Задача A. Масив і ще кілька масивів

Автор задачі: **Антон Тригуб**
Задачу підготував: **Владислав Денисюк**
Розбір написав: **Владислав Денисюк**

Блок 1

Оскільки ми лише можемо додати однакове число d до усіх елементів масиву, то додамо таке число d, щоб максимальний елемент масиву став рівний n, а мінімальний — 1, якщо це неможливо, то відповідь «`No`». Інакше потрібно додати d до усіх елементів масиву і перевірити, чи дійсно після такої операції усі числа будуть різними, якщо так, то відповідь «`Yes`», інакше «`No`».

Блок 2

У цій групі кожен масив можна уявити як інтервал чисел. Фактично ми маємо k інтервалів, ми завжди можемо додати для кожного з них числа так, щоб усі числа на першому інтервалі були від 1 до l_1, на другому - від $l_1 + 1$ до $l_1 + l_2$, і так далі. Ми уявляємо собі масиви як інтервали чисел які ми можемо рухати, і наша ціль — скласти з них інтервал $[1; n]$.

Блок 3

У нас є дві групи, оскільки наша ціль — утворити n різних цілих чисел, то така множина чисел матиме лише один мінімум — число 1. Далі помітимо, що після додавання відповідних чисел, якщо вони формують правильну відповідь, 1 з'явиться або в першому або в другому масиві. Переберемо обидва варіанти. Нехай ми розглядаємо варіант, де мінімум після додавання буде у першому масиві. Нехай мінімум у першому масиві до додавання був m_1, тоді потрібно буде додати $1 - m_1$ до першого масиву, щоб його мінімум став рівний 1. Потім у нас залишиться другий масив і множина чисел, якої нам не вистачає — числа від 1 до n, яких немає в першому масиві. Нехай найменше таке число — x. Тоді потрібно буде додати $x - m_2$ (де m_2 — мінімум у другому масиві до додавання) до другого масиву і перевірити, чи така комбінація чисел підходить. Аналогічно розглянемо варіант, де мінімум у другому масиві. Якщо жоден варіант не підійшов, то відповідь — «`No`».

Блок 4

Рішення схоже на рішення блоку `3`. Будемо перебирати порядок мінімумів серед цих трьох масивів, тобто будемо перебирати, який масив матиме найменший мінімум, другий найменший, і найбільший. Аналогічно по черзі прирівнюватимемо мінімум першому числу, яке відсутнє у вже оброблених масивах і так будувати відповідь. Всього є 6 варіантів такого порядку `[1 2 3]`, `[1 3 2]`, `[2 1 3]`, `[2 3 1]`, `[3 1 2]`, `[3 2 1]`.

Блок 5

Рішення цього блоку схоже на рішення блоку `2`. Масиви діляться на дві групи — масиви, де після додавання усі числа будуть парні, і масиви, де після додавання усі числа будуть непарні. Групу кожного масиву можна перебрати — всього буде 2^k варіантів. Для кожної з цих груп нам потрібно знайти такі числа d_i, щоб множина чисел в одній групі була рівна `[1 3 5 7...]`, а друга - `[2 4 6 8...]`. Фактично можемо знову уявити масиви як інтервали, як ми це робили в блоці `2`, але тут уже потрібно зважати на те, що у нас просто немає в одному випадку парних точок, а в іншому - непарних.

Блок 6

Помітимо, що усі числа в кінцевій множині повинні бути різні, це означає, що максимальний мінімум рівний k, оскільки максимум рівний n, і в якому масиві він би не був — мінімумом такого масиву буде $n-(n-k) = k$. Тому мінімум кожного масиву після додавання буде від 1 до k. Переберемо значення кожного мінімуму і перевіримо кожен варіант. Складність такого рішення — $O(n \cdot k!)$.

Блок 7

Кожен мінімум буде рівний числу від 1 до n. Переберемо, чому рівний мінімум кожного масиву і перевіримо кожен варіант. Складність такого алгоритму буде $O(n^{k+1})$.

Блок 8

Будемо робити те, що ми робимо в блоці 4 для трьох масивів, але для будь-якої кількості до п'яти. Перебиратимемо перестановку чисел від 1 до k, якою ми задаватимемо відносний порядок мінімумів масивів за зростанням. Для кожної перестановки використаємо алгоритм описаний в блоці 4 — знаходимо мінімальне число, що не зустрічалось у вже пройдених масивах і перетворюємо на нього мінімум масиву, що розглядається. При перевірці всіх таких варіантів ми точно перевіримо усі комбінації чисел що могли б бути відповіддю, оскільки у правильній відповіді усі числа різні, а отже мають відносний порядок. Складність такого рішення — $O(n \cdot k!)$.

