Розв'язання задачі «Альтернативні шляхи»

Автор: Ілля Порубльов

Основні ідеї кількома словами: максимальна сума знаходиться динамічним програмуванням, кількості «гарних» шляхів рахуються через рекурентну формулу від трьох параметрів: номера рядка, номера стовпчика та «відставань» суми шляху від максимально можливої.

1. Знаходження максимально можливої суми

Знаходження (лише) максимально можливої суми — відома задача динамічного програмування.

Прочитаємо вхідні дані у двовимірний масив `mass` (*таблицю даних*) і заведемо масив `est` (*таблицю оцінок*), елементи котрого будуватимемо за правилами:

1. елементи 1-го рядка та 1-го стовпчика — за формулами

$$\texttt{est}[1,j] = \sum_{k=1}^{j} \texttt{mass}[1,k], \quad \texttt{est}[i,1] = \sum_{k=1}^{i} \texttt{mass}[k,1], \tag{1}$$

   тобто суми всіх елементів рядка (стовпчика) таблиці даних від першого по поточний включно;

2. решту елементів — за формулою

$$\texttt{est}[i,j] = \max\left(\texttt{est}[i-1,j], \texttt{est}[i,j-1]\right) + \texttt{mass}[i,j] \quad \text{при } i \geqslant 2,\ j \geqslant 2, \tag{2}$$

   тобто вибирається більший серед верхнього сусіднього та лівого сусіднього елементів таблиці оцінок, і до нього додається поточний елемент таблиці даних.

Наприклад, на рис. 1 $\texttt{est}[2,2] = 8$ будується як максимум з $\texttt{est}[2,1] = 6$ та $\texttt{est}[1,2] = 7$ плюс число самóї клітинки $\texttt{mass}[2,2] = 1$.

Лема 1. *Кожна клітинка таблиці оцінок, побудованої згідно (1)–(2), зберігає максимально можливу суму, яку можна набрати, дійшовши до даної клітинки.*

2	5	7	2
4	1	2	8
7	5	9	7

Таблиця даних

2	7	14	16
6	8	16	24
13	18	27	34

Таблиця оцінок

Рис. 1. Приклад

2. Знаходження кількості «гарних» шляхів

Очевидно, що: при $(i = 1)$ `or` $(j = 1)$ кількість «гарних» шляхів дорівнює 1; при $\texttt{est}[i-1,j] > \texttt{est}[i,j-1] + K$ усі «гарні» шляхи до (i,j)-ої клітинки проходять через $(i-1,j)$-шу, при $\texttt{est}[i-1,j] < \texttt{est}[i,j-1] - K$ — через $(i,j-1)$-шу; у випадку $|\texttt{est}[i-1,j] - \texttt{est}[i,j-1]| \leqslant K$, серед «гарних» шляхів є i шляхи через $(i-1,j)$-у, i через $(i,j-1)$-у.

Це наштовхує на думку, ніби кількість «гарних» шляхів рівна (відповідно) або кількості для $(i-1,j)$-ої клітинки, або $(i,j-1)$-ої, або сумі цих кількостей. Але насправді остання «гілка» цих міркувань неправильна.

На рис. 2, прийти у клітинку $(1;3)$ можна одним «гарним» шляхом (RR), у $(2;2)$ — двома (DR та RD). А якщо спробувати продовжити ці шляхи до клітинки $(2;3)$, то RRD (сума 15) та DRR (сума 13) виявляються «гарними», а RDR (сума 11) — ні: сумарне «відставання» суми шляху від максимально можливої склало $15-11 = 4$ $(> 3 = K)$.

знач. (`mass`) 1 оцінка (`est`) 1 шлях | сума · | 1	знач. (`mass`) 3 оцінка (`est`) 4 шлях | сума R | 4	знач. (`mass`) 8 оцінка (`est`) 12 шлях | сума RR | 12
знач. (`mass`) 5 оцінка (`est`) 6 шлях | сума D | 6	знач. (`mass`) 4 оцінка (`est`) 10 шлях | сума DR | 10 RD | 8	знач. (`mass`) 3 оцінка (`est`) 15 шлях | сума RRD | 15 DRR | 13

(при $K = 3$)

Рис. 2. Приклад переліку «гарних» шляхів. Вказані значення числа у клітинці, оцінка клітинки згідно (1)–(2) та повний перелік «гарних» шляхів до кожної клітинки (при K=3). "D" та "R" означають перехід на одну клітинку вниз та на одну клітинку праворуч відповідно

З'ясуємо, *звідки взялося* це «відставання». При побудові $\texttt{est}[2,2]$ маємо $\texttt{est}[1,2] < \texttt{est}[2,1]$; отже, шлях, що приходить до $(2;2)$ через $(1;2)$ буде гіршим за оптимальний на $\texttt{est}[2,1] - \texttt{est}[1,2] = 2$. Таким чином, шлях RD має «відставання» 2, а для шляхів DR та RR (всі переходи котрих відбуваються відповідно до (1) та (2)) «відставання» рівні 0.

