АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА
ПЕТЪК 13-ТИ

Задачата Friday може би е една от по-трудните задачи, давани в Б група. Макар и състезателите да могат да хванат 50 точки само чрез имплементация на стандартен алгоритъм, решението за 100 беше значително по-трудно.

Нека разгледам какво ни е дадено в задачата – граф, пътища с цени по тях и се търси най-къс път между два върха в графа. Това си е стандартна Dijkstra или в случая дори по-бавен алгоритъм, катонапример алгоритъма на Floyd. И всъщност тези алгоритми са достатъчни за намиране на отговор на задачата, когато денят не е 13-ти петък. Така състезателите можеха да хванат 50 сигурни точки (графът е достатъчно малък всеки от тях да работи достатъчно бързо). С някакви други допълнения (примерно изчерпване или greedy) можеха да се хванат и някои от другите тестове, така че задачата, макар и трудна, даваше възможност за изява дори с разнообразни (неавторски) решения.

Нека разгледаме какво можем да направим когато денят е 13-ти петък. Първото наблюдение, което можем да направим е, че намереният път ще ползва не повече от 12 ребра. Защо? Нека сме посетили K върха, като времената, в които сме пристигали са $T_1, T_2, \ldots, T_K$. Ако имаме два различни върха, за които $T_i \% 13 == T_j \% 13$, $i < j$, то можем да твърдим, че пътят между тях е с дължина, която се дели на 13. Нека, например, сме използвали ребра с дължини 4, 3, 8 и 2. Така $T_1 = 4$, $T_2 = 7$, $T_3 = 15$, T4 = 17. $T_1 \% 13 == 4 == T_4 \% 13$. Наистина, пътят между тях е с дължина 3 + 8 + 2 == 13, което се дели на 13. Де факто, това лесно правило в модулната аритметика не би трябвало да затрудни състезателите в тази група, но по-скоро трудното в случая е да се сетят да го ползват. Как ни помага това? Ами очевидно, времето на пристигане във всеки връх ще дава различно число по модул 13 – в противен случай бихме имали отсечка от пътя, която Ели не може да използва. Тъй като преди да ползваме първия транспорт сме били във време 0, то в най-лошия случай ще ползваме най-много 12 ребра (и така ще изчерпим всички възможни числа по модул 13.

Как можем да ползваме факта, че оптималният път (ако има такъв) е сравнително къс? Една от възможностите е да пробваме някакъв тип изчерпване, което би могло да хване точки (а даже и да реши задачата, ако е достатъчно умно). Но авторското решение не разчита на такива „несигурни" методи.

Вариантът, който ще разгледаме, е базиран на малко „хакове" в графи и динамично оптимиране. Преди да направим динамичното, трябва да „разширим" графа – тоест да мултиплицираме върховете и ребрата му, като така го правим по-голям, но и по-удобен за ползване. Нека видим какво ни трябва да знаем, докато сме във всеки връх: освен, логично, кой е въпросният връх, също така и какви остатъци по модул 13 сме ползвали в досегашния път (както казахме, не можем да ползваме един остатък повече от веднъж). Тези остатъци са 13, но вече споменахме, че в началото сме винаги с време 0 -- тоест валидните остатъци всъщност са 12. Един стандартен начин да пазим множество от сравнително малко елементи е чрез битова маска. В случая тя ще има 12 бита, тоест ще е число от 0 до 4095. Така разделяме всеки от началните ни върхове на 4095 нови (ихаа) в който граф можем или да пуснем алгоритъм на Dijkstra (който на някои от тестовете би могъл да time limit-не), или да приложим динамично оптимиране. Защо можем да приложим динамично в разширения граф, но не и в началния? Първо, в началния нямахме информация кои остатъци сме ползвали в досегашния път и кои не. Второ, защото началният

граф би могъл да е цикличен, докато разширеният не е (след разширяването вече е DAG). Наистина, при ползването на ребро добавяме (винаги) по един бит в битовата маска на използваните остатъци по модул 13. Така стейтът на динамичното би бил (връх)(текущ_остатък)(битова_маска_на_използваните_остатъци). От всеки стейт трябва да проверим всички излизащи ребра и да отидем в съответните нови стейтове. Забележете, че не пазим само най-късите ребра, както бихме направили при Dijkstra или Floyd, ами за всеки от възможните остатъци пазим най-късото ребро, което дава този остатък. Така за всеки връх пазим най-много N * 13 ребра. Така сложността на алгоритъма е равна на броя стейтове умножен по броя ребра, излизащи от връх в разширения граф. Броят стейтове е, очевидно, $N * 13 * 2^{12}$, от което стигаме до извода, че сложността на този алгоритъм би била $O(N * 13 * 2^{12} * N * 13)$ или $O(N^2 * 2^{12} * 13^2)$. Спорно е дали можем да игнорираме тези „константи" в сложността, тъй като (в случая на задачата) N може да расте най-много до 50, така че те са съизмерими с него. Макар и това число да е сравнително голямо, на практика алгоритъмът работи значително по-бързо (тъй като ограничението на ребрата 2500 прави невъзможно от всеки връх да излизат 50 * 13 ребра).

Автор: Александър Георгиев