ZESZYTY NAUKOWE POLITECHNIKI ŁÓDZKIEJ
Nr 1078 ELEKTRYKA, z. 121 2010

SZYMON GRABOWSKI
Wydział Elektrotechniki, Elektroniki, Informatyki i Automatyki
Politechniki Łódzkiej

NOWE ALGORYTMY WYSZUKIWANIA DOKŁADNEGO I PRZYBLIŻONEGO

Recenzent: **doc. dr hab. Adam Jóźwik**

Maszynopis dostarczono 1. 10. 2010

Praca przedstawia główne wyniki z tematyki algorytmów tekstowych otrzymane w Katedrze Informatyki Stosowanej w latach 2004–2009. Algorytmy te dotyczą wybranych rozmaitych problemów wyszukiwania dokładnego i przybliżonego, również w intensywnie w ostatnich latach badanym scenariuszu z wykorzystaniem kompresji.

1. WPROWADZENIE

Można argumentować, że szeroko rozumiane *wyszukiwanie* jest najważniejszym, najbardziej fundamentalnym z problemów, jakimi zajmuje się informatyka. Problem ten polega na wskazaniu wystąpień danego obiektu (klucza) w bazie danych. Wskazanie (raportowanie) wystąpień w praktyce oznacza zwykle zwrócenie liczby wystąpień danego obiektu lub zwrócenie lokacji (indeksów) jego wystąpień. Użyte pojęcie bazy danych jest bardzo szerokie: może chodzić o relacyjną bazę danych opartą na tabelach zawierających rekordy, może to być kolekcja dokumentów tekstowych (np. webowych), hierarchiczna struktura np. XML, sekwencja DNA itd. Rzecz jasna, tak ogólnie rozumianego problemu nie można atakować w sposób uniwersalny, dlatego partykularne problemy wyszukiwania pojawiają w rozmaitych działach

informatyki (algorytmiki). Jedną z ważnych dziedzin są tu algorytmy tekstowe (ang. *string matching*, *text matching*) [1].

W Katedrze Informatyki Stosowanej, w latach 2004–2009 opracowano szereg algorytmów dla rozmaitych problemów wyszukiwania dokładnego (ang. *exact matching*) i przybliżonego (*approximate matching*) w tekście, także w intensywnie w ostatnich latach badanym scenariuszu z wykorzystaniem kompresji (online lub w wersji z indeksem). W poniższych sekcjach zarysowano tematykę badawczą, przedstawiono główne wyniki i ich zastosowania. Materiał ten został opisany w monografii „New algorithms for exact and approximate text matching" autora niniejszego artykułu, przedłożonej jako rozprawa habilitacyjna [2].

2. ALGORYTM AVARAGE-OPTIMAL SHIFT-OR

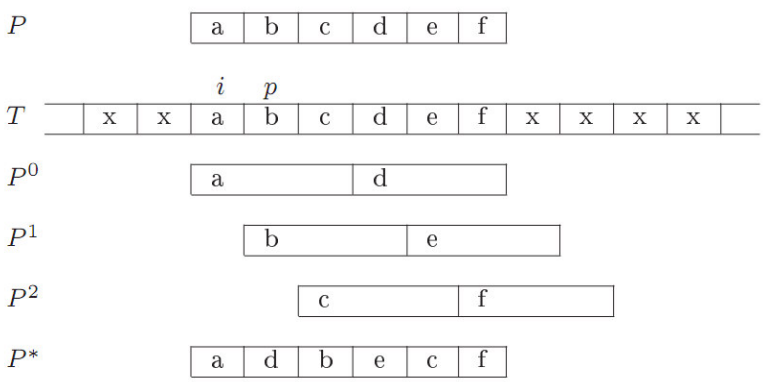
Algorytm opisany w tej sekcji rozwiązuje problem wyszukiwania dokładnego, ale aplikacje algorytmiczne kluczowej (nowatorskiej) techniki tego algorytmu mają też zastosowania dla innych problemów wyszukiwania w tekście.

