Завдання та розв'язки XXI Всеукраїнської олімпіади з інформатики

Бондаренко В.В., Галковський Т.О., Коротков А.С., Нейтер Д.Ю., Ягіяєв Ш.І.

1 Задача «Гірлянда»

Новорічну ялинку прикрашено гірляндою нескінченної довжини, що складається з послідовно з'єднаних лампочок. Коли гірлянду вмикають, загоряється лише перша лампочка, рахуючи від вимикача, яка горить одну секунду. Далі гірлянда починає мигати за таким правилом. Щосекунди для кожної лампочки перевіряється умова: якщо рівно одна із її сусідніх лампочок горить, то ця лампочка буде горіти на наступній секунді; інакше — не буде горіти. Перша лампочка має лише одну сусідню.

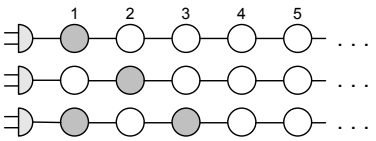


Рис. 1: Послідовність горіння лампочок

Завдання Напишіть програму `GARLAND`, яка за номером секунди знаходить кількість лампочок гірлянди, що будуть горіти протягом цієї секунди.

Вхідні дані Єдиний рядок вхідного файлу `GARLAND.DAT` містить одне ціле число N ($1 \le N \le 10^9$) — номер секунди.

Вихідні дані Єдиний рядок вихідного файлу `GARLAND.SOL` має містити ціле число — кількість лампочок, що будуть горіти на секунді N.

Приклад вхідних та вихідних даних

GARLAND.DAT
5

GARLAND.SOL
2

2 Задача «Острови»

Аби не відставати від сучасної світової тенденції, уряд країни Олімпія планує побудувати декілька островів для залучення туристів. Карта островів вже підготовлена та являє собою *таблицю* розміром $N \times M$ клітинок. Кожна клітинка може бути водою або сушею. Набір клітинок, що представляють сушу, є островом, коли з будь-якої з них можна потрапити в будь-яку іншу, переміщуючись сусідніми по горизонталі або вертикалі клітинами, та не існує інших таких клітин поза набором.

Для зручності було вирішено побудувати *мости* між деякими островами так, щоб усі острови стали сполученими між собою. Мости повинні будуватися лише по вертикалі чи горизонталі, проходити лише по клітинах з водою, починатись та закінчуватися клітинами з сушею. За *вартість* будівництва моста можна вважати кількість клітин води, через яку він проходить. Треба знайти мінімальну можливу загальну вартість будівництва групи мостів, які б сполучали між собою усі острови. Іншими словами, щоб з кожної клітини суші можна було досягнути будь-якої іншої, переміщуючись до сусідніх по вертикалі та горизонталі клітин суші, або мостам. Два різні мости можуть перетинатися між собою, тобто проходити через одну й ту саму клітину води на різних рівнях.

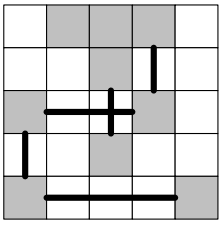


Рис. 2: Приклад з'єднаних островів

Завдання Напишіть програму `ISLANDS`, що за картою островів знаходить мінімальну вартість будівництва групи мостів, що з'єднують всі острови.

Вхідні дані Перший рядок вхідного файлу `ISLANDS.DAT` містить два цілих числа N та M ($1 \le N, M \le 500$) — розміри карти островів. Кожен з наступних N рядків містить M символів `0` (вода) або `1` (суша).

Вихідні дані Єдиний рядок вихідного файлу `ISLANDS.SOL` має містити одне ціле число — знайдену мінімальну вартість будівництва мостів. Якщо сполучити острова мостами неможливо, потрібно вивести число `-1`.

Приклад вхідних та вихідних даних

ISLANDS.DAT
5 5 01110 00100 10010 00100 10001

ISLANDS.SOL
8

3 Задача «Геном Ньютона»

На планеті Олімпія завершено вивчення геному мешканців Олімпійської галактики. Виявилося, що розшифрований геном може бути поданий у вигляді набору цілих чисел, що можуть повторюватися. У поданні геному талановитої особистості серед інших міститься єдине число, яке зустрічається непарну кількість разів та визначає номер певного генетично обумовленого таланту.

