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

Бондаренко Віталій Вікторович,
асистент факультету кібернетики Київського Національного університету ім. Тараса Шевченка.
Мисак Данило Петрович,
керівник гуртка СШ №52 м. Києва.
Ягіяєв Шаміль Ігорович,
менеджер проектів компанії «Арісент Україна».

ЗАВДАННЯ ПЕРШОГО ТУРУ

1. Мутація (Роман Рубаненко)

Умова. Вчені планети Олімпія кожен рік досліджують різноманітні мутації геномів примітивних організмів. Геном таких організмів може бути представлений як послідовність N невід'ємних цілих чисел, які занумеровані зліва направо від одиниці до N та не перевищують число N. Геноми підлягають постійним мутаціям. На кожному етапі мутації геном змінюється таким чином:

- на перше місце записується кількість одиниць у вхідному геномі;
- на друге місце записується кількість двійок у вхідному геномі;
- . . . ;
- на місце номер N записується кількість чисел, які дорівнюють N, у вхідному геномі.

Наприклад, геном [1, 2, 3] з трьох чисел після мутації перетвориться на [1, 1, 1] — по одній одиниці, двійці та трійці. Інші приклади:

- [1,2,2,3,3,3] → [1,2,3,0,0,0];
- [7,7,7,4,7,4,4] → [0,0,0,3,0,0,4].

Далі геном продовжує змінюватися за тим самим принципом.

Завдання. Напишіть програму mutation, яка за інформацією про початковий вигляд геному визначить його стан після K мутацій.

Вхідні дані. Перший рядок вхідного файла mutation.dat містить два цілих числа N і K ($1 \le N \le 10^5$, $1 \le K \le 10^9$), що задають початковий розмір геному та кількість мутацій, які геном переживе. Другий рядок містить N невід'ємних цілих чисел, що не перевищують N, — початковий вигляд геному.

Вихідні дані. У вихідний файл mutation.sol слід записати геном після K мутацій у тому ж форматі, що й у вхідному файлі: N чисел, розділені пропуском.

Оцінювання. Набір тестів складається з 3 блоків, для яких додатково виконуються такі умови:

- 40% балів: $1 \le N \le 100$, $1 \le K \le 100$;
- 30% балів: $1 \le N \le 1000$;
- 30% балів: $1 \le N \le 10^5$.

Приклад вхідних та вихідних даних

mutation.dat	mutation.sol
4 2 1 3 1 4	2 1 0 0

Пояснення. Спочатку [1,3,1,4] мутує в геном [2,0,1,1], який у свою чергу мутує в [2,1,0,0].

Розв'язання

Лема. Якщо геном містить принаймні одне додатне число, то через $O(\log\log N)$ мутацій він набуде вигляду [1,0,0, ,0] та перестане змінюватися.

Доведення. Розглянемо дві ітерації послідовних мутацій: $A \to B \to C$. Нехай $z(X)$ позначає кількість ненульових елементів у геномі X. Тоді $z(C)$ — кількість різних ненульових елементів B, тобто кількість різних частот ненульових чисел геному A. Щоб отримати $z(C)=P$ різних частот, потрібно мати хоча б $1+2+\ldots+P = P(P+1)/2$ ненульових елементів. Таким чином, $z(A) \ge P(P+1)/2 > z^2(C)/2$, звідки $z(C) < \sqrt{2z(A)}$. Оскільки $\sqrt{2z(A)} \le z(A)$ при $A \ge 2$, кількість ненульових елементів у геномі кожні дві ітерації зменшується, і рано чи пізно в ньому залишиться тільки один ненульовий елемент. А тоді вже за два кроки геном набуде вигляду [1,0,0,…,0]. Залишається оцінити кількість ітерацій, після яких це станеться.

Якщо після двох ітерацій кількість ненульових елементів стає меншою за $\sqrt{2z(A)}$, то після $2K$ ітерацій вона стане меншою за $\sqrt{2\sqrt{2\ldots\sqrt{2\sqrt{2z(A)}}}}$, де кількість радикалів дорівнює K.