Задача B. Масив монет і запити зважування

Автор задачі: **Денисов Костянтин**
Задачу підготував: **Денисов Костянтин**
Розбір написав: **Денисов Костянтин**

Блок 1

З обмеження $k < n$ випливає, що хоча б одна з монет з номерами n та 1 є справжньою. Тому першим запитом, поклавши на чаші першу та n-ту монету, дізнаємося номер однієї справжньої монети. Далі пройдемося по всім монетам зліва направо і порівняємо знайдену справжню монету з поточною монетою, якщо поточна монета легша, то вона і буде першою фальшивою монетою. Отже, маємо рішення, яке використовує не більше n запитів.

Блок 2

Нехай $[l, r]$ — поточний відрізок монет, на якому шукаємо фальшиву монету. Початково $[l, r]$=$[1, n]$. Нехай $m = \lfloor \frac{r-l+1}{2} \rfloor$. Виділимо два підвідрізка $[l, r]$, які не перетинаються: $[l, l + m - 1]$, $[r - m + 1, r]$. Покладемо на першу чашу терезів монетки з номерами з першого відрізка і на другу чашу монети з другого відрізка. Якщо монети з номерами з першого відрізка легші, то на цьому відрізку знаходиться фальшива монета і перейдемо до задачі $[l, r] := [l, l + m - 1]$. Аналогічно коли монетки з номерами з другого відрізка легші. Якщо ж ваги врівноважені, то монетка, що знаходиться між цими відрізками (така буде рівно одна) і є фальшивою.

Отже, кожного разу ми зменшуємо довжину відрізка пошуку принаймні вдвічі, тому маємо рішення, яке використовує не більше $\lceil log_2(n) \rceil \leqslant \lceil log_2(10^4) \rceil \leqslant 14$ запитів.

Блок 3

Використаємо схожу ідею як у попередньому блоці, але будемо ділити відрізок не на дві частини, а на три частини. Нехай $m = \lceil \frac{r-l+1}{3} \rceil$. Покладемо на першу чашу терезів монетки з номерами з відрізка $[l, l + m - 1]$ і на другу чашу монети з відрізка $[r - m + 1, r]$ (такі ж відрізки як у другому блоці, але з іншим m). Коли терези врівноважені, то фальшива монета знаходиться поза даними відрізками, тому продовжуємо пошук на відрізку $[l+m, r-m]$. Випадки, коли терези неврівноважені є аналогічними до другого блоку.

Отже, кожного разу ми зменшуємо довжину відрізка пошуку принаймні втричі, тому маємо рішення, яке використовує не більше $\lceil log_3(n) \rceil \leqslant \lceil log_3(10^4) \rceil \leqslant 9$ запитів.

Блок 4

Нехай $A = \{1, k + 1, 2 \cdot k + 1, ..., d \cdot k + 1\}$, де d — максимальне ціле число, що $d \cdot k + 1 \leqslant n$. Очевидно, що у цій множині є **рівно** одна фальшива монета. Знайдемо її алгоритмом з третього блоку. Нехай r — номер фальшивої монети, що ми отримали.

Тоді перша фальшива монета знаходиться на відрізку $[l, r]$, де $l = \max(1, r-k+1)$. Нехай $real$ — номер якоїсь справжньої монети (таку можна знайти, наприклад, додатковим зважуванням першої та останньої монети або з вже зроблених зважувань, використовуючи, що $|A| > 1$). Тоді можна запустити бінарний пошук по відрізку $[l, r]$, який буде порівнювати монету з номером, що відповідає середині поточного відрізка пошуку, з монетою з номером $real$. Якщо терези врівноважені, то перша фальшива монета буде у правій частині відрізка, інакше — у лівій.

Отже, перша частина рішення використовує не більше $\lceil log_3(\lceil \frac{n}{k} \rceil) \rceil$, а друга частина — $\lceil log_2(k) \rceil$. Отже, маємо рішення, яке використовує не більше $\lceil log_3(\lceil \frac{n}{k} \rceil) \rceil + \lceil log_2(k) \rceil \leqslant 11$ запитів.

