## 3. Перестановка (Данило Мисак)

Розглядатимемо неорієнтований граф на $N$ вершинах, ребро між вершинами $v$ та $w$ якого проведено тоді й лише тоді, коли серед записаних у вхідному файлі пар є або $(v, w)$, або $(w, v)$, або обидві. Задачу можна розв’язати, скориставшись пошуком у ширину чи пошуком у глибину на такому графі. При цьому можна використати два підходи: або жадібний (з пошуком у ширину чи в глибину), або рекурсивний (тут підійде лише пошук у глибину).

Розв’язання за допомогою **жадібного підходу** таке: спочатку маємо послідовність з одного-єдиного (довільного) числа $a$, після чого додаємо до неї числа в тому порядку, в якому проходимо їх пошуком у ширину чи в глибину (початкова вершина пошуку — число $a$). Кожне нове число ми відразу ж розміщуємо в послідовності або безпосередньо зліва, або безпосередньо справа від «суміжного» числа, з якого ми до даного числа дійшли. Зліва чи справа розміщувати число, визначається тим, у якому порядку дані два числа утворювали пару у вхідному файлі. Якщо граф зв’язний, утворена в кінці послідовність буде перестановкою чисел від $1$ до $N$ і, здійснивши операції у зворотному порядку, зможемо витерти з неї всі числа, крім останнього (числа $a$). А якщо граф незв’язний, то потрібної перестановки існувати не може. У цьому випадку утворена послідовність міститиме менше за $N$ елементів, що означатиме, що у вихідний файл слід вивести нуль.

Якщо при додаванні кожного нового числа перебудовувати всю послідовність (принаймні з місця, куди додано число), матимемо складність $O(N^{2})$. Замість цього можна зберігати не саму поточну послідовність, а індексний масив, у якому для кожного числа вказано число зліва від нього та справа від нього (або індикатор того, що зліва чи справа від числа немає іншого числа). Така структура подібна до двозв’язного списку. Для того щоб її модифікувати, додаючи нове число, не потрібно робити пошук: достатньо взяти інформацію з комірки індексного масиву, що відповідає числу, поряд з яким буде додано нове. Залежно від реалізації цей підхід вимагає $O(N+M)$ або $O(M)$ часу.

Інший спосіб розв’язати задачу полягає в **рекурсивному нарощуванні** масивів з перестановкою та порядком витирання чисел для піддерев. Почавши з довільної вершини-числа, робимо дії в такому порядку: рекурсивно заглиблюємось в усі вершини, що відповідають числам, які стоять у парах ліворуч від даного; додаємо до порядку витирання чисел усі ті дочірні вершини, в які ми заглиблювались, але в зворотному порядку; додаємо до перестановки-результату поточне число; рекурсивно заглиблюємось в усі вершини, що відповідають числам, які стоять у парах праворуч від даного, і відразу після заглиблення в кожну нову вершину і повернення звідти додаємо цю вершину до порядку витирання чисел. Якщо в результаті до перестановки додано всі $N$ чисел, маємо відповідь. Інакше граф незв’язний, а у вихідний файл треба вивести нуль. Як і у випадку з жадібним підходом, час виконання алгоритму складає $O(N+M)$ або $O(M)$.

Зауважимо, що через обмеження, яке накладає компілятор GCC на деяких архітектурах, програма, що використовує рекурсію, може не витримати великого рівня заглиблення на одному-двох тестах. Це стосується як рекурсивного підходу, так і жадібного з використанням пошуку в глибину.

Насамкінець пропонуємо читачу самостійно розв’язати пов’язану задачу, з якою довелося зіткнутися автору при підготовці чекера: перевірити правильність виведення програми учасника за час $O(M)$.