

## ЗАДАЧІ XXIII ВСЕУКРАЇНСЬКОЇ ОЛІМПІАДИ З ІНФОРМАТИКИ ТА РЕКОМЕНДАЦІЇ ЩОДО ЇХ РОЗВ'ЯЗУВАННЯ

**Бондаренко В.В., Галковський Т.О.,  
Коротков А.С., Нейтер Д.Ю.,  
Твердохліб Я.О., Ягієв Ш.І.**

(Закінчення, початок у №3 за 2010 рік)

### ЗАВДАННЯ ДРУГОГО ТУРУ

#### 1. Відстань між числами

Нехай числа  $a$  і  $b$  записані в десятковій системі числення. Визначимо відстань між ними як:  $(a_1 - b_1)^2 + (a_2 - b_2)^2 + (a_3 - b_3)^2 + \dots$ , де  $a_i$  позначає  $i$ -ту цифру числа  $a$ , а  $b_i$  позначає  $i$ -ту цифру числа  $b$ . Нумерація цифр починається із молодшого розряду числа, якому відповідає номер 1. Якщо значення  $i$  більше, ніж довжина числа, то вважається, що  $i$ -та цифра, рівна нулеві.

**Завдання.** Напишіть програму NUMBERS, яка за трьома цілими невід'ємними числами  $A$ ,  $B$  та  $C$  знайде такі числа  $X$  і  $Y$ , для яких виконуються умови:

1.  $A \leq X \leq B$  і  $A \leq Y \leq B$ .

2.  $X$  є мінімальним серед таких чисел, від яких відстань до  $C$  найменш можлива.

3.  $Y$  є максимальним серед таких чисел, від яких відстань до  $C$  найбільш можлива.

**Вхідні дані.** Перші три рядки вхідного файлу NUMBERS.DAT містять відповідно цілі числа  $A$ ,  $B$  та  $C$  ( $0 \leq A \leq B \leq 10^{18}$ ,  $0 \leq C \leq 10^{18}$ ).

**Вихідні дані.** Вихідний файл NUMBERS.SOL має містити два рядки, які містять відповідно цілі числа  $X$  та  $Y$ .

**Оцінювання.** Щонайменше у 50% тестів журі числа  $A$ ,  $B$  та  $C$  не будуть перевищувати 100 000.

#### *Приклад вхідних та вихідних даних*

| NUMBERS.DAT |
|-------------|
| 11          |
| 25          |
| 130         |

| NUMBERS.SOL |
|-------------|
| 20          |
| 19          |

Відстань від числа 20 до числа 130:  $(1-0)^2 + (2-3)^2 + (0-0)^2 = 1 + 1 + 0 = 2$ .

Відстань від числа 19 до числа 130:  $(1-0)^2 + (1-3)^2 + (9-0)^2 = 1 + 22 + 92 = 1 + 4 + 81 = 86$ .

#### Рекомендації щодо розв'язування

**Зведення до однакової довжини.** Виберемо найдовше серед чисел  $A$ ,  $B$  та  $C$ . Якщо одне або два числа мають меншу довжину ніж третє, доповнимо їх ведучими нулями так, щоб усі 3 числа стали однакової довжини.

**Видалення однакових перших цифр.** Поки перші цифри у  $A$  та  $B$  співпадають, видалятимемо їх разом з першою цифрою  $C$ , запам'ятовуючи ці цифри для відтворення відповіді. Ми можемо це зробити, не змінюючи відповіді, тому що у всякому разі на відповідних позиціях у  $X$  та  $Y$  цифри визначаються однозначно.

