АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА ПАРК

Нека първо разгледаме задачата, когато няма заявки от тип 2. Даден ни е претеглен ориентиран граф със специфична структура: от всеки връх излиза насочено ребро към точно един друг връх. Пита се дали съществува път с тегло поне *K*, който завършва в даден червен връх и започва от някой жълт или зелен връх. Ключово наблюдение е, че от всеки връх съществува най-много един път до даден друг. Това е така, защото когато сме в даден връх, имаме точно две възможности -- или следваме единственото ребро, което излиза от него, или спираме.

Това наблюдение е достатъчно за решение на Подзадача 1 -- достатъчно е да изпробваме всеки от останалите N-1 върха като начален, и да проверим дали, ако започнем от него ще стигнем до крайния червен като пътят е с дължина поне *K*. За да решим тази подзадача ни е нужно да може да класифицираме върховете по цветове - това всъщност не е никак трудно. Червени са върховете, които са част от цикъл. Жълти са тези, които са на разстояние от едно ребро от цикъл, но не принадлежат на цикъл. Всички останали са зелени. Лесно може да класифицираме върховете по цветове с O(N) обхождане в дълбочина. След като имаме тази информация, може да отговорим на една заявка със сложност O(N^2) като за всеки от останалите най-много N-1 нечервени върха намерим дължината до зададения червен като просто следваме ребрата докато стигнем зададения връх. Това води до решение със сложност O(Q * N^2).

За да решим Подзадача 2, е достатъчно да сметнем разстоянието от всеки нечервен до всеки червен връх за O(N^2), след което просто да отговорим на всички заявки, защото общата сложност става O(N^2 + Q). Това може да направим като забележим, че може да сметнем всички разстояния до даден червен връх за O(N) като обърнем посоката на ребрата и започнем обхождане от дълбочина от дадения червен връх. След това е достатъчно да намерим най-отдалечения жълт или зелен връх от дадения червен, защото ако съществува път с дължина поне K, то най-отдалеченият жълт или зелен връх винаги ни върши работа (това, че е достатъчно, да знаем най-отдалечения нечервен връх за всеки червен връх е едно от ключовите наблюдения в задачата).

За да решим Подзадачи 3 и 4 е нужно да разгледаме структурата на графа по-обстойно. Не е трудно да се забележи (и докаже), че графът е съставен от множество компоненти, всяка от които се състои от един единствен цикъл, като на всеки връх може да има "закачени" няколко дървета. Реално всеки цикъл е съставен от червените върхове, за които има закачени жълти върхове, за които пък има закачени дървета от зелени върхове. Както забелязахме преди, достатъчно е да знаем най-отдалечения нечервен връх за всеки червен. Лесно може да опростим задачата, като забележим, че за всеки жълт връх, лесно може да намерим най-отдалечения зелен в неговото поддърво -- това може да направим с едно обхождане в дълбочина от даден жълт връх в "обърнатия по ребра" граф. След това опростяване, може просто да добавим дължината на най-отдалечения зелен връх за всеки жълт. Така, в задачата оставаме единствено с червени и жълти върхове; забележете, че все пак трябва да помним кой е

бил най-отдалечения зелен връх за даден жълт, защото иначе не бихме могли да отговаряме на заявките правилно. Така, реално ни остава да отговаряме на въпроса -- "от кой жълт връх да тръгнем, така че да стигнем до даден червен като пътя между тях има дължина поне K?". Както вече решихме, достатъчно е да намерим следното - за всеки червен връх, кой е най-далечния жълт. Понеже е бавно да проверим по отделно всяка двойка червен и жълт, е необходимо по някакъв начин да построим структура, така, че да може да използваме информацията без да гледаме всяка двойка.

