

# Алгоритми розбиття ізотетичного багатокутника

Дзюменко Артем

Механіко-математичний факультет

4 грудня 2025 р.

## 1 Вступ

Розбиття прямокутного (ортогонального або ізотетичного) багатокутника на підмножину прямокутників є класичною задачею в обчислювальній геометрії. У найпростішому варіанті маємо простий багатокутник із прямокутними кутами (тобто всі внутрішні кути мають величину  $90^\circ$  або  $270^\circ$ ), і потрібно побудувати множину попарно непересічних прямокутників, які повністю покривають область багатокутника. При цьому прямокутники повинні бути вирізані з багатокутника відрізками, паралельними осям координат, а їх об'єднання має точно збігатися з початковою фігурою.

Задача виникає в різноманітних застосуваннях: від автоматизованого проектування мікросхем (VLSI layout), де області провідників і контактів часто подаються у вигляді ортогональних багатокутників [4], до комп'ютерної графіки, геоінформаційних систем і задач обробки зображень, де складні області зручно подати у вигляді набору прямокутників, що спрощує їх подання та полегшує аналіз форми або стиснення. У цих застосуваннях зазвичай мета стоїть знайти оптимальне чи мінімальне таке розбиття. Наприклад, така задача виникає природним чином у проектуванні схем “місто кварталів” (англ. Manhattan layouts) [2].

Задача подібних розбиттів має цікаву історію з точки зору обчислювальної складності. Довгий час залишалося відкритим питання, чи є задача знаходження мінімального за кількістю прямокутників розбиття ортогонального багатокутника поліноміально розв'язною, чи ж вона є NP-складною. Деякі ранні роботи припускали, що задача може бути NP-важкою [3]. Однак Імаї та Асано [1] показали, що для класу ізотетичних багатокутників (тобто таких, де всі ребра паралельні лише двом координатним осям) мінімальне прямокутне розбиття можна знайти за час  $O(n^{3/2} \log n)$ , де  $n$  — кількість вершин багатокутника. Згодом було показано, що задача мінімального прямокутного розбиття загального ортогонального багатокутника з отворами є NP-складною, тоді як для багатокутників без отворів існують поліноміальні алгоритми [5]. Алгоритм Імаї–Асано є ключовим результатом у цьому напрямку і є центральною темою даної роботи.

Мета цієї роботи — описати основні ідеї алгоритму Імаї–Асано для розбиття ізотетичних багатокутників,

розглянути необхідні геометричні та графові структури, а також обговорити його реалізацію. У подальших розділах послідовно розглядаються поняття вгнутих вершин, кандидатних хордів, побудова графа перетинів, зведення до задачі максимального парування в дводольному графі та алгоритм знаходження максимальної множини непересічних хорд, яка й визначає прямокутне розбиття мінімального розміру.

## 2 Структури даних та методи

Мета полягає в тому, щоб отримати таке розбиття з мінімально можливою кількістю прямокутників. Для простих ізотетичних багатокутників без отворів відомо, що оптимальне розбиття можна сформулювати в термінах розміщення спеціальних відрізків — хорд, які з'єднують певні пари вгнутих вершин багатокутника. Таким чином, центральною ланкою алгоритму Імаї–Асано є побудова та аналіз множини хордів-кандидатів, а також побудова графа їхніх перетинів.

Ізотетичним багатокутником (англ. rectilinear polygon) називають багатокутник на площині, чийі всі ребра паралельні лише осям  $x$  та  $y$  (тобто кожне ребро є або горизонтальним, або вертикальним). Внутрішні кути такого багатокутника можуть мати величину  $90^\circ$  або  $270^\circ$ . Вершини з внутрішнім кутом  $270^\circ$  називають вігнутими (або рефлексними), а з кутом  $90^\circ$  — опуклими.

Стандартним підходом є перетворення задачі розбиття багатокутника на задачу знаходження максимальної множини попарно непересічних хордів, де кожна хорда з'єднує дві вігнуті вершини і цілком лежить у внутрішній області багатокутника. Після розміщення всіх таких хордів внутрішня область багатокутника розбивається на прямокутники.

