Състезание 2 – Златна дивизия

Задача 3 – Анализ

Най-простото решение на тази задача би било да се „симулират“ всички възможни обхождания, ходейки само по валидни пътища. Това обаче би било твърде бавно и ненужно. Самият факт, че в условието пише, че отговорите не винаги ще се събират в 64-битов тип, подсказва това. Прехвърлено и като времева сложност, това става немислимо много.

Забелязва се обаче, че през много от клетките се минава повече от веднъж при няколко такива позволени обхождания. Нещо повече през част от тях се минава много повече от веднъж (при достатъчно големи дъски), което подсказва, че по някакъв начин може да се използва пресметнатият вече резултат за „орязана“ част от въведената дъска (в съответния тест). И действително: най-ефективна се оказва стратегията да се пресмята, като се използва вече полученият резултат за по-малки подзадачи.

Всичко това води до техниката на динамичното оптимиране, като тази задача е един доста ясен пример за нейното приложение.

За решаването на конкретния проблем, ще създадем таблица с размерите на дадената (n x n), която да отговаря на следния въпрос: „По колко позволени пътя мога да достигна от тази клетка (която гледаме) до търсената (тази, която е най-долу и вдясно), движейки се само надолу и надясно?“ По този начин, когато намерим стойността в най-горната лява клетка (тази, от която тръгваме), ще можем да отговорим на въпроса от задачата.

Какво знаем в самото начало? Ами очевидно от „крайната“ клетка (тази, която искаме да достигнем) до самата нея може да се достигне по точно един единствен начин => в клетката с координати за съответно ред и колона (n-1,n-1) (ако броим от 0) можем да запишем 1, а всички останали клетки на новосъздадената таблица да инициализираме с 0.

Обхождането на останалите клетки ще започнем от нашата финална цел до началото (защото само в края ни е известна търсената бройка). Независимо дали обхождането ще се направи последователно по редове или по колони, крайната клетка винаги ще е (0,0).

За яснота, ще покажем обхождане по редове. Първо ще се обходи последният ред от клетката с координати (n-1,n-2) до (n-1,0), тъй като клетката с координати (n-1,n-1) вече е попълнена. Следва предишният ред от (n-2,n-1) до (n-2,0) и така нататък до най-горния ред – от (0,n-1) до (0,0), където ще се съдържа и отговорът на задачата.

За всяка клетка се задават следните въпроси:

1. Ако се отиде надясно с толкова стъпки, с колкото е указано във входната таблица, ще се стигне ли до валидна клетка (т.е. такава, която е в рамките на таблицата)
2. Ако се отиде надолу с толкова стъпки, с колкото е указано във входната таблица, ще се стигне ли до валидна клетка

Ако отговорът на кой да е от тези въпроси е положителен, то към стойността на текущата клетка трябва да се добави стойността на съответната валидна клетка (съответно надясно и/или надолу).

Това решение без никакви оптимизации би проработило добре дотогава, докато отговорите не започнат да излизат от използвания тип.

За да се избегне този проблем, е нужна имплементация на т.нар. „дълги числа“, използвайки ограничението от условието, че дължината им няма да бъде по-голяма от 100 знака (или пресмятайки в най-лошия случай колко става тази дължина). Една примерна имплементация е да се използват масиви от символи, на които всеки елемент отговаря на съответната цифра на даденото число. Събирането на такива „дълги числа“ става аналогично на правилата, учени в училище.

Такова решение би имало сложност не по-лоша от $O(N^2 x MAX\_LEN)$, където MAX_LEN е максималната допустима стойност на дължината на числата. Това е практически недостижима горна граница на сложността, тъй като не всички числа ще имат по максимален брой (сто) цифри, но и в този си вид се събира в дадените ограничения.

Груба оценка на използваната памет би била $N^2 x 32bits$ за входните данни (заб. може да се използват и по-малки типове, тъй като са числа в интервала от 0 до 9 включително) и $N^2 x MAX\_LEN x 32bits$ (ако се използва числов тип за „дългите числа“; отново могат да се ползват и по-малки). Това при дадените ограничения за n е под 4MB, което също е идеално за нашия случай.