Розроблене обладнання отримує подання геному у вигляді набору множин чисел. Кожна множина задається четвіркою чисел s, f, a, b. Такій множині належать a послідовних цілих чисел починаючи з s, наступні b чисел множині не належать, наступні a знову належать, і т.д. Усі числа у множині не більші за f. Наприклад, множина ($s = 1, f = 10, a = 2, b = 1$) містить числа: 1, 2, 4, 5, 7, 8, 10, а множина ($s = 5, f = 50, a = 1, b = 19$) числа: 5, 25, 45.

Завдання Напишіть програму `GENOME`, що за поданням геному у вигляді набору множин чисел встановить, чи має його власник якийсь генетично обумовлений талант, та визначить його номер.

Вхідні дані Перший рядок вхідного файлу `GENOME.DAT` містить кількість множин N ($1 \le N \le 10\ 000$) у наборі. Наступні N рядків задають самі множини. Кожна множина задається четвіркою чисел — s, f, a, b, ($1 \le s, f, a, b < 10^9; s \le f$). Гарантується, що подання геному містить не більше одного числа, яке зустрічається непарну кількість разів.

Вихідні дані Єдиний рядок вихідного файлу `GENOME.SOL` має містити ціле число, яке зустрічається непарну кількість разів у поданні геному, або `0`, якщо такого числа не існує.

Приклад вхідних та вихідних даних

ORDER.DAT
4 7 59 1 9 7 82 1 49 17 50 1 29 27 27 1 1

ORDER.SOL
37

4 Задача «Знижки»

Відвідавши перед Новим роком великий магазин, ви обрали багато подарунків рідним та друзям. Зекономити певну кількість грошей вам можуть допомогти два типи передноворічних знижок, що діють у магазині:

1. При купівлі трьох товарів ви платите за них як за два найдорожчих з них.
2. При купівлі чотирьох товарів ви платите за них як за три найдорожчих з них.

Таким чином, певні товари можна об'єднати у трійки або четвірки і заплатити за них менше. Треба визначити найменшу можливу суму грошей, яка буде витрачена на придбання усіх подарунків. Наприклад, якщо ціни п'яти обраних подарунків складають: 50, 80, 50, 100, 20, то можна окремо придбати чотири перших товари, отримати за них знижку, та потім купити подарунок, що залишився за його номінальну ціну. Загалом вся покупка буде коштувати 250 грошових одиниць, замість 300.

Завдання Напишіть програму `DISCOUNT`, що за цінами усіх подарунків, знаходить мінімальну суму грошей, якої вистачить на їх купівлю.

Вхідні дані Перший рядок вхідного файлу `DISCOUNT.DAT` містить одне ціле число N ($0 \leq N \leq 10\ 000$). Другий рядок містить N натуральних чисел — ціни подарунків. Сума цін усіх подарунків менша за 10^9. Об'єднувати можна не лише ті товари, що йдуть підряд у вхідних даних.

Вихідні дані Єдиний рядок вихідного файлу `DISCOUNT.SOL` має містити одне ціле число — знайдену мінімальну суму грошей, за яку можна купити усі подарунки.

Приклад вхідних та вихідних даних

DISCOUNT.DAT
5 50 80 50 100 20

DISCOUNT.SOL
250

5 Задача «Точки»

На площині задано N точок. Окремо на площині задані дві *базові* точки.

Завдання Напишіть програму `POINTS`, що знаходить максимальну кількість точок, що потраплять у *смугу* створену парою паралельних прямих довільно проведених через базові точки. Базові точки не потрібно включати до суми точок. Якщо точка лежить на прямій — її потрібно врахувати у сумі.

Вхідні дані Перший рядок вхідного файлу `POINTS.DAT` містить одне ціле число N ($0 \leq N \leq 10\ 000$) — кількість точок. Другий рядок містить координати двох базових точок у форматі $x_1\ y_1\ x_2\ y_2$. Кожен з наступних N рядків містить координати точки площини у форматі $x\ y$. Координати точок — цілі числа, що за модулем не перевищують 10 000. Базові точки відрізняються принаймні однією координатою.