Klasyczny problem wyszukiwania dokładnego można sformułować następująco: dany jest tekst $T[0..n–1]$ o długości n symboli nad skończonym alfabetem $\Sigma = \{0, 1, \ldots, \sigma - 1\}$ oraz wzorzec $P[0..m–1]$ o długości m (nad tym samym alfabetem); należy zwrócić wszystkie indeksy $0 \le j \le n–m$, takie że $T[j..j+m–1] = P[0..m–1]$. Dwa słynne i o doniosłym znaczeniu dla całej dziedziny badawczej algorytmy rozwiązujące ten problem, to algorytm Knutha–Morrisa–Pratta (KMP) oraz algorytm Boyera–Moore'a (BM), oba z lat 70-tych XX wieku [3]. KMP był pierwszym algorytmem znajdującym wzorzec w czasie $O(n)$ w najgorszym przypadku, natomiast algorytm BM to pierwsza technika działająca w średnim przypadku w czasie subliniowym w długości tekstu. Analiza przypadku średniego, o której będzie tu mowa, zakłada (zgodnie ze zwyczajami przyjętymi w literaturze przedmiotu), iż rozkład symboli w tekście i wzorcu jest jednostajny nad danym alfabetem, zaś poszczególne znaki są „wylosowane" niezależnie od siebie. Mimo ponad 30 lat badań i ważnych wyników teoretycznych otrzymanych w tym czasie, wciąż jeszcze proponowane są nowe, zorientowane głównie na rezultaty praktyczne, algorytmy wyszukiwania dokładnego. Jedną z popularnych w ostatnich latach technik jest wykorzystanie tzw. równoległości bitowej (ang. *bit-parallelism*) do symulowania niedeterministycznego automatu skończonego (NFA) rozpoznającego zadany wzorzec. Do znanych z literatury algorytmów tego typu należą m. in. Shift-Or [4] i (szybszy od niego) BNDM [5]. Wadą algorytmu Shift-Or jest niemożność

wykonywania „przeskoków” (ang. *skips*); nawet w najlepszym przypadku działa on w czasie $O(n)$.

W pracy [6] przedstawiono nowatorską ideę wykorzystania techniki Shift-Or do wyszukiwania wzorców w tekście z przeskokami, co prowadzi do subliniowej średniej złożoności wyszukiwania.

Pokazano, że jeśli długość wzorca $m \leq w$, gdzie w jest długością słowa maszynowego, wyrażoną w bitach (w praktyce 32 lub 64), to algorytm ten osiąga optymalną złożoność w przypadku średnim, tj. $O(n \log_\sigma(m) / m)$, co znalazło odbicie w nazwie nadanej algorytmowi, Average-Optimal Shift-Or (AOSO). Pokazano także proste (choć jak się wydaje, nie stosowane wcześniej w tym kontekście) rozwiązanie implementacyjne, przyspieszające znacznie zarówno algorytm Shift-Or, jak i AOSO (w tej wersji algorytm nazwano Fast AOSO = FAOSO).



Rys. 1. Tworzenie sztucznego wzorca w algorytmie AOSO. P = abcdef, q = 3

Algorytm AOSO próbkuje tekst T w regularnych odstępach, o wielkości q znaków, zastępując oryginalny wzorzec P konkatenacją odpowiednio wybranych podsekwencji P, dla której budowany jest wariant automatu Shift-Or (rys. 1). W oryginalnym algorytmie Shift-Or trafienie odczytywane jest poprzez sprawdzenie pojedynczego bitu w wektorze stanu algorytmu; w algorytmie zaproponowanym test ten obejmuje grupę $\lfloor m/q \rfloor$ bitów. Nowy algorytm wymaga modyfikacji przetwarzania wstępnego algorytmu Shift-Or (przy zachowaniu jego złożoności czasowej i pamięciowej) oraz weryfikacji potencjalnych trafień. Rys. 2 przedstawia główną procedurę algorytmu AOSO, dla tekstu T o długości n, wzorca P o długości m oraz parametru przeskoku q. Tablica B, tworzona w czasie przetwarzania wstępnego, ma rozmiar $\sigma * m$ bitów. D jest wektorem stanu o długości m bitów.

```
1   for i ← 0 to σ − 1 do B[i] ← ∼0
2   h ← 0; mm ← 0
3   for j ← 0 to q − 1 do
4       for i ← 0 to ⌊m/q⌋ − 1 do
5           B[P[iq + j]] ← B[P[iq + j]] & ∼(1 << h)
6           h ← h + 1
7       mm ← mm | (1 << (h − 1))
8   D ← ∼0; i ← 0
9   do
10      D ← ((D & ∼mm) << 1) | B[T[i]]
11      if (D & mm) ≠ mm then Verify(T, i, n, P, m, q, D)
12      i ← i + q
13  while i < n
```