**Пошук числа  $X$ .** Вважатимемо що  $A$ ,  $B$  та  $C$  мають однакову довжину, а також перші цифри у  $A$  та  $B$  не співпадають. Окремо розглянемо випадки  $X=A$  та  $X=B$ . Виберемо першу цифру числа  $X$ . Маємо такі 3 випадки:

1. Ця цифра співпадає з першою цифрою числа  $A$ . У такому випадку, якими б не були наступні цифри числа  $X$ , воно буде строго меншим за  $B$ . Отже, нам треба вибрати їх так, щоб  $X$  було більшим за  $A$ . Отже, декілька перших цифр  $A$  та  $X$  співпадають, наступна цифра має бути строго більшою за відповідну цифру числа  $A$ , а решта цифр може бути обрана довільно. Оскільки ми шукаємо число з найменшою відстанню до  $C$ , то решту цифр вигідно обрати так, щоб вони співпадали з відповідними їм цифрами числа  $C$ .

2. Ця цифра більша першої цифри числа  $A$  та менша першої цифри числа  $B$ . У такому випадку інші цифри можуть бути обрані довільно.

3. Ця цифра співпадає з першою цифрою числа  $B$ . У такому випадку, якими б не були наступні цифри числа  $X$ , воно буде строго більшим за  $A$ . Отже, нам треба вибрати їх так, щоб  $X$  було меншим за  $B$ . Отже, декілька перших цифр  $B$  та  $X$  співпадають, наступна цифра має бути строго меншою за відповідну цифру числа  $B$ , а решта цифр може бути обрана довільно.

**Алгоритм пошуку  $X$ .** Отже, маємо такий алгоритм:

- Доповнимо деякі з чисел  $A$ ,  $B$  та  $C$  ведучими нулями так, щоб усі вони стали однакової довжини.
- Видалятимемо перші цифри з усіх трьох чисел, до тих пір поки перші цифри чисел  $A$  та  $B$  не стануть різними.
- Переберемо кількість перших цифр числа  $X$ , які співпадатимуть з відповідними цифрами числа  $A$ . Наступну цифру (якщо така є) виберемо якомога ближчою до відповідної цифри числа  $C$ , але так, щоб вона була строго більшою відповідної цифри числа  $A$ . Якщо таких цифр декілька, виберемо найменшу. Решту цифр виберемо рівними відповідним цифрам числа  $C$ . Якщо отримане число ближче до  $C$ , ніж уже знайдене або воно має таку саме відстань до  $C$  і водночас воно менше, ніж вже знайдене, то замінимо знайдений  $X$  на дане число.
- Переберемо кількість перших цифр числа  $X$ , які співпадають з відповідними цифрами числа  $B$ . Наступну цифру (якщо така є) виберемо якомога ближчою до відповідної цифри числа  $C$ , але так, щоб вона була строго меншою відповідної цифри числа  $B$ . Якщо таких цифр декілька, виберемо найменшу. Решту цифр виберемо рівними відповідним цифрам числа  $C$ . Як-

що отримане число ближче до  $C$ , ніж уже знайдене або воно має таку саме відстань до  $C$  і водночас воно менше, ніж уже знайдене, то замінімо знайдений  $X$  на дане число.

Окремо треба відзначити, що при виборі першої цифри потрібно враховувати, що вона повинна одночасно не перевищувати першу цифру числа  $B$  і не бути меншою за першу цифру числа  $A$ .

**Пошук числа  $Y$ .** Пошук числа  $Y$  повністю аналогічний пошуку числа  $X$ , за виключенням того, що коли потрібно вибрати декілька останніх цифр числа  $Y$ , нам треба максимізувати їх сумарну відстань до відповідних цифр числа  $C$ . Очевидно, що для кожної цифри найвіддаленішою від неї буде або цифра 0 або цифра 9.

