Zaawansowane algorytmy

Projekt 1

Analiza czasów działania algorytmów Sundaya, Morrisa-Pratta i Boyera Moora.

Adrian Świderek

367867

Grupa 23

Treść zadania:

Zaimplementuj i przebadaj algorytmy wyszukiwania przy następującym ograniczeniu:  
- Wczytujemy dane z pliku partiami - dostępny bufor to 2048 znaków (zakładamy, że wzorzec jest krótszy, a tekst może być dłuższy)  
Zaimplementuj w tym kontekście algorytmy Sundaya, Morrisa-Pratta i Boyera-Moora. Przetestuj je dla różnego rodzaju danych wejściowych.  
Należy zbadać czas działania algorytmów w zależności od wielkości alfabetu i wzorca.  
Przetestuj wzorce/tekst wygenerowane losowo, kody DNA i alfabet mówiony.

Uogólnij algorytm Sundaya tak, aby sprawdzał dwa kolejne znaki w tekście występujące za wzorcem. zamiast jednego. Dodatkowo zaimplementuj możliwość wyszukiwania wzorca zawierającego znak uogólniony '?' - pasuje każda litera.

Opis działania programu:

Poza algorytmami zajmującymi się wyszukiwaniem wzorca w tekście zaimplementowanymi na zajęciach, stworzyłem klasę o nazwie FileProcessing. Stworzona klasa posiada między innymi funkcję generateText – funkcja ta tworzy plik tekstowy z wygenerowanymi losowo znakami. Wielkość alfabetu w tym tekscie jest zależna od parametru który można w każdej chwili zmienić a długość tekstu jest wybierana losowo z przedziału który również podajemy w paramtrach funkcji. Kolejną ważną funkcją jest generatePattern. W tej funkcji z naszego tekstu wybieramy losowo ciąg znaków o długości podanej w parametrze funkcji. Ostatnią istotną funkcją jest readFromFileToBuffer. Tutaj wczytujemy z pliku 2048 znaków to naszego bufora następnie na tym buforze używamy naszych algorytmów wyszukiwania wzorca w tekście oraz mierzymy czas wykonywania za pomocą funkcji System.nanoTime() i zapisujemy nasz wynik w nanosekundach do pliku results.txt.

Tekst mówiony został wygenerowany przy pomocy strony: <http://lipsum.pl/index.php> a kod DNA: http://faculty.ucr.edu/~mmaduro/random.htm

Wyniki przebrowadzonych badań:

Pełne wyniki znajdują się w dwóch plikach excela w folderze z projektem.

Czas działania algorytmów zależny od wielkości alfabetu dla stałej długości wzorca:

Wnioski:

Możemy zaobserwować, że algorytm Sundaya jest przez zdecydowaną większość czasu algorytmem najszybszym jedynie w pierwszym przypadku gdzie rozmiar alfabetu był równy 2 czas jego wynonywania był bardzo zbliżony z czasem wykonywania algorytmu Boyera Moora. Im większy alfabet tym ta różnica na korzyść algorytmu Sundaya się zwiększała. Co ciekawe dla wiekości alfabetu = 2 średni czas wykonywania dla algorytmu Sundaya był zdecydowanie największy w porównaniu do innych wielkości alfabetów. Generalnie im większy alfabet tym czas działania spadał jednak nie tak drastycznie jak z 2 znaków na na 8 znaków – tam średni czas spadł ze średniej wartości 143790 nano sekund do 32740 nanosekund (średnia została obliczona z pominięciem największych odchyleń od normy. Co do pozostałych dwóch algorytmów, w większości przypadków szybszym okazywał się algorytm Boyera Moora jednak przy większych alfabetach różnica ta stawała się zdecydowanie mniejsza, czasem na korzyść algorytmu Morrisa-Pratta.

Czas działania algorytmów zależny od długosci wzorca i dla wielkości alfabetu uzależnionej od tekstu który jest badany (dla tekstu wygenerowanego losowo jest to 10).

Wnioski:

Najszybszym z algorytmów w większości przypadków był algorytm Sundaya a najwolniejszym algorytm Morrisa-Pratta. Im długość wzorca była większa tym czas wykonywania algorytmów spadał jednak w przypadku Morrisa-Pratta początkowo czas zaczął spadać – w tym przypadku widać to dla długości wzorca < 15 a od długości równej 24 czas wykonywania dla tego algorytmu drastyczcnie wzrastał. Kolejną ciekawą rzeczą którą możemy zauważyć jest fakt, że dla najdłuższych wzorców przy losowym tekście i DNA czas wykonywania algorytmu Boyera Moora był zawsze niższy od czasu wykonywania algorytmu Sundaya.