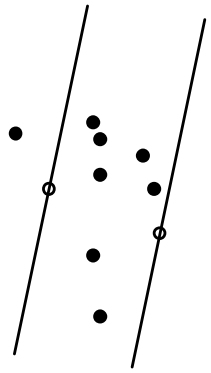


Рис. 3: Приклад смуги

Вихідні дані Єдиний рядок вихідного файлу `POINTS.SOL` має містити ціле число — знайдену максимальну кількість точок, що потраплять у смугу, яка буде утворена оптимально проведеними паралельними прямими через базові точки.

Приклади вхідних та вихідних даних

POINTS.DAT
4 0 0 50 0 0 -50 -1 0 50 0 100 50

POINTS.SOL
3

6 Задача «Катакомби»

Піратські катакомби на острові Скарбів було вирито за таким принципом. Після прихованого входу розташована печера, з якої виходять два тунелі — наліво і направо. Кожен із тунелів закінчується печерою, з якої також виходить два тунелі, і т.д. Довжина кожного тунелю рівна одиниці. *Кінцеві* печери, що знаходяться на відстані D від входу, не мають подальших виходів. Ніякі тунелі між собою не перетинаються, та не ведуть до одної печери. Число D називають *глибиною* катакомб.

У кожній з кінцевих печер приховано один *сундук зі скарбом*. Перед прибуттям на острів Капітана Джека Сперроу пірати вирішили перемістити ці сундуки згідно з останніми вказівками Капітана. Пірати намалювали план катакомб та пронумерували кінцеві печери зліва направо. Потім для кожного скарбу було встановлено номер печери, в якій він повинен опинитися перед прибуттям Капітана. Після переміщення в кожній печері знову опиниться лише один сундук.

Щоб забезпечити безпеку скарбів, пірати можуть лише обмінювати між собою сундуки з двох печер. Тільки після закінчення одного обміну можна починати інший. Необхідно знайти найменшу сумарну відстань яку піратам потрібно буде нести сундуки, аби розмістити їх потрібним чином.

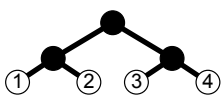


Рис. 4: Приклад карти катакомб

Завдання За наданими вхідними файлами, що містять опис печери зі скарбами, створіть відповідні вихідні файли, що містять мінімальну сумарну відстань, яку піратам доведеться нести сундуки, та послідовність обмінів.

Вхідні дані Вам надано 10 вхідних файлів, що мають назви `CATACOMB.D01`, `CATACOMB.D02`,..., `CATACOMB.D10`, у такому форматі. Перший рядок містить одне ціле число D — глибину катакомб. Другий рядок містить 2^D різних цілих чисел від 1 до 2^D. Кожне i-те з них ідентифікує номер печери до якої повинен попасти сундук, що находиться спочатку у печері i.

Вихідні дані Створіть 10 файлів `CATACOMB.S01`,..., `CATACOMB.S10`. Ці файли повинні містити відповіді для відповідних вхідних файлів. Перший рядок файлу має містити єдине ціле число — мінімальну сумарну відстань, яку пройдуть пірати зі скарбами. Другий рядок: ціле число K — відповідна кількість обмінів. Кожен наступний з K рядків: два числа, що є номерами *печер*, між якими відбувається обмін. Обміни повинні бути вказані у тому порядку, в якому вони мають відбуватися.

Приклади вхідних та вихідних даних

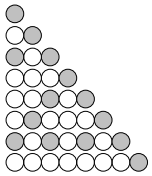
CATACOMB.D00
2 4 3 1 2

CATACOMB.S00
20 3 3 4 1 4 3 2

Наприклад, можна проводити обміни таким чином. Спочатку поміняти місцями скарби у печерах 3 та 4. Пройдена відстань 4 (по 2 для кожного сундука). Потім поміняти скарби у печерах 4 і 1, та 3 і 2. Відстань в обох випадках — 8. Таким чином — усі встануть на свої місця, а сумарна відстань буде 20.