Нека за момент приемем, че към всеки червен има точно един жълт връх. Нека разгледаме даден червен връх r_1. Всички жълти върхове в неговата компонента водят до него. Лесно може да ги подредим в масив, и да намерим дължината от всички до дадения червен за O(N). Нека след това разгледаме следващия червен връх r_2 (този, към който води r_1). Какво е различното, ако искаме да сметнем същата тази информация за r_2? Със сигурност е различно разстоянието от жълтия връх към r_1, защото трябва да преминем по всички червени, докато стигнем r_2. Другото, което е различно, е че трябва да премахнем реброто (r_1, r_2) от всички пътища, за да имаме същата информация за връх r_2. Лесен начин да направим това е да копираме цикъла 2 пъти, след което намирането на най-отдалечения връх за даден червен връх всъщност се свежда до намирането на максимум от N последователни елемента в масив от 2N елемента, за който съществуват множество от решения, един от които е да ползваме сегментни дървета. Друго възможно решение е използването на някое от известните решения за Range Minimum Query (RMQ). Нека си спомним, че приехме, че към всеки червен връх има точно един жълт. Какво трябва да направим ако има повече от един - достатъчно е да изберем жълтия с най- дълго ребро. А ако има нула -- тогава може да добавим ребро с достатъчно малко отрицателно число, което да ни сигнализира за липсата на жълт връх. За да решим Подзадача 4 е достатъчно да имаме решение със сложност O((N + Q) log N). Всъщност, описаното решение е със сложност O(N log N + Q), защото може да решим RMQ за O(N log N) сравнително лесно. За Подзадача 3 е предвидено решение със сложност O((N + Q) * sqrt(N)), което се базира на подобни техники, но не е нужно познаването нито на сегментни (или индексни дървета), нито на някое от бързите решения на RMQ -- достатъчно е да намираме максимума за интервал от елементи със сложност O(sqrt(N)), което се постига като разделим масива на sqrt(N) части от по sqrt(N) елемента.

За Подзадача 5, е нужно да поддържаме заявки от тип 2. Достатъчно е да зебележим, че swap(next[a], next[b]), е достатъчно за да прекачим станциите при кръстосване. Решението, описано за Подзадача 1, с тази промяна, решава и Подзадача 5.

За Подзадача 6, е достатъчно да намираме разстоянията от даден червен до всички останали жълти за O(N) както е описано в Подзадача 2. Такова решение работи за O(N * Q).

Нека разгледаме Подзадачи 7 и 8. Наблюдението, че може да пазим за всеки червен единствено жълтия закачен към него с най-дълго ребро, вече не работи, тъй като заявки от тип 2 може да презакачат поддървета. Например, ако имаме даден червен връх,

който има 2 жълти поддървета, с дължини на ребрата 100 и 200, и пазим единствено това за 200, то ако има заявка от тип 2, която да премести това ребро за 200, и върне такова за 50, то имаме проблем, защото ни е нужно реброто с дължина 100, а имаме единствено това, което сме взели при презакачането - 50. За да се справим с този проблем е достатъчно да пазим по една структура за всеки червен връх, в която имаме всички жълти ребра свързани към него и подредени по дължина. Така, винаги, когато се интересуваме от най-дългото жълто ребро, закачено за даден червен връх, може бързо -- O(log N), да го намерим дори и след кръстосвания. В предложеното решение използвам STL set, който дори позволява намирането на най-големия елемент за време O(1). Така, презакачането на жълти поддървета реално е местене на елементи между два set-а. Тази промяна също така променя решението за Подзадачи 3 и 4, защото сега се нуждаем от динамична структура, която да позволява намирането на максимум в последователност от динамичен масив. Нуждата от динамичността се основава на следното наблюдение -- след като кръстосаме две станции е възможно, най-отдалечения жълт връх за даден червен да се промени. Докато за подзадача 7 е достатъчна идеята от Подзадача 3 със сложност O(sqrt(N)), то за Подзадача 8 е наложително използването на бърза структура с логаритмична сложност както на update така и на намиране на максимум в последователност от елементи. Предложеното авторско решение използва сегментно дърво (segment tree). Това е и решението за 100 точки.

При съставянето на задачата, авторът измисли и доста по-интересна модификация на задачата, чието решение обаче е твърде предизвикателно за 5 часово състезание, на което има и други 2 задачи. Модификацията е следната: пита се не дали има път с дължина поне K, а колко такива пътя има? Лесно се вижда, че предложената задача е подзадача на текущата, защото ако знаем колко пътя има, е елементарно да кажем дали има поне един такъв. Докато park има решение със сложност O((N + Q) log N), то генерализацията има решение със сложност O((N + Q) log N^2). Можете ли да го откриете? А да го подобрите? Ако имате въпроси/предложения по тази задача, пишете на anastasovbg@gmail.com или aanastasov@alum.mit.edu.

Автор: Антон Анастасов