Algorytmy i struktury danych

Instytut Sterowania i Systemów Informatycznych
Wydział Elektrotechniki, Informatyki i Telekomunikacji
Uniwersytet Zielonogórski

Algorytmy przeszukiwania wzorca

1 Wstęp

Algorytmy przeszukiwania wzorca w tekście służą do odnajdywania w zadanym tekście określonego ciągu znaków. Ciąg ten może oczywiście występować więcej, niż jeden raz. Możliwe jest również, że przeszukiwany ciąg nie występuje w tekście. W opisie algorytmów zakłada się, że tekst i wzorzec są jednowymiarowymi tablicami znaków, indeksowanymi od zera.

2 Algorytm naiwny — brute force

Algorytm ten jest najprostszą metodą na znalezienie wzorca w tekście. Zasada działania opiera się na porównywaniu odpowiednich liter tekstu i wzorca, zaczynając od pierwszej litery tekstu i wzorca. Jeśli fragmenty tekstu są zgodne, porównywany jest następny znak tekstu z następnym znakiem wzorca itd. Jeśli wystąpiła niezgodność w dowolnym miejscu — algorytm rozpoczyna całą procedurę przeszukiwania od drugiego znaku tekstu i pierwszego znaku wzorca. Jeśli zostanie znaleziony cały szukany wzorzec, algorytm rozpoczyna przeszukiwanie od następnego znaku. Innymi słowy, wzorzec jest zawsze „przesuwany" o jeden znak. Jest to podstawowa wada tego algorytmu. Przykład działania algorytmu po odnalezieniu wzorca (przeszukiwany tekst to BAABBAB, wzorzec: AAB)

				↓			
Tekst	B	**A**	**A**	**B**	B	A	B
Wzorzec		A	A	B			
				↑			

Następna iteracja algorytmu rozpoczyna się od kolejnego znaku w badanym tekście:

			↓				
Tekst	B	**A**	**A**	**B**	B	A	B
Wzorzec			A	A	B		
			↑				

Złożoność obliczeniowa w najgorszym przypadku wynosi O(N*M), gdzie N jest długością tekstu, M zaś — długością wzorca.

2.1 Przykład

Tekst do przeszukania ma postać: „AACBAAB", szukany wzorzec: „AAB". Przeszukiwanie rozpoczyna się od pierwszego znaku tekstu i pierwszego znaku wzorca.

	↓						
Tekst	A	A	C	B	A	A	B
Wzorzec	A	A	B				
	↑						

Ponieważ występuje zgodność, porównuje się następne znaki:

		↓					
Tekst	A	A	C	B	A	A	B
Wzorzec	A	A	B				
		↑					

Ponieważ występuje zgodność, porównuje się następne znaki:

			↓				
Tekst	A	A	C	B	A	A	B
Wzorzec	A	A	B				
			↑				

Ponieważ C≠ B, więc algorytm rozpoczyna się od początku, jednakże zaczynając od 2. litery tekstu:

		↓					
Tekst	A	A	C	B	A	A	B
Wzorzec		A	A	B			
		↑					

Ponieważ występuje zgodność, porównuje się następne znaki:

			↓				
Tekst	A	A	C	B	A	A	B
Wzorzec		A	A	B			
			↑				

Ponieważ C≠ A, więc algorytm rozpoczyna się od początku, jednakże zaczynając od 3. litery tekstu:

			↓				
Tekst	A	A	C	B	A	A	B
Wzorzec			A	A	B		
			↑				

Ponieważ C≠ A, więc algorytm rozpoczyna się od początku, jednakże zaczynając od 4. litery tekstu:

				↓			
Tekst	A	A	C	B	A	A	B
Wzorzec				A	A	B	
				↑			

Ponieważ B≠ A, więc algorytm rozpoczyna się od początku, jednakże zaczynając od 5. litery tekstu:

					↓		
Tekst	A	A	C	B	A	A	B
Wzorzec					A	A	B
					↑		

Ponieważ wszystkie 3 litery są zgodne, więc został odnaleziony poszukiwany ciąg. Algorytm kontynuuje działanie od miejsca $k+1$ w badanym tekście, gdzie k oznacza miejsce „startu" algorytmu, Ponieważ w opisywanym przykładzie liczba nieprzebadanych znaków tekstu jest mniejsza od długości przeszukiwanego tekstu, przeszukiwanie kończy się.