Головні кроки алгоритму можна коротко описати так:

1. Визначити всі вігнуті вершини багатокутника.
2. Побудувати множину горизонтальних і вертикальних хорд-кандидатів, кожна з яких з'єднує пару вігнутих вершин, що лежать на одній горизонтальній або вертикальній прямій.

3. Побудувати граф, вершинами якого є хорди-кандидати, а ребра з'єднують пари хорд, що перетинаються.
4. Знайти максимальну за розміром множину попарно непересічних хорд; це еквівалентно знаходженню максимального незалежного множини вершин у графі перетинів (або максимального парування в певному похідному дводольному графі).
5. На основі обраних хорд побудувати прямокутне розбиття багатокутника.

У подальшому припускаємо, що багатокутник задано впорядкованим списком його вершин, і окремо задані (за потреби) межі отворів. Для коректного визначення вігнутих вершини важливо фіксувати орієнтацію обходу (за або проти годинникової стрілки) для зовнішньої межі та отворів.

### 3 Утворення хорд-кандидатів

Перший важливий крок — визначити всі вігнуті вершини багатокутника. Нехай вершини зовнішньої межі впорядковані за орієнтованим обходом. Для кожної вершини  $v_i = (x_i, y_i)$  розглянемо її сусідів  $v_{i-1}$  та  $v_{i+1}$ . Напрямки векторів

$$\vec{e}_{i-1} = v_i - v_{i-1}, \quad \vec{e}_i = v_{i+1} - v_i$$

дозволяють визначити знак повороту (через векторний добуток або орієнтований трикутник). Для ізотетичних багатокутників ребра завжди паралельні осям, тому кут між послідовними ребрами легко визначити. Якщо при фіксованій орієнтації обходу (скажімо, за годинниковою стрілкою) внутрішній кут у вершині  $v_i$  дорівнює  $270^\circ$ , кажуть, що вершина вігнута.

Визначивши множину вігнутих вершин, алгоритм Імаї–Асано розглядає кандидати на хорди, які з'єднують пари вігнутих вершин зі спільною координатою  $x$  (вертикальні хорди) або зі спільною координатою  $y$  (горизонтальні хорди). Ідея полягає в тому, що будь-яку внутрішню межу нескладно подати як послідовність таких вігнутих “кутів”, тож потенційні хорди виникають природно з геометрії багатокутника.

Визначення хорди (розрізу). Алгоритм зосереджується на кандидатах на розрізи, які називаються хордами. Хорда — це відрізок, що:

- сполучає дві вігнуті вершини багатокутника;
- паралельний одній з координатних осей (тобто горизонтальний або вертикальний);
- повністю лежить усередині багатокутника, включаючи кінці, за винятком, можливо, точкових доділів до межі.

Якщо дві вігнуті вершини мають однакову координату  $y = Y_0$  і лежать одна над одною ( $x = X_0$ ,  $x = X_1$ ) і між ними є вільний внутрішній простір, їх можна з'єднати горизонтальною хордою. Аналогічно, вігнуті вершини на одній вертикальній прямій  $x = X_0$  можуть бути з'єднані вертикальною хордою, якщо між ними немає перешкод.

Ці хорди-кандидати утворюють надлишкову множину; не всі з них входять до оптимального розбиття. Однак важливим є те, що кожне оптимальне розбиття може бути отримане як підмножина такої множини кандидатів [1]. Тому головне завдання полягає в тому, щоб із усіх можливих хорд-кандидатів вибрати максимальну множину попарно непересічних, що максимізує число відрізків (або еквівалентно мінімізує кількість прямокутників у розбитті).

