ОСОБЛИВОСТІ АЛГОРИТМІЧНО ДОЦІЛЬНОГО ОБЧИСЛЕННЯ РОЗВ'ЯЗКІВ КОМБІНАТОРНИХ ЗАДАЧ

Мазурок І. Є., Крачилова В. Д.
Одеський національний університете імені І. І. Мечникова

Для низки добре відомих комбінаторних задач існують (або можуть бути отримані) розв'язки у вигляді формули з використанням факторіалів та інших

швидко зростаючих функцій. Незважаючи на абсолютну математичну коректність таких розв'язків, безпосереднє їх кодування не є доцільним. Причиною є отримання великих або складно обчислювальних проміжних значень. У той же час спрощення формул не є можливим, через відсутність математичної символіки для запису формули у алгоритмічно доцільному вигляді. Таким чином виникає доволі складна задача алгоритмічно доцільного кодування комбінаторних формул [1].

Для прикладу, розглянемо задачу про кількість шляхів на прямокутному полі [2]. Припустимо, що в нас є прямокутна дошка розміром $n \times m$. Нам потрібно знайти кількість різних маршрутів з лівого нижнього кута у верхній правий. При чому рухатися можна лише у напрямку цілі (вправо або вверх). Таке обмеження робить кількість шляхів скінченною.

Розв'язок цієї добре відомої задачі легко отримати у вигляді формули:

$$\frac{(n+m-2)!}{(n-1)\times(m-1)}.$$

Оскільки чисельник раціонального виразу є факторіалом а знаменник добутком факторіалів, обидва значення є досить великими. У той же час ми знаємо, що відповіддю є натуральне число. Тобто чисельник можна поділити на знаменник і отримати частку, яка буде не порівняно невеликою. Окрім того безпосереднє підрахування чисельника і знаменника є обчислювально складною задачею яка не є необхідною для отримання відповіді. Причина такого явища у тому, що чисельник і знаменник складаються з великої кількості однакових множників які можуть бути скорочені.

Для вирішення задачі доцільно зайти $\max(m, n)$, нехай це буде n. Тепер виконаєм скорочення на $(n-1)!$

$$R = \frac{n \times (n+1) \times \ldots \times (n+m-2)}{1 \times 2 \times \ldots \times (m-1)}.$$

Обчислення цього раціонального виразу треба вести за схемою двох покажчиків: $i_1 = n \ldots (n+m-2)$ та $i_2 = 1 \ldots (m-1)$. При цьому ведучим є перший індекс. Якщо частковий добуток $R(i_1, i_2) \vdots i_2$ - ділиться націло на i_2, виконуємо ділення та виконуємо інкреент i_2. Якщо ні, виконуємо інкремент i_1 та переходимо до $R(i_1 + 1, i_2)$. Використанні такого способу дає можливість виконати обчислення за $O\big(\max(n, m)\big)$ та майже не використовує зайвих обчислень.

Якщо потрібно вирішити масову задачу для усіх $m \le M, n \le N$, то цей спосіб потребує $O\big(m \cdot n \cdot \max(n, m)\big)$. Обчислювальну складність можна значно зменшити використанням динамічного програмування [3]. Для збереження відповідей використаємо симетричну матрицю $R = \{r_{m,n} | 1 \le m \le M, 1 \le n \le N\}$, де $r_{m,n}$ – розв'язок задачі для заданих значень m, n. Для першого стовбця, та

першої строки матриці розв'язок очевидний $r_{m,1} = r_{1,n} = 1$. Інші елементи матриці будемо обчислювати послідовно за цілком очевидною формулою

$$r_{m,n} = r_{m-1,n} + r_{m,n-1}$$

оскільки рух можливий лише за одним напрямком по вертикалі та горизонталі. Таким чином для масової задачі ми отримуємо алгоритм, що потребує лише $O(m \cdot n)$ операцій. Цей алгоритм зовсім не використовую проміжних обчислень.

На прикладі однієї комбінаторної задачі ми розглянули підходи для скорочення обчислень та алгоритмічно доцільного кодування математично отриманих розв'язків.

Література

1. Tucker, A., Applied Combinatorics, (6rd. ed.), John Wiley, New York, 2012, 480pp.
2. Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2009) [1990]. Introduction to Algorithms (3rd ed.). MIT Press and McGraw-Hill, 2016, 1080pp.
3. Dimitri P. Bertsekas. Abstract Dynamic Programming, (2nd ed.), Athena Scientific, 2018, 324pp.