Streszczenie w języku polskim

Praca jest kompilacją serii wyników z dziedziny algorytmów tekstowych. W szczególności rozważamy warianty problemu znajdowania szablonu i wyszukiwania wzorca w tekście z niedopasowaniami. Jako n będziemy oznaczać długość wejściowego tekstu.

W pierwszej części rozpatrujemy dokładne warianty problemu znajdowania szablonów. W oryginalnym problemie szukamy takiego słowa, którego wystąpienia pokrywają dany tekst.

W pracy „Efficient Computation of 2-covers of a String" analizowane jest pojęcie 2-szablonów, czyli par słów o tej samej długości, których wystąpienia pokrywają dany tekst. W pracy przedstawiamy algorytm znajdujący dla każdej długości k przykładowy 2-szablon długości k (jeśli dla niej takowy istnieje) w oczekiwanym czasie O(n log n log log n) i pamięci O(n).

„Shortest Covers of All Cyclic Shifts of a String" rozwiązuje problem znalezienia najkrótszych szablonów dla wszystkich cyklicznych przesunięć w czasie O(n log n) i pamięci O(n). Dodatkowo, opracowany jest tam bardzo dokładny opis zbioru długości najkrótszych szablonów wszystkich cyklicznych przesunięć słów Fibonacciego.

W pracy „Internal Quasiperiod Queries" pokazujemy jak efektywnie odpowiadać na zapytania wewnętrzne o najkrótsze i wszystkie szablony. Opracowana struktura danych pozwala na znalezienie najkrótszego szablonu na przedziale w czasie O(log n log log n) lub wszystkich szablonów w czasie O(log n (log log n)^2). Struktura danych potrzebuje O(n log n) pamięci i O(n log n) czasu obliczeń wstępnych. Dodatkowo, przedstawione są efektywne rozwiązania w modelu offline.

W pracy „String Covers of a Tree" analizowane jest naturalne uogólnienie szablonu, w którym pokrywamy nie słowo, a drzewo. Krawędzie są etykietowane literami. Etykietą ścieżki jest słowo będące sekwencją liter na przechodzonych krawędziach. Słowo nazywamy szablonem drzewa, jeśli ścieżki etykietowane tym słowem pokrywają całe drzewo. Dla skierowanego drzewa przedstawiamy algorytm znajdujący wszystkie szablony w czasie O(n log n log log n), a w przypadku nieskierowanym w czasie i pamięci O(n^2). Dodatkowo, w przypadku nieskierowanym przedstawiamy alternatywny algorytm używający tylko O(n) pamięci, lecz działający w czasie O(n^2 log n).

W drugiej części rozprawy przedstawiamy asymptotycznie efektywne rozwiązania do praktycznych problemów.

W pracy ``Efficient Computation of Sequence Mappability'' rozważany jest problem mapowalności. Musimy w nim policzyć dla każdego indeksu i, będącego początkiem podsłowa długości m, na ilu innych indeksach znajduje się pasujące podsłowo tej samej długości, dopuszczając do k niedopasowań. Przedstawiamy kilka wyników, z których głównym jest algorytm działający w czasie O(n min{log^k n, m^k}) i pamięci O(n) dla ustalonego k. Dodatkowo, pokazujemy, że nie można rozwiązać problemu mapowalności w ściśle subkwadratowym czasie, chyba że hipoteza Strong Exponential Time jest nieprawdziwa.

W pracy „Circular Pattern Matching with k Mismatches" analizujemy wariant wyszukiwania wzorca, w którym wzorzec może być dowolnie przesuwany cyklicznie i dopuszczamy do k niedopasowań. Przedstawiamy prosty algorytm działający w czasie O(nk) i pamięci O(m) i bardziej skomplikowany algorytm O(n + nk^4/m), działający również w pamięci O(m).

Wyniki często bazują na analizie okresowości i użyciu zarówno klasycznych narzędzi stringologicznych, jak i najnowocześniejszych, zaawansowanych struktur danych.