### 4 Побудова графа перетинів кандидатних розрізів

Для формального аналізу множини хорд-кандидатів вводять граф перетинів. Нехай  $C$  — множина всіх кандидатних хорд (як горизонтальних, так і вертикальних). Визначимо неорієнтований граф  $G_C = (V_C, E_C)$ , де:

- кожній хорді  $c \in C$  відповідає вершина  $v_c \in V_C$ ;
- між вершинами  $v_{c_1}$  і  $v_{c_2}$  існує ребро тоді й лише тоді, коли відповідні хорди  $c_1$  та  $c_2$  перетинаються в межах внутрішньої області багатокутника.

За означенням, дві хорди перетинаються, якщо вони мають одну спільну внутрішню точку (або перетинаються в інтер'єрі, або один із кінців лежить на іншій хорді). Оскільки всі хорди паралельні осям, перетин можливий лише між горизонтальною та вертикальною хордою. Отже, граф перетинів природним чином є дводольним: усі його ребра з'єднують вершину з множини горизонтальних хорд з вершиною з множини вертикальних.

Визначення графа перетинів дозволяє перевести геометричну задачу у суто графову: знайти таку максимальну за розміром підмножину вершин  $I \subseteq V_C$ , що жодні дві вершини з  $I$  не з'єднані ребром (тобто  $I$  є незалежною множиною в графі  $G_C$ ). Елементи  $I$  відповідають множині попарно непересічних хорд, які можуть бути одночасно присутніми в розбитті. Саме ця множина визначає розбиття на прямокутники.

Наївний пошук максимальної незалежної множини в загальному графі є NP-складною задачею. Однак граф перетинів для хорд-кандидатів має спеціальну структуру, яку можна використати, щоб звести задачу до більш відомої задачі максимального парування в дводольному графі, для якої існують поліноміальні алгоритми (Hopcroft–Karp, тощо).

## 5 Структура графа та її геометричні властивості

Щоб уявити структуру графа, можна розташувати всі горизонтальні хорди в одній частці дводольного графа (назвемо її  $H$ ), а всі вертикальні — в іншій частці (назвемо її  $V$ ). Ребро з'єднує хорд  $h \in H$  та  $v \in V$ , якщо вони перетинаються. Важливою властивістю є те, що зв'язність між вершинами  $H$  та  $V$  має інтервальний характер.

Ідея інтервальності полягає в тому, що якщо зафіксувати порядок, у якому вертикальні хорди з'являються при русі вздовж осі  $x$ , а також порядок горизонтальних хорд при русі вздовж осі  $y$ , то для кожної хорди з  $H$  (або  $V$ ) множина її сусідів у протилежній частці утворює неперервний інтервал індексів. Іншими словами, якщо горизонтальна хорда  $h$  перетинає вертикальні хорди з індексами від  $i$  до  $j$ , то вона перетинає усі вертикальні хорди між  $i$  та  $j$ , а не вибірково піднабір. Така інтервальна структура означає, що граф має структуру інтервального дводольного графа.

Такі графи називають опуклими дводольними графами, оскільки в упорядкуванні вершин однієї частки кожна вершина іншої частки має множину сусідів, яка утворює безперервний блок (інтервал) цих впорядкованих вершин. Це впливає з геометричного факту: якщо горизонтальна хорда “сканує” вертикальні хорди вздовж осі  $x$ , то перетини виникають лише в певному діапазоні координат між крайніми вертикальними хордами, що перетинають дану горизонтальну. Такі діапазони завжди утворюють неперервні інтервали.

Опуклість графа не є критичною для коректності алгоритму, але вона відіграє ключову роль у прискоренні наступного кроку — пошуку максимального парування. Інтервальна структура дозволяє ефективно реалізувати алгоритм Hopcroft–Karp або його модифікацію так, щоб досягти часу роботи  $O(n^{3/2} \log n)$ , як у класичній роботі Імаї–Асано [1].

## 6 Перетворення до задачі дводольного зіставлення

Задача пошуку максимальної множини попарно непересічних хорд-кандидатів відображається в задачу максимального парування або максимального незалежного множини у спеціально побудованому дводольному графі. Один зі способів — розглянути граф перетинів  $G_C$  і побудувати його доповнення в деякій частці, однак пряме перетворення може бути складним.

