Browse
(решение)

Задачата Browse беше по-сложна версия на задачата в C група Graze (подсказка за което беше, че и двете думи означават „паса"). И докато там и овцете и кошарите бяха зададени като точки в 1D пространството, тук бяха зададени в 2D. С какво задачата е по-сложна в този вариант? Ами грийдито, което приложихме там не работи тук. Нещо повече, изисква се относително по-сложен алгоритъм за решаването на тази задача.

Какви са общите неща между задачите? И в двете се изисква двоично търсене по отговора, за да сведем задачата до по-лесна такава. Ако не ползваме binary search тази задача ще изисква алгоритъм поне толкова сложен, колкото максимален поток с минимална цена (min-cost max-flow). Но дори с него задачата не е тривиална. За сметка на това ако ползваме двоично търсене, на всяка итерация можем да проверим дали има валиден matching между овцете и кошарите, който да спазва ограниченията (до K овце в кошара). Това става със значително по-простия (и бързия) алгоритъм за max flow (в случая max matching). Коя от неговите имплементации да ползваме ще обсъдим малко по-нататък.

Нека първо видим защо и как това решение би работило. Нека използвайки двоичното търсене сме фиксирали отговора L – тоест всяка овца може да измине най-много L единици. Тогава строим следния граф. От изкуствен връх (source) слагаме ребро с капацитет 1 до всяка от овцете (всяка трябва да бъде ползвана точно веднъж). От всяка овца слагаме ребро до всяка ДОСТИЖИМА кошара (тоест до всички кошари, които са най-много на L единици) с капацитет отново 1 (всъщност този капацитет няма значение, стига да е по-голям или равен на 1). От всяка кошара строим по едно ребро към втори изкуствен връх (sink) с капацитет K. Това ни гарантира, че всяка кошара ще бъде „ползвана" най-много K пъти. Сега намираме максималния поток от source до sink. Ако той е равен на N (тоест равен на броя на овцете), то всяка овца е успяла да „намери" достатъчно близка кошара, която не е препълнена. Така връщаме true, тоест разстоянието L стига за да изпълним задачата. Ако пък флоуът е по-малко от N, то поне една овца не е намерила подслон. В този случай връщаме false, тоест разстоянието L е недостатъчно. Продължаваме binary search-а докато намерим минималната стойност на L, за която максималният поток е равен на N.

Така, вече намерихме всички основни части от алгоритъма, който решава задачата за 100 точки. Нека сега разгледаме детайлите и потенциалните „спънки".

Първата спънка е, че алгоритъмът за намиране на максимален поток е относително бавен. Тъй като не търсим поток а мачинг има няколко

алгоритъма, със сложност около O(N^3). Най-простият от тях е да пускаме DFS-та от source-а до sink-а, по пътища, които все още имат капацитет (augmenting paths). Тъй като на всеки такъв път намираме кошара за точно 1 овца, а едно DFS е със сложност O(M), то сложността е O(N*M). За съжаление задачата е така зададена, че в някои тестове графът става почти или изцяло пълен (тоест M = N*N). Тогава се достига и въпросната сложност O(N^3). Съществуват други алтернативи. Например да ползваме BFS вместо DFS (много лоша идея, макар и да е теоретично със същата сложност). Друга алтернатива е алгоритъмът на Диниц. Той е по-бърз от гореописания и би хванал ако не пълен брой, то почти пълен брой точки (без един тест). Алгоритъмът Push-Relabel (поне този, който аз написах) не се представи особено добре и ще time limit-не на 2-3 теста.
Често когато имаме задачи с мачинг най-простият алгоритъм е най-добър. В случая можем да ползваме една евристика, която значително подобрява бързодействието на алгоритъма със DFS-тата. Тя е преди да почнем да търсим augmenting paths да свържем възможно най-много върхове по greedy начин. Това не би дало оптимален отговор, но в доста случаи би дало близък до оптимален отговор (гарантирано поне ½ от него). Но това ни намаля броя на DFS-тата поне на половина! Така този алгоритъм става най-бърз от изброените 4.

Все още задачата обаче не (би трябвало да) върви за 100 точки. Получава TL на 1 до 3 теста. Какво да правим? Нека разгледаме binary search-а. Какво можем да оптимизираме в него? Можем ли да оптимизираме binary seach!? Както се оказва – можем. Изискания отговор е с 6 цифри след десетичната точка, като може да има до 3 цифри преди нея. Това прави 9 цифри, или приблизително 32 итерации за да постигнем желаната точност. Реално за сигурност опитните състезатели обикновено използват малко повече – примерно 50 итерации. Да видим как можем да оптимизираме този брой. Трябва да направим наблюдението, че отговорът ще е равен на времето нужно на най-бавната овца. Тогава ако разгледаме ВСИЧКИ възможни времена (на всяка овца до всяка кошара) едно от тях ще бъде и отговорът на задачата. Тъй като и овцете и кошарите бяха до 500, то трябва да разгледаме 500 * 500 = 250000 възможности. Ако ги сортираме в масив можем отново да правим binary search по тях. Но забележете, че този път правим binary search по ИНДЕКС (който е цяло число) вместо по отговора (който е реално число). Така броят итерации нужен за изцяло прецизен отговор би бил $\log_2(250000) = 18$. Току-що с това лесно наблюдение ускорихме програмата си двойно. Допълнително можем да разглеждаме само уникални стойности от този масив (очаква се, че доста от разстоянията ще се повтарят). Но това не ни помага за worst case-а, когато всички разстояния са различни. Все пак е подобрение за някои от тестовете (например този, в който всички овце и всички кошари са в една и съща точка). Тогава всяка стъпка от binary search-а създава пълен граф и итерациите са възможно най-бавни.

Така постигнахме алгоритъм със сложност O(logN * N^3). И да, това на пръв поглед е много, но с дадените оптимизации (и имайки предвид, че флоуът доста често върви значително по-бързо) е достатъчно решението ни да хване пълен брой точки.

Защо беше дадено ограничението за 50% от тестовете N, M, K <= 15? Съществува алтернативно решение, базирано на динамично оптимиране . Него няма да описваме подробно, само ще споменем, че стейтът е [кошара][свободни_места_в_кошарата][маска_на_овцете].

Очакването ми е задачата да не затрудни особено добрите състезатели (тъй като и двете части от нея са сравнително стандартни). Въпреки всичко по-неопитните може да не се досетят за binary-то или да не могат да имплементират flow-а. Също така очаквам някои от добрите състезатели да не хванат всички тестове, поради лоша имплементация и пропускане на някои от наблюденията, които ползвахме. За това те ще бъдат ощетени с най-много 15 точки.

Автор: Александър Георгиев