Анализ

Първата подзадача е само за 10 точки. Тя е предвидена за хората, които не измислят нищо по задачата. Прави се пълно изчерпване за всеки вариант за разпределение на играчите за всяка от заявките. Сложността тук е $O(\boldsymbol{Q} * 2^{\boldsymbol{N}} * (\boldsymbol{N} + \boldsymbol{M}))$.

Втората подзадача е за 35 точки. Тук вече се почва същинското решаване на задачата. Това, което трябва да се измисли е как сравнително бързо да се намира цената на оптимално разпределение. Очевидно в задачата е зададен свързан неориентиран граф с върхове играчите и ребра – стойностите на приятелствата. Първо ще улесним това, което търсим. Нека означим с двумерния масив $val[2][\boldsymbol{N}]$ – колко допринася всеки играч за съответния отбор и с матрицата $a[\boldsymbol{N}][\boldsymbol{N}]$ – стойностите на приятелствата. Нека с $teams[\boldsymbol{N}]$ означим разпределението на играчите по отбори. Искаме да максимизираме следното: $\sum_{i=0}^{N-1} val[teams[i]][i] - \sum_{i,j=0}^{N-1} a[i][j]$, ако $teams[i] \neq teams[j]$. За да стане този израз по-удобен, ще го извадим от константната $\sum_{i=0}^{N-1}(val[0][i] + val[1][i])$, така ще получим, че трябва да минимизираме $\sum_{i=0}^{N-1} val[1 - teams[i]][i] + \sum_{i,j=0}^{N-1} a[i][j]$, ако $teams[i] \neq teams[j]$. Новата сума е по-подходяща за минимизиране, защото няма разлики. Нека променим началния граф малко, за да стане по-удобен за намирането на оптимума. Добавяме два нови върха, които ще „отговарят" за отборите. Върхът за отбора на добрите свързваме с всеки играч, като теглата на ребрата са стойностите за допринасянето на играчите за този отбор. Аналогично свързваме върхът за отбора на лошите с всеки играч, като теглата на ребрата са допринасянето на играчите за отбора на лошите. Върху новия граф задачата изисква намиране на следното: как да разделим върховете на две компоненти на свързаност (в първата компонента ще е върхът за отбора но добрите, а във втората – върхът за отбора на лошите), така че сумата на ребрата, които свързват върхове от едната и другата, да е минимална? Това се нарича минимален срез ($minimum - cut$) с фиксиран връх за всяка компонента. Да помислим първо защо наистина това ни дава търсеното разпределение. Понеже върховете, които отговарят за двата отбора, ще са в различни компоненти, то за всеки от върховете-играчи в среза ще участва точно едно от ребрата, които го свързват със специалните върхове. Освен това за върхове-играчи, които са в различни компоненти на свързаност (съответстващо на различни отбори), ребрата, които ги свързват се явяват точно такива, участващи в среза. Така получаваме, че наистина среза съответства на сумата, която искаме да минимизираме. Задачата, до която достигнахме е с по-известно решение. Има теорема, която твърди, че минимален срез с фиксиран връх за всяка компонента е точно еквивалентен на максималния поток в графа с източник и приемник – тези два върха. Това означава, че за да решим тази част от задачата е достатъчно да построим потоковата мрежа от видоизменения граф и да намерим максималния поток в него. След което, за да получим отговора, трябва да извадим това число от $\sum_{i=0}^{N-1}(val[0][i] + val[1][i])$. Тук сложността ще е тази на максималния поток, която с реализация на Диниц е $O(\boldsymbol{N}^2\boldsymbol{M})$, но в действителност алгоритъма върви много по-бързо, защото графите не са много пълни и ребрата – не чак с толкова голяма тежест.

Третата подзадача е само за 10 точки. Тя е дадена за хората, които не могат да измислят последната голяма и важна част в задачата. Вече се включват и заявки, които променят множеството на играчите. Специалното тук е, че има само заявки за махане на играчи, а от тип 3 са пренебрежимо малко на брой. Заявките, които са подходящи с максимален поток, са такива за добавяне на ребра, върхове и т.н. Понеже задачата е в офлайн вариант за заявките, можем първо да прочетем дадените заявки. След това започваме да ги обхождаме отзад-напред. Така махането на върхове ще стане добавяне на нови върхове. Отново сложността ще е тази, която е на максималния поток, като в предната подзадача.

Последната подзадача е за 45 точки. Тук вече трябва да решим задачата и когато възстановяваме

началното множество на играчите от време на време. Наивният подход би бил при всяка заявка за махане на играч да пускаме максимален поток в новата мрежа от начало (нулиран), но той не е особено бърз. Отново ще се възползваме от офлайн заявките и това, че добавянето на върхове е съвместима операция с максималния поток. Първо прочитаме заявките и ги запаметяваме с допълнителна информация за времето на заявката (поредността). Всяка заявка от трети и четвърти тип я превръщаме на приблизително $\boldsymbol{N}$ заявки за добавяне или премахване на играчи (техните времена слагаме едни и същи!). След това сортираме заявките по номер на играчите и при равни номера – по времето. Така можем да видим интервалите от време, в които присъства всеки от играчите. Нека представим това, като заявки с интервали. Ако имаме едно интервално дърво (с листа – моментите от време), можем да разположим тези заявки във върховете му. При обхождането му, когато достигнем листо, ще сме добавили нужните върхове. Но както казахме, максималния поток не е добър при махане на върхове, а когато се връщаме нагоре по дървото, трябва да правим точно това. За тази цел ще използваме стек, в който ще пазим промените. Промените, които пазим са всъщност за кои ребра се променя потока (това е единственото, което се променя при прогреса на алгоритъма). Оптимизация, която можем да приложим, е за едно пускане на алгоритъма да не добавяме в стека едно ребро няколко пъти, защото е достатъчно само първия път да знаем каква е била старата стойност на потока. Освен това всеки елемент на стека ще носи и информация за индекса на върха в интервалното дърво, при който е извикан потока. Така ще знаем при напускането на върха в интервалното дърво точно кои са направените промени при него. Също за всеки връх от интервалното дърво пускаме потока само 1 път – след като сме добавили всички играчи за върха. Една малко подробност е, че за да работи правилно алгоритъма с интервалите на присъствие на всеки играч, трябва преди алгоритъма да изпълним фиктивна заявка от тип 1 с време 0 (така добавяме всички играчи в началото). Така описаният алгоритъм, все още не е напълно оптимален. Частта, която ще подобрим е добавянето на новите върхове. Няма нужда да добавяме всеки път ребрата (и после съответно да ги махаме). Достатъчно е в началото в потоковата мрежа да включим всички върхове и ребра, като просто пазим кои върхове участват в текущия граф. Модификацията при алгоритъма на максималния поток е просто да не минаваме през върхове, които не участват в текущия граф. Сложността тук е по-неприятна за описване, защото зависи много от примерите и сложността на потока, но е по-добра от наивното решение тук, защото не пускаме празен поток всеки път и не променяме ненужно ребрата на потоковата мрежа, за всяко ново пускане. Примерна сложност е: $O(\boldsymbol{QN} + \boldsymbol{QN^2M})$, като както казахме второто събираемо е прекалено груба оценка за времето, което отнема за потока по принцип, а в случая още повече.

Автор: Илиян Йорданов