АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА ПЪТИЩА

Задачата Roads съчетава два алгоритъма, всеки от които даден в най-простия си вариант. За да хванат пълен брой точки на задачата, състезателите от C група трябваше да съумеят да напишат безгрешно и двата и да ги съчетаят. Все пак бях предвидени и 50 (даже 55-60 точки, при добра имплементация) при написване на изчерпване, което наистина брои пътищата.

Нека разгледаме първо как бихме могли да хванем половината точки. Ограничението, което ни интересува, е че най-късият път ще има най-много 14 стъпки. Ако пробваме ВСИЧКИ възможни пътища (тоест на всяка стъпка пробваме 4-те посоки) бихме могли лесно да намерим не само колко дълъг е най-късият път, ами също така колко пътища с тази дължина има. Това обаче изглежда сравнително бавно, тъй като $4^{14} = 2^{28} = 268435456$, което най-вероятно би time limit-нало. Нека обаче забележим, че в един най-къс път не бихме отишли в дадена клетка и после да се върнем в тази, от която сме отишли. Така че ако си пазим от коя посока сме дошли в дадена клетка, не трябва да разглеждаме всички 4 продължения, ами само 3. Така броят нужни операции става $3^{14} = 4782969$ – достатъчно малко за да влязат многократно в time limit-а. Де факто имаше няколко теста, в които най-късият път е малко над 14, които също биха могли да бъдат хванати с това решение. Сложността на описания алгоритъм е експоненциална (тоест $O(3^L)$, където L е дължината на най-късия път).

Нека сега разгледаме и истинското решение. Първо, трябва да изчислим колко дълъг е най-късият път от Ели до изхода. Това става с просто пускане на BFS. Трябва да внимаваме за случая, когато от Ели до изхода няма път – в този случай очевидно броят най-къси пътища е 0. Ако това е случаят, можем директно да изпечатаме 0 и да прекратим програмата. Нека има път и той е с дължина L. Трябва да открием колко пътища от Ели до изхода има, които са точно с дължина L. Както често се случва в задачи, в които има „по колко начина еди какво си", и тук решението беше чрез динамично оптимиране.

Да разгледаме как ще построим динамичното в тази задача. Един от вариантите е да пазим стейт от тип [row][col][remMoves] – тоест къде се намира Ели в момента и колко хода й остават докато стигне до изхода. Ако броят оставащи ходове стане нула, но клетката (row, col) не е изхода, то трябва да върнем нула. В противен случай трябва да пробваме всички възможни продължения (които не излизат от дъската или не отиват в непроходима клетка) с remMoves – 1. Това решение е хубаво и достатъчно бързо, но би заемало твърде много памет. Наистина, най-дългият път би бил с дължина около N * M / 2, тоест около 5000 в най-лошия случай (ако замъкът е с формата на змия или спирала). Тъй като ограниченията за N и M са до 100, това решение би изисквало 50,000,000 клетки, всяка от които от тип int, тоест около 200 мегабайта. Разбира се, в условието е дадено ограничение за 70% от тестовете, където N и M са до 50. Това ограничение беше предвидено именно за това решение, като състезателите можеха да направят таблица от типа на [50][50][1250], която да хване всички тези тестове. По-находчивите състезатели даже биха могли да ползват най-голямата таблица, която се събира в дадения Memory Limit, като например [100][100][400]. Наистина, почти винаги най-късият път не надхвърля N + M, така че, в интерес на истината, те биха могли да хванат 85 точки! Оценката по сложност на алгоритъма е лесна - тъй като от всеки стейти пробваме константен брой продължения (по-точно 4), то сложността на алгоритъма е броят клетки на динамичното, умножене по константа. Очевидно най-късият път няма да е по-дълъг от броя полета в замъка, които са най-много N * M,

следователно цялата сложност по време, а и по памет на това решение е $O(N^2 * M^2)$ (игнорирайки четенето на входа и пускането на BFS-то за да намерим най-късия път, които са O(N * M)).
Последните два теста (дъска 100 на 100 под формата на змия и под формата на спирала) бяха единствените два, където минималната дължина е по-голяма от 200.
Как да оптимизираме паметта на динамичното, така че да можем да хванем дори тези два теста? Трябва да забележим, че когато се движим по оптимален път, то винаги минималното разстояние до изхода от клетката, в която сме и от клетката, в която отиваме, намалява. Тоест, при оптимален път, никога не бихме отишли в клетка, която е по-далеч (или на същото разстояние) от изхода. Така можем да пуснем BFS не от Ели, ами от изхода, като за всяка клетка в замъка бихме намерили на какво разстояние се намира от него тя. Стейтът на динамичното ни може да се трансформира до [row][col], като следим да отиваме от текущата клетка само в клетки със стриктно по-малко разстояние до изхода (тези клетки ще са с разстояние точно 1 по-малко от текущата). Тази оптимизация се ползва в значително по-сложни алгоритми, като например алгоритъма на Dinitz за намиране на максимален поток в граф. Разбира се, тук се ползва за нещо много по-просто, така че не се очакваше от състезателите да са се сблъсквали с Диниц преди – просто малко конструктивно мислене. Сложността на това решение пък (както по време, така и по памет) е O(N * M), като имаме веднъж N * M за да прочетем входа, веднъж за да пуснем BFS-то, и веднъж за да пуснем динамичното.

Автор: Александър Георгиев