7 Розв'язок задачі «Гірлянда» (Андрій Коротков)

За допомогою моделювання можна встановити, що на кожному кроці горіти буде кількість лампочок, що є степенем двійки. Це можна бачити із симетричності структури лампочок, що горять, на кожній секунді.



Розв'язок задачі зводиться до обчислення кількості b одиниць у двійковому представленні числа N. Відповіддю буде 2^{b-1}.

Обґрунтуємо цей факт. Зазначимо, що на n-ій секунді горітиме лампочка під номером n і не горітимуть ті, у яких номер більше за n.

Твердження. *На секундах виду* $2^k - 1$ *горіти будуть тільки лампочки з непарними номерами від 1 до* $2^k - 1$, $k \geq 1$.

Доведення. Зазначимо, що якщо для якогось k це твердження виконується, то, як наслідок, на секундах з номерами 2^k горіти буде лише одна лампочка з номером 2^k. Для обґрунтування варто лише проаналізувати зміни між секундами $2^k - 1$ і 2^k за правилами з умови.

Доведемо за індукцією по k.

База При $k = 1$ на секунді $2^1 - 1 = 1$ горить лампочка з номером 1, на секунді $2^1 = 2$ горить лампочка з номером 2. Для k=1 твердження виконується.

Крок Нехай для k вірно. Доведемо для $k+1$. За припущенням на 2^k-ій секунді буде горіти лише 2^k-та лампочка. Проаналізуємо подальші зміни.

Зліва і справа від лампочки 2^k утворюються симетричні картини. На кожному наступному кроці симетричність ламп зліва і справа від 2^k зберігається, а із симетричності на попередньому кроці одержуємо, що сусіди лампочки 2^k зліва і справа знаходяться у одному стані, тому лампа 2^k на наступному кроці горіти не буде.

У наступні 2^k-1 секунди лампочки на позиціях (2^k+1), (2^k+2),..., (2^k+2^k-1), а також симетричні ним на позиціях (2^k-1), (2^k-2), ..., 1 будуть вести себе так само, як і лампочки на позиціях 1, 2,..., 2^k-1 у перші 2^k-1-ну секунду. Лампа на позиції 2^k буде відігравати роль початку гірлянди.

Відповідно за припущенням індукції на секунді $2^k+2^k-1=2^{k+1}-1$ будуть горіти, з одного боку, лампи $2^k+1, 2^k+3, \ldots, 2^k+2^k-1$; а з іншого боку, лампи 2^k-1, $2^k-3, \ldots, 3, 1$, і лише вони. А це якраз те, що треба було довести. □

Перейдемо безпосередньо до розв'язання. Розглянемо найбільший одиничний біт у числі N. Нехай він відповідає числу 2^m. Оскільки це старший одиничний біт, то $N-2^m<2^m$. Якщо $N=2^m$, то буде горіти лише одна лампочка під номером 2^m, а інакше ця лампочка горіти не буде, і стан лампочок буде відповідати стану двох симетричних частин після $N-2^m$ секунди (як уже зазначалося, $N-2^m<2^m$). Позначимо за $f(N)$ відповідь для числа N. Тоді, якщо у числі N один біт, то $f(N)=1$, інакше $f(N)=2\cdot f(N-2^m)$, і у числі $N-2^m=N\operatorname{xor}2^m$ на один біт менше, ніж у числі N. Звідси неважко зрозуміти, що $f(N)=2^{b-1}$.

8 Розв'язок задачі «Острови» (Шаміль Ягіяєв)

Розв'язок задачі можна розбити на три етапи.

1. Виділення островів.
2. Пошук мостів найменшої довжини серед пар островів. Побудування графа, де вершинам відповідають острова, а ребрам — знайдені мости.
3. Пошук мінімального остовного дерева.

Перший етап задачі найпростіше зробити методом заливки. Цей алгоритм варто робити не рекурсивно, тому що при наведених об'ємах вхідних даних може переповнитися стек. Альтернативні підходи — пошук у ширину, або у глибину з використанням власного стеку. Для зручності кожен острів нумерується безпосередньо у карті. Це також використовується на наступному етапі.

Побудову графа можна зробити за два проходи карти: горизонтальний та вертикальний. Базовим алгоритмом буде прохід однієї смуги шириною одиниця, та аналіз потенційних мостів у цій смузі.