**Алгоритм пошуку  $Y$ .** Отже, маємо такий алгоритм:

- Доповнимо деякі з чисел  $A$ ,  $B$  та  $C$  ведучими нулями так, щоб усі вони стали однакової довжини.
- Видалятимемо перші цифри з усіх трьох чисел, доки перші цифри чисел  $A$  та  $B$  не стануть різними.
- Переберемо кількість перших цифр числа  $Y$ , які співпадають з відповідними цифрами числа  $A$ . Наступну цифру (якщо така є) виберемо якомога більше віддаленою від відповідної цифри числа  $C$ , але так, щоб вона була строго більшою відповідної цифри числа  $A$ . Якщо таких цифр декілька, виберемо найбільшу. Решту цифр виберемо так, щоб вони були якомога більше віддалені від відповідних цифр числа  $C$ . Якщо отримане число ближче до  $C$ , ніж уже знайдене або воно має таку саму відстань до  $C$  і водночас воно більше, ніж уже знайдене, то замінімо знайдений  $Y$  на дане число.
- Переберемо кількість перших цифр числа  $Y$ , які співпадають з відповідними цифрами числа  $B$ . Наступну цифру (якщо така є) виберемо якомога більше віддаленою від відповідної цифри числа  $C$ , але так, щоб вона була строго меншою відповідної цифри числа  $B$ . Якщо таких цифр декілька, виберемо найбільшу. Решту цифр виберемо так, щоб вони були якомога більше віддалені від відповідних цифр числа  $C$ . Якщо отримане число ближче до  $C$ , ніж уже знайдене або воно має таку саму відстань до  $C$  і водночас воно більше, ніж уже знайдене, то замінімо знайдений  $Y$  на дане число.

Окремо треба відзначити, що при виборі першої цифри потрібно враховувати, що вона повинна одночасно не перевищувати першу цифру числа  $B$  і не бути меншою за першу цифру числа  $A$ .

## 2. Криза

Унаслідок кризи курс національної валюти планети Олімпія (*олімп*) почав коливатися відносно міжгалактичної валюти (*галакт*). Після закінчення кризи, Петро — один з мешканців планети — вирішив підрахувати, скільки він міг би заробити, маючи початкові заощадження, якщо б наперед знав щоденний курс.

**Завдання.** Напишіть програму CRISIS, що за інформацією про суму заощаджень Петра на момент початку кризи в олімпіах та коливання курсу олімпів відносно галактів визначить максимальну кількість грошей, яку Петро міг би мати на кінець кризи. Не можна покупати та продавати нецілу кількість галактів. На кінець кризи всі гроші повинні бути переведені в олімпі.

**Вхідні дані.** Перший рядок вхідного файлу CRISIS.DAT містить два цілих числа  $N$  ( $1 \leq N \leq 50\,000$ ) — довжина кризи у днях та  $S$  ( $1 \leq S \leq 100\,000$ ) — сума початкових заощаджень Петра. Наступні  $N$  рядків містять по два натуральних числа, які не перевищують 1 000 000:

1. Перше число — кількість олімпів, за які у цей день можна було придбати один галакт.

2. Друге — кількість олімпів, які можна отримати, продавши один галакт цього дня. Друге число не перевищує перше.

**Вихідні дані.** Єдиний рядок вихідного файлу CRISIS.SOL має містити одне ціле число — найбільшу суму грошей в олімпіах, яку міг би мати Петро після закінчення останнього дня кризи. Гарантовано, що відповідь на кожен з тестів журі не буде перевищувати  $10^{18}$ .

**Оцінювання.** Щонайменше у 25% тестів буде виконуватись додаткове обмеження  $N \leq 20$ . Щонайменше у 75% тестів буде виконуватись додаткове обмеження  $N \leq 2000$ .

### Приклад вхідних та вихідних даних

| CRISIS.DAT | CRISIS.SOL |
|------------|------------|
| 3 1000     | 1050       |
| 100 99     |            |
| 110 105    |            |
| 90 80      |            |

Станом на початок кризи в Петра було 1000 олімпів. Першого дня Петро купив 10 галактів. Наступного дня Петро обміняв 10 галактів на 1050 олімпів. На третій день він не проводив ніяких операцій. Отже, на кінець кризи Петро мав би 1050 олімпів.

### Рекомендації щодо розв'язування

**Формалізація задачі і загальна ідея розв'язку.** Позначимо  $u_i$  — кількість олімпів, за яку можна купити один галакт,  $l_i$  — кількість олімпів, за яку можна продати один галакт. Під оптимальною послідовністю операцій будемо розуміти таку послідовність операцій купівлі на продажі галактів, яка максимізує суму грошей наприкінці кризи.

