Słowa, słowa, słowa.

12 czerwca 2018

1 Przydatny kod

Kod	Opis

`#include<iostream> // cin, cout`
`#include<string> // string`
 	
`using namespace std;`
`...`
`string s;`
`...`
	Przydatne biblioteki i deklaracja.

`cin >> s;` `cout << s;`	Wczytaj z stdin, wypisz na stdout.

`s = "Hello!\n" + '#' + "lo";`
`if(s.empty()) { ... }`
`size_t len = s.length();`
`char c = s[4];`
	Zapisz napis do zmiennej.
	Czy string jest pusty?
	Jaka jest długość stringa?
	Jaki jest 5-ty znak?

`size_t p = s.find("lo");`
`size_t p1 = s.find("H", 1);`
`if(p != string::npos) { ... }`
	Znajdź pierwsze wystąpienie stringa.
	Znajdź pierwsze wystąpienie stringa w sufiksie s[i,...]
	Czy powyższy znak występuje w stringu?

Uwagi:

1. Nie należy odwoływać się do znaków o złych indeksach.
2. Znak '\n' może być elementem stringa.
3. find zwraca std::string::npos jeśli znak nie należy do stringa.

Linki:

1. http://www.cplusplus.com/reference/string/string/

2 Definicje cz. I

2.1 Alfabet

Alfabetem nazywamy zbiór znaków.

$$\Sigma = \{a, b, c, d, e\}$$

2.2 Słowo

Słowem nazywamy ciąg znaków (najczęściej skończonej długości.

$$w = a_1a_2a_3 \dots a_n$$

Istnieje również słowo puste (długości 0), które oznaczamy jako ϵ.
Długość słowa oznaczamy przez $|w| = n$

2.3 Język

Językiem nazwiemy zbiór słów.

$$L = ab, abbc, abbbd$$

Np. Niech $\Sigma = \{0, 1\}$. Wówczas językiem złożonym ze słów długości 2 nad językiem Σ będzie $\Sigma^2 = \Sigma \times \Sigma = \{00, 01, 10, 11\}$. Ogólniej Σ^n jest zbiorem słów długości n.

2.4 Podsłowo

Podsłowem jest spójny podciąg znaków danego słowa.

$$w(i, j) = w_iw_{i+1} \dots w_j$$

Np. Dla słowa $w = baaccd$, $aacc$ jest podsłowem, ale bd nie jest.

2.5 Prefiks

Prefiksem nazywamy podsłowo które zaczyna się wraz z początkiem słowa. Tj, są to wszystkie podsłowa postaci $w(1, j)$.
Np. $w = abbc$, abb jest prefiksem, ale c nim nie jest.

2.6 Sufiks

Jeśli słowo w ma długość n, to sufiksami są wszystkie słowa postaci $w(i, n)$. (Wszystkie znaki od pewnego momemntu do końca słowa.)

2.7 Prefiksosufiks

Dla słowa w prefiksosufiksem nazwiemy słowo x takie, że x jest sufiksem słowa w, oraz x jest prefiksem słowa w.
Np. $w = albatrosjordialba$, $x_1 = alba$ $x_2 = a$ są prefiksosufiksami, ale $x_3 = ba$ nie jest.

3 Zadania

3.1 Języki $O(z)$

n - liczba słów do sprawdzenia
z - suma długości słów
str_i - słowo (dowolny ciąg cyfr, długości przynajmniej 1)
t_i - "TAK" lub "NIE"
Należy sprawdzić czy dane słowa należą do języka L. Jeśli i-te słowo należy do języka w i-tej linii wyjścia należy wypisać TAK, w przeciwnym wypadku należy wypisać NIE.

$$L = \{0^n 1^n : n \geqslant 1\}$$
$$01 \in L$$
$$0011 \in L$$
$$011 \notin L$$

Wejście	Wyjście
n str_1 str_2 ... str_n	 t_1 t_2 ... t_n
Przykład	
3 01 0011 011	 TAK TAK NIE

3.2 KMP

Zdefiniujmy funkcję ps obliczającą długość najdłuższego prefiksosufiksu dla kolejnych prefiksów danego słowa. (takie które nie są tym słowem, co znaczy, że długość prefiksosufiksu jest mniejsza od długości prefiksu).

$$ps(n) = \begin{cases} max_{1<i<n}(w(1,i) = w(n-i+1,n)), & \text{gdy istnieje prefiksosufiks} \\ 0, & \text{gdy jedynym prefiksosufiksem jest } \epsilon \end{cases}$$

Zbadajmy kilka własności funkcji ps.

1. Klasycznie warto sprawdzić kilka małych wartości. ps(1) = 0, bo $i < n = 1$.

2. Można zacząć się zastanawiać, czy da się wyliczać funkcję ps szybicej niż poprzez sprawdzanie wszystkich możliwości dla wszystkich długości. Okazuje się, że tak.

Lemat 1 (Prefiksosufiks mojego prefiksosufiksu jest moim prefiksosufiksem)
Rozpatrzmy prefiks w(1, n), oraz jego prefiksosufiksy $w(1, k_1), w(1, k_2), w(1, k_3), \dots w(1, k_r)$. $(k_1 < k_2 < \dots < k_r)$

Wówczas dla dowolnego słowa $w(1, z)$*, takiego, że* $w(1, n)$ *jest prefiksosufiksem wszystkie słowa* $w(1, k_i)$ *także są prefiksosufiksami* $w(1, z)$.

