**II (районний) етап Всеукраїнської учнівської олімпіади з інформатики  
Київ, 2019/20 н. р.**

Максимальна оцінка за кожну з чотирьох задач — 100 балів.

Для всіх задач обмеження на час — 1 секунда / тест; обмеження на пам’ять — 256 МБ.

Матеріали олімпіади буде оприлюднено на сайті [**kievoi.ippo.kubg.edu.ua**](http://www.kievoi.ippo.kubg.edu.ua), а також на [**soi.org.ua**](https://soi.org.ua/).  
Автор задач — Данило Мисак.

**1. Комікси** (назва програми: **comics.cpp** / **comics.pas** / **comics.\***)

Ярослав і Мирослава колекціонують комікси. Ярослав має коробок по коміксів у кожній та коробок по комік­сів у кожній, а Мирослава — коробок по коміксів у кожній та коробок по коміксів у кожній. У кого з дітей коміксів більше?

**Вхідні дані**

У вхідному файлі через пробіл вказано чотири числа: , , та (саме в такому порядку). Усі чотири числа натуральні та не перевищують .

**Вихідні дані**

У вихідний файл виведіть 1, якщо більшу кількість коміксів має Ярослав; 2, якщо більше коміксів у Миро­слави; 0, якщо Ярослав та Мирослава мають однакову кількість коміксів.

**Приклади**

|  |  |
| --- | --- |
| Вхідний файл **comics.in** | Вихідний файл **comics.out** |
| 2 3 5 4 | 2 |

**Коментар до прикладу**

Ярослав має комікси, а Мирослава — комікси, тобто на один комікс більше, ніж Ярослав.

**2. Монети** (назва програми: **coins.cpp** / **coins.pas** / **coins.\***)

Ярослав та Мирослава мають спільну колекцію з монет. Як символ своєї дружби вони хочуть окремо збе­ріга­ти таку пару монет, що в сумі номінальна вартість цих двох монет дає особливе число . Підрахуйте кіль­кість різних способів вибрати потрібну пару.

**Вхідні дані**

У першому рядку вхідного файлу вказано натуральні числа та , не менші за . У другому рядку записано натуральних чисел — номінальні вартості монет із колекції. Усі числа у вхідному файлі (включно з чис­лами та ) не перевищують .

**Вихідні дані**

У вихідний файл виведіть єдине число — кількість способів вибрати дві монети з сумарною номінальною вартістю . Відо­мо, що шукана кількість не перевищує .

**Приклади**

|  |  |
| --- | --- |
| Вхідний файл **coins.in** | Вихідний файл **coins.out** |
| 4 5  2 2 3 2 1 | 4 |
| 10 3  6 2 10 | 0 |

**Коментарі до прикладів**

У першому прикладі діти можуть вибрати пару в один із чотирьох способів: взяти першу і другу монети; або першу й четверту; або другу й четверту; або третю та п’яту.

У другому прикладі жодні дві монети, на жаль, не дають у сумі вартість .

**3. Марки** (назва програми: **stamps.cpp** / **stamps.pas** / **stamps.\***)

Нещодавно на уроці математики Ярослав і Мирослава вивчили, що арифметичною прогресією називають пос­лідовність чисел, у якій різниця між кожними двома сусідніми членами однакова. А невдовзі після того діти дізнали­ся, що на честь ювілею математичного товариства столиці було випущено дві серії марок. Кож­на серія скла­дається з марок різної номінальної вартості, і ці номіналів утворюють арифметичну прогре­сію. Для своєї колекції марок Ярослав придбав одну з цих серій, а Мирослава — іншу. Однак, роздивляю­чись придбання од­не одного, діти ненароком перемішали всі марки.

Знаючи номінали марок — попарно різних чисел, — допоможіть дітям розділити марки на дві серії. Ві­домо, що це можна зробити рівно в один спосіб.

**Вхідні дані**

У першому рядку вхідного файлу вказано натуральне число — кількість марок у серії, . У другому рядку записано різних натуральних чисел, менших за , — перемішані номінали марок.

**Вихідні дані**

У перший рядок вихідного файлу виведіть в порядку зростання всі номінали марок Ярославової серії, а в другий рядок — усі номінали марок Мирославиної серії (так само в порядку зростання). Діти пам’ятають, що найдешевша марка Ярослава має менший номінал, ніж найдешевша марка Мирослави.

