Wyszukiwanie wzorca w tekście

Adam Kraś 325177 Krzysztof Król 325178

Implementacja algorytmów wyszukiwania wzorca

W ramach zadania laboratoryjnego wykonaliśmy implementacje trzech rodzajów algorytmów typu wyszukiwanie wzorca w tekście. Podział pracy wyglądał w następujący sposób:

1. Adam Kraś
    a. Algorytm naiwny
    b. Algorytm KMP
    c. test_pattern.py
2. Krzysztof Król
    a. Algorytm KR
    b. Funkcja main odpowiedzialna za mierzenie czasu i rysowanie wykresów
    c. graph_maker.py

Sprawdzenie poprawności implementacji

W celu sprawdzenia poprawności implementacji naszych algorytmów wykonaliśmy testy dla zaproponowanych przypadków brzegowych:

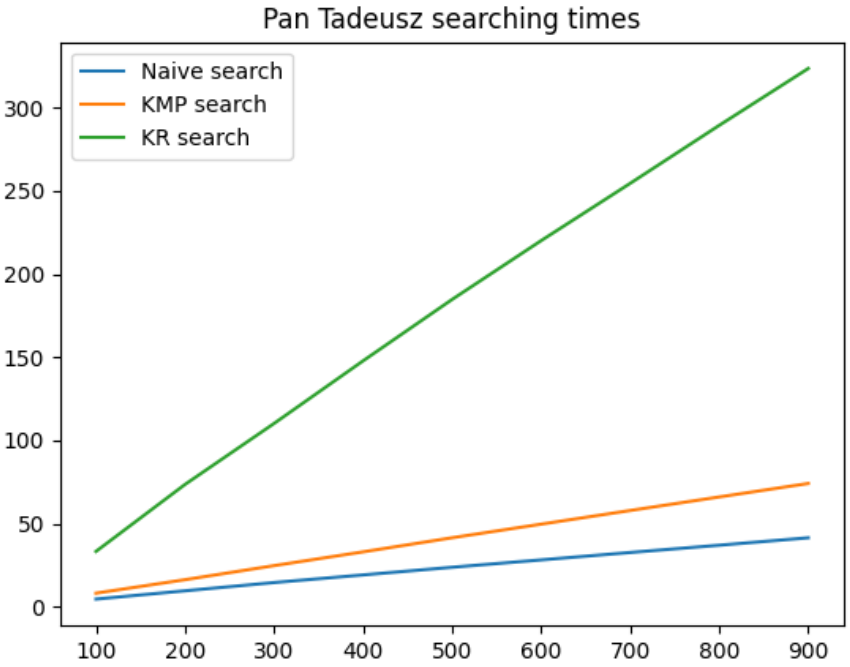
- Pusty jeden lub oba napisy
- Napis *pattern* równy napisowi *text*
- Napis *pattern* dłuższy od napisu *text*
- Napis *pattern* nie występuje w *text*

Wszystkie z naszych algorytmów dla dwóch pierwszych przypadków zwracają jednoelementową tablice, której jedynym elementem jest „0” (czyli początek *text,* czyli wszystko się zgadza. Dla dwóch ostatnich przykładów najbardziej logiczną opcją jest, żeby wyszukiwania nic nie zwracały (null) i tak też robią. Aby sprawdzić działanie algorytmów na tychże przypadkach brzegowych należy odpalić poszczególne moduły z implementacją danego wyszukiwania jako główny plik wykonywalny.

Oprócz tego w funkcji main, przed rozpoczęciem pomiarów czasowych, wykonujemy sprawdzenie dla losowego tekstu (*text)* i losowego wzorca (*pattern)* czy wszystkie algorytmy wskazują na te same miejsca w tekście, czyli czy wszystkie poprawnie znajdują wzorce tam gdzie one rzeczywiście są. Tak też jest. Za losowanie tekstu i wzorca odpowiada moduł test_pattern.py.

Porównanie algorytmów wyszukiwania wzorca

W dalszej części funkcji main mierzymy czasy w jakich dany algorytm wyszukuje wzorzec w tekście. W tym celu użyliśmy danego pliku pan_tadeusz.txt z tekstem jednej z ulubionych lektur każdego maturzysty. Dla każdego z zaimplementowanych sortowań mierzymy czasy wyszukiwania pierwszych 100, 200, 300, … 1000 słów *Pana Tadeusza* w całości utworu, a następnie prezentujemy je na wykresie takim jak ten:



Jak więc widać w przypadku przeszukiwania *Pana Tadeusza* najwydajniejszy jest algorytm naiwny, a najgorzej radzi sobie algorytm Karpa-Rabina. Jest tak dlatego, iż algorytm Karpa-Rabina sprawdza wszystkie hashe fragmentów oryginalnego tekstu o długości wzorca, a algorytm naiwny i Knutha-Morrisa-Pratta w niektórych przypadkach wykonują sprytne skoki o długość wzorca.