Lidia Sobońska 310903
Arkadiusz Kojtek 310746

AISDI

Wyszukiwanie wzorców

Sprawozdanie

Oświadczamy że poniższa praca, będąca podstawą do ocenienia wyników laboratorium z AISDI została wykonana przez nas samodzielnie.

Biblioteki wykorzystane w projekcie

Do stworzenia naszego projektu korzystaliśmy z następujących modułów:

1. **pyplot** z **Matplotlib,** który został wykorzystany do narysowania wykresów przebiegów czasowych.
2. **sys** służący do przyjęcia informacji o pliku źródłowym.
3. **timeit** wykorzystany do zmierzenia czasu działania każdej z funkcji.
4. **random**, który posłużył do wygenerowania losowych tekstów do celów testowych.

Algorytm tworzenia kopca

W pliku **pattern_search.py** znajdują się funkcję odpowiedzialne za wyszukiwanie wzorca w tekście. W pliku **pattern_search_test.py** znajdują się funkcje obsługujące testowanie algorytmów.
naive_pattern_search() - wyszukiwanie wzorców w tekście z wykorzystaniem metody naiwnej
kmp_pattern_search() - wyszukiwanie wzorców w tekście z wykorzystaniem algorytmu Knutha-Morrisa-Pratta.
kr_pattern_search() - wyszukiwanie wzorców w tekście z wykorzystaniem algorytmu Karpa-Rabina

Podział prac

Algorytm naiwny oraz algorytm KMP zostały zaimplementowane przez Arkadiusza Kojtek, a algorytm Karpa-Rabina oraz testowanie zostało napisane przez Lidię Sobońską.

Sposób uruchomienia

Plikiem obsługującym projekt jest **main.py**, uruchomienie projektu polega na

wywołaniu pliku **main.py** z poziomu terminala z dopisanym opcjonalnym argumentem reprezentującym plik zawierający dane do przeszukania. W przypadku, gdy nazwa pliku z danymi nie zostanie podana, domyślnie będzie brany plik '*pan-tadeusz.txt*'.

Przebieg działania programu

Najpierw uruchamiane są przypadki testowe, których wynik wypisywany jest na ekran oraz do pliku .tsv. Następnie program próbuje wczytać dane, na podstawie których mają pracować algorytmy, po czym przechodzi do mierzenia czasu w jakim każdy z algorytmów będzie szukał wzorca w tekście o długości dwa razy większej. Czas mierzony jest z użyciem funkcji **timeit()**. Informacje na temat czasu przeszukiwania są na bieżąco zapisywane do list, które następnie wykorzystywane są do utworzenia wykresu z wykorzystaniem obiektu **pyplot.** Na podstawie zebranych informacji tworzony jest plik .png z wykresem oraz .tsv ze zmierzonym czasem.

Pliki wynikowe

Jako plik wynikowy zostaną utworzone pliki:
times.tsv - czas przeszukiwania tekstu z użyciem poszczególnych algorytmów
search_times.png - wykres obrazujący przebieg czasowy każdego algorytmów
test_results.tsv - wyniki przeprowadzonych testów

Analiza danych

Analizując przebieg czasowy oraz powstałe wykresy można łatwo zauważyć, że czas działania algorytmu naiwnego okazał się najkrótszy. Wyniki okazały się bardzo zaskakujące, oczekiwaliśmy raczej odwrotnych wyników, jednak domyślamy się co może być przyczyną tych różnic.
Zacznijmy od algorytmu Knutha-Morrisa-Pratta, jego działanie polega na "zapamiętywaniu" postępu w trakcie pracy programu, algorytm powinien rozpoznać czy prefiks i sufiks są ze sobą zgodne i na podstawie tego ocenić czy konieczne jest sprawdzanie wzorca od początku. Niestety ta funkcjonalność nie jest tak korzystna przy przeglądaniu książki poszukując w niej poszczególnych słów, występowanie pasujących prefiksów i sufiksów jest zbyt rzadkie, aby korzyści były zauważalne, pozostałe działanie niewiele różni się od przeszukiwania naiwnego, obciążonego dodatkowe innymi instrukcjami przez co ogólna praca algorytmu w tym przypadku jest nieznacznie ale jednak wolniejsza.
Algorytm Karpa-Rabina z kolei pozwala uniknąć porównywania każdorazowo wszystkich znaków wzorca z danym fragmentem tekstu, lecz przy obecnej implementacji, nie jest to robione w żadnym z algorytmów. Korzyści płynące z tego skrócenia nie są więc zauważalne. Co więcej, dochodzi czas potrzebny do obliczeń, przez co ostatecznie czas wychodzi w tych testach najgorszy.
Dodatkowo, algorytm Karpa-Rabina jest bardzo wygodny, gdy porównanie nie musi być dokładne, na przykład gdy powinno się ignorować interpunkcję lub wielkość liter. Lecz znowu, w przypadku tych testów nie ma to miejsca, więc te korzyści także nie są widoczne.

Tabela ze zmierzonymi czasami dla jednego z testów

Number of words:	Naive Algorithm:	Knuth-Morris-Pratt Algorithm:	Karp-Rabin Algorithm:
1000	0.07193760000154725	0.08153029999812134	0.2206421999981103
2000	0.08007790000192472	0.16599789999963832	0.39084379999985686
3000	0.11816370000087772	0.18265690000043833	0.588319300000876
4000	0.15755780000108643	0.2587324999985867	0.8066629000022658
5000	0.18843749999723514	0.32136030000037863	1.0483426000027976
6000	0.22255020000011427	0.3772540999998455	1.3560169000011228
7000	0.2714893000011216	0.4346971000013582	1.454111600000033
8000	0.32577889999811305	0.5098105999968539	1.6710515999984636
9000	0.345288000000437	0.5585530000025756	1.8222419999983686
10000	0.3957274000022153	0.6453478000003088	2.022241999999096

Wykres z przebiegiem czasowym algorytmów dla jednego z testów