Rys. 2. Procedura AOSO(*T*, *n*, *P*, *m*, *q*)

Interesujące są konkretne wyniki, tj. szybkości działania zaproponowanego algorytmu. Zależą one (podobnie jak przy innych efektywnych algorytmach dla tego problemu) od typu danych, na jakich wykonuje się testy (głównie chodzi o wielkość alfabetu). Dla tekstu w języku naturalnym (angielskim, choć dla innych języków różnice nie będą istotne), na komputerze wyposażonym w procesor Intel Core 2 Duo o zegarze 3 GHz (wykorzystywano w testach tylko jeden rdzeń procesora), algorytm FAOSO osiągnął szybkość wyszukiwania od 1,45 GB/s (krótkie wzorce, $m = 4$) do 5,28 GB/s (długie wzorce, $m = 28$) [7]. Dla porównania, algorytm BNDM osiągnął w tym teście od 0,71 GB/s do 2,12 GB/s. Na sekwencji DNA szybkości były ogólnie niższe, ale i tu FAOSO dominował (1,47 GB/s – 3,72 GB/s), poza przypadkiem wzorców najkrótszych ($m = 4$), mało jednak przy DNA interesujących (niewielka selektywność wzorca), gdzie szybkość 0,39 GB/s nie należała do wyników najlepszych. Innymi słowy, w porównaniu z algorytmem BNDM algorytm FAOSO był zwykle ok. 2–2,5x szybszy. W porównaniu z najszybszym z testowanych algorytmów literaturowych (BNDM2 [8] z 2005 r. autorstwa Holuba i Duriana), był on zwykle szybszy o 10–20%.

W świetle przedstawionych wyników, współopracowany przez autora (współautor koncepcji: Kimmo Fredriksson z University of Joensuu w Finlandii) algorytm FAOSO należy do najszybszych w praktyce znanych rozwiązań wyszukiwania dokładnego. Sukces ten, jak się wydaje, ma dwie przyczyny: optymalną złożoność czasową w przypadku średnim (dla krótkich wzorców; dla wzorców dłuższych złożoność ta niewiele odbiega od optymalnej) oraz wykorzystanie prostej logiki bitowej, prowadzącej do efektywnego kodu zwłaszcza na współczesnych procesorach. Podkreślić należy, że przewaga algorytmu FAOSO nad konkurencją była większa na nowszym procesorze Core 2 Duo niż na procesorach Pentium4 i UltraSPARC IIIi [7]. Pewną wadą tego

algorytmu jest zależność od parametru q, który wprawdzie dla konkretnego typu danych można wyliczyć teoretycznie, ale w praktyce (rozkład często nie jest jednostajny!) jego wybór jest kwestią wstępnych eksperymentów. Aby pominąć tę niedogodność, opracowano również adaptacyjną odmianę tego algorytmu [7], gdzie parametr q może zmieniać się w trakcie wyszukiwania, dostosowując się do właściwości statystycznych tekstu. Wariant ten jest jednak o ok. 30% wolniejszy od FAOSO.

Opracowana przez autora technika równoległości bitowej posłużyła [7] do opracowania optymalnej w przypadku średnim (bez względu na długość wzorca) odmiany klasycznego algorytmu Aho–Corasick (AC) służącego do wyszukiwania wielu wzorców jednocześnie (ang. *multiple matching*). Również i w tym przypadku wyniki praktyczne są konkurencyjne względem rozwiązań istniejących. (Zaproponowana technika ma ponadto zastosowanie do jeszcze kilku problemów, m.in. *matching with swaps* [7].)

Zastosowania szybkich algorytmów wyszukiwania dokładnego (jednego lub większej liczby wzorców) są dość oczywiste: edytory tekstu, skanery antywirusowe, systemy plików, bioinformatyka; wyszukiwanie dokładne (*online*) jest też często jednym z etapów wyszukiwania przy pomocy tzw. indeksów odwrotnych (ang. *inverted indexes*), które wykorzystuje się powszechnie w wyszukiwarkach internetowych i innych systemach wyszukujących i katalogujących dane.