Ключем до ідеї розв'язку задачі є таке твердження:

**Твердження 2.1.** Існує оптимальна послідовність купівлі-продажу галактів, для якої виконується умова «локальної жадібності»: у кожен день або купляється максимально можлива кількість галактів, або продаються всі наявні галакти, або не виконується ніяких операцій.

**Доведення.** Припустимо, що існує оптимальна послідовність операцій, у якій порушується умова «ло-

кальної жадібності». І припустимо, що вперше ця умова порушується у день  $i$ . Покажемо, що цю послідовність можна модифікувати так, що умова «локальної жадібності» буде виконуватися для днів  $1 \dots i$  включно, а кінцева сума не зменшиться.

Припустимо, що в  $i$ -й день відбувалася купівля галактів, але було куплено менше, ніж максимально можлива кількість. Тоді:

1. Існує день  $j$  такий, що  $u_i < l_j$ ,  $j > i$ ,  $l_k \leq u_i$ , для всіх  $i < k < j$ . Якщо це було б не так, то відмовившись від купівлі галактів в  $i$ -й день, ми б отримали не гіршу послідовність.

2. Існує день  $k$  такий, що  $u_k < u_i$ ,  $i < k < j$ . Якщо такого дня не існує, то максимізувавши кількість галактів, що купляються  $i$ -го дня, ми отримаємо не гіршу послідовність.

3. Виходячи з пунктів 1 і 2, можна зробити висновки, що розглядану послідовність можна покращити, якщо відмовитися від купівлі галактів  $i$ -го, а замість цього купити ту саму кількість галактів  $k$ -го дня.

Аналогічно, якщо в  $i$ -й день відбувся продаж галактів, але були продані не всі наявні галакти, то:

1. Існує день  $j$  такий, що  $l_i < l_j$ ,  $j > i$ ,  $l_k \leq l_i$ , для всіх  $i < k < j$ . Якщо це не так, то продавши всі наявні галакти на  $i$ -й день, ми отримаємо не гіршу послідовність.

2. Існує день  $k$  такий, що  $u_k < l_i$ ,  $i < k < j$ . Якщо це не так, то у разі перенесення продажу галактів з дня  $i$  на день  $j$ , ми отримаємо не гіршу послідовність.

3. Виходячи з пунктів 1 і 2, поточну послідовність можна покращити, якщо продати залишок галактів у день  $i$ , а потім купити таку саму кількість галактів у день  $k$ .

У кожному з випадків ми можемо модифікувати розглядану послідовність так, що кінцева сума не зменшиться, а у  $i$ -й день буде виконуватися умова «локальної жадібності». Повторюючи міркування скінченну кількість разів, прийдемо до оптимальної послідовності, для якої виконується правило «локальної жадібності».

**Твердження 2.2.** В оптимальній послідовності операції купівлі та продажу галактів повинні чергуватися.

*Доведення.* Очевидне.

**Перебірний розв'язок.** Найпростіший розв'язок, що спадає на думку: за допомогою перебору з поверненням перебрати всі можливі послідовності операцій і вибрати ту, яка дає найбільшу суму. Виходячи з доведеного вище твердження, для кожного дня треба розглянути 3 альтернативи: у цей день відбувається купівля галактів, продаж галактів, або не відбувається ніяких операцій. Враховуючи твердження 2.2, для кожного дня достатньо перевіряти лише 2 альтернативи: відбувається чергова операція (тип операції буде однозначно визначатися кількістю операцій, зроблених до цього), або не відбувається ніяких дій.

Складність алгоритму  $O(2^N)$ .

**Розв'язок за допомогою динамічного програмування.** Введемо до розгляду функцію  $f(i)$  — максимальну кількість олімпів, яку може мати наприкінці  $i$ -го

дня Петро. Для зручності означимо  $f(0) = S$ . Тоді відповідь до задачі буде  $f(N)$ .