3 Algorytm KMP

Algorytm Knutha-Morrisa-Pratta, opublikowany w 1976 roku, stanowi rozwinięcie algorytmu naiwnego. Algorytm naiwny wykonuje wiele powtarzających się przeszukiwań, nie wykorzystując w przypadku niezgodności którejś z liter wzorca informacji wcześniej przebadanej (np. dla tekstu 'ABAAAABC' i wzorca 'ABAAAC' marnuje się przy pierwszej kolizji 6 porównań). W algorytmie KMP wykorzystywana jest informacja o możliwości wykonania przesunięcia wzorca o więcej, niż jeden znak.

3.1 Tworzenie tablicy przesunięć

Tablica przesunięć (o zwyczajowej nazwie *next*) wykorzystywana jest w algorytmie KMP do wyznaczania maksymalnych bezpiecznych przesunięć wzorca względem przeszukiwanego tekstu. Tworzona jest według następującego algorytmu:

1) $next[0] = -1;\ \ j = 1$
2) umieść kopię pierwszych j znaków wzorca pod fragmentem wzorca złożonym z pierwszych j liter wzorca, przesuniętych w prawo o jeden znak
3) przesuwaj tę kopię w prawo do chwili gdy:
   - wszystkie nakładające się znaki są identyczne, lub
   - nie ma nakładających się znaków
4) $next[j]$ jest równe liczbie nakładających się znaków
   $j = j + 1$
   jeśli j jest równe długości wzorca - stop,
   w przeciwnym przypadku skocz do punktu 2)

3.1.1 Przykład

Przyjmijmy, że mamy następujący wzorzec:
wzorzec=„BABAABBB",
zgodnie z krokiem 1) mamy, że: $next[0] = -1$, $j = 1$,
wykonując krok 2) otrzymujemy:

B							
	B						

Ponieważ nie ma nakładających się fragmentów wzorca, więc: $next[1] = 0$. Zgodnie z krokiem 4) algorytmu mamy: $j = 2$, więc

B	A						
	B	A					

(nakładające się fragmenty wzorca zaznaczono kolorem szarym). Ponieważ nie wszystkie nakładające się elementy wzorca są identyczne, więc konieczne jest przesunięcie dolnego fragmentu wzorca o jeden znak w prawo

B	A						
		B	A				

Ponieważ nie ma nakładających się fragmentów wzorca, więc: $next[2] = 0$.
Zgodnie z krokiem 4), $j = 3$, więc

B	A	B					
	B	A	B				

Ponieważ nie wszystkie nakładające się elementy wzorca są identyczne, więc konieczne jest przesunięcie dolnego fragmentu wzorca o jeden znak w prawo

B	A	B					
		B	A	B			

Ponieważ wszyskie nakładające się znaki są identyczne, więc $next[3]$ jest równe liczbie nakładających się znaków. W przykładzie będzie to: $next[3] = 1$. Postępując analogicznie dla pozostałych wartości $j = 4, 5, 6, 7$ otrzymuje się:

j	0	1	2	3	4	5	6	7
$next[j]$	-1	0	0	1	2	0	1	1

3.2 Algorytm KMP - wykorzystanie tablicy next

Celem prezentacji algorytmu konieczne jest wprowadzenie oznaczeń: $T[x]$ - przeszukiwany tekst, x - indeks znaku w tekście T, $x \geqslant 0$, $W[y]$ - szukany wzorzec o długości M, y - indeks znaku we wzorcu W, $y \geqslant 0$. Algorytm KMP możne zostać zapisany przy pomocy następującego pseudokodu:

1) $x = 0;\ y = 0$
2) Wyznaczyć tablicę przesunięć $next$ dla danego wzorca
3) Dopóki $T[x] = W[y]$ i nie przeszukano całego tekstu,
   zwiększ x i y o jeden
4) Jeśli przeszukano cały tekst, wtedy możliwe są dwa przypadki:
   Jeśli $y = M - 1$ to wzorzec został znaleziony na pozycji $x - M + 1$,
   w przeciwnym przypadku wzorzec nie został znaleziony
5) Ponieważ $T[x] <> W[y]$, więc należy skoczyć do 3) zmieniając y
   na wartość $next[y]$. Jeśli $y = -1$, należy zwiększyć x i y o jeden
   i również skoczyć do 3)

Krok 5) odpowiada przesunięciu wzorca tak, aby litera o indeksie $next[y]$ znalazła się pod znakiem o indeksie x. Wszystkie znaki wzorca na lewo od znaku x tekstu są zgodne z tekstem.

Algorytm KMP działa w czasie $O(M + N)$ w najgorszym przypadku.