Блок 5

Рішення п'ятого блоку повторюється з четвертим блоком крім частини з бінарним пошуком. Виберемо на поточному відрізку пошуку дві монети з номерами c та $d (c < d)$, щоб відрізки, на які ці номери поділяють відрізок були майже рівними за розмірами. Тоді покладемо на кожну чашу терезів дві монети. На першу — монети з номерами c та d. На другу — справжню монету з номером

real та фальшиву монету (одну фальшиву монету ми вже знаємо). Тоді оскільки фальшиві монети у цьому блоці мають однакову вагу, то:

- якщо ваги врівноважені, то монета з номером c — справжня та з номером d — фальшива і треба продовжувати пошук на відрізку $[c+1, d]$;
- якщо перша чаша важче, то монети з номерами c та d справжні і далі шукаємо на відрізку $[d+1, r]$;
- якщо друга чаша важче, то монети з номерами c та d фальшиві і далі шукаємо на відрізку $[l, c-1]$;

Отже, кожного разу відрізок пошуку зменшується принаймні втричі, тоді маємо рішення, що використовує не більше $\lceil log_3(\lceil \frac{n}{k} \rceil) \rceil + \lceil log_3(k) \rceil \leqslant 11$ запитів.

Блок 6

Нехай будемо підтримувати відрізок $[l, r]$, на якому знаходяться всі фальшиві монети. Тоді очевидно, що перша фальшива монета буде знаходитися на відрізку $[l, r-k+1]$. Нехай $m = \lceil \frac{r-k+1-l}{3} \rceil$. Покладемо на першу чашу вагів монетки з номерами з відрізка $[l, l+m-1]$ і на другу чашу монети з відрізка $[r-m+1, r]$. Помітимо, що не може бути такого, що на кожному з цих відрізків є фальшиві монети, оскільки довжина відрізка $[l+m-1, r-m+1] > k$. Тому маємо, що якщо ваги врівноважені, то фальшиві монети знаходяться на відрізку $[l+m, r-m]$. Якщо перша чаша важче, то — на відрізку $[r-m+1-k+1, r]$. Якщо друга чаша важче, то — на відрізку $[l, l+m-1+k-1]$. Будемо продовжувати зменшувати відрізок пошуку, поки його довжина більша за k.

Помітити, що кожного разу ми зменшуємо довжину відрізка, де знаходиться перша фальшива монета принаймні втричі, тому маємо рішення, що використовує не більше $\lceil log_3(n) \rceil \leqslant \lceil log_3(10^4) \rceil \leqslant 9$ запитів.

Задача C. Масив і додавання на відрізку

Автор задачі: **Антон Тригуб**
Задачу підготував: **Владислав Денисюк**
Розбір написав: **Владислав Денисюк**

Блок 1

Перше спостереження — це те, що ми можемо додавати нескінченно великі числа на відрізках і відповідно якщо ми додали на відрізку, то числа, які входять та не входять у відрізок, точно стали відрізнятись між собою. Розглянемо оптимальний вибір відрізків. Тоді назвемо кінці таких відрізків - перегородками. Фактичний зміст перегородки — це те, що інший кінець відрізка буде або зліва або справа від перегородки й відповідно змінить лише одне з чисел які вона розділяє. Тому нам потрібно мінімум $n-1$ перегородок між числами. Якщо жоден з відрізків не має кінця в точках 1 або n - то значення в точках 1 і n будуть рівними оскільки жоден з відрізків їх не чіпає. Тому потрібно принаймні n перегородок. Кожен відрізок має два кінці, тому може утворювати не більше двох перегородок. Відповідно нам потрібно $\lceil \frac{n}{2} \rceil$ перегородок. Покажемо, що такої кількості достатньо. Додаватимемо на відрізках $[1; \lceil \frac{n}{2} \rceil]$, $[2; \lceil \frac{n}{2} \rceil + 1]$, ... Якщо n парне - то в кінці ми додамо на відрізку $[\frac{n}{2}, n-1]$. Якщо ні, то на відрізках $[\lceil \frac{n}{2} \rceil; n-2]$ та $[n-1, n-1]$.