Оскільки $\sqrt{2\sqrt{2\ldots\sqrt{2\sqrt{2}}}} < 2$ (це нескладно довести за індукцією),

$$\sqrt{2\sqrt{2\ldots\sqrt{2\sqrt{2z(A)}}}} < 2(z(A))^{1/2^K}.$$

Маємо $z(A) \le N$, тому після $K = O(\log\log N)$ ітерацій число $1/2^K$ стане меншим за $1/\log N$, а логарифм числа $(z(A))^{1/2^K}$ — меншим за одиницю. Залишається додати сталу кількість ітерацій, щоб дійти до фінального вигляду. Моделювання однієї ітерації може бути просто реалізовано за лінійну складність відносно N з використанням допоміжного масиву. Умовна реалізація на псевдокоді може виглядати так:

```
b[] = [0, 0, …, 0]
 for i = 1 to N do
     b[a[i]] = b[a[i]] + 1
```

Після кожної ітерації треба перевірити, чи перетворився геном на [1,0,0,…,0]. Якщо це так, то подальше виконання програми не має сенсу: геном залишатиметься незмінним. В результаті програма виконає $O(N \cdot \min(K, \log\log N))$ операцій.

2. Лінія (Данило Мисак)

Умова. Орися розставила на аркуші в клітинку N^2 літер у формі квадрата $N \times N$ і хоче викреслити однією лінією деякі літери у такий спосіб: вона починає викреслювати літери, починаючи з лівої верхньої букви, і веде лінію то праворуч, то вниз; останньою лі-

терою вона викреслює праву нижню. Таким чином, дівчина викреслить рівно $2N-1$ літеру. При цьому Орися хоче, щоб уздовж лінії, яку вона проведе, було записано певне чарівне слово.

Завдання. Напишіть програму **line**, яка для заданих розташування літер і чарівного слова визначить, у скільки різних способів Орися може його викреслити, та виведе відповідь за модулем простого числа 1 000 003.

Вхідні дані. У першому рядку вхідного файла **line.dat** записано натуральне число N ($2 \le N \le 1000$) — довжину сторони квадрата з літер. У наступних N рядках записано по N малих літер латинської абетки (не обов'язково різних), що задають розташування літер. Пробілів між літерами немає. Далі записано чарівне слово, що складається з $2N-1$ літери (усі — малі літери латинської абетки, не обов'язково різні).

Вихідні дані. Вихідний файл **line.sol** повинен містити єдине число — остачу від ділення кількості способів, у які Орися може викреслити чарівне слово, на число 1 000 003.

Оцінювання. Набір тестів складається з 3 блоків, для яких додатково виконуються такі умови:

- 25 % балів: $2 \le N \le 10$;
- 30 % балів: $10 < N \le 100$;
- 45 % балів: $100 < N \le 1000$.

Приклад вхідних та вихідних даних

line.dat	line.sol
3	5
loc	
ogo	
gos	
logos	

Пояснення. Є рівно 5 способів викреслити слово logos (рис. 1).

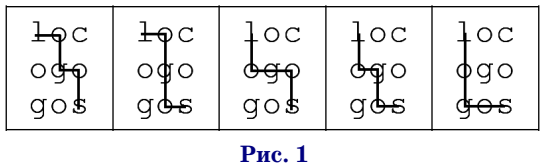

Рис. 1

Розв'язання

Будемо вважати, що літери записано в комірках квадратної таблиці, а замість ліній розглядатимемо шляхи з лівої верхньої клітинки, що йдуть праворуч та вниз. ***Гарним*** шляхом з лівої верхньої клітинки в довільну клітинку з літерою називатимемо такий шлях, уздовж якого послідовність літер збігається з початком чарівного слова — тобто який теоретично можна продовжити до правильного шляху в праву нижню клітинку (але лише теоретично: якщо такого шляху насправді немає, бо внизу і справа не знайдеться потрібних літер, шлях з лівої верхньої клітинки в дану все одно вважаємо гарним).