Проаналізувавши умову задачі можна зрозуміти, що максимальна кількість островів буде близько $N^2/2$ (де N — максимальна розмірність). Максимальна кількість ребер N^2.

Існує декілька підходів до побудови мінімального остовного дерева. Алгоритми Пріма і Крускула мають часову складність порядку $O(N^2\log N)$ для наших позначень. Для досягнення такої оцінки потрібно використовувати чергу з пріоритетами у алгоритмі Пріма, або одну з ефективних реалізацій систем множин, що не перетинаються. В іншому випадку оцінка може досягти $O(N^4)$. Алгоритм Крускала у нашому випадку має трохи меншу константу.

9 Розв'язок задачі «Геном Ньютона» (Тарас Галковський)

Визначимо функцію $F(l,r)$, що обчислить загальну кількість чисел в діапазоні $[l..r]$, що входять в розшифрований код геному. У разі відсутності числа, що зустрічається непарну кількість разів у геномі, для будь-якого діапазону чисел функція F буде приймати парні значення. Якщо ж ця функція приймає непарне значення для деякого діапазону, то це є необхідною і достатньою умовою існування шуканого числа в обраному діапазоні.

Користуючись цією властивістю функції F опишемо алгоритм пошуку відповіді. Почнемо з найбільшого діапазону $[1..10^9]$. Порахувавши значення функції F окремо для першої та другої половин діапазону ми можемо визначити в якій саме половині міститься число з непарною кількістю входжень. Відповідно, отримавши цю інформацію, ми зменшуємо діапазон пошуку вдвічі, і повторюємо крок алгоритму. Алгоритм завершує роботу, прийшовши до діапазону, що складається з одного числа. Кількість кроків алгоритму для діапазону $[l..r]$ складає $\log_2(r-l)$.

Тепер опишемо процес обчислення функції F.

$$F(l,r) = \sum_{i=1}^{M} G(set_i, [l..r])$$

де set_i це одна з множин отриманих на вході, що кодують набір чисел у поданні геному. $G(set, [l..r])$ — повертає кількість всіх чисел з множини set у діапазоні $[l..r]$.

Перейдемо до обчислення функції G. Для вказаного в умові представлення множин чисел таку функцію можна реалізовувати таким чином.

Якщо число $l < s$ приймемо за ліву границю l найменший елемент множини; також якщо $f < r$ приймемо $r = f$. Це дозволить скоротити наступні операції.

Оскільки, всі такі множини мають період $a+b$ та складаються з інтервалів однакової структури (перші a чисел належать множині, а b наступних — не належать). Можна порахувати номер інтервалу, який містить у собі число l — інтервал N_l, та номер інтервалу, що містить число r — інтервал N_r. Кожен інтервал з номером між числами N_l та N_r (не включаючи) містить рівно a чисел, що входять в set, і окремого розгляду вимагають лише крайні інтервалу N_l та N_r. Отримали, що часова складність функцій є $O(1)$. Отже, часова складність функції F пропорційна $O(M)$ (бо залежить від кількості множин, що утворюють код геному).

Часова складність описаного алгоритму складає $O(M \log N)$.

Розв'язок XOR Відомо, що для пошуку одного непарного числа з набору чисел достатньо послідовно здійснити операцію xor для всіх чисел набору. В нашому випадку цей розв'язок має місце, але, якщо брати до уваги обмеження, буде працювати лише на невеликій кількості наборів тестів. Складність алгоритму лінійна за числом чисел у представленні геному.

Існують певні оптимізації цієї ідеї, що базуються на ефективному підрахунку функції xor для послідовних чисел.

10 Розв'язок задачі «Знижки» (Шаміль Ягіяєв)

По-перше, треба помітити, що використання другого типу знижок не може покращити оптимальний розв'язок отриманий із використанням лише першого типу знижки.

Доведемо цей факт. Нехай існує оптимальний розв'язок, в якому використано знижку другого типу, та не можна побудувати не гірший за вартістю з використанням лише першого типу. Розглянемо ціни тих чотирьох товарів, що були використані для отримання знижки другого типу. Ми не заплатимо за найдешевший з цих чотирьох товарів. Але ж якщо ми застосуємо знижку першого типу для трьох найдорожчих товарів з четвірки, а останній купимо окремо, то це обійдеться не дорожче, ніж із використанням другого типу знижок. Це протиріччя доводить гіпотезу.