**Приклад**

|  |  |
| --- | --- |
| Вхідний файл **stamps.in** | Вихідний файл **stamps.out** |
| 4 7 9 23 3 16 15 11 2 | 2 9 16 23 3 7 11 15 |

**Коментар до прикладу**

Виведені у вихідний файл послідовності утворюють шукані серії марок, адже є арифметичними прогресія­ми: та . Серії виведено в правильному порядку, бо .

**4. Фантики** (назва програми: **wrappers.cpp** / **wrappers.pas** / **wrappers.\***)

За останній час Ярослав і Мирослава назбирали разом фантиків. Як відомо, кожен колекціонер прагне, щоб його колекція була максимально розмаїтою. Тому діти хочуть ­розподілити між собою фантики так, щоб жодні два чимось схожих між собою фантики не потрапили до одного власника. Для цього Ярослав та Ми­рослава занумерували фантики числами від до та виписали, які саме пари фантиків виглядають подіб­но. Усього в них вийшло пар, причому номери деяких фантиків могли бути виписані в кількох різних па­рах.

Допоможіть дітям розподілити фантики бажаним чином або визначте, що це неможливо.

**Вхідні дані**

У першому рядку вхідного файлу вказано два натуральних числа та — кількість фантиків та кількість їх подібних пар; , . У кожному з наступних рядків задано по два числа та — номери схожих між собою фантиків, , . Жодна пара номерів , у вхідному файлі не повторюється. Крім того, вхідні дані гарантують, що є не більше ніж один спосіб розподі­лити фан­тики між дітьми, щоб жодні два схожих між собою фантики не опинилися в одного власника.

**Вихідні дані**

У першому рядку вихідного файлу виведіть у порядку зростання номери фантиків, які мають опинитися в того ж власника, що й фантик під номером (включно з самим числом ). У другому рядку виведіть у по­рядку зростання номери фантиків, які повинні опинитися в іншого власника.

Якщо жоден роз­по­діл фантиків не задовольняє умову задачі, в обох рядках виведіть по нулю.

**Приклади**

|  |  |
| --- | --- |
| Вхідний файл **wrappers.in** | Вихідний файл **wrappers.out** |
| 5 6 2 5 2 3 1 5 4 5 3 4 1 3 | 1 2 4 3 5 |
| 3 3 1 2 2 3 1 3 | 0 0 |

**Ідеї розв’язання**

**1. Комікси**

Слід порівняти два числа та . Якщо більшим є перше число, слід вивести 1; якщо друге число — 2; якщо числа рівні, треба вивести 0. Слід також врахувати, що числа можуть вийти за межі двобайтових змінних і використати відповідний тип даних.

**2. Монети**

Безпосередній перебір принесе лише частковий бал, бо для великих програма не встигне за від­ведений час (секунду) перебрати всі можливі пари чисел. Замість цього можна скористатися іншим під­хо­дом. Для кожного числа від до слід підрахувати, скільки монет мають дану номінальну вартість: завести ма­сив , у комірці якого буде зберігатися кількість монет вартості (), і після зчиту­вання кожного наступного числа збільшувати відповідну комірку масиву на . Далі через ми позна­чи­мо найбільше ціле число, що не перевищує . Для всіх значень у межах від до включ­но добуток дорівнюватиме кількості пар монет, одна з яких має вартість , а інша — вартість ; при цьому . Сума всіх таких добутків буде кількістю варіантів вибрати пару мо­нет із сумар­ною вартістю за умови, що вартості цих двох монет різні. Залишається врахувати пари монет однакової вартості: якщо непарне, таких пар немає, а інакше їх рівно .