Більш зручний підхід, описаний у [1], полягає в тому, щоб побудувати похідний граф, у якому:

- вершини відповідають інтервалам перетину хорд з протилежної частки;

- ребра виражають можливість “узгодженого” вибору хорд, що не конфлікують одна з одною.

У цьому графі задача обрання максимальної множини непересічних хорд зводиться до задачі пошуку максимального парування. Завдяки опуклій структурі, побудова цього графа та реалізація алгоритмів пошуку парування можуть бути виконані значно ефективніше, ніж у загальному випадку.

Інтуїтивно, якщо кожна хорда задається інтервалом координат, на яких вона перетинає хорди протилежної орієнтації, то конфлікуючі хорди мають перетин цих інтервалів. Завдання — обрати максимально можливу кількість інтервалів, які не перетинаються. Така задача близька до класичної задачі про максимальну за розміром множину непересічних інтервалів, однак з додатковими обмеженнями, які виникають із дводольної структури.

## 7 Алгоритм максимального парування (Гопкрофт–Карп та підхід Імаї–Асано)

Алгоритм Гопкрофта–Карпа є класичним методом пошуку максимального парування в загальному дводольному графі за час  $O(\sqrt{|V|}|E|)$ , де  $|V|$  і  $|E|$  — кількість вершин і ребер відповідно. Для графа перетинів хорд-кандидатів або похідного від нього графа цей підхід можна пристосувати, використовуючи особливості опуклої (інтервальної) структури.

Загальна схема алгоритму:

1. Ініціалізувати парування як порожнє.
2. Повторювати, поки існує збільшувальний шлях (augmenting path):
  - виконати пошук у ширину (BFS) від усіх непарних вершин однієї частки, щоб знайти всі найкоротші збільшувальні шляхи;
  - виконати поглиблений пошук (DFS), щоб знайти й “збільшити” всі такі шляхи, оновлюючи парування.

У контексті алгоритму Імаї–Асано пропонується використовувати модифіковану реалізацію, де опуклість дводольного графа дозволяє обмежити кількість ребер, що розглядаються на кожному кроці, та забезпечити загальну оцінку часу  $O(n^{3/2} \log n)$  для всього алгоритму [1].

## 8 Отримання максимальної множини непересічних хорд

Коли максимально парування в відповідному дводольному графі знайдено, його можна інтерпретувати як

множину хорд, що “конфліктують” найменшим можливим чином. На основі цього парування визначається максимальна незалежна множина в графі перетинів, що, у свою чергу, відповідає максимальній множині попарно непересічних хорд.

Формально, між задачами максимального парування та максимальної незалежної множини в дводольному графі існують добре відомі зв'язки. У нашому випадку, опукла структура та спосіб побудови графа дозволяють явно відновити множину хорд, які можуть співіснувати без перетинів. Кожна така хорда додається до розбиття, а їхня сукупність задає повну систему внутрішніх розрізів.

## 9 Процедура розбиття

Після того як знайдено максимальну множину непересічних хорд, багатокутник можна розбити на прямокутники в такий спосіб:

1. Розмістити всі хорди з отриманої множини всередині багатокутника.
2. Розглядати отриманий планарний розбитий граф: його вершинами є кути багатокутника та точки перетину хорд із межами, а ребрами — частини початкових ребер і хорд.
3. Виділити всі грані цього графа, які належать внутрішній області багатокутника; кожна така грань є прямокутником.

Алгоритмічно, це можна реалізувати за допомогою структури даних для планарних графів або, в спеціальному випадку прямокутних хордів, за допомогою сканування по координатній сітці. Важливо переконатися, що кожна грань має чотири кути по  $90^\circ$  (тобто є прямокутником) і що об'єднання всіх таких прямокутників точно покриває багатокутник без перекриттів.

Проведення хорд і фінальне розбиття. У реалізації зазвичай вважають, що хорди розміщені як відрізки, паралельні осям, і належать внутрішній області багатокутника. Перетини між хордою та межею багатокутника або іншими хордами відповідно додаються як вершини до внутрішнього графа. Після цього можна виконати обхід усіх граней і перевірити, що кожна грань є прямокутником.

