**Анализ**

Първата подзадача е за 5 точки. Това е от серията подзадачи, където *k*=***N***. Очевидно при тях този параметър няма значение, т.е. се търси най-често срещания подниз на даден с дължина поне *m*. Лесно се вижда, че той е с дължина *m* – ако допуснем че е с по-голяма, то този низ с по-голямата дължина ще съдържа някакъв подниз с дължина *m*, който ще се среща поне толкова пъти. Това означава, че просто трябва да намерим най-често срещания подниз с дължина *m*. Ограниченията позволяват наивно решение – всеки подниз сравняваме с предишния докато ненамерим сходство (или е уникален от досегашните), за да видим колко пъти се среща такъв подниз досега. Сложността на този подход е , но това е твърде силна оценка и дефакто е .

Втората подзадача е за 7 точки. Тук наивният вариант е твърде бавен. Ще го забързаме – можем да сравняваме два подниза чрез хеширане. Освен това можем в *hash map* да пазим колко срещания има даден низ. Интересен факт е, че при правенето на тестове трудно се намериха, работещи тестове при единичен хеш. Това не означава, че състезателите ще вземат невъзможно тези точки, защото имат пълен фийдбек, а могат и да направят двоен хеш. Сложността за тази подзадача се подобри значително - .

Третата подзадача е за 12 точки. Тези точки също се взимат сравнително лесно. Ограниченията позволяват да правим следното. За всяка възможна дължина в заявката от *m* до *k*, ще гледаме всички поднизове с тази дължина. Разделяме ги на групи от равни. Това отново можем да правим с *hash map*. След което трябва да видим във всяка група дали е възможно продължение отляво или отдясно. Ако някъде не е възможно, то нейната дължина е потенциален отговор. Тук има граничен случай – ако някой подниз започва от най-ляво или завършва в най-дясно, то той не може съответно да се продължи отляво или отдясно. Описаният алгоритъм има следната сложност .

Четвъртата подзадача е за 18 точки. Пак се връщаме на условието *k*=***N***. Тя е първата по-съществена подзадача. Вече ограниченията изискват да направим нещо преди заявките, за да можем да отговаряме на тях. Нещо, което помага в по-сложни задачи с низове е суфиксния масив. Случаят и тук е такъв. Нека разгледаме суфиксния масив на дадения низ (за тази подзадача можем да го намираме и със сложност ). Ще докажем следното твърдение – индексите на поднизовете, които са отговор за дадено *m* се явяват подмасив на суфиксния масив. Нека разгледаме срещанията на даден подниз *s* с дължина *m*. Ако допуснем, че те не заемат подмасив на суфиксния масив, то между два от тях ще има някакъв друг подниз. Но понеже са подредени лексикографски, този подниз също трябва да има префикс *s*. Това е противоречие, т.е. наистина срещанията на *s* заемат подмасив на суфиксния масив. Сега да разгледаме *LCP* масив, който получаваме от суфиксния (в него сме записали най-дългия общ префикс на всеки два съседни подниза в суфиксния). Него можем да намираме тук и със сложност чрез *LCP* функцията. Задачата става следната: каква е дължината на най-дългия подмасив на *LCP* масива, така че всяко число е по-голямо или равно на *m*. Ще покажем следния начин да намерим този отговор за всяко *m* от 1 до ***N***. Нека добавяме числата в *LCP* масива малко, по-малко – от най-малките числа към по-големите. Като добавяме числата, остават някакви подмасиви, в които все още няма числа. Понеже ги добавяме от най-малките към по-големите, то ако сме добавили всички числа, които са по-малки от дадено *m*, тези празни подмасиви ще са с числа по-големи или равни на *m*. Ако намерим дължината на най-дългия, това очевидно ще е отговорът за това *m*. Как ще реализираме тази процедура най-ефикасно? Можем да използваме сегментно дърво, в което да пазим най-дългия празен подмасив. Ще ни трябва допълнителна информация за всеки връх – най-дългия празен префикс и най-дългия даден суфикс. Така, когато добавяме някакво число, за да видим колко е отговорът за даден връх – трябва да видим колко е бил за лявото и дясното дете и освен това дължината на най-дългия празен подмасив, който минава през средата. Последното можем да изчисляваме, като съберем дължините на най-дългия празен суфикс на лявото дете и най-дългия празен префикс на дясното дете. Лесно се вижда как се смятат и най-дългия празен префикс и суфикс на даден връх. Нека обобщим крайното решение за подзадачата – намираме суфиксния масив, след това *LCP* масива, после сортираме числата в него в нарастващ ред и накрая започваме да ги добавяме (на позициите, на които са били в *LCP* масива) и всеки път, когато свършим да добавяме числа с дадена стойност *m* имаме отговора за *m*+1 (гледаме корена на дървото). Сложността за тази подзадача е .

