Omówienie zadania Marudny Bajtazar

Filip Konieczny

15 maja 2022

1 Analiza

Pierwszą i kluczową obserwacją jest to, że wynik jest ograniczony przez małą liczbę, dokładniej przez 17. Różnych słów długości 17 nad $\{0, 1\}$ jest $2^{17} = 131072$, a podsłów długości 17 w łańcuchu długości n jest co najwyżej n (dokładnie $n - 16$), a $n \leqslant 100000$, więc nie ma możliwości, żeby słowo długości n wygenerowało wszystkie 2^{17} podsłów, więc wynik rzeczywiście będzie co najwyżej 17.

Czyli musimy pilnować jedynie czy w naszym słowie występują podsłowa do długości 17. Zrobimy trochę więcej, dla każdego podsłowa długości $\leqslant 17$ będziemy trzymać ile razy występuje w aktualnym słowie.

Rozważmy moment zmiany pojedynczej literki - ile podsłów długości co najwyżej 17 się zmienia? Zmieniają się podsłowa, które mają w środku zmienianą pozycję: istnieje co najwyżej k takich podsłów długości k dla pojedynczej pozycji. Czyli jak zmienia się jedna literka, to zmienia się co najwyżej $1 + 2 + \ldots + 17 = 153$ interesujących nas słów, czyli nie za dużo (liczba zmian jest $m \leqslant 10^4$).

2 Implementacja

Pomysł jest taki, że trzymamy tablicę `int cnt[l][N]`, gdzie $l \leqslant 17, N \leqslant 2^17$, gdzie trzymamy ile razy słowo długości l wyglądające jak zapis binarny liczby N występuje w całym słowie z wejścia. Dodatkowo chcemy trzymać tablicę `int used[l]`, gdzie trzymamy ile różnych podsłów długości l jest w słowie (zauważmy, że jest to równe liczbie niezerowych pól w tablicy `cnt[l][*]`).

Na początku obie je inicjalizujemy słowem z wejścia (lecimy po kolei po wszystkich podsłowach długości $1, 2, \ldots, 17$).

Przy aktualizacji lecimy po wszystkich podsłowach, które zahacza wybrany indeks, usuwamy z `cnt` wszystkie wystąpienia, które znikają, jednocześnie pilnując tablicy `used`, tzn. gdy element z pierwszej tych tablic spada do zera, trzeba zmiejszyć odpowiedni licznik w tej drugiej.

Następnie, symetrycznie, dodajemy wszystkie nowe słowa, które generuje zmieniony indeks, pamiętając, żeby zmienić też tablicę `used`, gdy z `cnt` znika zero.

Po aktualizacji musimy wypisać najmniejszą długość, dla której istnieje słowo, które nie występuje: będzie to najmniejsza liczba k spełniająca $used[k] \neq 2^k$.

Całość można zaimplementować w czasie $O(n \log n + m \log n^2 + m \log n)$ (odpowiednio, inicjalizacja, aktualizacja tablic, odpowiedź po aktualizacji, gdzie $\log n$ reprezentuje 17), aczkolwiek wolniejsze implementacje typu $O(n \log n^2 + m \log n^3 + m \log n)$ też mogą wejść (dotyczy to m.in. implementacji, gdzie wartość liczby na podstawie podsłowa jest liczona od zera za każdym razem, zamiast na podstawie wcześniejszych wartości).

Powodzenia!