

Тема 20. Алгоритми комбінаторики

В минулих темах з комбінаторного аналізу було розглянуто основні задачі комбінаторики – знаходження кількості можливих сполучень, розміщень, перестановок та розбиттів. Але поруч із цією задачею постає наступна: систематично перебрати (або згенерувати) всі можливі сполучення, розміщення, перестановки або розбиття відповідно. Саме цьому питанню й присвячена дана тема.

20.1. Генерування перестановок

Кожній n -елементній множині A можна поставити у взаємно-однозначну відповідність множину $A' = \{1, 2, \dots, n\}$. Зручно спочатку генерувати перестановки n перших натуральних чисел, а потім замінити кожне число відповідним елементом множини A . Унаслідок цього отримаємо всі перестановки елементів даної множини A .

Існують різні алгоритми побудови всіх перестановок множини $A' = \{1, 2, \dots, n\}$. Розглянемо один з них. Цей алгоритм ґрунтується на послідовній побудові перестановок множини A' у лексикографічному порядку. Далі перестановку (a_1, a_2, \dots, a_n) для спрощення записів позначатимемо як $a_1a_2\dots a_n$.

На множині всіх перестановок (загальніше – на множині всіх кортежів довжиною n з елементами множини $A' = \{1, 2, \dots, n\}$) означимо **лексикографічний порядок**: $a_1a_2\dots a_n < b_1b_2\dots b_n$, якщо для якогось k , $1 \leq k \leq n$, виконуються співвідношення $a_1=b_1, a_2=b_2, \dots, a_{k-1}=b_{k-1}$, але $a_k < b_k$. У такому разі говорять, що перестановка $a_1a_2\dots a_n$ менша від перестановки $b_1b_2\dots b_n$, або перестановка $b_1b_2\dots b_n$ більша від перестановки $a_1a_2\dots a_n$. Якщо замість чисел $1, 2, \dots, n$ взяти літери a, b, \dots, z із природнім порядком $a < b < \dots < z$, то лексикографічний порядок – це стандартна послідовність, у якій слова довжиною n наведено в словнику.

Означення 20.1. Перестановку $b_1b_2\dots b_n$ називають **лексикографічно наступною** за $a_1a_2\dots a_n$, якщо не існує такої перестановки $c_1c_2\dots c_n$, що $a_1a_2\dots a_n < c_1c_2\dots c_n < b_1b_2\dots b_n$.

Наприклад, перестановка 23415 множини $\{1, 2, 3, 4, 5\}$ менша від перестановки 23514.

Алгоритм генерування перестановок множини $A' = \{1, 2, \dots, n\}$ ґрунтується на процедурі, яка будує перестановку, лексикографічно наступну за даною перестановкою $a_1a_2\dots a_n$. Покажемо, як це можна зробити. Спочатку припустимо, що $a_{n-1} < a_n$. Поміняємо місцями a_{n-1} та a_n і отримуємо більшу перестановку. Вона лексикографічно наступна, бо ніяка інша перестановка не більша за дану перестановку й не менша за отриману.

Наприклад, нехай 234156 – задана перестановка; тоді перестановка 234165 лексикографічно наступна.

Тепер розглянемо випадок $a_{n-1} > a_n$. Переглянемо останні три члени перестановки. Якщо $a_{n-2} < a_{n-1}$, то останні три члени можна переставити для отримання наступної перестановки. Поставимо менше з двох чисел a_{n-1} і a_n , яке, однак, більше, ніж a_{n-2} , на позицію $n-2$. Потім розмістимо число, яке залишилося, та a_{n-2} на останніх двох позиціях у висхідному порядку.

Наприклад, нехай 234165 – задана перестановка; тоді перестановка 234516 лексикографічно наступна.

Узагальнивши ці міркування, отримуємо такий алгоритм.

Алгоритм побудови лексикографічно наступної перестановки

1. Знайти такі числа a_j і a_{j+1} , що $(a_j < a_{j+1})$ та $(a_{j+1} > a_{j+2} > \dots > a_n)$. Для цього потрібно знайти в перестановці першу справа пару сусідніх чисел, у якій число, що ліворуч, менше від числа, що праворуч.
2. Записати в j -ту позицію таке найменше з чисел $a_{j+1}, a_{j+2}, \dots, a_n$, яке водночас більше, ніж a_j .
3. Записати у висхідному порядку число a_j і решту чисел $a_{j+1}, a_{j+2}, \dots, a_n$ у позиції $j+1, \dots, n$.