## 10 Аналіз часу виконання

Аналіз складності алгоритму Імаї–Асано спирається на кілька ключових спостережень:

- кількість вігнутих вершин ізотетичного багатокутника є  $O(n)$ ;
- кількість хордів-кандидатів, утворених на основі цих вершин, також є  $O(n)$ ;

- граф перетинів має спеціальну опуклу (інтервальну) структуру, яка дозволяє зменшити число ребер, що розглядаються явно.

Оцінка часу включає такі етапи:

1. Визначення вігнутих вершин та побудова кандидатних хорд може бути здійснена за  $O(n \log n)$  часу (або навіть за  $O(n)$  при акуратній реалізації сканувальних процедур).
2. Побудова графа перетинів наївно потребує  $O(n^2)$  часу, але завдяки сканувальним алгоритмам та інтервальної структурі можна досягти меншої асимптотики.
3. Застосування алгоритму Гопкрофта–Карпа або його модифікацій до опуклого дводольного графа дозволяє реалізувати пошук максимального парування за  $O(n^{3/2} \log n)$  часу.

Сумарно це дає оцінку  $O(n^{3/2} \log n)$  для всього алгоритму мінімального прямокутного розбиття ізотетичного багатокутника [1]. Цей результат є суттєвим, оскільки показує, що задача, яку спершу підозрювали в NP-складності, насправді має досить ефективний точний алгоритм для важливого класу багатокутників без отворів.

## 11 Алгоритмічна реалізація та оцінки складності

Реалізація алгоритму Імаї–Асано складається з кількох послідовних етапів. Нехай  $n$  – загальна кількість вершин прямокутного багатокутника (разом з отворами),  $r$  – кількість вігнутих (рефлексних) вершин,  $H$  – кількість горизонтальних хорд (відрізків видимості),  $V$  – кількість вертикальних хорд, а  $M$  – кількість ребер у дводольному графі їхніх перетинів. Для простоти припускається, що полігон простий, ортогональний, з фіксованою кількістю отворів, тобто всі асимптотичні оцінки подані відносно  $n$ .

Попередня обробка полігона. На першому кроці виконується перевірка вхідних даних функцією `validate`, яка проходить по всіх вершинах кожного контуру й перевіряє, що кожне ребро є або горизонтальним, або вертикальним (тобто не допускаються діагональні відрізки). Цей крок має час  $O(n)$  та потребує  $O(1)$  додаткової пам'яті (окрім збереження самого полігона). Далі за допомогою `fix_orientation` для кожного контуру обчислюється орієнтований “псевдоплоща”  $\sum_i x_i y_{i+1} - y_i x_{i+1}$  і зовнішня межа приводиться до канонічної орієнтації проти годинникової стрілки, а отвори – за годинниковою. Це також лінійний за часом крок  $O(n)$  з

константною додатковою пам'яттю. Функція `determine_convexity` для кожної вершини обчислює знак векторного добутку сусідніх ребер і, з урахуванням орієнтації контуру, класифікує вершину як рефлексну (кут  $270^\circ$ ) або опуклу (кут  $90^\circ$ ), після чого записує цю ознаку в структуру даних вершини. Весь цей етап займає час  $O(n)$  і додає лише одну логічну змінну на вершину, тобто не змінює асимптотично обсяг пам'яті.

Після цього багатокутник розкладається на дві множини ребер – вертикальні та горизонтальні – а також формується список усіх рефлексних вершин. Це реалізує функція `classify_edges`, яка для кожного ребра зберігає нормалізовані координати (наприклад, `y_low`, `y_high` для вертикальних ребер, `x_low`, `x_high` для горизонтальних) та додаткову службову інформацію (належність до зовнішнього контуру чи отвору, локальний індекс вершини, напрямок ребра). Цей крок також має час  $O(n)$  і формує три масиви сумарного розміру  $O(n)$ , які використовуються на подальших етапах.

