Notatka z AiSD 28 czerwca 2022

Shift-And i KMR

IIUWr. II rok informatyki.

0.1 Algorytm Shift–AND

Idea algorytmu:

- W trakcie czytania tekstu pamiętamy informację o wszystkich prefiksach wzorca, które są sufiksami przeczytanego fragmentu tekstu.
- Algorytm ten przeznaczony jest do wyszukiwania krótkich wzorców, więc powyższa informacja może być przechowywana w jednym słowie maszynowym i w prosty sposób, kilkoma rozkazami, uaktualniana po wczytania kolejnego znaku.

Niech $C_j[0..m]$ będzie wektorem charakterystycznym zbioru prefiksów wzorca, które są sufiksami $t_1...t_j$, tj. $C_j[k] \equiv (P_k \sqsupset T_j)$.

Obserwacje:

O1. Wektor C_j można w prosty sposób wyznaczyć na podstawie wektora C_{j-1}, wzorca oraz j-tego znaku tekstu.

Mamy bowiem:

$$C_j[k] = \begin{cases} true & \text{dla } k = 0 \\ C_{j-1}[k-1] \wedge (p_k = t_j) & \text{dla } k > 0 \end{cases}$$

O2. Wzorzec występuje z przesunięciem $j - m$ wtedy i tylko wtedy, gdy $C_j[m] = true$.

Uwagi implementacyjne:

- Jeśli wzorzec jest krótki ($m <$ długość słowa maszynowego), do uaktualnienia wektora charakterystycznego możemy wykorzystać długie operacje logiczne. W tym celu dla każdej litery d alfabetu tworzymy wektor R_d taki, że $R_d[i] \equiv (p_i = d)$. Wówczas

  $$C_j = Shift(C_{j-1})\ AND\ R_{p_j},$$

  gdzie operacja $Shift$ oznacza przesunięcie w prawo o jeden bit z ustawieniem skrajnie lewego bitu na 1.
- Wystarczy pamiętać jeden (bieżący) wektor charakterystyczny zbioru prefiksów i uaktualniać go po każdym przeczytanym znaku.

1 Algorytm Karpa–Millera–Rosenberga (KMR)

Idea algorytmu:

- Niech $w = PT$, a więc w jest konkatenacją wzorca P i tekstu T.
- Numerujemy wszystkie podsłowa słowa w o długości m w jednoznaczny sposób, tj. taki, że takie same podsłowa otrzymują ten sam numer, a różne podsłowa - różne numery.
- Wypisujemy wszystkie pozycje większe od m, na których zaczynają się podsłowa o takim samym numerze co podsłowo zaczynające się na pozycji 1 (a więc wzorzec).

Numerowanie podsłów

- Do numerowania wykorzystujemy kolejne liczby naturalne. W ten sposób zawsze będziemy mieli do czynienia z numerami nie większymi od n (bo różnych podsłów danej długości jest nie więcej niż pozycji, na których mogą się one zaczynać).
- Startujemy od ponumerowania podsłów długości 1. W tym celu sortujemy w czasie liniowym litery występujące w słowie.
- Jeśli mamy ustaloną numerację słów długości k, możemy w prosty sposób znaleźć numerację podsłów długości k' dla dowolnego $k' \in \{k+1, \ldots, 2k\}$:
  - Dla każdego $i = 1, \ldots, |PT| - k'$ tworzymy parę $\langle nr_k(i), nr_k(i + k' - k + 1)\rangle$, gdzie $nr_s(j)$ jest numerem s-literowego podsłowa zaczynającego się od pozycji j (w obliczonej przez nas numeracji podsłów s-literowych).
  - Sortujemy leksykograficznie utworzone pary. Przeglądając ciąg par z lewa na prawo nadajemy im numery = " liczba różnych par na lewo".

PRZYKŁAD
Załóżmy, że ponumerowaliśmy podsłowa 2 literowe w słowie $w = bbaabbaaaabbaa$ w następujący sposób:

Pozycja	1	2	3	4	5	6	7	8	9	10	11	12	13
Podsłowo	*bb*	*ba*	*aa*	*ab*	*bb*	*ba*	*aa*	*aa*	*aa*	*ab*	*bb*	*ba*	*aa*
Numer	4	3	1	2	4	3	1	1	1	2	4	3	1

Tworząc numerację podsłów 4 literowych przypisujemy kolejnym pozycjom słowa w następujące pary:

Pozycja	1	2	3	4	5	6	7	8	9	10	11
Podsłowo	*bbaa*	*baab*	*aabb*	*abba*	*bbaa*	*baaa*	*aaaa*	*aaab*	*aabb*	*abba*	*bbaa*
Para	4,1	3,2	1,4	2,3	4,1	3,1	1,1	1,2	1,4	2,3	4,1

Po posortowaniu par otrzymujemy ciąg:

$$(1,1),\ (1,2),\ (1,4),\ (1,4),\ (2,3),\ (2,3),\ (3,1),\ (3,2),\ (4,1),\ (4,1),\ (4,1),$$

co umożliwia nam łatwe nadanie numerów parom:

Para	(1,1)	(1,2)	(1,4)	(2,3)	(3,1)	(3,2)	(4,1)
Numer	1	2	3	4	5	6	7

i przypisanie ich podsłowom z kolejnym pozycji słowa w:

Pozycja	1	2	3	4	5	6	7	8	9	10	11
Podsłowo	*bbaa*	*baab*	*aabb*	*abba*	*bbaa*	*baaa*	*aaaa*	*aaab*	*aabb*	*abba*	*bbaa*
Numer	7	6	3	4	7	5	1	2	3	4	3

□

Fakt 1 *Algorytm KMR działa w czasie $O(n \log n)$.*

DOWÓD: Chcąc znaleźć numerację słów m literowych wystarczy obliczyć numerację dla $\lceil \log m \rceil$ różnych długości. Obliczenie numeracji dla każdej z długości może być wykonane w czasie liniowym. □

Uwaga: Algorytm KMR może być zastosowany do wielu problemów związanych z wyszukiwaniem takich samych podsłów, w szczególności do problemu znajdowania najdłuższego powtarzającego się podsłowa.