Тепер розглянемо шлях RDR. На частині "RD" вже було «відставання» 2, до котрого додалася ще неоптимальність переходу $(2;2) \to (2;3)$ (величиною $\texttt{est}[1,3] - \texttt{est}[2,2] = 2$). Сумарно $2 + 2 = 4 > 3 = K$.

Тож введемо поняття *«затримки»*, що стосується кожного одиничного переходу ("R" або "D"):

1. якщо даний перехід єдиний (R у 1-му рядку або D у 1-му стовпчику), «затримка» рівна 0;
2. При $i \geqslant 2$, $j \geqslant 2$ для знаходження «затримок» достатньо обчислити:
   (a) більшу з попередніх оцінок $\texttt{max\_est} := \max(\texttt{est}[i-1,j], \texttt{est}[i,j-1])$,
   (b) нову оцінку $\texttt{est}[i,j] := \texttt{max\_est} + \texttt{mass}[i,j]$,
   (c) «затримки» $\texttt{add\_sh\_U} := \texttt{max\_est} - \texttt{est}[i-1,j]$ та $\texttt{add\_sh\_L} := \texttt{max\_est} - \texttt{est}[i,j-1]$.

Іншими словами, якщо даний перехід оптимальний (са́ме він вибирається згідно (2)), «затримка» рівна 0; якщо даний перехід *не*оптимальний (при виборі згідно (2) його оцінка строго менша оцінки іншого можливого переходу), «затримка» рівна різниці оцінок.

Цілком очевидна наступна лема:

Лема 2. *«Відставання» будь-якого шляху дорівнює сумі «затримок» усіх його переходів.*

2.1. Алгоритм підрахунку кількості «гарних» шляхів на основі рекурсії з запам'ятовуваннями

Введемо величину $T(i,j,d)$ — кількість шляхів з клітинки $(1;1)$ до клітинки $(i;j)$, «відставання» котрих не перевищує d (при $0 \leqslant d \leqslant K$).

При приході до клітинки згори, «відставання» шляхів збільшуються на `add_sh_U`, приході справа — на `add_sh_L`. Тому має місце співвідношення

$$T(i,j,d) = T(i-1,j,d-\texttt{add\_sh\_U}) + T(i,j-1,d-\texttt{add\_sh\_R}). \tag{3}$$

Звичайно, потрібно задати ще початкові умови:

1. «відставання» не може бути від'ємним, тому при $d<0$ вважаємо $T(i,j,d)=0$;
2. у 1-му рядку та 1-му стовпчику шлях єдиний, тому при $((i=1)\,\texttt{or}\,(j=1))\,\texttt{and}\,(d \geqslant 0)$ маємо $T(i,j,d)=1$.

Формулу (3) можна реалізувати просто рекурсивною функцією, але такий розв'язок, звичайно, не буде ефективним.[1] А от якщо зробити «мемоізовану» версію, тобто запам'ятовувати відповіді вже розв'язаних підзадач — щоб у разі повторного звернення до такої ж підзадачі не розв'язувати її заново, а підставляти готову відповідь, отримаємо *дуже* ефективний *за часом* алгоритм.

Звичайно, алгоритм, що працює з великим *три*вимірним масивом, не можна вважати ефективним за об'ємом пам'яті. Але виявляється, що (при вказаних в умові задачі та пам'ятці учасника обмеженнях) пам'яті все-таки вистачає.

Можна розробити ітеративний алгоритм, значно ефективніший за пам'яттю і не менш ефективний за часом роботи. Але він істотно громіздкіший, тому його опис буде поданий лише в більш повній версії, що буде роздана в електронному вигляді.

3. Оцінки часу роботи та об'єму пам'яті

Час роботи на різних вхідних даних однакового розміру може сильно відрізнятися. Тому доведеться сказати, що для найшвидших алгоритмів («мемоізованої» реалізації формули (3) та найкращої ітераційної реалізації кількість дій у гіршому випадку оцінюється як $O(M{\cdot}N{\cdot}K)$, але для багатьох вхідних даних оцінка не набагато перевищує $O(M{\cdot}N)$.

Об'єм пам'яті мемоізованої рекурсії становить $O(M{\cdot}N{\cdot}K)$; для ітеративних алгоритмів— $\theta(N{\cdot}(M{+}K))$.

[1] час роботи буде приблизно пропорційний добутку кількості *(!!!)* «гарних» шляхів на розміри поля