Для підрахунку  $f(i)$  використаємо динамічне програмування. Припустимо, що  $j$  — останній день продажу галактів ( $0 \leq j < i$ ).

Тоді станом на кінець  $i$ -го дня Петро матиме

$$g(j, i) = \max\{f(j), \lfloor \frac{f(j)}{\min\{u_k : j < k < i\}} \rfloor \cdot l_i\} \text{ олімпів:}$$

він або взагалі не робитиме ніяких операцій, або купить галакти за найменшою на проміжку ціною, а потім продасть їх в день  $i$ . Перебираючи можливі значення  $j$  — останнього дня покупки, отримаємо:  $f(i) = \max\{g(j, i) : 0 \leq j < i\}$ .

За отриманими вище співвідношеннями можна підрахувати  $f(i)$  для всіх  $i$  від 1 до  $N$  за час  $O(N^2)$ .

**Ефективний розв'язок.** Покажемо, як можна розв'язати задачу за час  $O(N)$  за допомогою жадібного алгоритму.

Будемо називати точками продажу такі дні, коли є сенс продавати всі наявні галакти, якщо вони є. Відповідно точками купівлі будемо називати дні, коли є сенс купувати максимально доступну кількість галактів. Будемо позначати через  $\text{next}(i)$  першу точку продажу, починаючи з дня  $i$ .

Будемо шукати точки продажу, починаючи з кінця. Введемо до розгляду також величину  $c_i$ , яку будемо обчислювати так:

$$c_i = \begin{cases} l_i, & \text{якщо день } i \text{ — точка продажу,} \\ \min\{u_i, c_{i+1}\}, & \text{у протилежному випадку.} \end{cases}$$

Величина  $c_i$  відображає мінімальну ціну, за якою можна купити галакти в дні  $i \dots \text{next}(i) - 1$ . Якщо в якійсь момент  $c_i = l_{\text{next}(i)}$ , то купувати галакти на проміжку  $i \dots \text{next}(i) - 1$  не вигідно взагалі.

**Твердження.** День  $i$  — точка продажу тоді і тільки тоді, коли або  $i = N$ , або  $l_i > c_{i+1}$ .

**Доведення.**  $i = N$  завжди є точкою продажу, бо за умовою на кінець кризи всі гроші повинні бути переведені в олімпи. Надалі припустимо, що  $i < N$ .

Припустимо, що  $l_i > c_{i+1}$  і покажемо, що  $i$  — точка продажу. Наступна після  $i$  точка продажу буде  $j = \text{next}(i+1)$ . З умови  $l_i > c_{i+1}$  слідує, що

- або  $l_i > l_j$ , а, отже вигідніше продати всі наявні галакти в день  $i$ , ніж продати їх у наступній точці продажу;
- або існує такий день  $k$ , що  $i < k < j$ ,  $u_k < l_i$ . У такому випадку вигідніше продати всі галакти у день  $i$ , а потім купити максимально можливу кількість галактів у день  $k$ , ніж не продавати галакти в день  $i$  взагалі.

В обох випадках виходить, що в день  $i$  вигідно продавати всі наявні галакти, тобто  $i$  — точка продажу.

Припустимо тепер, що  $l_i \leq c_{i+1}$ . Звідси слідує, що, по-перше,  $l_i \leq l_j$ , а, по-друге, не існує такого дня  $k$ , щоб  $i < k < j$  і  $u_k < l_i$ . Тобто, якщо продати галакти в день  $i$ , то Петро отримає менше олімпів, ніж якщо продати галакти в день  $j$ , а крім того, якщо продати галакти в

день  $i$ , то не буде можливості вигідно купити галакти до настання дня  $j$ . Отже, у день  $i$  немає сенсу нічого продавати. Тобто  $i$  — не є точка продажу.

