АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА СКОРОСТ

Задачата трябва да напомня на състезателите за един от най-популярните алгоритми в графи – тази за намиране на минимално покриващо дърво. Само че в случая дървото, което търсим, не е минимално. Не е и произволно, за съжаление – отново има наложени известни ограничения и изисквания.

Първото нещо, което ни ограничава е, че трябва да намерим не едно, а две неща – както минималната, така и максималната скорост (или, погледнато по друг начин, минималната скорост и минималната разлика между скоростите, която всъщност искаме да оптимизираме).

Добре де, нямаме *толкова* много възможни отговори. При 10000 ребра сме сигурни, че едно от тях ще фиксира минималната скорост, а друго ще фиксира максималната такава. Това прави около 100,000,000 възможности, за всяка от които трябва все пак да проверим дали е възможно. Това, макар и много, не е някакво невероятно много. Това решение е със сложност O(M^3) и би хванало някакви точки (около 30-40). Да видим, обаче, как можем да го оптимизираме!

Състезателите в тази група (а и всяка по-горна, for that matter) трябва веднага да се замислят за двоично търсене. Наистина, фиксирайки минималната скорост, бихме могли да правим двоично търсене по максималната! Така трябва:

1. Да фиксираме една от цените на ребрата и да кажем, че тя ще е минималната.
2. Да направим двоично търсене по горната граница.
3. За фиксирания интервал [долна граница, горна граница] да проверим дали графът (с премахнати всички ребра, които не влизат в този интервал) е свързан.

Това наистина работи, като постига сложност O(M*log(M)*M). Това решение вече хваща около 60 точки. И все пак има накъде да подобрим!

След като задачата ни напомня на минимално покриващо дърво, дали не бихме могли да ползваме някой от алгоритмите за минимално покриващо дърво? Прим би бил относително неудачен в случая, но Крускал, както се оказва, подхожда перфектно!

Сортираме ребрата по тяхната цена (със сложност O(M*logM)) и отново, както и при миналите идеи, фиксираме долната граница. След като знаем минималната скорост, можем да обхождаме ребрата с по-големи цени и да ползваме структурата данни Disjoint-Set Forest (която се ползва в алгоритъма на Крускал) за да знаем свързаността на графа. По-точно, ще броим от колко компоненти е изграден той. В началото тази бройка е N, а при срещане на ребро, което свързва различни компоненти, намаляме тази бройка с единица (и обединяваме компонентите, естествено). В момента, в който броят компоненти стане 1, то вече графът е свързан и знаем, че горната граница е цената на последното ребро, което сме ползвали.

Всяко "свързване" в тази структура данни е почти константно, така че можем да очакваме това решение да е съвсем малко по-зле от O(M*M). За да улесня малко състезателите (а и да дам малко повече точки на по-бавните решения), в авторското решение нарочно не ползвах по-оптимизираната версия на Disjoint-set Forest, а съвсем тривиалната такава. В нея операциите на теория са O(logN) (вместо почти O(1)), но на практика можете да очаквате да са доста по-бързи, отново почти константни.

Цялата сложност на това решение е O(M*logM + M*M), което се доминира от O(M*M). (Ако счетем, че операциите в Disjoint-Set Forest са O(1)).

Автор: Александър Георгиев

Нека за двойка позволени скорости (S_min, S_max) графът с ребра в този интервал е свързан. При увеличаване на горната граница S_max графът ще продължава да бъде свързан, а при увеличаване на долната граница S_min свързаността може да се наруши. Свързаността на графа може да бъде проверена за сложност O(N+M) чрез обхождане в ширина или дълбочина. Тъй като търсим минимална разлика S_max-S_min, нека увеличим S_min колкото можем повече. В момента, в който графът престане да бъде свързан, да увеличим горната граница S_max докато графът не стане свързан отново. Можем да използваме структура от данни „Опашка", която ще ни позволи да добавяме в опашката последователно-увеличаващите се горни граници S_max, както и да изваждаме от нея последователно-увеличаващите се долни граници. Всяка скорост за горна граница добавяме в опашката точно веднъж и я изваждаме също веднъж: общо M операции, след всяка от които бива проверявана свързаността на графа за O(N+M). Обща сложност: O(M^2).

Автор: Петър Иванов