Обґрунтування алгоритму. Доведемо, що не існує перестановки, яка водночас більша від $a_1a_2\dots a_n$, але менша від побудованої за цим алгоритмом. Це означає, що побудована перестановка дійсно лексикографічно наступна за даною перестановкою $a_1a_2\dots a_n$. Справді, за

наведеним алгоритмом нова перестановка збігається зі старою в позиціях $1, \dots, j-1$. У j -й позиції нова перестановка містить a_k , а стара – a_j , причому $a_k > a_j$. Отже, нова перестановка лексикографічно більша від старої. Окрім того, вона перша в лексикографічному порядку з $a_1, a_2, \dots, a_{j-1}, a_k$ у позиціях з 1 до j . Стара перестановка остання з a_1, a_2, \dots, a_{j-1} у цих самих позиціях. Згідно з алгоритмом a_k вибирають найменшим з $a_{j+1}, a_{j+2}, \dots, a_n$, але більшим, ніж a_j . Отже, не існує жодної перестановки між старою та новою. ►

Наприклад, побудуємо перестановку, наступну в лексикографічному порядку за 362541. Остання пара чисел, у якій перше число менше за друге, – 25. Отже, розглянемо послідовність чисел 541. Серед них найменше число, яке більше від 2, це – 4. Тепер 4 запишемо на місце 2, а решту чисел 251 розмістимо на останніх трьох позиціях у висхідному порядку: 364125.

Щоб побудувати всі $n!$ перестановок множини $A' = \{1, 2, \dots, n\}$, починаємо з лексикографічно найменшої перестановки 123... n і послідовно $n! - 1$ разів виконуємо алгоритм побудови лексикографічно наступної перестановки.

Продемонструємо це на прикладі множини $A' = \{1, 2, 3, 4\}$. За наведеним алгоритмом буде побудована наступна послідовність перестановок: 1234, 1243, 1324, 1342, 1423, 1432, 2134, 2143, 2314, 2341, 2413, 2431, 3124, 3142, 3214, 3241, 3412, 3421, 4123, 4132, 4213, 4231, 4312, 4321. Як і очікувалось, кількість перестановок становить $4! = 24$.

20.2. Генерування сполучень

Розглянемо множину $A' = \{1, 2, \dots, n\}$. Сполучення без повторень з n елементів по r – це r -елементна підмножина множини A' . Позаяк порядок запису елементів множини неістотний, то домовимось записувати елементи в кожному сполученні у висхідному порядку: наприклад, $\{3, 5, 1\}$ будемо записувати як $\{1, 3, 5\}$. Отже, сполучення $\{a_1, a_2, \dots, a_r\}$ розглядатимемо як рядок чисел $a_1 a_2 \dots a_r$, причому $a_1 < a_2 < \dots < a_r$.

Як і для перестановок, покажемо, як за даним сполученням знайти наступне відповідно до лексикографічного порядку. Припустимо, що $n = 5$ та $r = 3$. Якщо можна збільшити останню цифру перестановки, то так і будемо робити. Тому, маючи рядок 123, його можна замінити на 124. Якщо ж маємо 125, останнє число збільшити не можна. Тому переходимо до наступного (справа) числа і дивимось, чи можна збільшити його. У даному разі це можна зробити: потрібно замінити 2 на 3. Проте ми прагнемо побудувати найменший рядок із тих, котрі більші 125. Тому збільшуємо останнє число (тобто 3) на 1 і записуємо результат у наступну позицію. Отже, перші два числа – 1 і 3, тому наступний рядок – 134. Припустимо, що є рядок 145. Останнє й передостаннє числа збільшити не можна. Проте перше число можна збільшити, тому замість 1 пишемо 2. Щоб зробити рядок мінімальним, як останні числа візьмемо 3 та 4, унаслідок чого отримаємо рядок 234.

Узагальнимо ці міркування. Значення останнього числа в рядку – найбільше можливе, якщо воно дорівнює $n - r + 1$. Якщо останнє число – найбільше можливе, то передостаннє – найбільше можливо, якщо воно дорівнює $n - r + (r-1)$ або $n - r + i$, де $i = r-1$ – позиція числа. Загалом, значення кожного i -го числа найбільше можливе, якщо числа праворуч від нього – найбільші можливі, і це значення дорівнює $n - r + i$. Отже, переглядаємо рядок справа наліво й визначаємо, чи дорівнює значення i -го елемента $n - r + i$ (це максимальне значення, яке може бути в позиції i). Перше значення, яке не задовольняє цю умову, можна збільшити. Нехай, наприклад, це значення дорівнює m і займає j -ту позицію. Збільшуємо m на 1, а значення кожного елемента, який стоїть після j -го, дорівнює значенню попереднього елемента плюс 1. Тепер можемо сформулювати потрібний алгоритм.

Алгоритм побудови лексикографічно наступного сполучення

1. Знайти в рядку перший справа елемент a_i такий, що $a_i \neq n - r + i$.
2. Збільшити знайдений елемент a_i на 1.
3. Встановити значення елементів в позиціях $j = i+1, i+2, \dots, r$ на $a_{j-1} + 1$.

Наприклад, нехай $A' = \{1, 2, 3, 4, 5, 6\}$. Знайдемо сполучення, наступне за $\{1, 2, 5, 6\}$ у лексикографічному порядку. Це сполучення подамо рядком 1256. Маємо $n=6, r=4$. Перший справа з таких елементів, що $a \neq 6 - 4 + i$, – це $a_2=2$. Для обчислення наступного більшого

сполучення збільшуємо a_2 на 1 й отримуємо $a_2=3$. Тепер нехай $a_3 = 3+1 = 4$ і $a_4 = 4+1 = 5$. Отже, наступне в лексикографічному порядку сполучення – те, що зображене рядком 1345.