Зауважимо, що якщо клітинка з літерою розташована на перетині i-го рядка та j-го стовпця, то шлях у неї з лівої верхньої клітинки матиме довжину $i+j-1$ незалежно від того, як саме він проходить. Отже, в послідовності літер ця клітинка відповідатиме $(i+j-1)$-й літері чарівного слова. Якщо літера, записана в даній комірці, відрізняється від літери з цим номером чарівного слова, кількість гарних шляхів у дану клітинку нульова, адже остання літера на шляху завжди буде неправильною. Якщо натомість літера в даній комірці збігається з $(i+j-1)$-ю літерою чарівного слова, то кількість гарних шляхів у неї дорівнює сумі кількості гарних шляхів у клітинку-сусіда ліворуч від даної і кількості гарних шляхів у клітинку-сусіда вище від даної. Справді: будь-який гарний шлях входить у дану клітинку або зліва, або зверху, причому він мав бути гарним і на попередньому кроці. І навпаки: якщо є якийсь гарний шлях, що закінчується в сусідній зліва або зверху клітинці, то його можна продовжити на одну літеру вправо або вниз відповідно.

Окремо слід розглянути випадок клітинок на верхній та лівій межах поля. Для клітинок у верхньому рядку кількість гарних шляхів дорівнює просто кількості гарних шляхів у клітинку зліва від даної. Аналогічно для комірок лівого стовпця кількість гарних шляхів дорівнює кількості таких шляхів у клітинку зверху від даної. Звичайно, за умови, що в даній клітинці стоїть літера, яка збігається з відповідною буквою чарівного слова.

Фактично нас цікавить кількість гарних шляхів у праву нижню клітинку. Щоб знайти її, послідовно порахуємо цю кількість для всіх клітинок поля, починаючи з лівої верхньої та йдучи зліва направо зверху вниз. Для лівої верхньої клітинки кількість гарних шляхів дорівнює або 1, якщо літера в ній збігається з першою літерою чарівного слова, або 0, якщо не збігається. Далі кількості шляхів можна рахувати за описаною вище схемою, адже на момент, коли ми розглядаємо довільну комірку, клітинки зліва та зверху від неї, якщо такі є, вже було розглянуто раніше.

Таким чином, у задачі застосовано класичний підхід динамічного програмування. Щоб уникнути розгляду додаткових випадків — коли клітинка розташована у верхньому рядку або лівому стовпці — можна ввести фіктивний додатковий рядок зверху та стовпець зліва від поля, для всіх комірок яких кількості гарних шляхів покласти рівними нулям.

Не слід, звичайно, забувати, що всі операції виконуються за модулем числа, вказаного в умові.

3. Східний кросворд (Данило Мисак)

Умова. Марися любить розв'язувати східні кросворди. Так називається головоломка, в якій потрібно зафарбувати деякі клітинки прямокутника $N \times M$ таким чином, щоб на кожній з N вертикалей і на кожній з M горизонталей кількість зафарбованих клітинок дорівнювала деякому наперед визначеному записаному на полях для даної вертикалі чи горизонталі числу. На жаль, інколи укладачі кросвордів помиляються, і кросворд розв'язку не має. Дівчина не хотіла б марнувати свій час, розв'язуючи такі кросворди.

Завдання. Напишіть програму **puzzle**, яка для заданих величин N та M, а також $N+M$ чисел, записаних на полях кросворда, визначить, чи є даний кросворд розв'язним.

Вхідні дані. У першому рядку вхідного файла **puzzle.dat** записано число T, $1 \le T \le 5$, — кількість кросвордів для перевірки. Кожен кросворд подається трьома рядками: в першому рядку записано натуральні числа N та M, що не перевищують 10^5, — ширину та висоту кросворда; у другому рядку подано N цілих невід'ємних чисел — кількість зафарбованих клітинок на першій, другій, N-й вертикалях відповідно; у третьому рядку подано M цілих невід'ємних чисел — кількість зафарбованих клітинок на першій, другій, M-й горизонталях відповідно. Жодне з чисел у другому і третьому рядках не перевищує загальної кількості клітинок на відповідній вертикалі чи горизонталі.

