**Анализ**

*Тагове: покриващи дървета, МПД, алгоритъм на Крускал, DSU*

Ясно е, че постановката задава неориентиран, непретеглен граф. Главната пътна мрежа всъщност представлява покриващо дърво на графа. Цикличната замяна по същество образува цикъл с новодобавеното ребро, от което маха едно от старите ребра.

Първата подзадача е за 13 точки. Тук улеснение идва от това, че графът е пълен. Понеже в даденото покриващо дърво няма връх, който е свързан с всеки друг, то можем да направим покриващо дърво, което не включва нито едно ребро от началното. Неговата цена ще е ***N***-1, а за да го открием трябва просто да пуснем един *DFS* в графа без ребрата от покриващото дърво. Сложността тук е .

Втората подзадача е за 30 точки. Трябва да направим някои основни наблюдения. Нека видим какво точно представлява цената за преобразуване от едно покриващо дърво в друго. Ясно е, че минимален брой смени ще направим, ако общите ребра на двете дървета не ги пипаме по възможност. Нека разглеждаме едно ребро, което не е в началното покриващо дърво. Тогава то затваря цикъл в началното дърво. Ако допуснем, че всички ребра от цикъла, участват в целевото покриващо дърво, получаваме цикъл в него, т.е. има поне едно, което не участва в целевото и е от началното. Заедно с него може да направим циклична замяна и да прогресираме с една стъпка. Но така можем да приложим абсолютно същите разсъждение за текущото покриващо дърво и целевото. По този начин извършвайки циклична замяна, добавяме едно нужно ребро и махаме едно излишно. Така ако двете дървета в началото имат *k* общи ребра, то е ясно, че ще ни трябват най-малко ***N***-1-*k* циклична замени за превръщане от едното дърво в другото. Тази формула ни помага и в контекста на най-скъпо дърво. Защото това означава, че то ще е такова, което има най-малко общи ребра с даденото. Тези разсъждения ни позволяват да направим една интуитивна грийди идея. Нека обхождаме последователно ребрата, които не се включват в даденото покриващо дърво. Тогава можем да пробваме да направим циклична замяна с текущото ребро само ако на пътят между неговите два върха в дървото има някое от оригиналните ребра. По този начин като минем през всички възможни ребра, които можем да добавим, ще сме извършили възможно най-много циклични замени и ще сме достигнали до най-скъпото покриващо дърво. За бързи промени по ребрата е най-удобно списъците със съседите да са реализирани чрез *set* или *unordered\_set*. Очакваната сложност за решение е .

Третата подзадача е за 57 точки. След като направихме ключовите разсъждения е време да оптимизираме процеса по намиране на покриващо дърво, което е най-скъпо. Можем да кажем, че търсим дърво, което има най-малко общи ребра с даденото. Една хубава идея е да направим графа претеглен – слагаме цена 0 на ребрата, които не са в покриващото дърво и цена 1 на ребрата, които са в него. След което търсим минимално покриващо дърво на този граф. Ясно е, че така ще получим покриващо дърво, в което има най-малък брой общи ребра с началното. Освен това неговата сума е точно броя общи ребра (тези, които са от оригиналното).Най-удачно е да се ползва алгоритъма на Крускал, за да се възползваме от цените на ребрата. Така можем да направим *count sort* на ребрата, с което намирането на покриващото дърво ще стане почти линейно. Друг вариант, който е по-ценен за тази задача е да забележим, че всъщност, когато първо обхождаме ребрата с цена 0, те са безплатни и ще добавим такива, колкото максимално да свържем върховете без да ползваме ребрата на даденото покриващо дърво. По друг начин казано, ако от началния граф махнем ребрата на даденото покриващо дърво, то в началото Крускал просто ще взема ребра, така че да направи покриващи дървета на свързаните компоненти, на които евентуално се разпада графа след махането на ребрата. Съответно ребрата, които ще се вземат с цена 1 е ясно точно колко ще са – колкото са свързаните компоненти минус 1, защото в крайна сметка трябва да свържем всички свързани компоненти в една компонента. Така всъщност за решение на тази подзадача е достатъчно да махнем ребрата на покриващото дърво и да преброим компонентите. Ако от ***N***-1 извадим техния брой ще получим отговора. За да намерим най-скъпо покриващо дърво все пак трябва да пуснем пълния алгоритъм на Крускал, за да открием точно ребрата, които ни трябват. Сложността е .

Възможно е задачата да се усложни, като се добавят заявки под формата на *циклични замени* за промяна на даденото покриващо дърво. Съответно в този случай трябва да извеждаме само цената на най-скъпото покриващо дърво за заявките. Лесно се забелязва, че търсената цената може да не се промени или да се промени с ± 1 в зависимост от това дали се маха мост от графа без ребрата на покриващото дърво и дали се добавя ребро, което обединява свързани компоненти. Така ако заявките са независими просто трябва да се открият мостовете, за да се отговоря на заявките. Но това не се очаква като материал, който да се знае от С група. Ако заявките не са независими, задачата става още по-интересна. За да намерим отговора трябва да знаем само броя свързани компоненти, на които се разпада граф след премахване на ребрата от покриващото дърво. Това е точно задачата за *DSU*. Една заявка (циклична замяна) всъщност добавя едно ново ребро и маха едно старо ребро. Лошото е, че по принцип *DSU* не поддържа триене на ребра. Понеже заявките за офлайн, може да се постигне решение със сложност , ако е броят заявки. Погледнете анализа на варианта на тази задача за А група, за малко по-голямо описание на тази част.

*Идея: Руско Шиков  
Автор: Илиян Йорданов*