Блок 2

У цьому блоці ми фактично розв'язуємо задачу на двох масивах у яких дві перегородки спільні. Тому відповідь тут - $\lceil \frac{n-2}{2} \rceil$. Але тут потрібно врахувати два крайніх випадки: коли усі числа рівні, тоді потрібно використати розв'язок блоку 1 і випадок, коли лише одне число відмінне від інших - тоді воно або в кінці або на початку і ми розв'язуємо блок 1, але в масиві $n-1$ чисел - тому відповідь $\lceil \frac{n-1}{2} \rceil$.

Блок 3

У цьому блоці ми можемо використати рекурсивний перебір. Додавання на відрізку буде відрізняти числа на відрізку від чисел за межами відрізка. Всього ми можемо так зробити не більше $\lceil \frac{n}{2} \rceil$ разів, оскільки відповідь коли всі числа однакові точно не краще за відповідь коли деякі з них - різні. Таким чином маємо рекурсію де ми перебираємо n^2 варіантів відрізка $\lceil \frac{n}{2} \rceil$. Складність $O(n^n)$ що при максимальному n - $8^8 = 16777216$, це повинно проходити по часу.

Блок 4

Оскільки числа на позиціях 1 і n рівні одна з перегородок буде на початку або в кінці, також можна уявити що елементи 1 і n - сусідні, тобто числа розташовані по колу, і тоді нам потрібно розставити перегородки по колу, це нічого не змінює для даного блоку, оскільки ми завжди ставимо перегородку між 1 і n, але у наступних блоках така інтуїція буде корисною. Щоб розставити всі інші перегородки для кожного значення від 1 до n згенеруємо інтервали позицій на яких би ми могли розставити перегородки для чисел з даним значенням. Тепер у нас є кілька інтервалів і нам потрібно вибрати мінімальну кількість точок, щоб кожному інтервалу належала хоча б одна точка. Посортуємо кінці відрізків і додаватимемо відрізок коли зустрічатимемо лівий кінець. Коли ми зустріли правий кінець відрізка, якому ще не належить жодна перегородка ми не можемо поставити найлівішу з ще не поставлених перегородок правіше від даної позиції. Якщо ж ми поставимо таку перегородку лівіше - то це буде точно не краще, оскільки поставивши перегородку в даній позиції ми задовольнимо всі незадовільнені відрізки, що мають кінці лівіше або в даній точці. Тому поставимо таку перегородку і відмітимо всі незадовільнені відрізки у яких ми вже зустріли ліві кінці - як задовільнені. Повторимо поки не пройдемо всі відрізки. Отримаємо P перегородок і ще мусимо поставити додаткову перегородку на початку або в кінці, в сумі матимемо $P+1$ перегородок. Аналогічно конструкції з блоку 1 ми можемо знайти відрізки що задовольнять умову з кінцями в цих перегородках. Відповідь $\lceil \frac{P+1}{2} \rceil$.

Блок 5

Для будь-якої групи чисел помітимо що конструкція на перегородках яку ми будуємо в блоці 1 фактично теж повинна бути витримана відносно перегородок, оскільки при її побудові підхід є аксіоматичним і всі інші конструкції містять в собі дану. Помітимо, що якщо зациклити масив і ставити перегородки на такому масиві то нічого не змінюється, як бонус - ми автоматично отримуємо виконання умови, що перегородка має бути перед першим елементом або після останнього. Якщо ж масив циклічний - то "перед першим"і "після останнього це одне й те ж. Ми можемо перебрати точку де ми точно ставимо перегородку, взяти її за початок масиву і запустити сканлайн подібний до рішення в блоці 4.

Блок 6

Цей блок був також було створено для неоптимальних реалізацій. Зокрема в ньому немає потреби зжимати числа в масиві.

Блок 7

Для кожної точки порахуємо наступну точку справа в якій ми муситимемо поставити нову перегородку, це можна зробити знайшовши найближчий справа (по колу звісно) кінець відрізка, який ми не перетинаємо наразі, наприклад це можна порахувати сканлайном. Потім побудуємо граф на таких переходах. Кожна позиція має однозначного наступника, тому побудуємо бінарні підйоми на такому графі. Для кожної позиції порахуємо скільки перегородок потрібно поставити, якщо одну з перегородок ми ставимо в даній позиції і переходами ми покриваємо весь масив по колу. Складність $O(n * log(n))$.

Задача D. Масив і часткові суми

Автор задачі: **Valerio Stancanelli**
Задачу підготував: **Alessandro Bortolin**
Розбір написав: **Роман Білий**

Блок 1

Якщо усі числа 0 або 1, то відповідь рівна 0. Якщо ж усі числа -1 або 0, то відповідь рівна 1 — можна виконати одну операцію першого типу.