Побудова відрізків видимості (хорд). Наступним кроком є побудова множини внутрішніх горизонтальних і вертикальних відрізків видимості від рефлексних вершин до найближчих меж полігона. Функція `horizontal_sweep` виконує сканування по осі  $y$ : попередньо формуються події початку/кінця вертикальних ребер та проходження через рефлексні вершини, які сортуються за координатою  $y$  та типом події. Сортування коштує  $O((n+r) \log(n+r))$ . У процесі сканування підтримується збалансована структура `SortedList` активних вертикальних ребер, відсортованих за координатою  $x$ . Оновлення цієї структури для кожної події (додавання/видалення ребра) виконується за  $O(\log n)$  часу, а кількість подій є  $O(n+r)$ . Для кожної рефлексної вершини виконується двійковий пошук за  $x$ -координатою у списку активних ребер та локальні операції для формування до двох горизонтальних відрізків; отже, сумарна складність цього етапу є порядку

$$O((n+r) \log n + S_H),$$

де  $S_H$  – кількість породжених горизонтальних відрізків (у типовому випадку  $S_H = O(n)$ ). Потрібна пам'ять –  $O(n)$  для списків подій, активних ребер та масиву відрізків.

Аналогічно, `vertical_sweep` сканує по осі  $x$  і будує вертикальні відрізки видимості, використовуючи `SortedList` активних горизонтальних ребер, відсортованих за  $y$ . Цей етап так само має час

$$O((n+r) \log n + S_V),$$

де  $S_V$  – кількість вертикальних відрізків, і використовує  $O(n)$  додаткової пам'яті. В підсумку множини горизонтальних та вертикальних хорд мають сумарний розмір  $O(H+V)$ , який у класичному аналізі для прямокутних полігонів оцінюється як  $O(n)$ .

Побудова дводольного графа перетинів. Маючи множини горизонтальних ( $H$ ) та вертикальних ( $V$ ) відрізків, реалізація будує дводольний граф  $G = (H \cup V, E)$ , де ребро з'єднує горизонтальний відрізок із вертикальним, якщо вони перетинаються всередині полігона. Функція `build_bipartite_graph` виконує sweep за координатою  $x$ : горизонтальні відрізки задаються інтервалами  $[x_{\text{low}}, x_{\text{high}}]$ , вертикальні виступають точковими запитами в певних значеннях  $x$ . Події (початок/кінець горизонтальних відрізків та вертикальні запити) сортуються за  $x$  та типом події, що дає  $O((H+V) \log(H+V))$  часу. У процесі сканування підтримується впорядкований за  $y$  список активних горизонтальних відрізків (`SortedList`), що дозволяє за  $O(\log H)$  часу додавати й видаляти відрізки.

Для кожного вертикального відрізка з діапазоном  $[y_{\text{min}}, y_{\text{max}}]$  через двійкові пошуки знаходиться піддіапазон активних горизонталей, які потрапляють у цей вертикальний інтервал. Кожна така пара “горизонтальний–вертикальний” породжує ребро дводольного графа. Нехай  $M = |E|$  – кількість таких перетинів. Тоді сумарний час побудови графа становить

$$O((H+V) \log(H+V) + M),$$

а потрібна пам'ять –  $O(H+V+M)$  для зберігання хорд та списків суміжності `adj_H`, `adj_V`. Відомо, що для множини осьово-орієнтованих відрізків, отриманих із прямокутного полігона, число перетинів  $M$  можна оцінити як  $O(n^{3/2})$ , що є ключовим для досягнення асимптотики Імай-Асано.

Максимальне пароспівставлення в конвексному дводольному графі. Ключова властивість побудованого графа полягає в тому, що після впорядкування вертикальних хорд за координатою  $x$  множина сусідів кожної горизонтальної хорди є неперервним інтервалом за індексами вертикальних вершин, тобто граф є конвексним щодо однієї долі. Функція `max_matching_convex` використовує цю конвексність: спочатку для кожної горизонталі  $h$  зі списком сусідів `adj_H[h]` обчислюються межі інтервалу  $[L_h, R_h]$  (найменший та найбільший індекс вертикальних сусідів). Це вимагає  $O(H+M)$  часу й  $O(H)$  пам'яті для збереження інтервалів.