3.2.1 Przykład

Znaleźć wzorzec „BABAABBB” w tekście: „BABABAABBABAABBB”.
Wykonanie algorytmu rozpoczyna się od ustawienia wartości początkowych: $x = 0$, $y = 0$. Następnie konieczne jest wyznaczenie tablicy *next*. Została ona wyznaczona w przykładzie 3.1.1.

indeks T	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x	↓															
tekst T	B	A	B	A	B	A	A	B	B	A	B	A	A	B	B	B
wzorzec W	B	A	B	A	A	B	B	B								
y	↑															
indeks W	0	1	2	3	4	5	6	7								
next	-1	0	0	1	2	0	1	1								

Ponieważ znaki $T[x]$ i $W[y]$ są zgodne, następuje przesunięcie x i y aż do momentu stwierdzenia niezgodności lub znalezienia całego wzorca.

W przykładzie niezgodność występuje na pozycji 4:

indeks T	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x					↓											
tekst T	B	A	B	A	B	A	A	B	B	A	B	A	A	B	B	B
wzorzec W	B	A	**B**	A	A	B	B	B								
y					↑											
indeks W	0	1	2	3	4	5	6	7								
next	-1	0	0	1	2	0	1	1								

Z tego powodu konieczne jest przesunięcie wzorca (poprzez zmianę y) tak, aby znak o indeksie next[4] (czyli 2), oznaczony pismem pogrubionym znalazł się pod znakiem o indeksie x (oznaczonym kolorem szarym):

indeks T	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x					↓											
tekst T	B	A	B	A	B	A	A	B	B	A	B	A	A	B	B	B
wzorzec W			B	A	**B**	A	A	B	B	B						
y					↑											
indeks W			0	1	2	3	4	5	6	7						
next			-1	0	0	1	2	0	1	1						

Ponieważ znaki tekstu o indeksach od 4 do 8 są zgodne z odpowiednimi znakami (od 2. do 6.) wzorca, więc następuje przesuwanie indeksów x i y zgodnie z krokiem 3). Różne znaki występują na pozycji 9:

indeks T	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x										↓						
tekst T	B	A	B	A	B	A	A	B	B	A	B	A	A	B	B	B
wzorzec W			B	**A**	B	A	A	B	B	B						
y										↑						
indeks W			0	1	2	3	4	5	6	7						
$next$			-1	0	0	1	2	0	1	1						

Z tego powodu konieczne jest przesunięcie wzorca (poprzez zmianę y) tak, aby znak o indeksie next[7] (czyli 1), oznaczony pismem pogrubionym znalazł się pod znakiem o indeksie x (oznaczonym kolorem szarym):

indeks T	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x										↓						
tekst T	B	A	B	A	B	A	A	B	B	A	B	A	A	B	B	B
wzorzec W									B	**A**	B	A	A	B	B	B
y										↑						
indeks W									0	1	2	3	4	5	6	7
next									-1	0	0	1	2	0	1	1

Ponieważ wszystkie pozostałe znaki są zgodne z wzorcem, więc na koniec x przyjmie wartość $x = 15$, długość wzorca wynosi $M = 8$, stąd też wzorzec został odnaleziony na pozycji $x - M + 1 = 15 - 8 + 1 = 8$. Zwróć uwagę, że tablice tekstu i wzorca są indeksowane od wartości 0.

4 Algorytm Boyera i Moore'a

W algorytmie tym w przeciwieństwie do algorytmu KMP porównywany jest ostatni znak wzorca. Jeśli badany znak o (indeksie x) tekstu nie wchodzi w skład wzorca, możliwe jest przesunięcie indeksu x o długość wzorca. Jeśli znak ten występuje - konieczne jest przesunięcie wzorca tak, aby znak ten znalazł się pod odpowiadającym mu znakiem wzorca.

4.1 Tablica przesunięć shift

Celem wyznaczenia optymalnego przesunięcia, konieczne jest wyznaczenie na podstawie wzorca tablicy przesunięć, zwanej zwyczajowo „$shift$". Tablica ta, o rozmiarze równym liczbie wszystkich możliwych znaków, które mogą wystąpić w tekście przeszukiwanym i we wzorcu, zawiera wartość zero w komórce odpowiadającej indeksowi ostatniego znaku wzorca, 1 - przedostatniego itd. Jeśli określony znak nie występuje we wzorcu, tablica shift przyjmuje wartość równą długości wzorca. Przy wyznaczaniu tej tablicy zakłada się ponadto, że jeśli określony znak istnieje we wzorcu więcej, niż jeden raz, pod uwagę bierze się tylko ostatnią pozycję (dążąc tym samym do minimalizacji wartości elementów tablicy $shift$).

