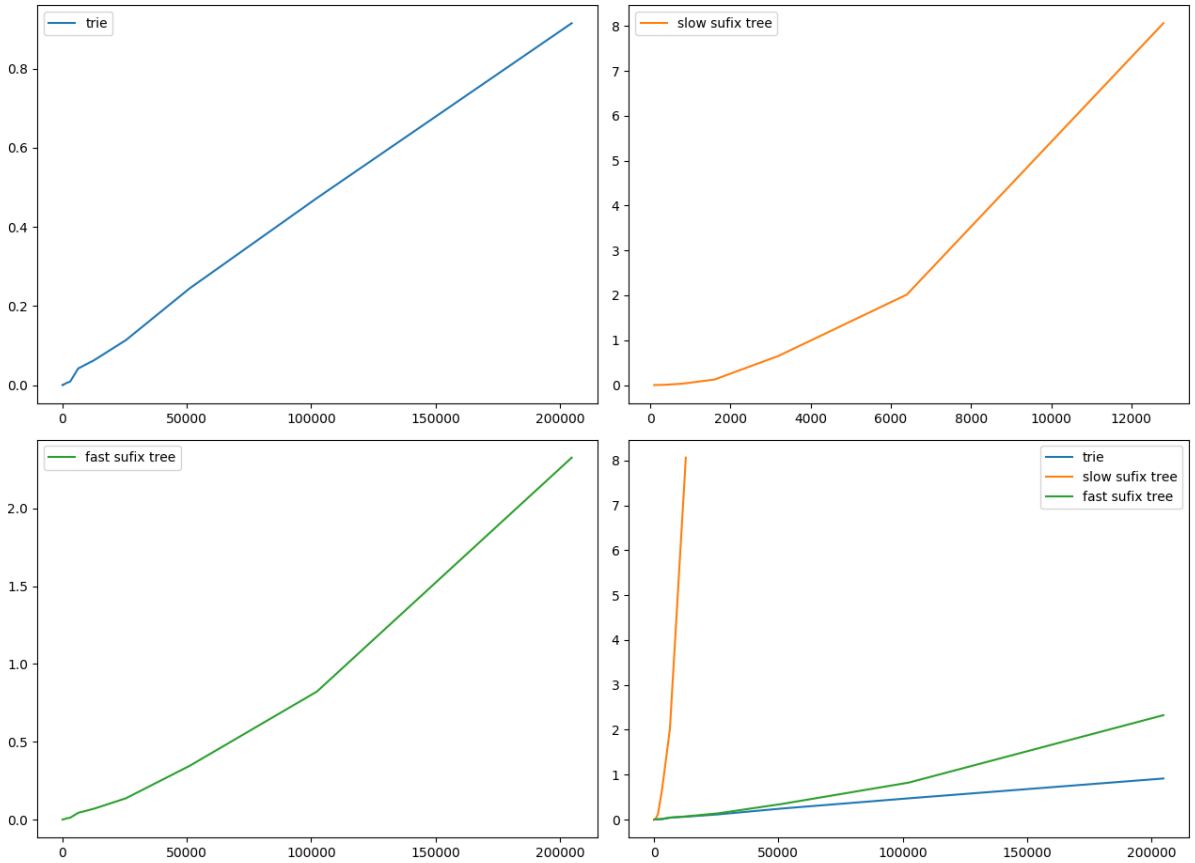


Rys. 1. Pomiary czasu na przykładzie tekstu ustawy. Oś OX to liczba znaków, natomiast oś OY – czas w sekundach.

Wersja *trie* działa zdecydowanie najwolniej. Już dla 3000 znaków czas wykonania przekracza $8\,s$, podczas gdy drzewa sufiksów dla tej liczby znaków są tworzone prawie natychmiastowo. Jest to zgodne z oczekiwaniami. Tekst ustawy napisany jest w języku naturalnym. Większość węzłów można więc zastąpić przedziałami. Dzieje się tak w wersjach drzew sufiksów. Dodatkowo, wersja *fast suffix tree* wykorzystuje m.in. linkowanie do zwiększenia wydajności, co także widać na rys. 1.

3. Tekst „a”

Wyniki pomiarów przedstawiono na rys. 2.

