Informatyczny kącik olimpijski (14) – słowa, słowa, słowa...

Mamy dany m-elementowy zbiór słów S nad t-literowym alfabetem oraz liczbę n. Rozważmy słowa („hasła"), składające się z n liter. Interesują nas te spośród tych słów, które zawierają, jako podsłowa, każde ze słów z S. Ile jest takich słów–haseł? I jak je wyznaczyć? Na przykład, jeśli $S = \{abc, cd\}$ oraz $n = 5$, to wszystkie dopuszczalne hasła to *abccd*, *cdabc* oraz *abcdx* i *xabcd*, gdzie x oznacza dowolną literę.

Jeśli któreś słowo w S jest podsłowem innego, to krótsze słowo można pominąć w rozważaniach – i tak musimy zagwarantować, żeby w haśle znalazło się dłuższe słowo, a wraz z nim jego krótsze podsłowo. Wystąpienia słów z S w poprawnym haśle mogą o siebie zahaczać, ale nie zawierają się jedno w drugim (w szczególności, żadne dwa podsłowa nie zaczynają się i nie kończą w tym samym miejscu).

Zacznijmy więc od najprostszego rozwiązania – generowania wszystkich haseł. Rekurencyjnie generujemy kolejne litery, co zajmie czas rzędu n^t, bo właśnie tyle jest słów długości n o literach z t-literowego alfabetu. Na koniec musimy jeszcze, dla każdego hasła, sprawdzić, czy zawiera ono wszystkie słowa z S – można do tego wykorzystać wersję *algorytmu Knutha–Morrisa–Pratta służącą do jednoczesnego wyszukiwania w tekście wielu wzorców*. Polecam zapoznanie się z tym algorytmem w literaturze lub Internecie. O jego podstawowej wersji pisaliśmy w *Delcie* 1/2008.

Jak ulepszyć ten algorytm? Zauważmy, że w każdym wywołaniu rekurencji pamiętamy całe dotychczas wygenerowane słowo. Może nie musimy? Na pewno musimy znać jego długość, musimy też wiedzieć, które słowa z S już dotychczas wystąpiły. Musimy jeszcze wiedzieć, jakie słowa z S są „otwarte" – tzn. mogą być częściowo zawarte w dotychczas wygenerowanym słowie, o ile dalsze litery będą pasowały. Na przykład jeśli już wygenerowaliśmy *hellowo*, to *world* jest takim otwartym podsłowem. Takich słów może być kilka. Żeby jednak je wszystkie scharakteryzować, wystarczy wybrać najdłuższy „pasujący prefiks" – tj. najdłuższy prefiks spośród prefiksów podsłów, które mają występować w całym słowie, który jednocześnie jest sufiksem aktualnie wygenerowanego słowa. Innymi słowy, wystarczy realizować jednocześnie wspomniany już algorytm KMP dla wielu wzorców. Na przykład, jeśli $S = \{bcdef, defgh, bcdz\}$, a już wygenerowaliśmy *abcd*, to najdłuższym pasującym prefiksem jest *bcd* (choć samo *d* również jest pasującym prefiksem).

W ten sposób ograniczyliśmy liczbę możliwych stanów – czyli opisów dotychczas wygenerowanego słowa – do iloczynu długości hasła (n), liczby podzbiorów S (2^m; musimy pamiętać, które słowa się już pojawiły), oraz liczby różnych pasujących prefiksów (ograniczone przez ml, gdzie l to maksymalna długość słowa w S). Tak więc liczba stanów jest ograniczona przez $nml2^m$.

Co nam to właściwie daje? Ograniczamy liczbę wywołań rekurencji, bo jeśli trafimy do stanu, w którym już byliśmy (w poprzednim przykładzie mógłby to być stan opisujący słowo *bbcd*), to możemy wykorzystać już obliczony wynik, ponieważ „ciąg dalszy" generowania będzie taki sam.

Teraz należy się jeszcze zastanowić, ile czasu może zająć pojedynczy krok rekurencji. Polega on na wydłużeniu już wygenerowanego słowa każdą możliwą literą. A więc ile trwa aktualizacja stanu pojedynczą literą? Tyle, co pojedynczy krok algorytmu KMP, z tym że mamy do uwzględnienia jeszcze jeden czynnik. Kiedy aktualizujemy pasujący prefiks, to przeglądamy coraz krótsze i krótsze prefiksy, aż natrafimy na taki, który można wydłużyć – czyli istnieje pasujący prefiks, równy temu z dodaną literą. W tym przypadku musimy sprawdzać jeszcze jedno kryterium – czy ten wydłużony prefiks należy do jakiegoś słowa z S, które jeszcze się nie pojawiło. Musimy więc skorzystać z jakiejś reprezentacji podzbiorów zbioru S, na przykład poprzez maski bitowe – ale szczegóły już w innym miejscu tego numeru...

Filip WOLSKI

Rozwiązanie zadania M 1224.
Przyjmijmy, że bok pojedynczego pola szachownicy ma długość 1 oraz środek każdego pola szachownicy ma współrzędne całkowite. Niech ponadto środek pola, na którym początkowo stoją cztery pionki, ma współrzędne $(0, 0)$.

Każdemu pionkowi stojącemu na polu, którego środek ma współrzędne (m, n), przyporządkujemy liczbę $1/2^{m+n}$. Wówczas z równości

$$\frac{1}{2^{(m+1)+n}} + \frac{1}{2^{m+(n+1)}} = \frac{1}{2^{m+n}}$$

wynika, że po wykonaniu każdego kroku suma S liczb przyporządkowanym wszystkim pionkom stojącym na szachownicy nie zmienia się. Zatem początkowa konfiguracja czterech pionków daje zależność $S = 4$.

Gdyby po skończonej liczbie kroków na każdym polu stał co najwyżej jeden pionek, to

$$S < \sum_{n=0}^{\infty}\sum_{m=0}^{\infty} \frac{1}{2^{m+n}} = \left(\sum_{n=0}^{\infty} \frac{1}{2^n}\right) \cdot \left(\sum_{m=0}^{\infty} \frac{1}{2^m}\right) = 4.$$

Otrzymaliśmy sprzeczność, która dowodzi, że po każdym kroku pewne dwa pionki stoją na tym samym polu.