Definicja:

Prefiksosufiksem słowa $s[0 \dots n-1]$ nazywamy taki fragment $s[0 \dots k-1]$, że $s[0 \dots k-1] = s[i-(k-1) \dots n-1]$.

Jeśli $k \neq n$ to taki prefiksosufiks nazywamy właściwym.

Przykład:

Niech $s =$ "abracaaabrac".

Wtedy "abrac" jest prefiksosufiksem, bo "abracaaabrac" (właściwym)

Definicja:

Niech $s[0 \dots n-1]$ – słowo.

Niech p będzie tablicą o długości n oraz niech:

$p[i]$ = długość najdłuższego prefiksosufiksu właściwego w słowie $s[0 \dots i]$

$p[0] = 0$.

Przykład:

$s =$ "abaaba" $\Rightarrow$ $p: 0, 0, 1, 1, 2, 3$

$s =$ "aabaaab" $\Rightarrow$ $p: 0, 1, 0, 1, 2, 2, 3$

Naszym zadaniem na dzisiejszy wykład jest wyznaczenie tablicy p w czasie $O(n)$.

Zacznijmy od naiwnego rozwiązania:

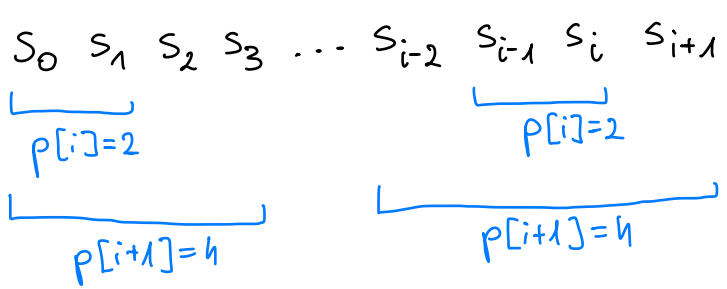
Dla każdego indeksu w s sprawdzamy wszystkie możliwe prefiksy $s[0 \dots i]$ i sprawdzamy, czy są one prefiksosufiksami właściwymi.

To rozwiązanie działa w czasie $O(n^3)$.

Obserwacje 1)

↳ wartości w p mogą rosnąć co najwyżej o 1.

Załóżmy przeciwnie:



Niech $p[i]=2$, $p[i+1]=4$,

$$\Downarrow$$

$$s[0 \dots 4-1] = s[i-2 \dots s_{i+1}]$$

więc w szczególności $s[0 \dots 2] = s[i-2 \dots i]$

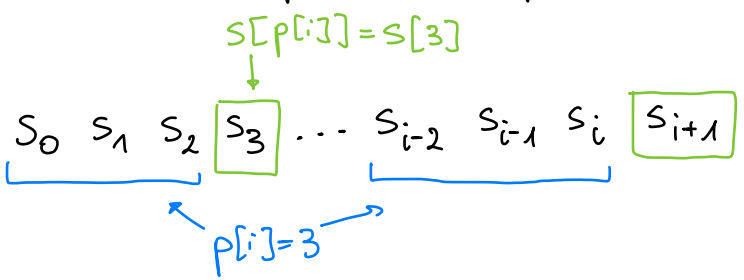
a zatem $s[0 \dots 2]$ jest wł. prefiksosufiksem w słowie $s[0 \dots i]$ więc $p[i]$ powinno się również 3. Sprzeczność.

Obserwacja 2)

↳ możemy skorzystać z już wyznaczonych wartości p to obliczenie kolejnych.

Załóżmy, że do i włącznie mamy wyliczone $p[i]$. Chcemy znaleźć $p[i+1]$.

1) Jeśli $s[i+1] = s[p[i]]$ to $p[i+1] = p[i]+1$



2) W przeciwnym wypadku:

↳ musimy spróbować 1) z innym prefiksosufiksem, który jest krótszy <=> jest zawarty w prefiksosufiksie o długości $p[i]$. Nie chcemy ominąć żadnego, zatem zależy nam na najdłuższym.

Jest to dokładnie definicja $p[p[i]-1]$.

Może się zdarzyć, że żaden z prefiksosufiksów nie spełni 1). Wtedy ostatnią nadzieją będzie czy $s[0] = s[i+1]$, jeśli i to zawiedzie, to $p[i+1] = 0$

Kod:

```
int n = s.size();
p[0] = 0; // wczesniej zainicjalizowalem tablice
for (int i=0; i<n-1; i++) {
    int j = p[i];
    while (j > 0 && s[j] != s[i+1]) {
        j = p[j-1];
    }
    if (s[j] == s[i+1]) {
        j++;
    }
    p[i+1] = j;
}
```

Dlaczego to dzieje w $O(n)$?

Ponieważ każdy $s[i]$ jest ewaluowany co najwyżej 2 razy:

1) raz jako kandydat na przedłużenie prefiksosufiksu
2) raz jako element do którego wracamy jeśli inne przedłużenie zawiodły

W i-tym kroku $p[i+1]$ może wzrosnąć co najwyżej o 1, nie zmienić wartości lub spaść nawet do zera.

Sumarycznie dodamy 1 co najwyżej n razy, a zatem sumarycznie możemy zmniejszyć o co najwyżej n. (bo $p[i]$ nie-ujemne)

Zadanie:

Niech $s[0 \ldots m-1]$ – wzorzec, $t[0 \ldots n-1]$ – tekst, $m \leq n$

Chcemy powiedzieć ile razy w t wystąpił s.

Rozwiązanie: [Algorytm KMP: Knuth-Morris-Pratt]

Rozpatrzmy nowe słowo postaci:

$$s_0 \, s_1 \ldots s_{m-1} \, \# \, t_0 \, t_1 \cdots t_{n-1}$$

, gdzie # – znak, który nie występuje ani w s, ani w t

Dla tak stworzonego słowa liczymy tablicę p i za każdym razem, gdy $i > m+1$ oraz $p[i] = m$ mamy wystąpienie wzorca w tekście.

Źródło: cp-algorithms.com