Algorytmy tekstowe

Laboratorium 1 – raport

Mateusz Kocot

Zadanie 1.

Zaimplementowano algorytmy wyszukiwania wzorców (*string_matching_algorithms.py*). Implementacja mocno opiera się na wykładzie. Do stworzenia tablicy przejścia dla automatu skończonego (finite automaton - fa) wykorzystano jednakże funkcję prefiksową *prefix_function*, co zmniejszyło złożoność czasową preprocessingu do $O(m|\Sigma|)$, gdzie m to długość wzorca, a $|\Sigma|$ – rozmiar alfabetu (zbioru znaków wzorca). Jest to możliwe dzięki następującej własności:

$$\delta(q, a) = \delta(\pi[q], a), \text{ jeśli } q = m \text{ lub } P[q+1] \neq a,$$

gdzie δ to funkcja przejść automatu, π – funkcja prefiksowa, P – wzorzec.

Zadanie 2.

W pliku *tests.py* zaimplementowano testy porównujące szybkość działania algorytmów.

Zadanie 3. / Zadanie 4.

Wyszukano wystąpienia wzorca *art* w załączonej ustawie. Czasy działania wszystkich algorytmów oscylują wokół $0.04\ s$. Brak znaczących różnic w szybkości działania spowodowany jest dużą różnicą w długości tekstu i wzorca. Najszybciej jednak zdaje się działać skończony automat.

Zadanie 5.

Wyszukano wystąpienia wzorca *kruszwil* w załączonym fragmencie polskiej Wikipedii. Czasy działania wszystkich algorytmów oscylują wokół $50\ s$. Tu także długość wzorca jest znacznie mniejsza od długości tekstu. Nie widać znacznej różnicy w czasie działania, jednakże fa i kmp wyprzedzają algorytm naiwny o kilka sekund.

Zadanie 6.

By czas wykonania fa i kmp znacznie zmalał w stosunku do algorytmu naiwnego, należy sprawić, by złożoność tego ostatniego ukwadratowiła się. Wystarczy więc znaleźć takie m, że $O\big(m(n-m+1)\big) = O(n^2)$. Widać, że za m można przyjąć wartość typu $n/10$ tak by w obu nawiasach występował czynnik n. Wówczas, przy założeniu alfabetu rozmiaru $O(1)$, złożoności fa i algorytmu kmp wynoszą $O(n) + O(n) = O(n)$. Zatem, dla odpowiednio długiego tekstu, założenie powinno zostać spełnione.

Wykorzystano załączony plik *ustawa.txt*, a za wzorzec przyjęto $(n/10)$-krotne powtórzenie litery *a* (n - długość tekstu), tak by utrzymać złożoność utworzenia tablicy przejścia - $O(n)$. Dla takich danych

czasy algorytmów: naiwnego, fa i kmp wyniosły odpowiednio ok. $0.36\ s$, $0.10\ s$, $0.09\ s$. Jak widać, algorytm naiwny został wykonany znacznie wolniej.

Zadanie 7.

Ponieważ obliczenie funkcji prefiksowej (kmp) ma złożoność $O(m)$, a obliczenie funkcji przejścia (fa) - $O(m|\Sigma|)$, by czas obliczenia funkcji przejścia był znacznie dłuższy od czasu obliczenia funkcji prefiksowej, należy wybrać odpowiednio długi wzorzec o odpowiednio dużej wielkości alfabetu.

Jako przykład wybrałem ciąg liczb i liter o długości 1000. Wówczas czasy policzenia funkcji przejścia i funkcji prefiksowej wynoszą odpowiednio ok. $0.018\ s$ oraz 0.001 s. Czas utworzenia tej pierwszej jest więc ok. 18 razy dłuższy. Dodatkowo, dla porównania, czas utworzenia funkcji przejścia za pomocą algorytmu z wykładu o złożoności $O(m|\Sigma|^3)$ wynosi ok. $18\ s$, czyli nieporównywalnie więcej.