**Твердження.** День  $i$  — точка покупки тоді і тільки тоді, коли  $i$  не є точкою продажу і  $c_i = u_i$ .

**Доведення.** Очевидно, що якщо  $i$  — точка продажу, то немає сенсу купляти галакти в день  $i$ , і навпаки. Припустимо надалі, що  $i$  — не є точкою продажу.

Припустимо, що  $c_i = u_i$ . Тоді, по-перше,  $u_i \leq l_{\text{next}(i)}$ , а, по-друге,  $u_i \leq u_k$  для всіх  $i < k < \text{next}(i)$ . Значить, якщо купити галакти в день  $i$ , то їх можна буде потім продати як мінімум без втрат. І крім того не має можливості купити галакти дешевше до настання наступної точки продажу. Отже, є сенс купувати максимально можливу кількість галактих у день  $i$ , тобто  $i$  — точка покупки.

Припустимо тепер, що  $c_i < u_i$  (бо за означенням  $c_i \geq u_i$ ). Тоді або  $u_i > l_{\text{next}(i)}$ , або існує  $k: i < k < \text{next}(i)$ ,  $u_i > u_k \leq l_{\text{next}(i)}$ . У першому випадку, якщо купити галакти, їх потім неможливо буде продати без втрат. У другому випадку вигідніше купити галакти в день  $k$ , аніж у день  $i$ . В обох випадках купувати галакти в день  $i$  не вигідно, тобто  $i$  не є точкою покупки.

Керуючись наведеними вище твердженнями, можна розв'язати задачу за час  $O(N)$ . Для цього спочатку, починаючи з останнього дня розраховуються величина  $c_i$  та положення точок продажу. Враховуючи, що положення точок купівлі за допомогою твердження 2.2 однозначно встановлюється за величиною  $c_i$  та положенням точок продажу, отримуємо оптимальну послідовність операцій. Залишається лише виконати цю послідовність операцій і підрахувати суму наприкінці кризи.

### 3. Пари точок

На площині своїми координатами задані  $N$  червоних та  $N$  синіх точок. Жодні три точки не належать одній прямій. Необхідно побудувати  $N$  відрізків таких, що ніякі два з них не перетинаються, кожен відрізок з'єднує точки різного кольору, і кожна точка належить рівно одному відрізку.

**Завдання.** Напишіть програму PAIRS, яка за інформацією про розташування точок знайде будь-яке розбиття множини точок на  $N$  відрізків, кожен з яких сполучає синю і червону точки. Причому, кожна точка належить лише одному відрізку і ніякі два відрізки не перетинаються.

**Вхідні дані.** Перший рядок вхідного файлу PAIRS.DAT містить єдине натуральне число — кількість блоків тестових даних, що задано у файлі. Блоки слідує один за одним. Кожен блок необхідно обробити окремо. Перший рядок кожного тестового блоку містить ціле число  $N$  ( $1 \leq N \leq 2500$ ) — кількість точок одного кольору. Далі слідує набори координат синіх точок, а потім — червоних. Набір координат точок одного кольору задається  $N$  рядками, кожен з яких містить пару цілих чисел —  $x$  і  $y$  координати точки ( $|x| \leq 10\,000$ ,  $|y| \leq 10\,000$ ).

**Вихідні дані.** Вихідний файл PAIRS.SOL має містити відповіді для всіх блоків тестових даних із вхідного файлу без розділення. Кожен з  $N$  рядків відповіді кожного з блоків має містити пару цілих чисел від 1 до  $N$  — номери синьої та червоної точок, сполучених відрізком. Точки пронумеровані в тому порядку, у якому вони слідує у вхідному файлі окремо по кожному кольору. У випадку, якщо не можна сполучити точки відрізками без перетинів, єдиний рядок відповіді для блоку має містити число 0.

**Оцінювання.** Щонайменше у 20% тестів буде виконуватись додаткове обмеження  $N \leq 10$  та кількість блоків не перевищуватиме 2.