Відсортуємо набір цін за спаданням, та обчислимо суму цін тих з них, що стоять на місцях, номери яких не діляться на три націло. Це означає що в групи будуть об'єднані товари, що після впорядкування стоять на місцях (1, 2, 3), (4, 5, 6),....

Доведемо за індукцією, що це розбиття — оптимальне. Для кількості товарів 0, 1 або 2 знижки застосувати не можна. Нехай для будь-якої кількості товарів, меншої за $N \geq 3$, твердження вірне. Базуючись на цьому, доведемо його вірність для кількості N. Будемо вважати, що ціни товарів впорядковано за незростанням. Доведемо, що можна знайти оптимальне розбиття, де буде виділена група з товарами з номерами (1, 2, 3). Припустимо супротивне. Нехай для цієї кількості існує набір у оптимальному розбитті, де першим за спаданням товаром, за який не буде заплачено, буде товар a_k, (k>3). Нехай його було об'єднано у групу з товарами a_i і a_j, де $(i < j < k)$. Тепер, об'єднаємо його у групу з товарами a_1 і a_2, а ті, що було об'єднано

з a_1 і a_2 — з a_i і a_j. (Якщо a_1 чи a_2 не належали групам, то вони обміняються ролями з цими товарами). Оскільки всі найдешевші товари у групах мають номери не менші k, то після цієї заміни вони і залишаться найдешевшими, тобто знижку не буде зменшено. Тепер аналогічним чином обміняємо ролями товари a_3 і a_k. Ця операція може лише збільшити знижку. Без цієї групи залишається $N-3$ товари, до яких можна застосувати припущення індукції.

Складність алгоритму розв'язку складатиме $O(N \log N)$, що є складністю «найважчої» операції алгоритму — сортування.

11 Розв'язок задачі «Точки» (Богдан Рубльов, Тарас Галковський)

Проведемо базову пряму через дві базові точки, а також через них проведемо дві паралельні прямі. Нехай ці прямі закріплені в своїх базових точках і починають замітати поверхню з однаковою швидкістю (аналогічно діям «двірників» на лобовому склі машини). Тобто виконується а) кути між кожною з прямих і базовою прямою в довільний момент часу однакові; б) кожна пряма обов'язково проходить через свою базову точку, змінюючи лише кут нахилу.

Під час такого руху прямі будуть час від часу перетинати точки на площині. Ці події ми класифікуємо за такими правилами:

1. Перша пряма перетнула точку. В залежності від того, в якій частині площини (відносно базової прямої) це сталося — ця точка або додається до області між паралельними прямими, або виходить з неї.

2. Друга пряма перетнула точку. В залежності від того, в якій частині площини (відносно базової прямої) це сталося — ця точка або додається до області між прямими, або виходить з неї (навпаки, в порівнянні із першою прямою).

Слідкуючи за всіма такими подіями в кожен момент часу можна відповідати на питання, скільки ж точок знаходиться в смузі між двома прямими. Отже, загальна складність алгоритму — складність підтримки структури, що дозволяє слідкувати за описаними подіями.

Описаний словами алгоритм природно реалізується злиттям двох списків точок, відсортованих за двома різними критеріями: перший список містить всі точки площини, в порядку обходу навколо першої базової точки, а другий містить список точок площини в порядку обходу навколо другої базової точки (важливо, щоб базова пряма, від якої починається відлік кутів співпадала для обох списків; також напрям обходу — за чи проти годинникової стрілки — має бути однаковим для обох списків). Маючи такі відсортовані списки точок, ми проведемо моделювання процесу замітання (стартове положення неважливе). У деякий момент часу ми можемо отримати подію, що є найближчою до поточного моменту, шляхом порівняння двох найменших елементів з відсортованих списків. Виконавши розгляд чергової події, вона знищується у відповідному масиві, звідки була взята, і крок алгоритму повторюється. Часова складність сортування N елементів складає $O(N \log N)$.