Вихідні дані. Вихідний файл **puzzle.sol** повинен містити відповідь для кожного з кросвордів в окремо-

му рядку: 1, якщо кросворд розв'язний, або 0, якщо ні. Відповіді потрібно подати в тому ж порядку, в якому у вхідному файлі подано самі кросворди.

Оцінювання. Набір тестів складається з 3 блоків, для яких додатково виконуються такі умови:

- 15% балів: жоден з розмірів (ні ширина, ні висота) жодного з кросвордів не перевищує 5;
- 35% балів: жоден з розмірів жодного з кросвордів не перевищує 1000, причому хоча б один з розмірів хоча б одного з кросвордів більший за 5;
- 50% балів: хоча б один з розмірів хоча б одного з кросвордів більший за 1000.

Приклад вхідних та вихідних даних.

puzzle.dat	puzzle.sol
2	1
3 2	0
1 2 1	
2 2	
4 5	
5 5 5 5	
0 0 0 0 0	

Пояснення. У першому випадку кросворд має, наприклад, таке розв'язання (рис. 2).

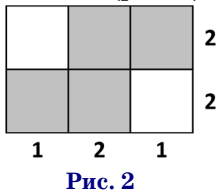

Рис. 2

У другому випадку кросворд нерозв'язний: з одного боку, рядок з п'ятірок свідчить, що всі вертикалі повністю зафарбовано; з іншого боку, рядок з нулів означає, що жодна з горизонталей не містить жодної зафарбованої клітинки. Такого, очевидно, бути не може.

Розв'язання

Замість «кросворда» казатимемо «матриця», замість «зафарбованих клітин» — «одиниці», а замість «незафарбованих клітин» — нулі. Позначимо кількість одиниць в i-му стовпці через c_i, $1 \le i \le N$, а кількість одиниць в i-му рядку — через r_i, $1 \le i \le M$. Уважатимемо, що

$$\sum_{i=1}^{N} c_i = \sum_{i=1}^{M} r_i$$

(обидві суми дорівнюють загальній кількості одиниць у матриці). Інакше можна відразу стверджувати, що потрібної матриці не існує.

Розглянемо набір з перших K стовпців. Для кожного з них ми знаємо, скільки в даному стовпці має бути одиниць, а отже, і загальну кількість одиниць у цих стовпцях:

$$\sum_{i=1}^{K} c_i.$$

З іншого боку, в i-му рядку в цих стовпцях буде не більше ніж $\min(r_i, K)$ одиниць, тому загальна кількість одиниць у цих стовпцях не може перевищувати

$$\sum_{i=1}^{M} \min(r_i, K).$$

Звідси маємо необхідну умову для існування матриці чисел, що задовольняє умову задачі:

$$\sum_{i=1}^{K} c_i \leqslant \sum_{i=1}^{M} \min(r_i, K),\ 1 \leqslant K \leqslant N.$$

Дану умову позначимо через **(*)**.

Оскільки завжди без втрати загальності можна переставити будь-які стовпці місцями, можемо розставити стовпці в порядку спадання (незростання) кількості одиниць у них. Таким чином ми досягнемо того, що для довільного K сума

$$\sum_{i=1}^{K} c_i$$

буде максимально можливою, а виведена вище умова необхідності — максимально суворою. При цьому для зручності вважатимемо, що рядки також впорядковані, але за зростанням (неспаданням) кількості одиниць, тобто r_M — максимальний рядок, r_{M-1} — другий за величиною і т. д.