Обґрунтування алгоритму. Доведемо, що наведений алгоритм дійсно будує наступне в лексикографічному порядку сполучення. Рядок чисел, яким подано лексикографічно наступне сполучення, відрізняється від рядка, що зображує дане сполучення, з позиції i , бо в даному сполученні в позиціях $i+1, i+2, \dots, r$ є максимально можливі числа. Отже, a_i+1 – найменше можливе число, яке можна записати в позицію i , якщо хочемо отримати сполучення, більше від даного. Тоді $a_2+2, \dots, a_i+r-i+1$ – найменші можливі числа, які можна записати в позиціях від $i+1$ до r . ►

Продемонструємо наведений алгоритм на прикладі множини $A' = \{1, 2, 3, 4, 5, 6\}$, коли потрібно знайти всі сполучення довжиною 4. За наведеним алгоритмом буде побудована наступна послідовність сполучень: 1234, 1235, 1236, 1245, 1246, 1256, 1345, 1346, 1356, 1456, 2345, 2346, 2356, 2456, 3456. Як і очікувалось, кількість сполучень становить $\frac{6!}{4! \cdot (6-4)!} = 15$.

Коротко зупинимось на питанні генерування всіх розміщень з n елементів по r . Знову розглядатимемо цю задачу лише для множини $A' = \{1, 2, 3, 4, 5, 6\}$. Один із можливих способів її розв'язання такий. Використаємо алгоритм генерування лексикографічно наступного сполучення для побудови r -елементних сполучень n -елементної множини A' . Після кожної стадії, коли побудовано чергове r -сполучення, застосуємо $r!-1$ разів алгоритм побудови перестановки за умови $n=r$ для побудови всіх перестановок елементів цього сполучення як r -елементної множини.

20.3. Генерування розбиттів множини

Опишемо алгоритм генерування всіх розбиттів множини. Ідею цього алгоритму найпростіше пояснити, сформулювавши його в рекурентній формі. Зазначимо спочатку, що кожне розбиття π множини $\{1, 2, \dots, n\}$ однозначно задає розбиття π_{n-1} множини $\{1, 2, \dots, n-1\}$, яке одержане з π після вилучення елемента n із відповідного блока (і вилучення порожнього блока, якщо елемент n утворював одноелементний блок). Навпаки, якщо дано розбиття $\sigma = \{A_1, \dots, A_k\}$ множини $\{1, 2, \dots, n-1\}$, то легко знайти всі такі розбиття π_n множини $\{1, 2, \dots, n-1, n\}$, що $\pi_{n-1} = \sigma$. Це розбиття:

$$\begin{array}{llll} \{A_1, & A_2, & \dots, & A_k, & \{n\}\} \\ \{A_1 \cup \{n\}, & A_2, & \dots, & A_k\} \\ \{A_1, & A_2 \cup \{n\}, & \dots, & A_k\} \\ \dots & \dots & \dots & \dots \\ \{A_1, & A_2, & \dots, & A_k \cup \{n\}\} \end{array}$$

Наведені міркування підказують простий рекурентний спосіб генерування всіх розбиттів. Якщо дано список L_{n-1} усіх розбиттів множини $\{1, 2, \dots, n-1\}$, то список L_n усіх розбиттів множини $\{1, 2, \dots, n-1, n\}$ утворюють заміною кожного розбиття σ в списку L_{n-1} на відповідну йому послідовність з наведених вище.

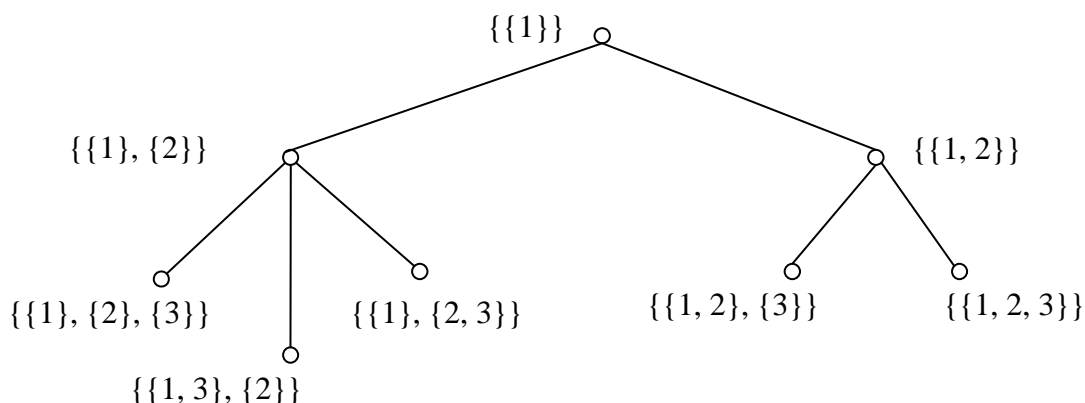


Рис. 20.1.

Наприклад, на рис. 20.1 показано формування списку всіх розбиттів множини $\{1, 2, 3\}$. Усього розбиттів $\Phi(3) = 5$, де $\Phi(n)$ – число Белла.