Далі застосовується жадібний алгоритм розкладу інтервалів по точках: вертикальні вершини перебираються у зростаючому порядку; у пріоритетній черзі (за правими кінцями  $R_h$ ) підтримуються всі “активні” інтервали з  $L_h \leq v$ . Для кожного вертикального індексу  $v$  видаляються з черги всі інтервали з  $R_h < v$ , після чого (якщо черга не порожня) вибирається інтервал з найменшим  $R_h$  і будується пара  $(h, v)$  в пароспівставленні, якщо відповідні вершини ще не зайняті. Такий алгоритм реалізується за  $O((H+V) \log H)$  часу, де фактор  $\log H$  походить від операцій над пріоритетною чергою. Додаткова пам'ять для цього етапу становить

$O(H+V)$  (масиви `mate_H`, `mate_V`, пріоритетна черга та список інтервалів).

Максимальна незалежна множина хорд і побудова розбиття. Отримане максимальне пароспівставлення перетворюється на максимальну незалежну множину вершин дводольного графа за допомогою стандартної процедури через мінімальне вершинне покриття. Функція `max_independent_set_from_matching` виконує пошук чергуючих шляхів від усіх вільних вершин лівої долі (горизонтальних хорд), позначаючи досяжні вершини у множинах  $Z_H$  та  $Z_V$ . Це BFS по ребрах графа з розмежуванням спарених і неспарених ребер, який має часову складність  $O(H+V+M)$  та використовує  $O(H+V)$  пам'яті. На основі отриманих множин формується мінімальне вершинне покриття  $C$ , а максимальна незалежна множина є його доповненням  $I = (H \cup V) \setminus C$ .

Нарешті, функція `augment_with_remaining_reflexes` добудовує відрізки від тих рефлексних вершин, які не мають жодної інцидентної хорди в побудованій незалежній множині. Для кожної такої вершини обирається будь-який наявний відрізок видимості (горизонтальний або вертикальний), що забезпечує охоплення всіх вігнутих кутів і відповідає кроку добу-дови “максимальних” вертикальних/горизонтальних ліній у класичних описах алгоритму. Цей етап у най-гіршому разі має час  $O(r+H+V)$  і потребує  $O(H+V)$  пам'яті для структурування хорд за `vertex_id`.

Загальна оцінка. Сумарна асимптотична складність реалізації домінується етапом побудови дводольного графа та пошуку пароспівставлення. За умови, що кількість перетинів між хордами задовольняє оцінку  $M = O(n^{3/2})$ , часову складність можна записати як

$$O(n \log n) + O(M) + O((H+V) \log(H+V)) = O(n^{3/2} \log n),$$

де  $H+V = O(n)$ . Оперативна пам'ять, необхідна для роботи алгоритму, також є поліноміально обмеженою і дорівнює  $O(n+M)$  (зберігання полігона, списків хорд, графа перетинів та допоміжних структур для sweeper-алгоритмів і пароспівставлення). Таким чином, імплементація відтворює теоретичну асимптотику алгоритму Імаї-Асано  $O(n^{3/2} \log n)$  для задачі оптимального розбиття прямокутного багатокутника на мінімальну кількість прямокутників.

## Література

- [1] Н. Imai та Т. Asano. «Efficient algorithm for geometric graph search problems». В: ().
- [2] W. Lipski. «Finding a Manhattan path and related problems». В: ().
- [3] W. Lipski та ін. «On two-dimensional data organization II». В: ().

[4] V. P. Rubtsov. «Realization of VLSI topology by rectangles». В: ().

[5] Alexei Gorpinevich Valeriu Soltan. «Minimum Dissection of a Rectilinear Polygon with Arbitrary Holes into Rectangles». В: ().