Wyniki te otrzymane zostały we współpracy z dr. Kimmo Fredrikssonem i zaprezentowano je w dwóch publikacjach: [6] z konferencji *String Processing and Information Retrieval* (SPIRE'05) oraz [7], artykule z pisma *Journal of Discrete Algorithms*.

3. TECHNIKA MATRYOSHKA COUNTERS OSZCZĘDNEJ REPREZENTACJI LICZNIKÓW

Innym rozważanym problemem (czy raczej klasą problemów) było wyszukiwanie przybliżone w trybie online, czyli takim, gdzie wzorzec P nie musi występować w postaci dokładnej w tekście T, a wystarczy jedynie, aby istniały obszary tekstu T wystarczająco podobne do P, w sensie zdefiniowanej miary podobieństwa i zadanego kryterium błędu. Tryb online oznacza, iż tekst T nie może zostać poddany uprzedniemu indeksowaniu. W wyniku badań autora przedstawiono nowatorską technikę użycia zagnieżdżonych liczników (nazwano ją *Matryoshka counters*) [9] w algorytmach wyszukiwania przybliżonego wykorzystujących liczniki. W szczególności autor wykazał jak zredukować złożoność czasową klasycznego algorytmu Shift-Add [10], służącego do wyszukiwania wg miary Hamminga z limitem błędów k, z $O(n\lceil m \log k / w \rceil)$ do $O(n\lceil m/w \rceil)$, gdzie w jest ponownie liczbą bitów w słowie maszynowym. I tym razem zaproponowana technika (tj. *Matroyshka counters*) należy do

elastycznych, mających liczne zastosowania. Są to, między innymi, wyszukiwanie przybliżone w sensie (δ, γ) (zastosowanie: wyszukiwanie w sekwencjach zapisów muzycznych, np. MIDI), problemy *k*-insercji (ang. *k-insertions*) i *episode matching*; oba ostatnie modele mają zastosowanie w wykrywaniu intruzów np. w ruchu sieciowym. Wyniki te opisane zostały w publikacji [11] z konferencji LATA'09.

4. INNE BADANE PROBLEMY WYSZUKIWANIA PRZYBLIŻONEGO

W bioinformatyce, analizie sekwencji muzycznych oraz ogólniej rozumianej analizie sygnałów wykorzystywane są globalne miary podobieństwa, z których bodaj najstarszą jest długość najdłuższej wspólnej podsekwencji (longest commone subsequence, LCS). Wyniki autora dotyczą głównie problemu LCTS (longest common transposition-invariant subsequence), gdzie szukamy najdłuższej podsekwencji dwóch danych sekwencji, takiej że odpowiednia para symboli różni się o stałą wartość. Problem ten ma zastosowanie w muzyce (ściślej, w dziedzinie określanej mianem music information retrieval), gdzie stała różnica między parami symboli to odpowiadające sobie pary nut w potencjalnie różnych tonacjach; podobieństwo oznacza tę samą (bądź podobną) melodię, ale np. zagwizdaną przez użytkownika bazy sekwencji melodycznych w innej tonacji niż ta, w której przechowywana jest dana melodia.

Dla problemu LCTS autor (we współpracy z pozostałymi autorami zacytowanych dalej publikacji) otrzymał dwa wyniki: algorytm o złożoności pesymistycznej $O(nm \log \log \min(n, m, \sigma))$ [12], gdzie n, m to długości pasowanych sekwencji, a σ to wielkość alfabetu, co jest najlepszym znanym wynikiem dla tego problemu, oraz algorytm hybrydowy łączący znane efektywne podejścia dla tego problemu (algorytm Hunta–Szymanskiego oraz algorytm bitowo-równoległy Hyyrö) w taki sposób, aby każdy z komponentów działał w obszarze swojej „kompetencji" [13,14]. Testy eksperymentalne wykazały, iż na danych muzycznych (MIDI) algorytm ten jest znacząco (do 2 razy) szybszy niż inne znane rozwiązania.

