

Algorytmy tekstowe

Algorytm wyszukiwania wzorca

Algorytm Knutha-Morrisa-Pratta wyszukiwania wzorca

Źródło: <https://zpe.gov.pl/a/przeczytaj/DcT6FBT2u>

Algorytm ten jest pierwszym odkrytym algorytmem, który rozwiązuje problem wyszukiwania wzorca w tekście, cechuje się złożonością liniową względem długości tekstu oraz wzorca. Jego nazwa pochodzi od trzech matematyków-informatyków, którzy wspólnie opublikowali ostateczną wersję algorytmu: **Donald Ervin Knuth**, **Vaughan Ronald Pratt** oraz **James Hiram Morris**. W skrócie algorytm nazywamy „KMP”.

Algorytm KMP sprawdza po kolei każdy znak tekstu, do którego możemy dopasować wzorec. W momencie, gdy pierwszy znak wzorca oraz aktualnie przetwarzany znak tekstu są takie same, rozpoczynamy sprawdzanie po kolei, czy następne znaki wzorca i tekstu są identyczne. Na pierwszy rzut oka metoda ta działa w taki sam sposób, jak metoda naiwna. Usprawnienie pojawia się jednak w momencie, gdy już znaleźliśmy potencjalne dopasowanie, które okazuje się błędne. W takiej sytuacji algorytm KMP pomija sprawdzanie dopasowania na tych następnych pozycjach, które na pewno okażą się błędne.

Pomocna w tym działaniu jest analiza wstępna wzorca, która polega na utworzeniu tablicy częściowych dopasowań. Przechowuje ona liczby całkowite. Długość takiej tablicy to długość wzorca powiększona o 1. Aby najłatwiej wytłumaczyć proces analizy wzorca w algorytmie KMP, posłużymy się wcześniejszą jego wersją (znaną jako algorytm MP), a następnie rozszerzymy ją o pewien dodatkowy przypadek, dzięki któremu powstała jego obecna wersja.

Chcąc dokładnie omówić tę procedurę, musimy wprowadzić kilka pojęć:

1. **prefiks** – składająca się z k znaków przednia część łańcucha znaków;
2. **sufiks** – składająca się z k znaków końcowa część łańcucha znaków;
3. **prefikso-sufiks** – składająca się z k znaków część łańcucha znaków, która występuje zarówno z przodu, jak i z tyłu;
4. **szerokość prefikso-sufiksu** – długość prefiksu lub sufiksu, z którego składa się prefikso-sufiks.

Budowa tablicy częściowych dopasowań w algorytmie MP polega na wyznaczeniu maksymalnego prefikso-sufiksu dla każdego możliwego prefiksu we wzorcu.

Oznaczmy ciąg znaków wzorec jako W oraz tablicę częściowych dopasowań jako T.

Jak odnaleźć wzorec w tekście?

Rozważmy następujący problem. Mamy ciąg znaków: jedno słowo (np. „kot”) lub kilka wyrazów – np. zdanie „ala ma kota” (dla uproszczenia pomijamy wielkości liter). Problem wyszukania wzorca w tekście polega na odnalezieniu takiej liczby, która określa, ile

początkowych znaków tekstu należy usunąć, aby rozpoczynał się on właśnie od wzorca. W przytoczonym przykładzie taką cyfrą będzie 7. Po usunięciu z ciągu „ala ma kota” 7 początkowych znaków, zostaniemy z wyrażeniem „kota”. Wówczas łatwo zauważyć, że wzorzec „kot” pasuje do początku zdania.

W jaki sposób odbywa się szukanie wzorca w tekście? Pomocna jest tzw. **metoda naiwna**, określana często mianem brute force. Polega ona na wykonywaniu porównania wzorca z początkiem tekstu dla każdej liczby znaków, które z niego usuniemy. Takie wyszukiwanie w podanym przykładzie będzie wyglądało następująco:

	a	l	a		m	a		k	o	t	a
1.								k	o	t	
2.									k	o	t
3.										k	o
4.											k
5.											
6.											
7.											
8.											
9.											
10.											

Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Podstawową wadą tej metody jest to, że nie jest ona optymalna. Nie nadaje się do poszukiwania długich wzorców w długich tekstach. Istnieją algorytmy, które dużo efektywniej radzą sobie w podobnych sytuacjach. Jednym z takich algorytmów jest **algorytm Knutha-Morrisa-Pratta**.

Algorytm Knutha-Morrisa-Pratta wyszukiwania wzorca

Specyfikacja problemu:

Dane:

- t – łańcuch znakowy
- w – poszukiwany w łańcuchu t wzorzec

Wynik:

Wypisanie kolejnych wystąpień wzorca w łańcuchu.

Algorytm ten jest pierwszym odkrytym algorytmem, który rozwiązuje problem wyszukiwania wzorca w tekście, cechuje się złożonością liniową względem długości tekstu oraz wzorca. Jego nazwa pochodzi od trzech matematyków-informatyków, którzy wspólnie opublikowali ostateczną wersję algorytmu: **Donald Ervin Knuth**, **Vaughan Ronald Pratt** oraz **James Hiram Morris**. W skrócie algorytm nazywamy „**KMP**”.

Algorytm KMP sprawdza po kolei każdy znak tekstu, do którego możemy dopasować wzorzec. W momencie, gdy pierwszy znak wzorca oraz aktualnie przetwarzany znak tekstu są takie same, rozpoczynamy sprawdzanie po kolei, czy następne znaki wzorca i tekstu są identyczne. Na pierwszy rzut oka metoda ta działa w taki sam sposób, jak metoda naiwna. Usprawnienie pojawia się jednak w momencie, gdy już znaleźliśmy potencjalne dopasowanie, które okazuje się błędne. W takiej sytuacji algorytm KMP pomija sprawdzanie dopasowania na tych następnych pozycjach, które na pewno okażą się błędne.

Pomocna w tym działaniu jest analiza wstępna wzorca, która polega na utworzeniu tablicy częściowych dopasowań. Przechowuje ona liczby całkowite. Długość takiej tablicy to długość wzorca powiększona o 1. Aby najłatwiej wytłumaczyć proces analizy wzorca w algorytmie KMP, posłużymy się wcześniejszą jego wersją (znaną jako algorytm MP), a następnie rozszerzymy ją o pewien dodatkowy przypadek, dzięki któremu powstała jego obecna wersja.

Chcąc dokładnie omówić tę procedurę, musimy wprowadzić kilka pojęć:

1. **prefiks** – składająca się z k znaków przednia część łańcucha znaków;
2. **sufiks** – składająca się z k znaków końcowa część łańcucha znaków;
3. **prefikso-sufiks** – składająca się z k znaków część łańcucha znaków, która występuje zarówno z przodu, jak i z tyłu;
4. **szerokość prefikso-sufiksu** – długość prefiksu lub sufiksu, z którego składa się prefikso-sufiks.

Budowa tablicy częściowych dopasowań w algorytmie MP polega na wyznaczeniu maksymalnego prefikso-sufiksu dla każdego możliwego prefiksu we wzorcu.

Ciekawostka

Algorytmy wyszukiwania wzorca w tekście stosowane są często w genetyce. Przykład, którym się posłużymy, będzie operował na znakach A, C, G, T, czyli na symbolach oznaczających zasady azotowe, z których zbudowane jest DNA. Więcej na temat DNA przeczytasz w e-materiale [DNA jako nośnik informacji genetycznej](#)Otwiera się w nowym oknie.

Oznaczmy ciąg znaków wzorzec jako W oraz tablicę częściowych dopasowań jako T.

Krok 0:

Na zerowej pozycji w tablicy T wstawimy liczbę -1, która będzie naszym [wartownikiem](#).

W		G	C	A	T	G	C	G	A	G	C
T	-1										

Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Krok 1:

Pierwszy możliwy prefiks wzorca to „G”. Wśród ciągu znaków „G” nie występuje żaden prefikso-sufiks, zatem kolejny element w tablicy T to 0 – prefikso-sufiks pusty.

W		G	C	A	T	G	C	G	A	G	C
T	-1	0									

Krok 2:

Kolejny prefiks wzorca to „GC”. Nie występuje żaden prefikso-sufiks, zatem jako kolejny element tablicy również wstawiamy 0 – prefikso-sufiks pusty.