Тепер нехай ми хочемо зробити одну операцію 2 типу. Покажемо, що ми можемо її виконати на всьому масиві. Нехай існує проміжок $[l..r]$, що виконавши операцію другого типу на цьому проміжку, всі елементи стали невід'ємними. Для цього необхідно, щоб усі елементи $a_1, a_2, \ldots, a_{l-1}$ та $a_{r+1}, \ldots, a_n$ були невід'ємними. А отже після виконання операції другого типу на всьому масиві всі елементи стануть не меншими, ніж якщо виконати операцію на $[l..r]$. Отже ми можемо спробувати виконати її на всьому масиві і подивитись, чи стали всі елементи невід'ємними.

Аналогічно, спробуємо зробити операцію 3 типу на всьому масиві.

Блок 2

Розглянемо будь-який елемент зі значенням 1 (якщо такого не існує, зробимо одну операцію першого типу). Нехай його позиція p. Будемо робити операції другого типу на проміжках $[p..r_i]$ так, щоб кожного разу усі елементи, які замінюються, були невід'ємними. Наприклад, якщо масив $[1, -1, 0, -1, 1, -1, -1]$ та $p = 1$ (нумерація від 1), то зробимо операції на відрізках [1..3], [1..6], [1..7]: $[1, -1, 0, -1, 1, -1, -1] \to [1, 0, 0, -1, 1, -1, -1] \to [1, 1, 1, 0, 1, 0, -1] \to [1, 2, 3, 3, 4, 4, 3]$.

Таким чином ми зможемо зробити усі елементи правіше p невід'ємними. Аналогічно операціями 3 типу зробимо усі елементи лівіше p невід'ємними.

Найгірший випадок для такого розв'язку — усі елементи крім одного рівні -1. Можна експериментально визначити, що в такому випадку потрібно не більше 7 операцій 2 типу і 7 операцій 3 типу. Тобто такий розв'язок виконує не більше 14 операцій.

Блок 3

Покажемо, що завжди можливо використати не більше 3 операцій. Розглянемо масив $s_0 = 0$, $s_i = a_1 + a_2 + \ldots + a_i$.

Тоді якщо ми зробимо операцію другого типу на відрізку $[l..r]$, то масив стане рівним $[a_1, a_2, \ldots, a_{l-1}, s_l - s_{l-1}, s_{l+1} - s_{l-1}, \ldots, s_r - s_{l-1}, a_{r+1}, \ldots, a_n]$. А якщо зробимо операцію третього типу, то масив стане рівним $[a_1, a_2, \ldots, a_{l-1}, s_r - s_{l-1}, s_r - s_l, \ldots, s_r - s_{r-2}, s_r - s_{r-1}, a_{r+1}, \ldots, a_n]$.

Розглянемо тепер s_x — мінімальне значення масиву s та s_y — максимальне значення масиву s. Якщо $x > y$, виконаємо операцію першого типу, тобто домножимо усі значення a на -1, відповідно усі значення s також домножаться на -1, а мінімальне та максимальне значення поміняються місцями.

Тепер припустимо ми виконаємо операцію другого типу на відрізку $[x+1..n]$. Усі значення на цьому проміжку стануть рівними $a'_i = s_i - s_x \geqslant 0$. А також $a'_i = s_i - s_x = s_{i-1} + a_i - s_x = a_i + (s_{i-1} - s_x) \geqslant a_i$.

Аналогічно, якщо ми виконаємо операцію третього типу на відрізку $[1..y]$. Усі значення на цьому проміжку стануть рівними $a'_i = s_y - s_i \geqslant 0$.

Тому зробимо спочатку операцію другого типу на проміжку $[x+1..n]$, а потім операцію третього типу на відрізку $[1..y]$. Якщо б ми робили ці операції окремо, то усі значення стали б невід'ємними. Але після першої з цих двох операцій всі елементи стали не меншими, тому ми можемо виконати їх в даному порядку і усі значення стануть невід'ємними.

Блок 4

Поєднаємо розв'язок блоку 3 з розв'язком блоку 1.

Блок 5

Ми вміємо розв'язувати за одну операцію. А також точно існує розв'язок за 3 операції, який робить операцію 3 типу. Тому нам потрібно визначити, чи можливо зробити усі елементи невід'ємними за 2 операції 2 типу.