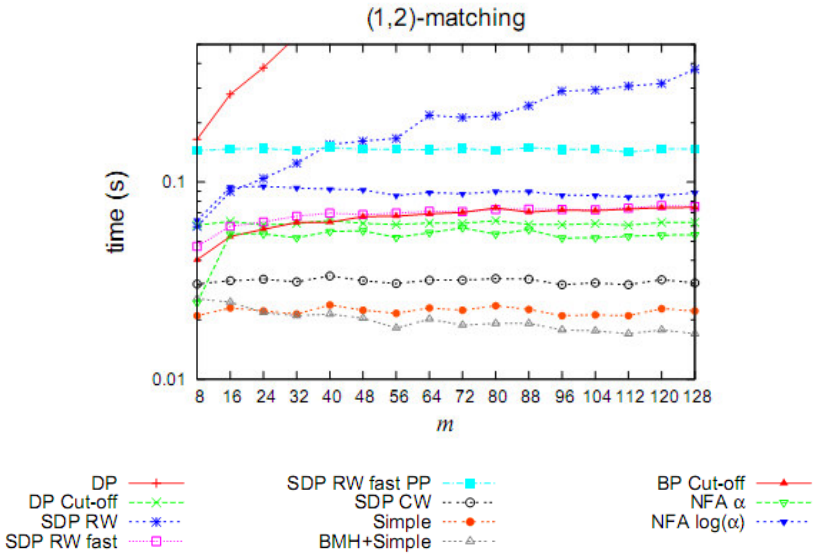
5. WYSZUKIWANIE Z PRZERWAMI

Kolejny badane zagadnienie dotyczy ważnej klasy problemów wyszukiwania, gdzie region tekstu może zawierać wzorzec z przerwami między poszczególnymi znakami. Bodaj najważniejsze zastosowanie wyszukiwania z przerwami (ang. *matching with gaps*) to bioinformatyka, w szczególności analiza sekwencji białkowych. Inne zastosowanie to analiza sekwencji muzycznych, gdzie przerwy odpowiadają pominiętym nutom w sekwencji z bazy melodii (pominięcia te mogą odpowiadać ozdobnikom czy innym elementom sekwencji

muzycznej nie wpływającym istotnie – lub w ogóle – na linię melodyczną danego utworu muzycznego). W ramach tych zagadnień przedstawiono szereg nowych algorytmów dla modeli (δ, α) i (δ, γ, α), gdzie δ odpowiada za dopuszczalną maksymalną różnicę wartości dla pary symboli (np. dopuszczalne „zafałszowanie" melodii), α za maksymalną dopuszczalną długość przerwy, zaś γ jest maksymalnym sumarycznym błędem w sensie δ. Dla protein przyjęty model jest nieco inny, kryterium δ zastępowane jest klasami symboli, zaś zakresy dopuszczalnych przerw są zwykle różne pomiędzy każdą parą znaków. Na szczęście, większość opracowanych algorytmów dla problemu (δ, α) daje się łatwo zaadoptować dla wyszukiwania w sekwencjach białkowych.

Nowe algorytmy dla wymienionych problemów (również w wersji *transposition-invariant*, tzn. z możliwą zmianą tonacji jednej z pary sekwencji muzycznych) wykorzystują następujące techniki algorytmiczne: rzadkie programowanie dynamiczne (ang. *sparse dynamic programming*), równoległość bitową, oszczędne pamięciowo symulacje automatów NFA oraz filtrację. Mnogość przedstawionych nowych wyników wiąże się z faktem, iż żaden algorytm nie dominuje, w sensie teoretycznym lub praktycznym, w pełnym spektrum scenariuszy; przeanalizowano szczegółowo zależności między złożonością czasową przetwarzania wstępnego (*preprocessingu*) a czasu właściwego szukania w opracowanych algorytmach, a także zależności między złożonością przypadku średniego a najgorszego. Przykładowo, optymalizacja algorytmu pod kątem średniej złożoności czasowej często pogarsza jego złożoność w przypadku najgorszym.

