Тагове	На пълното решение	На частичните решения
	BFS Trie	Двоично повдигане

Анализ

Подзадача №1

Както винаги оставих подзадача с тестовите примери за обратна връзка от системата.

Подзадача №2

В подзадачата единствено се търси дължината на най-краткия път от 1 до N. Тази задача е основна и се решава чрез Обхождане в ширина (BFS).

Постигната сложност: $O(N)$

Имплементация: `escape_11p.cpp`

Подзадача №3

Нека за всеки връх освен дължината на най-краткия път търсим и лексикографски най-малкия низ, с такава дължина, по който може да се стигне до него. Нека разстоянието от 1 до връх u да е d_u. Нека *родителите* на връх u са тези върхове, които са съседи на u и са на разстояние от 1 точно равно на $d_u - 1$. Така, индуктивно, като трябва да изберем от кой родител v да достигнем u, ние трябва да следваме критериите:

1. Низът до v е лексикографски най-малкият измежду тези до всички родители на u.
2. При равни низове се избира този с най-малка буква на реброто до u.

Тази идея е основна за нататъчните решения. Най-наивно може да се приложи като за всеки връх директно се намери всеки кандидат-низ и се избере лексикографски най-малкият.

Постигната сложност: $O(N^2)$

Имплементация: `escape_22p.cpp`

Подзадача №4

За по-добри резултати, ние трябва да измислим добър начин за проверка на критерии 1 и 2. Докато критерии 2 е директен след установяване равенство по критерии 1, другият е по-предизвикателен. Може да забележим, че правейки нашето BFS обхождане, ние винаги избираме точно 1 ребро, от което да се достига връх u от негов родител v. Разглеждайки само тези ребра, ние получаваме дърво (иначе наричано BFS-дърво), което покрива всички оптимални пътища в графа. За бързо сравняване на низове S, T стандартно се използва двоично търсене по техните префиксни хешове, за да може за $O(log_2 min(|S|, |T|))$ време да се намери първата позиция, в която те се различават. Подобна идея може и да се приложи в случая – като използваме еквивалента на префиксни хешове в дърво, а именно двоичното повдигане (`binary lifting`) и поддържаме хеша на низа, образуван от връх в дървото до всеки 2^k-ти родител, ние бихме могли отново чрез двоично търсене да намерим точно тази позиция, в която се различават. За повече подробности може да погледнете имплементацията.

Постигната сложност: $O(Nlog_2^2 N)$

Имплементация: `escape_59p.cpp`

Подзадача №5

За хората, запознати със структурата от данни Trie

Следвайки главното правило, че намаляването на броя логаритми в решението се постига с опростяването му, ние може да достигнем до по-лесна идея. Представете си чисто идейно как добавяме всички оптимални низове в `Trie` и за всеки връх записваме неговата позиция в `BFS-order`-а на трая, като за всеки връх преминаваме първо през ребрата с по-малка буква. Така, за всички низове с равна дължина, един низ ще е по-малък от друг, когато неговата позиция в този `BFS-order` е по-малка. Ако успеем да приложим имплицитно идеята на трая в решението ни, ние лесно ще може да се справим с критерии 1, като го свеждаме до сравнение на две числа. Удобното е, че ние абстрактно погледнато чрез нашето BFS, ние обхождаме трая точно в този `BFS-order`, като за да намерим позициите в `BFS-order`-а, ние може като достигнем връх u на по-голямо разстояние $d_u > d_v$ от предишния обходен връх v, да съпоставим позициите на низовете в `BFS-order`-а на всички върхове на разстояние d_u от 1. Това може да го постигнем като си сортираме низовете на всички върхове като pair-чета по критериите и приложим разделяне на последователности. Готините от вас може да приложат `radix sort`, което ще доведе до решение с крайна сложност $O(N)$, но това не беше нужно за 100 точки на самото състезание.

За хората, **не***запознати със структурата от данни Trie*

Желаем за всеки низ до връх u да съпоставим номер n_u, така че за два върха x, y, за които $d_x = d_y$, низът до връх x да е по-малък до низът до y, тогава и само тогава, когато $n_x < n_y$. Индуктивно, нека да сме избрали номерата на всички върхове на разстояние $d - 1$. За да съпоставим номерата на тези на разстояние d, ние може за всеки връх да си изберем от кой родител да бъдем достигнати по критерии 1 и 2, да сортираме върховете точно по тези критерии, и съпоставим номера чрез разделяне на последователности. По-готините от вас може да са се сетили, че сортирането в този му вид е сортиране на точки в двумерно пространство с малки координати и да са приложили radix sort, който би довел до сложност от $O(N)$ за крайното решение, но това не беше нужно за 100 точки на самото състезание.

Постигната сложност: $O(N\ log_2 N)$

Имплементация: `escape_100p.cpp`

Автор: Борис Михов