78)

Wystarczy konkatenacja jednego z ~~dwóch~~ tekstów ze sobą i wystąpienie drugiego w ~~tym~~ konkatenacji.

a, b

$a' = a + a$

komp: $a' \# b$

77)

Oznaczmy sobie słowa we wzorze pamięciowy '◇' jako $a_1, a_2, \ldots, a_n$.

Wzorzec występuje w tekście jeśli znajdziemy niemechodzące na siebie wystąpienia wszystkich a_i w poprawnej kolejności.

~~Niech~~ Niech długość tekstu to n. Weźmy uniwersalną rodzinę funkcji haszujących w $\{0, \ldots, m-1\}$ tak, że $m > n$.

Mamy teraz policzyć hasze dla poszczególnych a_i.

Algorytm:

$i = 1$, $j = 0$

1) Oblicz h = hasz (tekst[j: len(a_i)])

2) dopóki h != hasz (a_i):

j++

3) i++, tekst = tekst[j:], go to 1)

Wersja z KMP:

e_1 # tekst

↓

e_2 # tekst [j:] , gdzie j pierwsze wystąpienie wzorca e_1

czyli: $\Pi[j] = |e_1|$

itd...

Sumarycznie O(n)

Z11

Wszystkie operacje Union wykonują się w czasie stałym.

Następnie, gdy wykonujemy operację find przechodzimy ścieżką do korzenia i wykonujemy kompresję.

Zatem sumarycznie każdą krawędź odwiedzimy maksymalnie raz, ponieważ przy kolejnym wywołaniu na wierzchołku, który leży na ścieżce, którą już szliśmy wcześniej odpowiedź uzyskujemy w O(1).