Можна використовувати дві черги з пріоритетами для виконання аналогічних дій, однак реалізація цього підходу складніша.

Розв'язки з часовою складністю $O(N^2)$ мають принципову відмінність. Перша частина — досягла квадратичної оцінки шляхом використання неефективних методів сортування із правильною ідеєю алгоритму. Інша ж частина таких розв'язків діяла шляхом фіксації конкретної смуги на площині і підрахунком кількості точок множини, що попали у вибрану смугу. Цей розв'язок ґрунтується на очевидній властивості задачі, що смуга обов'язково проходить через обидві базові точки, її сторони паралельні, і, нескладно перевірити, що не треба розглядати такі смуги, на границі яких не лежить жодна точка з множини.

12 Розв'язок задачі «Катакомби» (Андрій Коротков)

Базові твердження На даний момент не вдалося знайти ефективного поліноміального розв'язку задачі. Відносно ефективна реалізація перебору варіантів базується на основі декількох фактів. Спочатку сформулюємо ці факти, а потім наведемо їх доведення.

Твердження (1). *Два підряд ідучих обміни по різних висотах k і l (у цьому порядку) можна замінити на два рівносильних обміни по висотах l і k.*

Наслідком з цього твердження є те, що існує оптимальна схема обмінів, у якій спочатку робляться усі обміни більшої довжини, а потім, меншої. Тобто, усі довжини обмінів впорядковані за незростанням.

Твердження (2). *Розглянемо мінімальне піддерево, у якому знаходиться скарб та печера, до якої він має потрапити. Не існує оптимальної стратегії обмінів, при якій скарб вийшов би за межі цього піддерева.*

Наслідком цього є такий факт. Якщо скарб та місце призначення знаходяться у різних піддеревах відносно якогось вузла, то скарб переходитиме у інше піддерево рівно один раз. З цього також слідує, що існує оптимальна послідовність обмінів, у якій по будь-якій висоті будь-який скарб буде обмінюватися не більше одного разу.

Твердження (3). *Якщо структуру можна розбити на піддерева, у кожному з яких рівно один скарб не має місця призначення у даному піддереві, а також якщо такий скарб у деякому піддереві A має опинитися у піддереві B, то і зайвий скарб піддерева B має потрапити у A, то у оптимальній послідовності обмінів будуть присутні попарні обміни через корінь між усіма парами таких піддерев A та B.*

Алгоритм перебору Ці факти дають нам можливість реалізувати такий алгоритм перебору. Для кореня дерева алгоритм з'ясовує, які скарби повинні перейти з лівого піддерева у праве, та навпаки. Далі він перебирає всі можливі варіанти їх попарного обміну. Після цього у лівому та правому піддеревах опиняться лише ті скарби, які там повинні і залишитися. Тобто, потрібно розв'язати таку ж саму задачу меншої розмірності для кожного з піддерев. Нехай кількість пар обмінів через корінь дорівнює k, тоді складність перебору рівна $k!$ помножене на суму складностей перебору для лівого та правого піддерев.

Цей алгоритм працює досить швидко для $D \leq 4$. В іншому випадку буде відігравати суттєву роль кількість обмінів, які потрібно здійснити через корінь. У певних випадках необхідно проводити аналіз структури дерева на предмет можливості застосування твердження 3.

Доведення тверджень

Твердження (1). *Два підряд ідучих обміни по різним висотах k і l (у цьому порядку) можна замінити на два рівносильних обміни по висотах l і k.*

Доведення. Якщо у цих обмінах приймають участь 4 попарно різні скарби, то твердження тривіальне. Якщо різних скарбів 2, то такі обміни взагалі зайві ($A \leftrightarrow B$, потім $B \leftrightarrow A$). Тепер нехай різних скарбів 3.

Позначимо той, що приймає участь у двох обмінах, через A, а інші через B (з першого обміну) і C (з другого обміну). Нехай обмін $A \leftrightarrow B$ іде через висоту k, а $A \leftrightarrow C$ — через l.