Переформулюємо виведену умову без алгебри: для довільного K, $1 \le K \le N$, сума K найбільших кількостей по стовпцях не перевищує суму всіх кількостей по рядках, якщо всі ці кількості попередньо «обрізати» до K, тобто всі числа, більші за K, зменшити до K. Покажемо, що ця умова є й достатньою для існування відповідної матриці. Для цього заповнимо перший стовпець (у якому має бути найбільша кількість одиниць) так: поставимо одиниці в тих c_1 рядках, де загальна кількість одиниць має бути найбільшою. По-перше, це зробити можливо, адже якщо в **(*)** підставити $K=1$, матимемо, що c_1 не перевищує кількості ненульових рядків. По-друге, коли ми це зробимо, то перейдемо до меншої підзадачі: сконструювати матрицю вже з $N-1$ стовпцями з сумами $c'_1=c_2$, $c'_2=c_3$, ..., $c'_{N-1}=c_N$ і з M рядками з сумами $r'_1=r_1$, $r'_2=r_2$, ..., $r'_{M-c_1}=r_{M-c_1}$, $r'_{M-c_1+1}=r_{M-c_1+1}-1$, $r'_{M-c_1+2}=r_{M-c_1+2}-1$, ..., $r'_M=r_M-1$ (при цьому впорядкованість $r'_1, r'_2, ..., r'_M$ за неспаданням може порушитись, але це не впливає на правильність розв'язання). Якщо ми покажемо, що відповідні числа знову задовольняють **(*)**, такі ж операції можна буде продовжити, поки ми не побудуємо матрицю повністю. Отже, нам треба довести таку нерівність:

$$\sum_{i=1}^{K} c'_i \leqslant \sum_{i=1}^{M} \min(r'_i, K),\ 1 \leqslant K \leqslant N-1.$$

Знайдемо таке найменше $C \ge 1$, що серед чисел $r_1, r_2, ..., r_M$ таких, які більші або рівні C, буде не менше за c_1, а чисел, які більші або рівні $C+1$, — не більше за c_1. Таке C обов'язково знайдеться і буде не більшим за N, адже за умовою **(*)** при $K=1$ чисел, не менших за 1, є не менше за c_1, а чисел, більших за N, бути не може в принципі.

Тоді для $K<C$ маємо:

$$\sum_{i=1}^{K} c'_i = \sum_{i=2}^{K+1} c_i \leqslant K \times c_1 = \sum_{i=M-c_1+1}^{M} \min(r_i - 1, K) =$$
$$= \sum_{i=M-c_1+1}^{M} \min(r'_i, K) \leqslant \sum_{i=1}^{M} \min(r'_i, K).$$

Водночас для $K \ge C$:

$$\sum_{i=1}^{K} c'_i = \sum_{i=2}^{K+1} c_i = \sum_{i=1}^{K+1} c_i - c_1 \leqslant \sum_{i=1}^{M} \min(r_i, K+1) - c_1 =$$
$$= \sum_{i=1}^{M-c_1} \min(r_i, K+1) + \sum_{i=M-c_1+1}^{M} \min(r_i, K+1) - c_1 =$$
$$= \sum_{i=1}^{M-c_1} \min(r_i, K) + \left(\sum_{i=M-c_1+1}^{M} \min(r_i, K+1) - c_1 \right) =$$

$$= \sum_{i=1}^{M-c_1} \min(r_i, K) + \sum_{i=M-c_1+1}^{M} \min(r_i - 1, K) =$$

$$= \sum_{i=1}^{M-c_1} \min(r'_i, K) + \sum_{i=M-c_1+1}^{M} \min(r'_i, K) = \sum_{i=1}^{M} \min(r'_i, K).$$

Отже, залишається відсортувати підрахунком, а точніше — підрахувати індексні масиви для обох послідовностей: кількостей одиниць по стовпцях і по рядках. Якщо в масиві кількостей по стовпцях достатньо йти і щораз додавати до суми S_c нові елементи в порядку їх спадання, то для масиву кількостей по рядках потрібно зберігати кількість на поточний момент часу чисел, не менших за дане (оновлювати його можна, щораз віднімаючи кількість чисел, які дорівнюють даному), та на кожному кроці додавати цю кількість до суми S_r. Далі порівнювати S_c та S_r.

4. Дерева (Роман Фурко)

Умова. У столиці країни Олімпія було визначено територію для будівництва нового Олімпійського парку. Границі території мають форму опуклого багатокутника. Ідея дизайнера парку полягає в тому, щоб засадити деревами певну кількість зелених зон, які мають на карті форму круга: кожну зелену зону можна задати координатами центра та її радіусом.