#### Приклад вхідних та вихідних даних

| PAIRS.DAT | PAIRS.SOL |
|-----------|-----------|
| 2         | 1 2       |
| 2         | 2 1       |
| 1 1       | 1 1       |
| 3 5       |           |
| 4 10      |           |
| 2 2       |           |
| 1         |           |
| 1 1       |           |
| 2 2       |           |

#### Рекомендації щодо розв'язування

Доведемо існування такого паросполучення.

**Лема 5.** Множини точок потужності  $N$  завжди можна сполучити  $N$  відрізками, такими що кожен

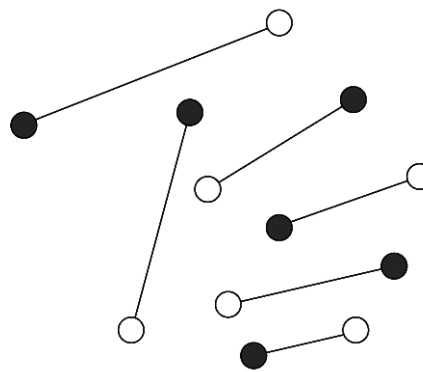


Рис. 2

відрізок з'єднує вершину першої і другої множини; при цьому жодні два відрізки не перетинаються.

#### Неконструктивне доведення

**Доведення.** Паросполучення мінімальної загальної довжини, очевидно, не має перетинів. Якщо припустити, що в такому паросполученні існує перетин, то можна обміняти сполучення пар вершин, відрізки яких перетинаються, і отримати паросполучення меншої довжини.

#### Конструктивне доведення

**Доведення.** Помітимо, що побудувавши опуклу оболонку множини точок, ребра оболонки, фактично, не перетинаються між собою; крім того, усі можливі відрізки між вершинами, що лежать всередині опуклої оболонки, не можуть перетнути ребра опуклої оболонки. Спро-

буємо використати цю властивість для розв'язання задачі. Проаналізуємо наступний алгоритм:

- Побудуємо опуклу оболонку всіх точок (з обох множин).

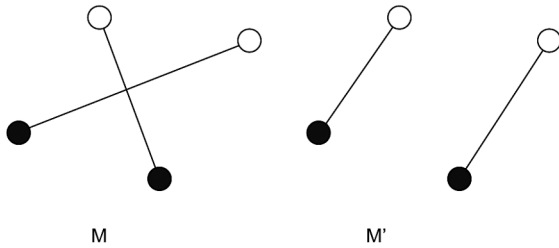


Рис. 3

- Оберемо довільне різнокольорове ребро з побудованої оболонки.
- Додамо обране ребро до паросполучення, та видаляємо дві вершини, що з'єднані ним.
- Повторюємо, поки є вершини.

Згідно наведеної властивості, слідує, що цей алгоритм не може утворити перетини серед відрізків, які він сполучає. Однак, необхідно довести, що цей алгоритм може з'єднати всі вершини. Тобто перевірити, чи завжди на оболонці існуватиме різнокольорове ребро. Нескладно перекоонатися, що таке ребро існуватиме на оболонці не завжди.

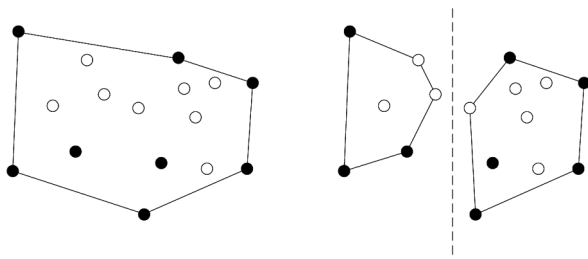


Рис. 4

У такому випадку ми можемо модифікувати задачу, і продовжити розв'язання наведеним вище алгоритмом: на такому кроці можливо розділити задачу на дві незалежні підзадачі. Знайдемо вертикальну пряму таку, що ліворуч від неї буде лежати однакова кількість синіх та червоних точок (очевидно, у такому випадку праворуч кількості вершин також будуть співпадати). Розв'яжемо отримані підзадачі рекурсивно. Розв'язки при цьому незалежні, оскільки не може в таких підзадачах виникнути відрізка, що перетне обрану розділяючу вертикальну пряму. Після розділення на підзадачі ми або отримаємо різнокольорові ребра в підзадачах, або знову прийдесться повторити розділення, але для задач меншої розмірності. З цього слідує, що алгоритм завершує роботу за скінчену кількість кроків.