Jednym z ważnych otrzymanych wyników jest algorytm bitowo-równoległy wyszukiwania w modelu (δ, α) o złożoności fazy szukania $O(n\lceil m/w \rceil)$, który można w naturalny sposób dostosować do scenariusza, w którym pojawiają się przerwy ujemne; niekonwencjonalny ten model ma zastosowanie w analizie białek i do tej pory uważano ów problem za trudny (wg wiedzy autora, niewiele istniało wcześniej algorytmów dla tego wariantu problemu i nie miały one interesującej złożoności w przypadku najgorszym). Również wyniki eksperymentalne opracowanych algorytmów potwierdzają ich zalety. Rys. 3 pokazuje zależności między długością wzorca m a czasem wyszukiwania dla modelu (δ, α) przy parametrach $\delta = 1$, $\alpha = 2$. Jak widać na rysunku (czas wyrażony w skali logarytmicznej), klasyczny algorytm programowania dynamicznego (DP) jest wielokrotnie wolniejszy od większości algorytmów zaproponowanych (m. in. Simple, BMH+Simple, Sparse Dynamic Programming Column-Wise (SDP CW)), a co gorsza, jego wydajność spada liniowo wraz z długością wzorca, w przeciwieństwie do większości nowych algorytmów.

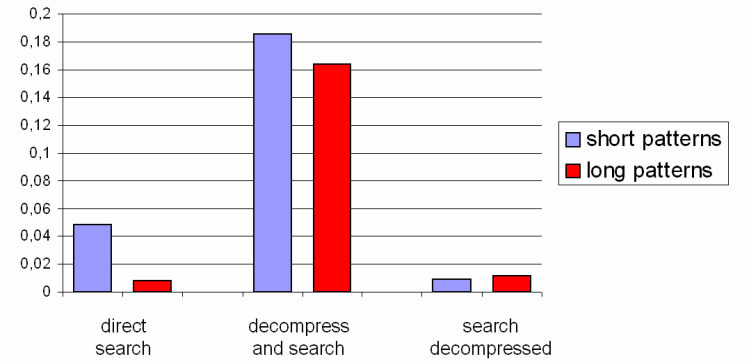


Rys. 3. Czasy szukania wzorca w modelu (δ, α) przy parametrach $\delta = 1$, $\alpha = 2$

Osiągnięte wyniki, otrzymane we współpracy z Kimmo Fredrikssonem, zostały opublikowane w pracach [15]–[19].

6. WYSZUKIWANIE W TEKŚCIE SKOMPRESOWANYM

Jeszcze inne zagadnienie badawcze to wyszukiwanie online w danych dostępnych w postaci skompresowanej, bez „jawnej" dekompresji. Scenariusz ten jest stosunkowo nowym, intensywnie badanym paradygmatem wyszukiwania. Osiągnięte w Katedrze wyniki to: prosty, lecz efektywny, algorytm kompresji oparty na q-gramach, pozwalający na wyszukiwanie pełnotekstowe (współpraca z Kimmo Fredrikssonem, publikacja [20] w piśmie *Information Processing Letters*) oraz poprawa efektywności istniejących kodów bajtowych, w przypadku tekstów statycznych [21]. Podkreślić należy, że wyszukiwanie w zaproponowanym algorytmie kompresji, dla „długich" wzorców (w praktyce dla tekstów w języku naturalnym oznacza to minimalną długość 7 lub 9 znaków) jest znacznie szybsze niż wyszukiwanie w równoważnym tekście nieskompresowanym (nie każdy z algorytmów tej kategorii cechuje się tą zaletą), co ukazuje rys. 4.



Rys. 4. Czasy szukania wzorców krótkich i długich w pliku *dickens* o rozmiarze 10.2 MB. Dla wzorców długich szukanie w tekście skompresowanym („direct search") trwa krócej niż w tekście w postaci oryginalnej („search decompressed")

Algorytm ten demonstruje, w jaki sposób znane wcześniej kody bajtowe, np. (*s*,*c*)-DC [22], używane w kompresji „na słowach", mogą być dostosowane do scenariusza pełnotekstowego, a zatem poszerza on znacznie zakres ich stosowania.