Ми вже знаємо, що другу операцію вигідно виконувати на всьому масиві. Нехай ми виконуємо першу операцію на проміжку $[l..r]$. Порахуємо для кожного r, яке повинне бути мінімальне значення a'_r після першої операції так, щоб усі значення на проміжку $[r+1..n]$ стали невід'ємними після другої операції.

Тепер переберемо l, після цього будемо перебирати r від l до n та підтримувати значення після першої та після другої операції. Відповідно ми можемо визначити, чи можливо розв'язати задачу за 2 операції другого типу.

Складність такого розв'язку $O(n^2)$.

Блок 6

Нехай ми виконуємо першу операцію на відрізку $[l..r]$. Тоді усі значення a_1, $a_1 + a_2$, ..., $a_1 + a_2 + \ldots + a_{l-1}$ повинні бути невід'ємними, а отже, якщо ми зробимо першу операцію з $l = 1$, то усі значення стануть не меншими. А отже ми можемо перебирати лише r.

Складність такого розв'язку $O(n)$.

Блок 7

Тепер потрібно визначити, чи можна розв'язати за 2 операції використовуючи операції не лише другого типу. Якщо ми виконуємо операцію першого типу, то операція другого чи третього типу повинна бути на всьому масиві. Переберемо ці варіанти.

Розв'язок для двох операцій третього типу аналогічний розв'язку з двома операціями другого типу.

Тепер розглянемо варіант, коли ми виконуємо спершу операцію третього типу, потім операцію другого типу. В іншому порядку розв'язок аналогічний.

Нехай ми робимо операцію 3 типу на проміжку $[l..r]$. Переберемо r, після цього переберемо l, рахуючи значення схожі до попередніх блоків. Складність такого розв'язку $O(n^2)$.

Блок 8

Будемо розв'язувати методом розділяй і володарюй. Нехай зараз ми розв'язуємо задачу, коли $L \leqslant l \leqslant r \leqslant R$. Нехай $\frac{M=L+R}{2}$.

Якщо $r \leqslant M$, то ми можемо перейти в меншу підзадачу L, M. Але також усі значення від $M+1$ повинні стати невід'ємними після другої операції. Для цього порахуємо, яке мінімальне значення повинне бути на позиції M після першої операції, щоб далі все було добре.

Аналогічно, якщо $l > M$, то ми можемо перейти в меншу підзадачу $M+1, R$. Але також усі значення до M повинні стати невід'ємними після другої операції. Для цього порахуємо, чи це дійсно станеться, а також яке значення буде на позиції M після другої операції.

Тепер, якщо $l \leqslant M < r$.

Переберемо r. Порахуємо значення $y1_r$ — яке буде значення на позиції $M+1$ після першої операції та $y2_r$ — мінімальне значення, яке повинне бути на позиції M після другої операції так, щоб справа всі значення стали невід'ємними. Щоб порахувати значення $y2_r$ необхідно виписати, як змінюються значення при даних операціях. Після чого потрібно виконати запити двох типів: додати пряму з заданими параметрами та порахувати максимальне значення в точці. Ці запити можна робити підтримуючи опуклу оболонку прямих.

Переберемо l. Порахуємо значення $x1_l$ — мінімальне значення на позиції $M+1$ після першої операції, щоб зліва усі значення стали невід'ємними. Припустимо y — значення на позиції $M+1$ після першої операції, тоді порахуємо такі 2 значення $x2_l$ та $x3_l$, що значення на позиції M після другої операції рівне $x2_l \cdot y + x3_l$. Для знаходження $x1_l$ також скористаємось методом опуклих оболонок.

Тепер потрібно знайти, чи існує така пара l, r, що $y1_r \geqslant x1_l$ та $x2_l \cdot y1_r + x3_l \geqslant y2_r$.

Посортуємо всі l, r разом по значеннях $x1_l$ та $y1_r$, починаючи від найбільших. Якщо ми розглядаємо наступне значення r, то додамо пряму з параметрами $y1_r, -y2_r$. Якщо ми розглядаємо наступне значення l, то потрібно перевірити, чи існує пряма, для якої значення в точці $x2_l$ більше за $-x3_l$. Для цього також будемо підтримувати опуклу оболонку прямих.

Складність такого розв'язку $O(n \log^2 n)$.