Петата подзадача е за 24 точки. Тук условието е малко по-различна от задачата (това беше началният вариант на условието). Сега трябва да отчитаме само продължаването вляво. Нека сме намерили отново суфиксния масив (този път ще ни трябва на обърнатия низ) и *LCP* масива, който му съответства. Старият подход очевидно не работи тук. Имаме следното лесно наблюдение, когато съществува отговор за дадена заявка. Нека поднизът *s* е отговор за дадена заявка и той не може да бъде продължен вляво (със сигурност има такъв отговор, защото ако допуснем противното, то ще продължаваме всеки възможен отговор все по-наляво, а знаем, че за заявката съществува отговор). Тогава ако разгледаме срещанията на *s* в *LCP* масива, те правят подмасив, където има подниз с число, което е дължината на *s*. Очевидно ако всички числа са по-големи от дължината на *s*, то той може да се продължи вляво, а ние взехме такъв, който не може. Това лесно наблюдение ни показва следния подход – можем за всеки подниз в *LCP* масива да определим интервала, в който всички числа са по-големи или равни на неговата стойност. Лесно се вижда, че с два стека можем да намерим линейно интервала за всяко число в *LCP* масива. Сега дадена заявка ни пита за следното – колко е дължината на най-дългия интервал (+1 по принцип, защото всяко число в *LCP* масива се получава на базата на два подниза), чието характеризиращо число е между *m* и *k* включително. Тези наблюдения в момента показват, че може и да има случаи, в които няма отговор. Това беше началният вариант на задачата, но видях, че отговор не само винаги съществува но има и с дължина *m*. Наистина да допуснем, че отговорът се получава за някакъв низ *s* с по-голяма дължина. Тъй като той не може да се продължи вляво (вече показахме, че винаги има такъв), то нищо не пречи да вземем префикса му с дължина *m*. Очевидно той ще се среща поне толкова пъти, колкото *s*, но също няма да може да се продължи вляво. Това означава, че се явява отговор на задачата. След като видим това, означава, че когато намерим интервалите и ги сортираме по характеризиращите числа, с едно обхождане можем да намерим отговорът за всяка дължина *m*. За дължини, които са по-големи от най-голямата в *LCP* масива, отговорът ще е 1 (ще представлява някакъв префикс на оригиналния низ), а лесно може да се докаже, че всяка по-малка дължина ще се среща в масива. Отговорът за дадена дължина получаваме, като намерим максималната дължина на интервалите с това характеризиращо число. Крайната сложност за тази подзадача е .

Шестата и последна подзадача е за 34 точки. Какво пречи да направим същото, като в предната подзадача, т.е. да имаме и отговор за всяко *m*? Ако следваме доказателството, проблемът е, че дори и да вземем префикс на *s* с дължина *m*, то той ще може да се продължи пък вдясно и нищо няма да постигнем. Няма как да оправим това доказателство, защото то в случая не е за верен факт (отговор тук може и да не същестува за дадена дължина, например в низа abcabc за дължина 2). Сега ще разглеждаме суфиксните масиви на низа и обърнатия низ и съответно техните *LCP* масиви. Очевидно пак можем да намерим отговор на заявка, който не може да се продължи вдясно. Нека разглеждаме само такива низове (можем да ги намираме по аналогичен начин с предната подзадача). Сега трябва за дадена заявка да видим има ли такива поднизове, така че са с дължина между *m* и *k* и вляво не могат да се продължат до дължина по-голяма от *k*. Имаме следното наблюдение – ако гледаме интервал в суфиксния масив на дадения низ с някакво характеризиращо число, то той има съответстващ в суфиксния масив на обърнатия низ. По-точно ако даден интервал *[l; r]* с характеризиращо число *d* се състои от индекси на поднизове: *ind[l]*, …, *ind[r]*, то за обърнатия низ ще имаме някакъв друг интервал *[l’; r’]*, състоящ се от индекси (тях гледаме спрямо дадения низ) на поднизове: (*ind[l]*-*d*+1), …, (*ind[r]*-*d*+1) (незадължително в този ред). Това също не се доказва особено трудно. Аналогично на предно доказателство, ако допуснем, че интервала *[l; r]* се разпада в *LCP* масива на обърнатия низ и имаме някакъв друг подниз там, то той трябваше да бъде някъде в интервала в суфиксния масив на дадения низ (поднизовете, които съвпадат в *d* знака, в която и посока да гледаме са еднакви). Имайки това твърдение знаем, че като имаме даден интервал с характеризиращо число *d*, можем да намерим съответстващ при обърнатия низ и неговото характеризиращо число ще е по-голямо или равно на *d*. Това число всъщност ще задава за тези поднизове, които не могат да се продължат вдясно, колко най-много могат да се продължат вляво. Тук имаме и аналогично твърдение на миналата подзадача – ако имаме отговор на заявка, то той ще за дължина *m*. Да допуснем противното, че имаме подниз *s* с по-голяма дължина. Тогава ако вземем суфикса му с дължина *m*, това ще са низове, които не могат да бъдат продължени вдясно и ще им съответства даден интервал в суфиксния масив на дадения низ. Съответно за тях ще можем да видим, че се продължават вляво с дължината на *s* или по–малка и че ще са отговор. Това означава, че ако подредим интервалите, които получаваме от дадения низ, първо по характеризиращото им число и второ по характеризиращото число на съответния им интервал при обърнатия низ, за да намерим отговор за дадена заявка *m* и *k* e достатъчно да видим максималната дължина в подмасива с първо характеризиращо число *m* и второ, което е по-малко или равно на *k*. Нека обобщим какъв ще е крайният алгоритъм за задачата след намирането на суфиксните и *LCP* масиви. Намираме интервалите на обърнатия низ и в един *hash* *map* пазим хеш за интервалите. Него можем да намираме, като хеш на множество от индекси. След това намираме интервалите на дадения низ и използвайки същото хеширане лесно намираме какъв трябва да е хеша на съответстващия интервал и с *hash* *map*-а и съответното характеризиращо число. После сортираме (тук използваме два пъти *count sort* за бързодействие) интервалите по признака. Накрая за всяка дължина намираме между кои индекси се среща в най-новия масив и за всеки интервал намираме максималната дължина на интервал от него до най-левия със същото първо характеризиращо число. Така можем да отговаряме на заявките с двоично търсене. Окончателната сложност на задачата е . Може да се направи и в (макар и с голяма константа) с линейно строене на суфиксен масив, но така задачата ще стане прекалено тегава. Има и алтернативни решения със суфиксно дърво и суфиксен автомат.

*Автор: Илиян Йорданов*