7. SKOMPRESOWANE INDEKSY PEŁNOTEKSTOWE

Indeksy pełnotekstowe, wg definicji, pozwalają na wyszukiwanie dowolnych fraz w tekście, a nie tylko fraz złożonych z pełnych wyrazów. Tym samym pozwalają one na „obsługę" np. języków orientalnych (gdzie nie istnieje odpowiednik spacji, tj. naturalna segmentacja tekstu), sekwencji bioinformatycznych, kodów języków programowania (gdzie segmentacja może utrudniona) czy muzycznych sekwencji nutowych. Rzecz jasna, indeksy pełnotekstowe działają również z językami, w których podział na słowa jest naturalny (np. polski, angielski). Indeksy pełnotekstowe znane są od lat 70-tych (drzewo sufiksowe), jednak w dopiero ok. roku 2000 pojawiły się pierwsze skompresowane indeksy pełnotekstowe i od tej pory zagadnienie to przeżywa niezwykle dynamiczny rozwój (kilkadziesiąt wartościowych prac, które pojawiły się w ostatnich 9–10 latach). Oszczędne pamięciowo indeksy można użyć np. dla sekwencji DNA liczonych w gigabajtach, gdzie użycie drzewa sufiksowego jest praktycznie wykluczone (potrzebuje ono ok. 20 razy więcej pamięci niż wymagana dla przechowania samej indeksowanej sekwencji).

W Katedrze Informatyki Stosowanej opracowano ideę bardzo prostego (niemniej efektywnego) indeksu skompresowanego z rodziny indeksów FM (wykorzystujących transformatę Burrowsa–Wheelera), o nazwie FM-Huffman. Indeks ten, realizujący operacje *count* (zliczanie trafień), *locate* (zwracanie pozycji trafienia) i *display* (wyświetlanie kontekstu wokół danego trafienia),

poddano szczegółowej analizie i zaproponowano szereg odmian o lepszych własnościach użytkowych niż rozwiązanie oryginalne. Współautorami byli (m.in.) prof. Gonzalo Navarro z Uniwersytetu Chilijskiego oraz dr Veli Mäkinen z Uniwersytetu w Helsinkach, czołowi eksperci w dziedzinie indeksów skompresowanych. Ta międzynarodowa współpraca w licznym gronie zaowocowała publikacjami [23]–[26] (artykuły z konferencji SPIRE'04, PSC'05, PSC'06 oraz artykuł w piśmie *International Journal of Foundations of Computer Science* (2006)).

8. PODSUMOWANIE

Opracowane algorytmy często wykorzystują „modne" w ostatnich latach podejścia: równoległość bitową oraz zastosowanie kompresji. Obok analiz teoretycznych, większość z zaproponowanych algorytmów zaimplementowano i poddano weryfikacji przy użyciu testów empirycznych, a osiągnięte wyniki zwykle pozwalają zaliczyć nowe metody do najefektywniejszych dla danych problemów.

LITERATURA

[1] **Navarro G., Raffinot M.:** Flexible Pattern Matching in Strings – Practical on-line search algorithms for texts and biological sequences. Cambridge University Press, 2002.

[2] **Grabowski Sz.:** New Algorithms for Exact and Approximate Text Matching. Wydawnictwo Politechniki Łódzkiej, 2009.

[3] **Cormen T.H., Leiserson C.E., Rivest R.L., Stein C.:** Wprowadzenie do algorytmów. Wydawnictwa Naukowo-Techniczne, Warszawa 2004.

[4] **Baeza-Yates R.A., Gonnet G.H.:** A New Approach to Text Searching. Proc. 12th Int. Conference on Research and Development in Information Retrieval (SIGIR), str. 168–175. ACM Press, 1989.

[5] **Navarro G., Raffinot M.:** Fast and flexible string matching by combining bit-parallelism and suffix automata. ACM Journal of Experimental Algorithmics, 5(4), 2000.

[6] **Fredriksson K., Grabowski Sz.:** Practical and optimal string matching. Proc. String Processing and Information Retrieval Conference (SPIRE), LNCS 3772, str. 374–385, Springer, 2005.

[7] **Fredriksson K., Grabowski Sz.:** Average-optimal string matching. Journal of Discrete Algorithms, 7(4):579–594, 2009.

[8] **Holub J., Durian B.:** Fast Variants of Bit Parallel Approach to Suffix Automata. Talk given in The Second Haifa Annual International Stringology Research Workshop of the Israeli Science Foundation, 2005.

[9] **Grabowski Sz., Fredriksson K.:** Bit parallel string matching under Hamming distance in $O(n\lceil m/w \rceil)$ worst case time. Information Processing Letters, 105(5):182–187, 2008.

[10] **Baeza-Yates R.A., Gonnet G.H.:** A New Approach to Text Searching. Communications of the ACM, 35(10): 74–82, 1992.

