АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА МАКСИМАЛНА СУМА

Задачата е базирана на една стандартна задача за динамично оптимиране. А именно, намирането на най-дългия (най-тежкия, с най-голяма сума) път от 1, 1 до N, N, вървейки **само** надолу и надясно (това е задачата в 20% от тестовете). Въпросната задача се решава чрез рекурентната зависимост:

dp(i, j) = a[i][j] + max(dp(i-1, j), dp(i, j-1)).

За целта на нашата задача е нужен още един аргумент – а именно, колко хода „наляво“ или „надясно“ сме направили до този момент. С други думи, можем да напишем горната зависимост по този начин:

dp(i, j, 0) = a[i][j] + max(dp(i-1, j, 0), dp(i, j-1, 0)).

Т.е. нашата задача е същата като гореспомената стандартна при K = 0. Както вече казахме, решение на задачата при K = 0 носи 20 точки. Други 20 точки носи пълно изчерпване (намиране на всички възможни пътища от 1, 1 до N, N, включително движейки пула и K пъти наляво или нагоре и сравнявайки дължините на тези пътища). Логично, комбинация от пълно изчерпване и решене на задачата при K = 0 би донесло 40 точки на състезателя.

Решението за 100 точки е отново базирано на метода на динамичното оптимиране. Нека сме решили задачата за K = 0. Т.е. ако имаме dp[i][j][k], което означава „най-дълъг път от 1, 1 до i, j с използвани най-много k на брой „наляво“ или „нагоре“, сме сметнали цялата тази таблица за k = 0. Тогава можем да сметнем dp(i, j, 1) по следния начин:

```
dp(i, j, 1) = a[i][j] + max(
        dp(i-1, j, 1),
        dp(i, j-1, 1),   // Тези две възможности са същите като при стандартната задача, т.е. не ползваме само ходове „надолу“ и „надясно“
        dp(i+1, j, 0),
        dp(i, j+1, 0)    // Тези две възможности са когато правим ход „наляво“ или ход „надясно“
)
```

Така можем да обобщим рекурнатната зависимост по следния начин: Ако трябва да сметнем dp(i, j, k), което ще рече да имаме пула в клетка (i, j) местейки го надолу и надясно колкото пъти искаме и местейки го наляво и нагоре точно k пъти:

```
dp(i, j, k) = a[i][j] + max(
        dp(i-1, j, k),
        dp(i, j-1, k),
        dp(i+1, j, k-1),
        dp(i, j+1, k-1)
)
```

Имайки тази рекурентна зависимост, започваме да смятаме задачата първо за k = 0, после за k = 1, … , като смятайки таблицата за всяко различно к, отново обхождаме (както при стандартната задача) ред по ред (отгоре-надолу), а всеки ред -- отляво-надясно (тъй като смятайки стойността за i, j, ще ни трябват вече сметнати стойностите (**за същото k**) i-1, j и i, j-1) Накрая трябва да вземем максималното dp(N, N, k) за к = 0, к = 1, ..., к = К за отговор на така зададената задача. Важен момент е състезателите да покажат добра техника и да не пазят таблиците за всички възможни k, а само тези които ни трябват (а те са именно – настоящата, която смятаме и предходната), иначе няма да се поберат в ограниченията за памет.

Описаният алгоритъм е със сложност O(N^2 * K).

Автор: Момчил Иванов