End-To-End

(Анализ)

Задачата *End-to-End* беше относително стандартна (алгоритъм на Дейкстра) като за състезание от сорта на контрола за национален отбор. Въпреки това, състезателите трябваше да се сетят как да разширят графа по такъв начин, че Дейкстра да намира желания отговор, а също и как да забързат решението си, тъй като стандартната имплементация с Heap беше твърде бавна.

Тъй като не знаем с коя цифра ще е оптималният отговор, за съжаление трябва да пробваме всяка от тях. От тук нататък в анализа ще разглеждаме решение за конкретна фиксирана цифра X, с която искаме да намерим отговор. Някои от по-наблюдателните състезатели сигурно са забелязали, че цифрите 4 и 5 имат далеч по-голям шанс да дадат оптимален отговор (тъй като разликата им във всяка посока е най-много 5). Въпреки това, има и много тестове, в които отговорът е друга цифра, а тъй като тестовете са групирани, това наблюдение не ни върши много работа.

Нека първо разгледаме под-задачата, в която пътищата на поне едно от оптималните решения (тъй като потенциално има много такива) се срещат в точно една клетка. При него можем да намерим най-късите пътища от всяка от четирите страни до всички клетки на матрицата, след което за всяка клетка да проверим каква е сумата на пътищата до всяка от страните. Това решение е относително просто, тъй като изисква единствено да напишем Дейкстра, в която в началото слагаме всички клетки от първия ред, най-дясната колона, последния ред или най-лявата колона (4 пускания на алгоритъма). Броят ребра е съизмерим с броя върхове в този граф (тъй като от всяка клетка имаме най-много 4 продължения). Така сложността на едно пускане на алгоритъма на Дейкстра е със сложност O(N * M * log(N * M)) по 4 пускания (за всяка страна) по 10 цифри, което в дадените ограничения по време би минало – за тестовете, в които има оптимален отговор, срещаш се в една клетка.

За да решим задачата в генералния случай ще трябва да "разширим" графа. Първо, ще изчислим какво е разстоянието от най-лявата и най-дясната колона до всяка клетка. След това ще почнем да строим път от най-горния ред до най-долния такъв (отново с Дейкстра), само че пазейки и информация дали вече към този път сме свързали лявата и дясната страна (по един бит допълнителна информация). Така стейтът ни става четиримерен: [ред][колона][връзка_в_ляво][връзка_в_дясно], което откъм размер е [500][500][2][2]. В този разширен граф отново пускаме Дейкстра, като от всеки стейт, освен да продължим в четирите посоки, трябва да пробваме и да свържем пътя с лявата и дясната страна на матрицата (ако вече не са). В момента, в който се озовем в клетка в най-долния ред, и сме свързани с лявата и дясната страна, сме намерили път и можем да спрем дейкстрата.

Сложността на този алгоритъм е O(N * M * log(N * M)) за първоначалните Дейкстри от най-лявата и най-дясната колона, плюс още O(N * M * 4 * log(N * M * 4)). Цялото това нещо трябва да направим 10 пъти (за всяка цифра), като имаме и по 6

продължения от всеки стейт (четирите съседни клетки, както и да свържем лявата или дясната колона).

Това вече е много (при все, че има и константа от приоритетната опашка, която не отчитаме). Макар и да би хванало тестовете, където N, M <= 250 (и да може да се съчетае с тези, в които отговорът може да бъде намерен от една клетка) – това ни дава само 60 точки. Как да ги добутаме до 100?

Най-важното нещо в задачата беше да забележим, че можем да направим *линейна* Дейкстра – което можем, когато цената на ребрата е цяло число и е относително малка. В случая всяко ребро е с цена най-много 9, тоест всеки път от клетка до клетка е с цена най-много (N + M) * 9. Реално и отговорът от произволна клетка е не по-голям от (N + M) * 5 – просто променяме реда и колоната на клетката в матрицата да съдържа петици и имаме отговор!

За да направим Дейкстрата линейна, ще заменим приоритетната опашка с масив от обикновени опашки. Размерът на масива ще е (N + M) * 9 (което е максималният отговор при избор на цифра 0 или 9). След това си представете, че разглеждаме стейт с цена X. Ако искаме да отидем на съседна клетка, чиято "цена" за конвертиране към цифрата, която ползваме е 4, новата цена ще стане X + 4. Ако пък искаме да свържем лявата страна и това е с цена 100 от текущата клетка, новата цена ще е X + 100. Вместо да вкарваме новия стейт в приоритетната опашка (както правим по принцип), просто го вкарваме в опашката, която е на индекс X + 4 и X + 100, съответно. Обхождаме индексите на масива в нарастващ ред (което реално са и цените в нарастващ ред). Така се отърваваме от логаритъма (и константата) на приоритетната опашка, като жертваме това, потенциално да разглеждаме празни опашки. Все пак, тъй като (N + M) * 9 (броят опашки) е значително по-малко от N * M * 2 * 2 (броят стейтове), сложността се доминира от броя разглеждани стейтове (тоест, логаритъмът наистина се маха).

Откъм сложност получаваме O(N * M), която обаче е твърде измамна, тъй като имаме много константи: x10 (за всяка цифра), x2 (за връзката с лявата страна), x2 (за връзката с дясната страна), x6 (за всяко възможно продължение от всеки стейт). Допълнително, имаме и стандартните константи, произлизащи от работата с опашки, масиви и памет. Дори без тях, знаем, че трябва да направим поне 60,000,000 сложни операции, което води и до относително големия тайм лимит (1-2 секунди, в зависимост от машината).

Няколко оптимизации, които можем да направим:

1. Да ползваме `short` вместо `int` за масивите с разстоянията, тъй като отговорът е относително малък. По-малкото ползвана памет позволява по-добро използване на кеша.
2. Да обхождаме цифрите в най-вероятния ред на намиране на оптимален отговор и да прекъсваме Дейкстрата в момента, в който стигнем до по-голям отговор. Такъв ред е например: {4, 5, 3, 6, 2, 7, 1, 8, 0, 9}.

Автор: Александър Георгиев