[11] **Fredriksson K., Grabowski Sz.:** Nested Counters in Bit-Parallel String Matching. Proc. 3rd Int. Conference on Language and Automata Theory and Applications (LATA), LNCS 5457, str. 338–349, Springer, 2009.

[12] **Navarro G., Grabowski Sz., Mäkinen V., Deorowicz S.:** Improved Time and Space Complexities for Transposition Invariant String Matching, Technical Report TR/DCC-2005-4, University of Chile, Department of Computer Science, 2005.

[13] **Grabowski Sz., Deorowicz S.:** Nice to be a chimera: A hybrid algorithm for the longest common transposition-invariant subsequence problem. Proc. Int. Conference on Modern Problems of Radio Engineering, Telecommunications, and Computer Science (TCSET), str. 50–54, Lviv, Ukraina, 2008.

[14] **Deorowicz S., Grabowski Sz.:** A hybrid algorithm for the longest common transposition-invariant subsequence problem. Computing and Informatics, 28, str. 729–774, 2009.

[15] **Fredriksson K., Grabowski Sz.:** Efficient Bit-parallel Algorithms for (δ, α)-matching. Proc. Int. Workshop on Experimental and Efficient Algorithms (WEA), LNCS 4007, str. 170–181, Springer, 2006.

[16] **Fredriksson K., Grabowski Sz.:** Efficient algorithms for (δ, γ, α)-matching. Proc. Prague Stringology Conference (PSC), str. 29–40, Prague, Czechy, 2006.

[17] **Fredriksson K., Grabowski Sz.:** Efficient algorithms for pattern matching with general gaps and character classes. Proc. String Processing and Information Retrieval Conference (SPIRE), LNCS 4209, str. 267–278, Springer, 2006. (Nagroda za najlepszy artykuł.)

[18] **Fredriksson K., Grabowski Sz.:** Efficient Algorithms for (δ, γ, α) and (δ, k_Δ, α)-matching. International Journal of Foundations of Computer Science, 19(1):163–183, 2008.

[19] **Fredriksson K., Grabowski Sz.:** Efficient algorithms for pattern matching with general gaps, character classes and transposition invariance. Information Retrieval, 11(4), str. 335–357, 2008.

[20] **Fredriksson K., Grabowski Sz.:** A general compression algorithm that supports fast searching. Information Processing Letters, 100(6):226–232, 2006.

[21] **Grabowski Sz.**: Making dense codes even denser. AGH Automatyka, tom 12, zeszyt 3, str. 769–779, 2008.

[22] **Brisaboa N., Fariña A., Navarro G., Esteller M.:** (S,C)-Dense Coding: An Optimized Compression Code for Natural Language Text Databases. Proc. String Processing and Information Retrieval Conference (SPIRE), LNCS 2857, str. 122–136, Springer, 2003.

[23] **Grabowski Sz., Mäkinen V., Navarro G.:** First Huffman, then Burrows-Wheeler: A Simple Alphabet-Independent FM-Index. Proc. String Processing and Information Retrieval Conference (SPIRE), LNCS 3246, str. 210–211, Springer, 2004.

[24] **Grabowski Sz., Mäkinen V., Navarro G., Salinger A.:** A Simple Alphabet-Independent FM-index, Proc. Prague Stringology Conference (PSC), str. 230–244, Prague, Czechy, 2005.

[25] **Przywarski R., Grabowski Sz., Navarro G., Salinger A.:** FM-KZ: An even simpler alphabet-independent FM-index. Proc. Prague Stringology Conference (PSC), str. 226–241, Prague, Czechy, 2006.

[26] **Grabowski Sz., Navarro G., Przywarski R., Salinger A., Mäkinen V.:** A Simple Alphabet-independent FM-index, International Journal of Foundations of Computer Science, 17(6):1365–1384, 2006.

NEW ALGORITHMS FOR EXACT AND APPROXIMATE TEXT MATCHING

Abstract

This work presents main results in the domain of text algorithms obtained in Computer Engineering Dept. in the years 2004-2009. The algorithms concern various exact and approximate string matching problems, also in the recently actively developed scenario involving compression.

Politechnika Łódzka
Katedra Informatyki Stosowanej