Dowód 1
$w(1, n)$ *jest prefiksosufiksem* $w(1, z) \implies w(1, n) = w(z - n + 1, z)$
$w(1, k_i)$ *jest prefiksosufiksem* $w(1, n) \implies w(1, k_i) = w(n - k_i + 1, n)$

Wówczas $w(1, k_i) = w(n-k_i+1, n) = w(z-n+1+(n-k_i+1), n) = w(z-k_i, n)$. *(W drugiej równości korzystamy z tego, że* $w(n - k_i + 1, n)$ *jest sufiksem* $w(1, n)$ *oraz* $w(z - n + 1, z)$.

Lemat 2
$ps(n + 1) \in \{0, ps(n) + 1, ps(ps(n)) + 1, ps(ps(ps(n))) + 1, \dots\}$

Dowód 2
Niech $w = a_1$*, wówczas słowo jedynym prefiksosufiksem słowa w jest słowo puste* ϵ.
Niech $w = a_1 \dots a_n a_{n+1}$*, oraz niech* $\{\epsilon, w(1, k_1), w(1, k_2), \dots w(1, k_r)\}$ *będzie zbiorem prefiksosufiksów słowa* $w(1, n)$.
Wówczas poniższe warunki są równoważne.

1. $w(1, k_i + 1)$ *jest prefiksosufiksem słowa w*

2. $w(1, k_i) = w(n - k_i, n) \wedge a_{k_i+1} = a_{n+1}$ *(Po prostu przedłużamy słowo o jedną literę, która musi się zgadzać.*

Lemat 3
Wartości $ps(1), \dots, ps(n)$ *da się wyliczyć w* $O(n)$.

Dowód 3 (Koszt zamortyzowany)
ps(n+1) może zmaleć kilka razy lub w ogóle, i urosnąć tylko raz lub w ogóle. Wiemy, również, że $ps(n + 1) < n + 1$ *oraz, że funkcja ps nie zmaleje bardziej niż wzrosła. W takim przypadku liczba operacji nie przekracza* $c * 2n = O(n)$.

3.2.1 Wystąpienie słowa

Rozpatrzmy alfabet Σ, taki, że $\# \notin \Sigma$ oraz słowa $p, x_1, x_2, \dots, x_r$ składają się ze znaków z Σ.
Wówczas dla słowa $w = p\#x_1\#x_2\#\dots\#x_r$ zachodzi warunek:
Jeśli $p = w(1, |p|) = w(n-|p|+1, n)$ (Słowo wzorcowe p wystąpiło). to zachodzi dokładnie jeden z poniższych warunków:

1. $n = |p|$
2. $ps(n) = |p|$
3. Po wartościach ps, zaczynając od ps(n), da się doskoczyć do $ps(ps(...))) = |p|$.

3.2.2 Minimalne pokrycie

Treść: Szablon (12 OI) https://main.edu.pl/pl/archive/oi/12/sza (Jest też na szkopule) Rozpatrzmy słowo $w = a_1 a_2 \ldots a_n$.

Lemat 4
Jeśli $w(l, r)$ pokrywa w to $w(l, r)$ jest prefiksosufiksem.

Dowód 4
Słowo w(l, r) musimy przyłożyć do pierwszego i ostatniego znaku.

Lemat 5
$w(1, k)$ pokrywa $w(1, n)$ i jest prefiksosufiksem $w(1, n+k)$ to pokrywa $w(1, n+k)$.

3.3 Karp-Rabin

Zdefiniujmy funkcję haszującą h przyporządkującą danemu podsłowu liczbę.

$$w = a_1 a_2 \ldots a_n$$
$$p, q \in \mathbb{P}$$
$$h(l, r) = (\sum_{i=l}^{r} a_i p^i) \mod q$$

W szczególności zachodzi:

1. $h(1, n) = (\sum_{i=1}^{n} a_i p^i) \mod q$.
2. $w(l, l+k) = w(r, r+k) \implies p^{r-l} hs(l, l+k) \equiv h(l, l+k) \pmod q$

3.3.1 Kolizje

O ile funkcja h jest dostatecznie losowa, to przypisuje wartości z przedziału $\{0, 1, \ldots, n-1\}$ z podobnym (równym) prawdopodobieństwem. Dla k słów:

$$P(\text{kolizja nie wystąpi}) = \frac{|A|}{|\Omega|} = \frac{n!}{(n-k)! k^n}$$

A - zbiór funkcji różnowartościowych z $\{0, 1, \ldots, n-1\}$ na $\{0, 1, \ldots, k-1\}$
Ω - zbiór funkcji z $\{0, 1, \ldots, n-1\}$ na $\{0, 1, \ldots, k-1\}$
Czasami można też wziąć więcej funkcji hashujących np. dwie, wówczas o ile są różne to prawdopodobieństwo kolizji maleje.

3.4 Zastosowanie

Mamy k wzorców o sumarycznej długości K oraz n słów (o sumarycznej długości N). Wówczas w czasie $O(nK)$ możemy sprawdzić wystąpienie wzorców w słowach. (o ile nie będziemy mieli pecha tzn. kolizji i nie będziemy ich sprawdzać).