**Твердження.** Вертикальна пряма, що розділює задачу на підзадачі завжди існує, за умови, що на оболонці немає різнокольорового ребра.

**Доведення.** Це можна довести конструктивно, призначивши кожній синій вершині вагу +1, а кожній червоній вершині -1. Почнемо замінити площину вертикальною прямою, підраховуючи при цьому суму ваг у вершинах, що замінюються. Оскільки на оболонці не існує різнокольорових ребер, опукла

оболонка утворена вершинами одного кольору (без обмеження загальності, синього); з цього слідує, що крайня ліва і крайня права точки, що будуть заметені, мають вагу +1. Отже, у крайній лівій точці значення такої функції (сумарна заметена вага) рівна +1, а у момент перед потраплянням у крайню праву точку -1. З наслідку теореми Коші про середні значення, така функція приймає значення нуль у певній точці посеред проміжку, що доводить існування шуканої розділяючої вертикальної прямої. Описане доведення також і є способом пошуку такої розділяючої прямої — замінюючи і обчислюючи значення функції сумарної ваги, ми зупинимося в точці, у якій значення функції буде рівним нулю.

**Реалізація.** Описаний алгоритм можна реалізувати, наприклад, так: на кожному кроці будемо опуклу оболонку методом загортання кута (Graham scan)  $O(N \log N)$ . Якщо різнокольорового ребра не існує, нам необхідно розділити множину точок на підзадачі. Для цього ми скористаємося алгоритмом, описаним у відповідному твердженні, який працює за час  $O(N \log N)$  — необхідне сортування точок за координатою  $X$ . Зауважимо, що таке сортування можна здійснити лише один раз при зчитуванні даних. Оскільки маємо рівно  $N$  кроків, за кількістю пар вершин, які необхідно сполучити відрізком, загальна складність описаного алгоритму  $O(N^2 \log N)$ .

Прискорити алгоритм можна, якщо скористатися іншим методом побудови опуклої оболонки. А саме замість побудови всієї опуклої оболонки методом загортання, можемо окремо побудувати верхній та нижній ланцюг опуклої оболонки (Andrew's algorithm). Це тим більш зручно, оскільки точки із самого початку відсортовані за координатою  $X$  для кроку розділення. У такому випадку, складність кожного кроку  $O(N)$ , а, отже, загальна складність  $O(N^2)$ .

Зауважимо, що існують алгоритми, що дозволяють підтримувати опуклу оболонку множини точок при дозволених операціях видалення точок із множини. Деякі з них базуються на методі верхніх та нижніх ланцюгів, і мають складність  $O(N \log N)$ , по  $O(\log N)$  на кожне видалення. Застосування цього алгоритму, однак, не пришвидчить загальну складність розв'язку, оскільки в запропонованому розв'язку пошук розділяючої прямої здійснюється за лінійний час на кожному кроці. Науковцями описаний алгоритм, який дозволяє шукати таку пряму за час  $O(\log N)$ , таким чином дозволивши знизити оцінку всього розв'язку до  $O(N \log N)$ . Журі пропонує учасникам олімпіади розібратися зі згаданими алгоритмами у якості вправи.

### Література

1. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. У вправі 35–3 згаданий інший варіант розв'язку задачі зі складністю  $O(N^2 \log N)$ .
2. Ratko Totic, Ivan Stojmenovic. On pairing points in the plane, 1991, NSJOM vol. 21–2, pp. 157–160.
3. John Hershberger and Subhash Suri. Applications of a Semi-Dynamic Convex Hull Algorithm, 1992, BIT vol. 32–2, pp. 249–267.