Sprawozdanie
AISDI, **wyszukiwanie**

wykonujący:
Jakub Rozkosz
Hubert Gołębiowski

Podział pracy:
Jakub Rozkosz:

- implementacja algorytmu Karpa-Rabina
- implementacja algorytmu naiwnego
- wykonanie testów algorytmu naiwnego oraz Karpa-Rabina
- napisanie pliku *read_file.py*
- pisanie sprawozdania

Hubert Gołębiowski:

- implementacja algorytmu Knutha-Morrisa-Pratta
- testy jednostkowe do algorytmu KMP
- plik time_measure.py
- plik drawing_plots.py
- pisanie sprawozdania

Wspólnie:

- pisanie main.py, wykonanie pomiarów oraz wygenerowanie wykresów

Pliki:
algorytm_naiwny.py - plik z implementacją naiwnego algorytmu wyszukiwania.

algorytm_KR.py - plik z implementacją algorytmu Karpa-Rabina. Zawiera funkcję find do wyszukiwania frazy w tekście.

algorytm_KMP.py - plik z implementacją algorytmu Knutha-Morrisa-Pratta. Znajduję się w nim funkcja KMP_find, która służy do wyszukiwania frazy w tekście. Korzysta ona z pomocniczej funkcji create_DFA, która tworzy Deterministyczny Automat Skończony dla wyszukiwanej frazy. DFA jest tworzony w postaci słownika, w którym kluczami są wszystkie znaki występujące w wyszukiwanej frazie, a wartościami są listy stanów.

test_naiwny.py - plik z testami jednostkowymi algorutmu naiwnego.
test_KR.py - plik z testami jednostkowymi algorytmu KR
test_KMP.py - plik z testami jednostkowymi algorytmu KMP

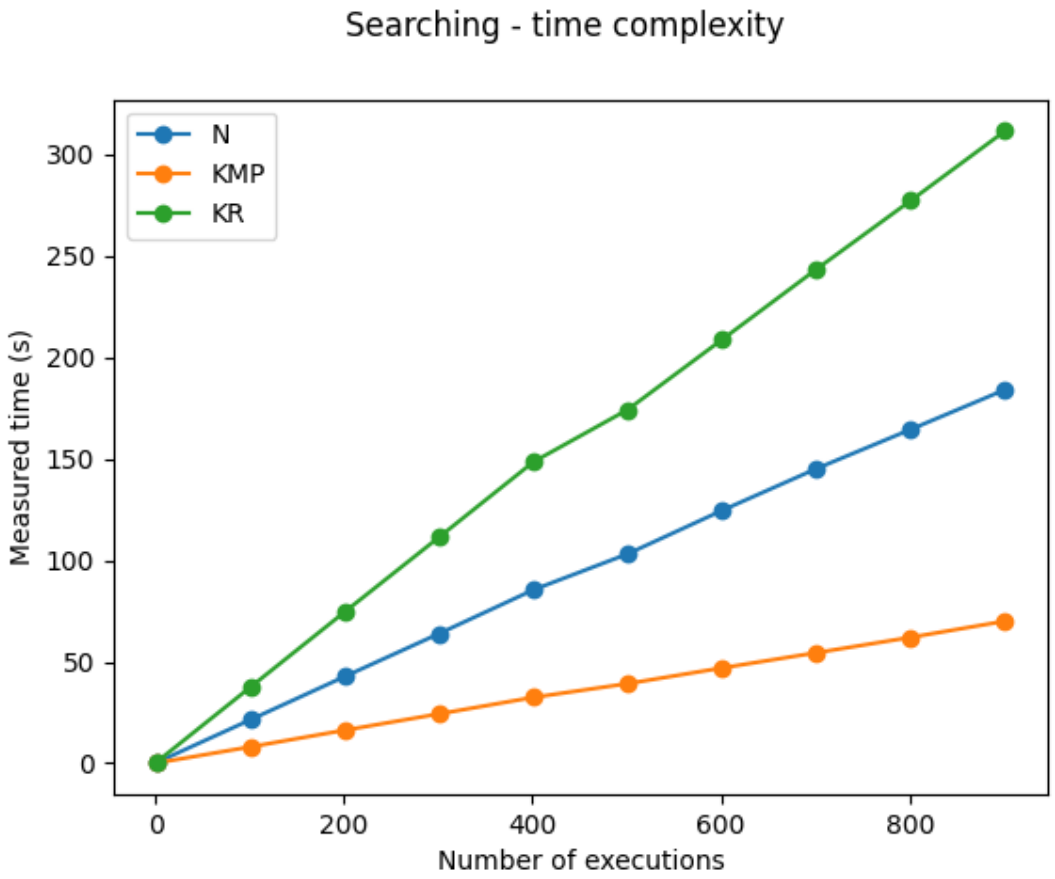
drawing_plots.py - plik z funkcją do rysowania wykresów na podstawie pomiarów czasowych.
time_measure.py - plik z funkcją do wykonywania pomiarów czasowych, w argumencie można podać ilość repetycji pomiarów, zwracana jest średnia z wszystkich pomiarów.

main.py - plik z zaimportowanymi, powyższymi modułami, w którym wykonywane są pomiary oraz generowane wykresy.

Instrukcja:
Program pisany w języku Python przy użyciu środowiska Visual Code Studio. Testy jednostkowe wykonywane przy użyciu modułu Pytest.
Aby przetestować każdy z algorytmów należy uruchomić testy jednostkowe w odpowiadających plikach testowych. Przetestowane zostały przypadki brzegowe oraz generowane losowo teksty i wzorce.

Opis:
W naszym projekcie porównaliśmy ze sobą czasy wykonywania trzech algorytmów wyszukiwania (naiwnego, Karpa-Rabina, Knutha-Morrisa-Pratta). Do pomiarów wykorzystaliśmy funkcje process_time() z biblioteki time. Mierzyliśmy czas wyszukiwania n pierwszych słów z pliku pan-tadeusz.txt, który został zmodyfikowany tak, aby nie zawierał polskich znaków ani znaków interpunkcyjnych. Liczba n pierwszych słów w naszych pomiarach zaczynała się od 1 a kończyła na 900. W naszych algorytmach małe i duże litery są rozróżniane, co oznacza że słowa zaczynające się z małej i dużej litery są traktowane jako odrębne.

Wyniki:



Wszystkie algorytmy mają ~liniową złożoność czasową.
Algorytm KMP okazał się najszybszym i to ze znaczną różnicą w prędkości działania w porównaniu do pozostałych algorytmów. Swoją szybkość zawdzięcza faktowi, że nigdy przy

przeszukiwaniu wzorca nie cofa się w iteracji. Potrzebuję on chwilę na początku na utworzenie DFA, jednak jest to jednorazowa akcja, a zysk czasowy, jaki w ten sposób uzyskujemy, rekompensuje stratę z nawiązką.
Drugi co do szybkości działania okazał się algorytm naiwny z nieco większym gradientem funkcji na wykresie.
Ostatecznie algorytm KR - według naszych pomiarów jego złożoność czasowa jest największa.