4.1.1 Przykład

Niech wzorzec=„ABGBD", zaś wszystkie możliwe znaki tekstu i wzorca to „ABCDEFGH". Znaki we wzorcu numeruje się od końca:

A	B	G	B	D
4	3	2	1	0

Długość wzorca wynosi 5. Wartości tablicy $shift$ wynoszą więc:

k	shift[k]	komentarz
'A'	4	jest na pozycji 4, licząc od końca
'B'	1	wynosi minimum$\{1, 3\} = 1$
'C'	5	nie występuje we wzorcu, więc wartość równa długości wzorca, czyli 5
'D'	0	na ostatniej pozycji wzorca, więc zero
'E'	5	nie występuje we wzorcu, więc wartość równa długości wzorca, czyli 5
'F'	5	nie występuje we wzorcu, więc wartość równa długości wzorca, czyli 5
'G'	2	jest na pozycji 2, licząc od końca
'H'	5	nie występuje we wzorcu, więc wartość równa długości wzorca, czyli 5

Przy wyznaczaniu elementów tablicy shift, należy pamiętać, iż najmniejszą jej wartością jest zero, na pozycji odpowiadającej ostatniemu znakowi wzorca.

4.2 Algorytm BM

Założenia:
M — długość wzorca W,
N — długość tekstu T.
Algorytm Boyera — Moore'a można przedstawić następująco (należy pamiętać, że tablice znaków indeksowane są od zera):

1) Wyznaczyć tablicę przesunięć $shift$ dla szukanego wzorca
2) Jeśli długość wzorca > długość tekstu - stop, nie znaleziono
3) Rozpocząć przeszukiwanie od ostatniej litery wzorca
   i odpowiadającej jej literze tekstu: $i = M - 1,\ j = M - 1$
4) Dopóki $j > 0$ wykonuj:
   5) Dopóki i-ty znak tekstu oraz j-ty znak wzorca różnią się, wykonuj:
      4a) Używając tablicy $shift$ wyznacz przesunięcie x odpowiadające znakowi $T[i]$
      4b) Jeśli $M - j > x$, zwiększ i o wartość $M - j$; w przeciwnym przypadku - zwiększ i o wartość przesunięcia x
      4c) Jeśli i jest większe od długości tekstu - stop, wzorca nie odnaleziono
      4d) Ustaw j na ostatni znak wzorca
   6) Zmniejsz i oraz j o jeden
7) Odnaleziono wzorzec na pozycji i

4.2.1 Przykład

Odnaleźć wzorzec „ABGBD" w tekście „ABGHHABGBDEH". Tablica $shift$ została wyznaczona w punkcie 4.1.1. W przykładzie ponadto $M = 5, N = 12$. Ponieważ $M \leqslant N$, więc poszukiwanie może się rozpocząć.

Przeszukiwanie rozpoczyna się od ostatniego znaku wzorca i odpowiadającego znaku tekstu: $i = 4$, $j = 4$.

indeks T	0	1	2	3	4	5	6	7	8	9	10	11
i					↓							
tekst T	A	B	G	H	H	A	B	G	B	D	E	H
wzorzec W	A	B	G	B	D							
j					↑							
indeks W	0	1	2	3	4							

Ponieważ $j > 0$, oraz znaki H i D różnią się, więc należy wyznaczyć przesunięcie $x = shift['H'] = 5$. Ponieważ warunek $5 - 4 > 5$ nie jest spełniony, $i = 4 + 5 = 9$:

indeks T	0	1	2	3	4	5	6	7	8	9	10	11
i										↓		
tekst T	A	B	G	H	H	A	B	G	B	D	E	H
wzorzec W						A	B	G	B	D		
j										↑		
indeks W						0	1	2	3	4		

Ponieważ i-ty i j-ty znak tekstu są identyczne, więc należy zbadać poprzedni znak, zmniejszając je o 1:

indeks T	0	1	2	3	4	5	6	7	8	9	10	11
i									↓			
tekst T	A	B	G	H	H	A	B	G	B	D	E	H
wzorzec W						A	B	G	B	D		
j									↑			
indeks W						0	1	2	3	4		

Postępując analogicznie okaże się, że wzorzec został znaleziony.