W		G	C	A	T	G	C	G	A	G	C
T	-1	0	0								

Krok 3:

Kolejny prefiks to „GCA”. Znowu występuje jedynie prefikso-sufiks pusty, więc do tablicy wstawiamy 0.

W		G	C	A	T	G	C	G	A	G	C
T	-1	0	0	0							

Krok 4:

Sytuacja jak poprzednio – wstawiamy 0.

W		G	C	A	T	G	C	G	A	G	C
T	-1	0	0	0	0						

Krok 5:

W aktualnie przeszukiwanym fiksie „GCATG” znaleźliśmy pierwszy prefikso-sufiks. Jego szerokość wynosi 1, zatem w tablicy umieszczamy element 1.

W		G	C	A	T	G	C	G	A	G	C
T	-1	0	0	0	0	1					

Krok 6:

W kolejnym prefiksie największy prefikso-sufiks, jaki znajdziemy, jest o szerokości 2. Zauważmy, że nie musimy w tym kroku przeszukiwać całego prefiksu „GCATGC”, gdyż możemy kontynuować wyszukiwanie, które rozpoczęliśmy w poprzednim kroku.

W		G	C	A	T	G	C	G	A	G	C
T	-1	0	0	0	0	1	2				

Krok 7:

W tym kroku największy prefikso-sufiks ma szerokość 1. Pamiętając o informacji z poprzedniego kroku, sprawdź czy uda się rozszerzyć poprzednio znaleziony prefikso-sufiks. Dopiero gdy próba się nie powiedzie, sprawdzamy, czy możemy znaleźć inny prefikso-sufiks. Zatem w tym kroku są wykonywane tak naprawdę 2 kroki.

W		G	C	A	T	G	C	G	A	G	C
T	-1	0	0	0	0	1	2	1			

Krok 8:

W prefiksie „GCATGCGA” nie ma żadnego prefikso-sufiksu. Również w tym kroku próbowaliśmy rozszerzyć poprzedni prefikso-sufiks.

W		G	C	A	T	G	C	G	A	G	C
T	-1	0	0	0	0	1	2	1	0		

Krok 9:

Znaleźliśmy prefikso-sufiks o długości 1 – w tablicy zapisujemy tę wartość.

W		G	C	A	T	G	C	G	A	G	C
T	-1	0	0	0	0	1	2	1	0	1	

Krok 10:

Możemy rozszerzyć poprzednio znaleziony prefikso-sufiks. Do tablicy wpisujemy wartość o 1 większą niż w poprzednim kroku.

W		G	C	A	T	G	C	G	A	G	C
T	-1	0	0	0	0	1	2	1	0	1	2

Wszystkie powyżej opisane kroki możemy przedstawić za pomocą poniższego pseudokodu. Jako w oznaczono wzorzec:

```
1 funkcja budowaTablicyMP(W)
2   rozmiarT = długość(W) + 1
3   stwórz tablicę T o długości rozmiarT
4   poz = -1
5   T[0] = -1
6
7   // dla każdego kolejnego prefiksu wzorca
8   dla i = 1, 2, ..., rozmiarT - 1 wykonuj:
9       // dopóki możesz znaleźć dłuższy prefikso-sufiks
10      dopóki poz > -1 i W[poz] != W[i - 1] wykonuj:
11          poz = T[poz]
12      poz += 1
13
14      // zapisz szerokość znalezionej prefikso-sufiksu
15      T[i] = poz
16
17  zwróć T
```

Zadania

Problem 1 Lekarzom udało się wyodrębnić łańcuch DNA odpowiedzialny za pewną mutację. W ramach badań zebrano grupę chętnych pacjentów, którzy poddali się badaniom genetycznym. Na podstawie fragmentów sekwencji DNA wskaż, czy wśród pacjentów są osoby ze wskazaną mutacją genową.

Dane:

GAGTTGTGCCAACCCTCGTCCTCACCGAAGCTTGCTGCCAATGATTAGGA

– pacjent nr 1

TCATTGCCTTGCGACAGACCTCCCACTCACACTCGCTCGCATTGAGCTAC

– pacjent nr 2

TCGATGGGCCATCAGCTTGACCCGCTCTGTAGGGTCGCGATTACGTGAGT

– pacjent nr 3

CGCCTTCT – mutacja