АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА ПОКРИТИЕ

Нека означим с $len(i)$ дължината на най-дългия подниз, имащ следните две свойства:

1) започва със символа, намиращ се на i-та позиция в оригиналния низ
2) среща се поне веднъж изцяло преди началната си позиция в оригиналния низ

За да се изчислят стойностите на $len(i)$, в авторовото решение се използва суфиксен масив. След построяването на суфиксния масив се намират и lcp-стойностите на всеки два съседни суфикса в лексикографската наредба, показващи дължината на най-дългия им общ префикс. Тази стъпка е реализирана с **алгоритъм на Kasai** с линейна сложност.

Разглеждаме суфиксите в реда на лексикографската им подредба. В един стек ще съхраняваме двойки числа – lcp-стойността на съответния суфикс и този след него в масива и началната му позиция в оригиналния низ. Целта ни е преди добавянето на i-тия суфикс в стека, да можем да намерим най-ляво разположения лексикографски по-малък суфикс, който има най-дълъг общ префикс с i-тия с конкретна дължина.

При добавянето на $i-1$-вия поред суфикс, чийто най-дълъг общ префикс с i-тия е с дължина x, е ясно, че най-дългият общ префикс между всеки от суфиксите, намиращи се в стека, и i-тия ще бъде най-много с дължина x. Това предполага, че в стека ще съхраняваме само суфикси с нарастващи lcp-стойности (най-голямата е на върха на стека). Когато добавяме нов суфикс, от върха на стека премахваме всички други, чиито lcp-стойности са по-големи или равни. Обърнете внимание, че премахнатите суфикси също имат най-дълъг общ префикс с i-тия с дължина новодобавената lcp-стойност. Ето защо, началната позиция на суфикса със съответната lcp-стойност ще бъде най-малката от всички премахнати и новодобавената.

Остава въпросът, как точно намираме $len(p[i])$ – $p[i]$ в случая е началната позиция на поредния суфикс, който разглеждаме. От всички суфикси в стека ни интересуват само два. Първият е този, който има най-дълъг общ префикс с i-тия суфикс и този префикс не го застъпва. Вторият е най-ляво разположеният, чийто най-дълъг общ префикс с i-тия суфикс се застъпва с него. От втория можем да използваме само частта от началото му до $p[i]-1$.

За да определим началните позиции и дължините на въпросните поднизове можем да построим **sparse table** върху елементите в стека (и да я променяме при всяко добавяне или премахване от него). С други думи, за всеки суфикс в стека, помним най-малката начална позиция на последните 2^k суфикса до него включително. Това ни позволява да направим двоично търсене по отговора. Започваме от последния суфикс в стека и избираме максимално голямо k. Проверяваме дали за последните 2^k суфикса най-дългият общ префикс на разположения най-вляво от тях се застъпва с текущия суфикс. Ако да – отчитаме неговата начална позиция за подниз от втория тип и продължаваме със суфиксите, които не сме проверили. Ако не – намаляваме избраното k с единица. Така описаният алгоритъм задължително ще достигне до суфикс, чийто най-дълъг общ префикс не се застъпва с i-тия суфикс, и той ще бъде потенциално по-дълъг от използваемата част на подниза от втори тип. Изключение прави случаят, когато такъв не съществува.

Разбира се, това ни дава непълна стойност за $len(p[i])$, защото разглеждаме само лексикографски по-малките суфикси. Важно е да отбележим, че сред тях има и суфикси, започващи след началото на i-тия, които алгоритъмът просто пренебрегва. За да намерим

окончателната стойност, трябва да приложим същия подход и за суфиксите отзад напред в суфиксния масив (от лексикографски най-големия до лексикографски най-малкия).

След като вече сме изчислили $len(i)$ за всяко i, можем да направим същото и за оригиналния низ, обърнат наобратно. Сега за всяка позиция i ще знаем и кой е най-дългият подниз, който завършва в i и се среща поне още веднъж след това.

Броят на търсените поднизове можем да намерим с едно обхождане на получените стойности. На i-тата стъпка ни е необходим броят на позициите j ($1 \leq j < i$), за които последната позиция на най-дългия намерен подниз, започващ от j, е с не повече от 1 по-малка от началото на най-дългия намерен подниз, завършващ в i. Това може да се постигне с **дърво на Фенуик**.

Сложността и на трите основни части от решението (строенето на суфиксния масив, изчисляването на $len(i)$ от ляво надясно и от дясно наляво и пребройването на покритите поднизове) е $O(N * \log_2 N)$.

Изготвил анализа:
Добрин Башев