5 Algorytm Rabina i Karpa

Algorytm Rabina i Karpa opiera się na porównywaniu ze sobę całości wzorca na raz i odpowiedniego fragmentu tekstu, po zamianie ich na wartości liczbowe przy użyciu odpowiedniej funkcji H. Poprawnie skonstruowana funkcja H powinna umożliwić wykorzystanie wartości z obliczeń w poprzednim kroku algorytmu. Wyznaczenie wartości H_w funkcji H dla wzorca wykonywane jest jednorazowo, na początku algorytmu. Wartość H_t, czyli wartość funkcji H dla określonego fragmentu tekstu wyznaczana jest w każdym kroku algorytmu.

5.1 Wybór funkcji H

Poszukiwanie wzorca w algorytmie RK polega na zamianie wzorca i fragmentu tekstu na liczbę. Oznacza to, że ciąg M znaków w kroku i będzie interpretowany jako pewna liczba x_i,

najlepiej całkowita (szybkość porównywania). Przyjmując za b — jako podstawę systemu — liczbę wszystkich możliwych znaków, otrzymuje się:

$$x_i = t[i]b^{M-1} + t[i+1]b^{M-2} + \ldots + t[i+M-1]$$

W kroku $i+1$ otrzymuje się

$$x_{i+1} = t[i+1]b^{M-1} + t[i+2]b^{M-2} + \ldots + t[i+M]$$

Funkcja H powinna umożliwiać wykorzystanie obliczeń wykonanych w kroku poprzednim do wyznaczenia wartości w kroku bieżącym. Zadanie polega więc na wyznaczeniu wartości x_{i+1} przy wykorzystaniu wartości x_i. Wartość ta wynosi:

$$x_{i+1} = (x_i - t[i]b^{M-1}) + t[i+M]$$

Jako funkcji H można wykorzystać funkcję $H(x) = x \ mod \ p$, gdzie mod oznacza resztę z dzielenia x przez p. Wartość p powinna być dużą liczbą pierwszą. Funkcja H może więc być więc wyznaczona jako

$$H(x_{i+1}) = \Big[(x_i - t[i]b^{M-1}) + t[i+M]\Big] \ mod \ p$$

Z własności operatora mod wynika, że zamiast wyznaczać resztę z dzielenia dla całego wyrażenia, możliwe jest jej wyznaczenie po każdej operacji cząstkowej, przenosząc wynik do następnej. Przykład: $(5*100+6*10+8) \ mod \ 7 = 568 \ mod \ 7 = 1$, co jest równoważne: $5*100 \ mod \ 7 = 3$, $(3+6*10) \ mod \ 7 = 0$, $(0+8) \ mod \ 7 = 1$.

Jako wartość b wskazane jest wybranie liczby będącej potęgą liczby 2 ze względu na możliwość implementacji mnożenia jako operacji przesunięcia bitowego.

5.2 Algorytm RK

Wartość H_w funkcji H dla wzorca można więc wyznaczyć przy pomocy pętli, gdzie i przebiega każdy znak wzorca, jako:

$Hw \ = \ 0;$
`dla` i `przyjmującej wartość od 0 do liczby znaków wzorca -1 wykonuj:`
    $Hw \ = \ (Hw * b + (i - ty \ \ znak \ \ wzorca))) \ \ mod \ \ p;$

Analogicznie wyznacza się wartość H_t tej funkcji dla fragmentu tekstu:

$Ht \ = \ 0;$
`dla` i `przyjmującej wartość od 0 do liczby znaków wzorca -1 wykonuj:`
    $Ht \ = \ (Ht + b * p - (i - ty \ \ znak \ \ tekstu)) * b\hat{}M - 1) \ \ mod \ \ p;$
    $Ht \ = \ (Ht * b + (i + M - ty \ \ znak \ \ tekstu)) \ \ mod \ \ p;$

Dodawanie wartości $b*p$ nie zmienia wartości wyrażenia (z uwagi na operator mod), pozwala uchronić się przed „wskoczeniem" w liczby ujemne.

Algorytm RK polega na wyznaczaniu wartości funkcji H_t i H_w dla początkowego fragmentu tekstu. Następnie należy porównać te wartości. Jeśli są równe - znaleziono wzorzec. W przeciwnym przypadku — należy wyznaczyć wartość funkcji H dla znaków od 2 do $M+1$ i porównać je z H_w. Czynności te należy powtarzać, używając odpowiednich znaków tekstu, aż do znalezienia wzorca lub zbadania całego tekstu.

Literatura

[1] P. Wróblewski. Algorytmy, struktury danych i techniki programowania, Helion, 1996.
[2] L. Banachowski, K. Diks, W. Rytter. Algorytmy i struktury danych, WNT 1996.
[3] T. H. Cormen, Ch. E. Leiserson, R. L. Rivest. Wprowadzenie do algorytmów, WNT 1997.