# Algorytm Knutha-Morrisa-Pratta

Spis treści

[Opis struktury 4](#_Toc408762615)

[Opis zadania 5](#_Toc408762616)

[Plik wejściowy 6](#_Toc408762617)

[Plik wyjściowy 7](#_Toc408762618)

[Uwagi 8](#_Toc408762619)

Paweł Borawski

# Opis algorytmu

Powiemy, że słowo *u*[1..*k*] jest prefiksem słowa *w*[1..*l*], jeśli

*u*[1+*j*]=*w*[1+*j*]dla*j*=0,1,…,*k*−1

tzn. słowo *w* zaczyna się słowem *u*, np. jest tak dla *u*=ABCAA, *w*=ABCAABBAABAABAAB.

Powiemy, że słowo *u*[1..*k*] jest sufiksem słowa *w*[1..*l*], jeśli

*u*[1+*j*]=*w*[*l*−*k*+1+*j*]dla*j*=0,1,…,*k*−1

tzn. słowo w kończy się słowem *u*, np. jest tak dla *u*=ABAAB, *w*=ABCAABBAABAABAAB.

Wreszcie powiemy, że słowo *u*[1..*k*] jest prefikso-sufiksem słowa *w*[1..*l*], jeśli  
słowo u jest zarówno jego prefiksem i sufiksem. Na przykład słowo *u*=AB jest prefikso-sufiksem słowa *w*=ABCAABBAABAABAAB.

Okazuje się, że przy wyszukiwaniu wzorca w tekście, bardzo przydatna jest znajomość najdłuższego prefikso-sufiksu całego wzorca i nie tylko. Oczywiście, każde słowo jest swoim prefiksem i sufiksem, zatem jest również prefikso-sufiksem. Dalej będą nas jednak interesowały właściwe prefikso-sufuksy, tj. takie, które nie są całym słowem.  
Dalej będziemy naywali je po prostu ramkami. Na przykład, najdłuższą ramką słowa  
Pattern.png  
jest słowo ABA. Oczywiście istnieje również druga ramka — słowo A, ale jest ona krótsza.

## Tablica prefikso-sufiksowa

Naszym celem jest wyznaczenie długości najdłuższej ramki dla każdego prefiksu wzorca *P*.  
Na przykład dla słowa ABAABAAABAAB, jego kolejne prefiksy to:

A  
AB  
ABA  
ABAA  
ABAAB  
ABAABA  
ABAABAA  
ABAABAAA  
ABAABAAAB  
ABAABAAABA  
ABAABAAABAA  
ABAABAAABAAB

a ich najdłuższe ramki mają długość odpowiednio:

|  |  |  |  |
| --- | --- | --- | --- |
| *j***(długość prefiksu)** | **Prefiks** | **Najdłuższa ramka prefiksu** | *π*[*j*]**(długość ramki)** |
| 1 | A | brak | 0 |
| 2 | AB | brak | 0 |
| 3 | ABA | A | 1 |
| 4 | ABAA | A | 1 |
| 5 | ABAAB | AB | 2 |
| 6 | ABAABA | ABA | 3 |
| 7 | ABAABAA | ABAA | 4 |
| 8 | ABAABAAA | A | 1 |
| 9 | ABAABAAAB | AB | 2 |
| 10 | ABAABAAABA | ABA | 3 |
| 11 | ABAABAAABAA | ABAA | 4 |
| 12 | ABAABAAABAAB | ABAAB | 5 |

## Obliczanie tablicy

Przejdźmy więc do opisu metody obliczania tablicy *π*[1..*m*] — tablicy długości najdłuższych właściwych prefikso-sufiksów dla wzorca *P*[1..*m*].  
Po pierwsze tablicę obliczamy dla kolejnych wartości *j*=1,2,…,*m*.

Oczywiście, zaczynamy od wartości

*π*[1]=0,

gdyż słowo jednoliterowe nie posiada żadnej ramki. Można myśleć, że posiada ramkę będącą słowem pustym o długości 0.  
Następnie przypuśćmy, że mamy już obliczone wartości *π*[1],*π*[2],…,*π*[*j*−1] i chcemy obliczyć *π*[*j*].  
Oznaczmy *t*=*π*[*j*−1], tzn. niech t oznacza długość poprzedniej ramki. Zauważmy, że

jeśli *P*[*t*+1]=*P*[*j*], to *π*[*j*]=*t*+1

Innymi słowy, jeśli nowa litera *P*[*j*] przedłuża poprzednią ramkę (o długości *t*=*π*[*j*−1]), to natychmiast otrzymujemy najdłuższą ramkę dla prefiksu j literowego. Dzieje się tak na przykład dla podanego wcześniej słowa *P* i dla *j*=3,5,6,7,9,10,11,12. Jeśli zaś tak nie jest, tzn. zachodzi *P*[*t*+1]≠*P*[*j*], to okazuje się, że wystarczy popatrzeć na najdłuższą ramkę poprzedniej ramki, a więc spróbować przedłużyć ramkę o długości *π*[*t*]

## Algorytm Morrisa-Pratta

Obliczona wcześniej funkcja π nazywana jest w literaturze funkcją **Knutha-Morrisa-Prata**. Poniższy algorytm, korzysta z tej tablicy dość istotnie i nazywa się algorytmem **Morrisa-Pratta**.  
Poniższy algorytm wypisuje wszystkie pozycje, na których występuje wzorzec *P*[1..*m*] w tekście *T*[1..*n*]. Algorytm korzysta z obliczonej wcześniej tablicy *π*[1..*m*].

# 

# Opis zadania

Moim zadaniem było zaimplementowanie wyszukiwania wzorca w tekście za pomocą Algorytmu Knutha-Morrisa-Pratta.

# Plik wejściowy

Format pliku wejściowego:

Tekst tekst test

======================  
tekst

Gdzie: Wszystkie linie do „======================” oznaczają tekst, w którym będziemy szukać wzorca. Linie po „======================” oznaczają wzorzec którego szukamy.

# Plik wyjściowy

Wynik:

Tekst znaleziono w następujacych linijkach:

test TEST TEST

# Uwagi

Brak