**3. Марки**

Один з можливих способів розв’язати задачу такий. [Відсортуємо](https://uk.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D1%81%D0%BE%D1%80%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F#.D0.92.D1.96.D0.B4.D0.BE.D0.BC.D1.96_.D0.B0.D0.BB.D0.B3.D0.BE.D1.80.D0.B8.D1.82.D0.BC.D0.B8_.D1.81.D0.BE.D1.80.D1.82.D1.83.D0.B2.D0.B0.D0.BD.D0.BD.D1.8F) всі чисел у порядку від наймен­шого до найбільшого. Серед трьох найменших елементів відсортованої послідовності принаймні два нале­жатимуть до однієї й тієї ж прогресії, причому будуть двома найменшими її членами (а якщо всі три най­менші числа належать одній і тій самій прогресії, то найменшими двома її членами є, очевидно, два перших числа). Тепер, послідовно розглядаючи гіпотези про те, що двома найменшими членами однієї з прогресій є перший і другий; перший і третій; другий і третій елементи відсортованої послідовності, встановимо, кот­ра з цих гіпотез є правильною. Для перевірки можемо скористатися таким підходом: перебиратимемо в по­рядку збільшення всі чисел; якщо чергове число, яке ми розглядаємо, є таким, що підходить до першої прогресії (а, беручи припущення гіпотези, ми вже знаємо і перший член цієї прогресії, і її різницю, і кількість елементів), то долучаємо це число до першої прогресії, інакше — до другої. Якщо обидві побудовані послі­довності дійсно є арифметичними прогресіями (з елементами в кожній), то маємо відповідь; інакше пере­ходимо до наступної гіпотези. Описаний процес перевірки можна втілити за один лінійний прохід послідов­ності.

Оскільки алгоритм складається з сортування і кількох лінійних проходів масиву на елементів, йо­го складність можна оцінити як . Це дозволяє заробити повний бал.

Утім, алгоритм можна оптимізувати і до лінійного. Для пошуку трьох найменших елементів замість сортування використаємо, наприклад, три послідовних лінійних проходи. А для перевірки гіпотези переби­ратимемо числа не в порядку збільшення, а в довільному. Це не завадить відібрати з набору ті й лише ті числа, що належать до першої прогресії. Щоб установити, чи решта чисел утворюють другу прогресію, знай­демо шляхом двох лінійних проходжень два найменших числа, що не потрапили до першої прогресії: вони якраз і мають становити два перших члени другої прогресії. Залишається ще раз пройтися по масиву і пере­вірити, чи решта чисел у ньому «узгоджуються» зі знайденими першими членами потенційної прогресії. При цьому, звичайно, не слід забувати, що в обох прогресіях повинно бути рівно по членів. Нарешті, щоб уникнути сортування при виведенні відповіді, можна конструювати прогресії безпосередньо з їхніх арифме­тичних властивостей (а не виводити у вихідний файл упорядковані елементи масиву).

**4. Фантики**

Задачу можна переформулювати в термінах теорії графів. Нехай фантики — це вершини графа, а ребро між двома вершинами проведене тоді й лише тоді, коли два відповідних фантики схожі. Нам необ­хідно розділити всі вершини графа на дві частини так, щоб кожне ребро графа сполучало вершини, що на­лежать до різних частин. Граф, для якого це вдається зробити, називається дводольним, а відповідні його частини — долями.

Задачу можна розв’язати з допомогою пошуку [в глибину](http://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D1%88%D1%83%D0%BA_%D0%B2_%D0%B3%D0%BB%D0%B8%D0%B1%D0%B8%D0%BD%D1%83) або [в ширину](http://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D1%88%D1%83%D0%BA_%D1%83_%D1%88%D0%B8%D1%80%D0%B8%D0%BD%D1%83). Присвоївши вершині влас­ника №  і запустивши будь-який із типів пошуку з цієї вершини, будемо кожній новій пройденій вер­шині графа присвоювати власника № , якщо ми прийшли у неї з вершини, якій було присвоєно власника № , і навпаки. При цьому якщо дана вершина сполучена ребром хоча б з однією іншою вершиною, якій раніше присвоїли того самого власника, що й даній вершині, то розбити граф на дві частини неможливо.

Після завершення пошуку в глибину або в ширину, якщо вдалося присвоїти власників всім верши­нам, маємо потрібний розподіл вершин у компоненті зв’язності графа, яка містить вершину , а інакше виводимо нулі. Якщо граф складається з єдиної компоненти зв’язності, виводимо знайдений розподіл. Як­що ж у графі є дві або більше компонент зв’язності, тобто якщо після завершення пошуку, запущеного з вер­шини , одна чи кілька вершин залишились невідвіданими, то слід також вивести нулі. Справді: якщо пот­рібний розподіл вершин усіх компонент зв’язності існує, то він не єдиний, адже розподіли різних ком­по­нент можна як завгодно комбінувати. А згідно з умовою задачі якби розподіл існував, то він мав би бути єдиним. Отже, у випадку двох і більше компонент зв’язності умова задачі гарантує відсутність потрібного розподілу.

Насамкінець зауважимо, що обмеження має суто технічний характер і пов’язане з не­обхідністю обмежити розмір вхідного файлу.