Ditchers
(решение)

Задачата "Скатавки" ("Ditchers" (жаргон от английски): хора, които пропускат училище или училищни часове) показваше сравнително често-срещана ситуация в училище. Колко пъти ви се е случвало да правите училищен проект с някои от съучениците ви, като ви се е щяло някои от тях изобщо да не участват в проекта? Ако не ви се е случвало – значи вие сте този човек.

Greedy
Нека първо видим как бихме решили задачата, ако имаше само един училищен проект. Едно възможно решение е грийди такова – премахваме хора, започвайки от по-горните нива на дървото към по-долните, като през цялото време следим да спазваме всички ограничения нагоре по дървото (включително текущия връх, който гледаме). Това решение е що-годе очевидно и вярно. Колкото по-надолу в дървото е даден връх, толкова повече "шефове" има той, съответно толкова повече други върхове зависят от него. Започвайки от по-горните нива махаме върхове, които биха могли да нарушат по-малко зависимости.

Какво става, обаче, когато имаме два проекта? Вече горните върхове в едното дърво могат да са листа в другото, съответно нямаме строго дефинирана "височина" на всеки връх. Това бива и причина грийдито вече да не работи (или поне ми се ще да съм направил тестовете достатъчно силни за да не работи) – зависимостите от едното дърво влияят на зависимостите в другото дърво и обратно.

Идея за поток
Нека разгледаме по-абстрактно задачата. Трябва да премахнем максимален брой върхове, от които зависят други елементи в насочен граф. Нещо повече, имаме лимит на върховете – всеки връх задава дадено ограничение (целочислено число). По-опитните състезатели биха надушили в подобна задача решение чрез поток. Точно това е и решението, което ще ползваме в задачата.

За хората, които не са срещали задачи с потоци, сравнително честа техника е, когато имаме лимит на върховете, а не на ребрата, да разбием всеки връх на два нови ("вход" и "изход"), като поставим насочено ребро от входа до изхода с капацитет равен на капацитета на върха. Всички входящи ребра насочкваме към "входа" на съответния връх, а всички изходящи излизат от неговия "изход". Така увеличихме двойно началния граф, сведохме задачата до стандартна само с капацитети на ребрата (но не и на върховете).
Тази идея можем да приложим и тук, но това не е нужно – тъй като графът е по-специфичен (дърво), то имаме или само едно входящо ребро към всеки връх или само едно изходящо (зависи в коя посока гледаме дървото).

Решение за едно дърво
Нека разгледаме отново по-простия вариант с едно дърво и да видим как бихме построили графа, за да намерим отговора, ползвайки flow. Оказва се, че има два начина, по които можем да направим това.

Първи начин
Флоуът ще започва от корена на дървото и ще се оттича през върховете (тоест ще върви в посока към листата). За да лимитираме капацитетите на всеки връх, ще направим входящото ребро в корена на всяко поддърво да е с капацитет, равен на

броя деца в това поддърво (броейки него самия) минус лимита, който е зададен за това поддърво.
Наистина, ако дадено поддърво има 7 върха, а лимитът на корена на поддървото е 4, то можем да премахнем най-много 3 върха от това поддърво – затова капацитетът на върха (и съответно на входящото ребро в него) ще е 3. Тъй като коренът на дървото няма входящо ребро, но все пак има ограничения, ще добавим изкуствен source с едно единствено ребро, сочещо към корена на дървото, с адекватния капацитет.

Допълнително, от всеки връх ще изведем едно ново ребро с капацитет 1 към изкуствен sink, където се оттича флоуа. Ползването на това ребро е еквивалентно на това да премахнем дадения връх. Намирайки максималния поток между source и sink би ни дало максималния брой върхове, които можем да премахнем.

Втори начин
Обръщаме дървото наобратно, като флоуът започва от върховете и се оттича в корена на дървото. Тъй като тук флоуът е в обратната посока, капацитетът на реброто от даден връх към бащата ще е просто лимитът на дадения връх. От изкуствен source добавяме ребро с капацитет 1 към всеки връх в дървото (чието ползване би означавало премахването на съответния връх). От корена на дървото добавяме ребро с капацитет, равен на лимита на корена към изкуствен sink. Отново, размерът на потока от source до sink дава максималния брой премахнати върха, които можем да имаме.
Този начин е малко по-прост (тъй като не се налага да броим децата във всяко поддърво, макар че това не е особено трудно).

Решение за две дървета
Guess what! За решението на задачата ще ползваме двата начина, прилагайки единия в едното дърво, а другия – в другото! Всеки връх от едното дърво ще бъде свързан със своя двойник в другото. Реално флоуът ще започва от корена на едното дърво и ще се оттича в корена на другото.
Това може да си го представите като нарисувате едното дърво в прав вид (строейки графа за поток по първия начин), след което нарисувате второто дърво, обърнато, под първото (строейки графа по втория начин). Nice, а?

Разбира се, нужна е (относително) бърза имплементация за флоу. Алгоритъмът на Диниц върши чудесна работа, но дори малко по-бавни биха се справили за дадените ограничения. Сложността на алгоритъма е O(N^3) (тъй като графът е обединение на две дървета, демек броят ребра е равен на броя върхове). Сложността на алгоритъма на Диниц е O(M * N^2), но тъй като M = N получаваме O(N^3).
На пръв поглед това е страшно много, но на практика това върви много по-бързо. Наистина, ако ползваме по-лесната имплементация с намиране на пътища с капацитет 1 между source-а и sink-а чрез DFS/BFS можем да видим, че максималният поток е N, като намирането на всеки път е O(N+M) = O(N). От тук можем да видим, че ще направим не повече от O(N^2) операции (а на практика – много по-малко, тъй като повечето пътища ще са относително кратки).

Автор: Александър Георгиев