1. $k > l$. Після обмінів $A \leftrightarrow B(k)$, $A \leftrightarrow C(l)$ одержимо розташування BAC. Шуканими рівносильними обмінами будуть $B \leftrightarrow C(l)$, $A \leftrightarrow B(k)$.

2. $k < l$. Після обмінів $A \leftrightarrow B(k)$, $A \leftrightarrow C(l)$ одержимо розташування BCA. Шуканими рівносильними обмінами будуть $A \leftrightarrow B(l)$, $B \leftrightarrow C(k)$.

Твердження доведено.

□

Твердження (2). *Розглянемо мінімальне піддерево, у якому знаходиться скарб та печера, до якої він має потрапити. Не існує оптимальної стратегії обмінів, при якій скарб вийшов би за межі цього піддерева.*

Доведення. Якщо ми перенесемо скарб у інше піддерево, то нам ще треба буде його повернути у початкове (бо він має потрапити на місце призначення, яке знаходиться у початковому). Назвемо висотою вузла (від низу) кількість тунелів, через які треба протягнути якийсь скарб із його піддерева, щоб досягнути вузла. Висота вузла зі скарбом, зрозуміло, 0.

Припустимо, що існує оптимальний перерозподіл, у якому скарб переходить у інше піддерево і повертається назад. Нехай висота відповідного вузла рівна k. Також позначимо скарб,

який так переходить, через A (у першому піддереві). І нехай він при переході у друге піддерево міняється зі скарбом B (у другому піддереві), а при поверненні — з C (у першому піддереві). Тепер покажемо, що існує більш оптимальна схема. Будемо повторювати всі обміни аж до обміну $A \leftrightarrow B$. Обмін $A \leftrightarrow B$ робити не будемо. Далі знову почнемо повторювати обміни; там де приймав участь A, прийматиме B, і навпаки; повторюємо до обміну $A \leftrightarrow C$ (у початковій схемі). Цей обмін не повторюємо, замість нього робимо обмін $B \leftrightarrow C$, а потім $A \leftrightarrow B$. І далі знову повторюємо обміни аж до кінця. Розташування буде еквівалентним. Як бачимо, порівняно з першою схемою зникло 2 обміни вартості $4k$ ($A \leftrightarrow B$ і $A \leftrightarrow C$) і з'явилося два обміни один вартістю $4k$ ($B \leftrightarrow C$) і один вартістю не більше $4 \cdot (k-1)$ ($A \leftrightarrow B$, вони у одному піддереві відносно вузла висоти k), решта обмінів не змінили вартостей. Нова схема краща за стару. Отримали протиріччя. □

Твердження (3). *Якщо структуру можна розбити на піддерева, у кожному з яких рівно один скарб не має місця призначення у даному піддереві, а також якщо такий скарб у деякому піддереві A має опинитися у піддереві B, то і зайвий скарб піддерева B має потрапити у A, то у оптимальній послідовності обмінів будуть присутні попарні обміни через корінь між усіма парами таких піддерев A та B.*

Доведення. Покажемо нижню оцінку на сумарну довжину обмінів. По-перше, обміни через корінь все одно треба буде робити, відстань при цьому буде добутком кількості пар та глибини катакомб. По-друге, схема переміщень для кожного піддерева буде виглядати наступним чином: відбуваються якісь переміщення, потім зайвий елемент обмінюється з тим, якого не вистачає, і знову робляться якісь переміщення всередині піддерева, бо елементи скарби, що мають призначення у цьому піддереві не вийдуть з нього (твердження 2), а коли, той якого не вистачало потрапить до піддерева, то надалі в ньому залишатиметься. Загалом, під час обмінів зайвий елемент може мінятися із зайвим. Важливо те, що зайвий елемент буде лише один. Отже, другий доданок оцінки знизу буде сумою довжин обмінів всередині кожного піддерева. Покажемо схему, при якій ця оцінка досягається, і тим самим покажемо оптимальність. Робимо вказані вище попарні обміни через корінь — добуток кількості пар та глибини катакомб. Далі всередині кожного піддерева робимо заплановані раніше обміни (різниця лише в тому, що невистачаючий скарб потрапить на місце зайвого відразу, а не пізніше), решту обмінів відміняємо — сума довжин обмінів у кожному піддереві. Твердження доведено. □