Piotr Adrian Brzeziński
Bartłomiej Patryk Rasztabiga
Laboratorium AISDI - grupa 104

Laboratorium AISDI - wyszukiwanie wzorca w tekście

Link do repozytorium: https://gitlab-stud.elka.pw.edu.pl/brasztab/brzezinski-rasztabiga-aisdi-lab/-/tree/master/zad5

Podział zadań

Piotr Brzeziński:

- testy jednostkowe (pkt. 2)
- raport (pkt. 4)

Bartłomiej Rasztabiga:

- implementacja algorytmów (pkt. 1)
- pomiar wydajności (pkt. 3)

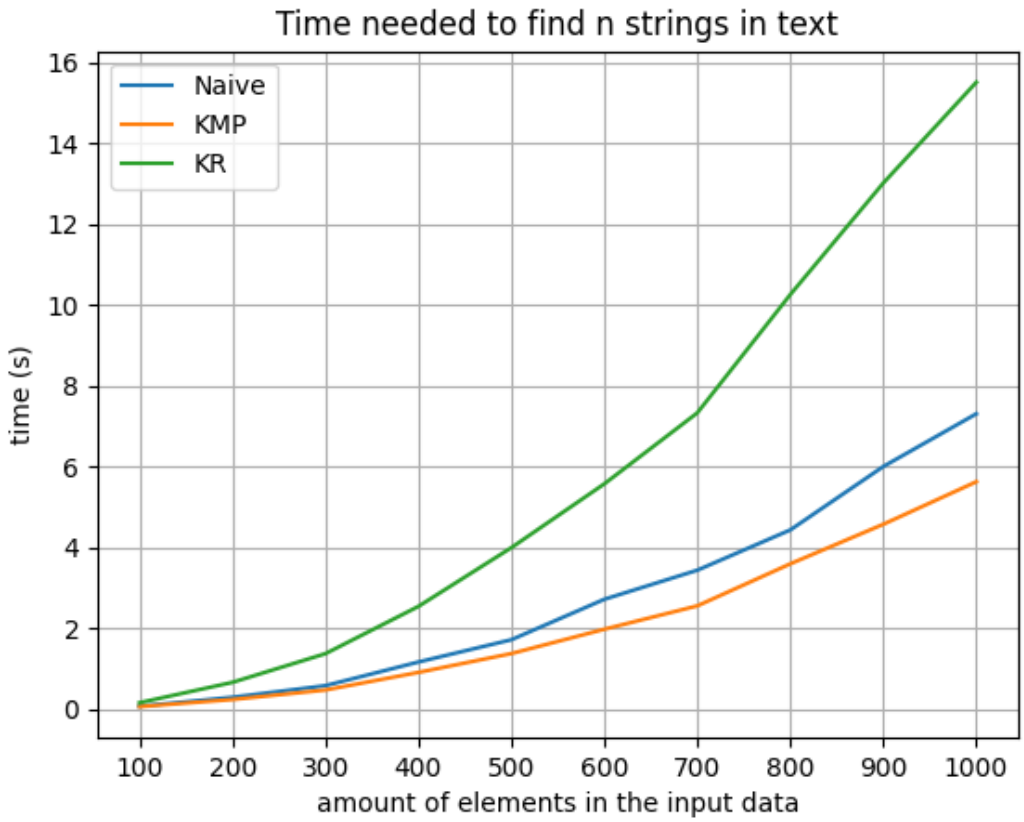
Struktura projektu

Plik `algorithms.py` zawiera zgodne z poleceniem implementacje wszystkich trzech wymaganych algorytmów.

Poprawność ich działania sprawdzana jest poprzez testy w pliku `test_algorithms.py`. Sprawdzane są tam dla każdego algorytmu przypadki brzegowe, a dla naiwnego dodatkowo kilka innych, przykładowych przypadków. W klasie `TestRandomData` znajduje się natomiast test losowy - generowane losowe ciągi - wzorzec oraz 'tekst' - o ustalonej długości, po czym sprawdzone zostaje, czy wszystkie algorytmy dają dla takich danych identyczne wyniki.

Plik `main.py` odpowiedzialny jest za przeprowadzenie testów wydajnościowych, zapisanie ich wyników do pliku tekstowego oraz utworzenie wykresu porównawczego. Pliki te znajdują się w folderze `results`.

Wyniki testów wydajnościowych




```
Wyniki pomiarów:

Szukanie n słów w tekście:
100:    Naive: 0.0607448s       KMP: 0.05131323s        KR: 0.14813064s
200:    Naive: 0.2833652s       KMP: 0.22776948s        KR: 0.65195114s
300:    Naive: 0.5708678s       KMP: 0.46326166s        KR: 1.3623694s
400:    Naive: 1.1565153s       KMP: 0.89880403s        KR: 2.53257943s
500:    Naive: 1.708179s        KMP: 1.3656263s         KR: 3.98764138s
600:    Naive: 2.7102589s       KMP: 1.96874819s        KR: 5.56899688s
700:    Naive: 3.4305389s       KMP: 2.54723089s        KR: 7.3272418s
800:    Naive: 4.4280429s       KMP: 3.58856137s        KR: 10.2571921s
900:    Naive: 5.9983129s       KMP: 4.56808297s        KR: 13.01965555s
1000:   Naive: 7.3039725s       KMP: 5.62126983s        KR: 15.51419585s
```

Wnioski

Zgodnie z teorią, algorytm Knutha-Morrisa-Pratta okazał się w testach szybszy, niż algorytm naiwny. Wynika to z m.in z faktu, że w przeciwieństwie do naiwnego, algorytm KMP potrafi wykorzystać informacje o już przetworzonych znakach, nawet jeśli okaże się, że nie są częścią szukanego wzorca.

Problem pojawia się w przypadku algorytmu Karpa-Rabina - w naszych testach jest on znacząco wolniejszy od dwóch pozostałych, co nie powinno mieć miejsca - w teorii przynajmniej algorytm naiwny powinien być od niego wolniejszy. Taki odchył w testach wynika najprawdopodobniej z faktu, że algorytm KR operuje na liczbach na tyle dużych, że Python wewnętrznie przechodzi na inny typ danych, znacząco spowalniając obliczenia i powodując inne od oczekiwanych wyniki.