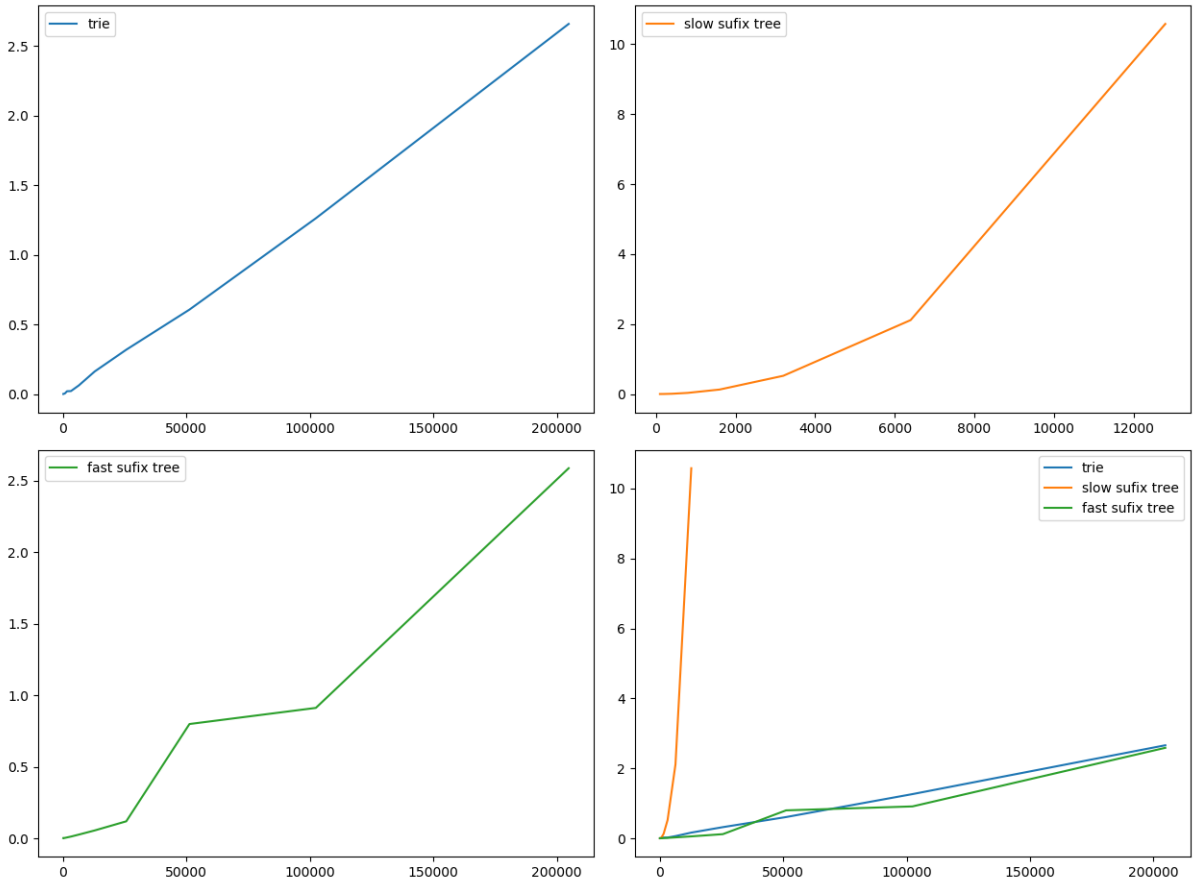


Rys. 2. Pomiary czasu na przykładzie tekstu „a”. Oś OX to liczba znaków, natomiast oś OY – czas w sekundach.

W tym przypadku, drzewo sufiksów wygląda niemal identycznie jak *trie*. Całe drzewo przypomina ścieżkę z jedno-węzłowymi rozwidleniami (znak */0*). W związku z tym, czasy działania wersji *trie* oraz *fast suffix tree* są bardzo podobne. Nieco szybciej działa *trie* prawdopodobnie po prostu dla tego, że jest to procedura prostsza i w tym przypadku wykonuje mniej operacji. Wersja *slow suffix tree* działa znacznie wolniej od dwóch pozostałych. Ponieważ nie wykorzystuje ona linkowania, za każdym razem poszukiwanie rozpoczyna z korzenia, co jak widać znacząco wpływa na czas wykonania.

4. Tekst „abcde”

Wyniki pomiarów przedstawiono na rys. 3.



Rys. 2. Pomiary czasu na przykładzie tekstu „abcde”. Oś OX to liczba znaków, natomiast oś OY – czas w sekundach.

Wykresy wyglądają bardzo podobnie do tych z poprzedniego punktu. Jest mniejsza różnica między *trie* i *fast suffix tree*. Wraz ze zwiększaniem alfabetu, różnica ta będzie coraz mniejsza, a w końcu *fast suffix tree* zacznie być szybsze.

5. Podsumowanie

Jeżeli tylko tekst nie jest samymi powtórzeniami kilku liter, *fast suffix tree* działa najszybciej. Szybszy od *trie* będzie ze względu na wydajniejsze etykietowanie węzłów, a od *slow suffix tree* – ze względu na zastosowanie procedury *fast find* oraz linkowania.