Отже, дизайнер доручив посадити дерева у точках, що мають цілі координати та розташовані в межах парку (можливо, на його границі) та хоча б однієї з зелених зон (можливо, на межі). Якщо певні зелені зони перетинаються, тобто одна й та сама точка на карті належить двом чи більшій кількості зелених зон, у цьому місці тим не менш можна посадити лише одне дерево.

Завдання. Напишіть програму trees, що за даними про координати вершин многокутника, який задає територію парку, а також за даними про координати центрів та радіуси зелених зон визначить кількість дерев, які має бути посаджено.

Вхідні дані. У першому рядку вхідного файла **trees.dat** вказано натуральне число N ($3 \le N \le 10^5$) — кількість вершин багатокутника, що задає територію парку. Наступні N рядків містять по два цілих числа — відповідно абсциси та ординати вершин у порядку обходу за чи проти годинникової стрілки. Наступний рядок містить ціле число M ($1 \le M \le 50\,000$) — кількість зелених зон. Далі йдуть M рядків, кожен з яких містить по три цілих числа: абсцису та ординату центра зеленої зони, а також її радіус. Усі координати, задані у вхідному файлі, є цілими числами в межах від -10^9 до 10^9 включно. Радіуси зелених зон цілі додатні, сума всіх радіусів не перевищує 10^5. Зверніть увагу, що деякі зелені зони можуть цілком лежати всередині інших зон; крім того, окремі зелені зони можуть лежати поза многокутником. Різні зони можуть мати спільні центри або й узагалі збігатися.

Вихідні дані. Єдиний рядок вихідного файла **trees.sol** повинен містити єдине ціле число — кількість дерев, що будуть посаджені за дорученням дизайнера.

Оцінювання. Набір тестів складається з блоків, для яких додатково виконуються такі умови:

- 60% балів: $N \le 100$. Зокрема, серед даного набору є такі групи тестів, що перетинаються:
  - 20% балів (від загальної кількості): парк має форму прямокутника, сторони якого паралельні осям координат;
  - 30% балів (від загальної кількості): $N \times M \times R^2 \le 10^7$, де через R позначено найбільший з радіусів;
  - 25% балів (від загальної кількості): існує квадрат зі сторонами довжини 2015, паралельними до осей координат, у якому або на межах якого повністю лежить територія парку;
- 40 % балів: на вхідні дані не накладено додаткових обмежень.

Приклад вхідних та вихідних даних

trees.dat	trees.sol
3	73
0 0	
15 0	
15 15	
3	
3 3 4	
5 5 7	
15 15 1	

Розв'язання

Спочатку розглянемо задачу ефективного знаходження перетину опуклого многокутника та вертикальної прямої. Очевидно, що кількість точок перетину не перевищує 2, якщо тільки пряма не містить вертикальну сторону многокутника. Розіб'ємо многокутник на верхній і нижній ланцюги: знаходимо найлівішу (а з усіх таких — найнижчу) та найправішу (з усіх таких — найвищу) вершину многокутника. Проводимо пряму між цими точками. Всі вершини многокутника, що лежать вище від проведеної прямої, належать верхньому ланцюгу, а решта — нижньому. Після такого поділу в кожному ланцюзі вершини впорядковані за абсцисою. Це дозволяє знаходити точки перетину вертикальної прямої з опуклим многокутником за $O(\log N)$ операцій за допомогою двійкового пошуку.

Тепер, оскільки круг довільного натурального радіуса R перетинається рівно з $2R+1$ цілочисельною вертикальною прямою, загальна кількість таких прямих, що перетинаються принаймні з одним кругом, не перевищує $2S+M=O(S)$, де S — сумарний радіус усіх M кругів. Розглянемо відповідні $2S+M$ відрізків перетину та розташуємо їх у порядку зростання абсциси, а при однаковій абсцисі — за зростанням ординати нижнього кінця. Для кожного відрізка за $O(\log N)$, скориставшись описаним у попередньому абзаці методом, знайдемо кількість цілих точок на відрізку, що лежать усередині многокутника. При цьому, щоб не рахувати по кілька разів точки, які лежать водночас на кількох відрізках, для кожної нової абсциси пам'ятатимемо ординату, на якій закінчився попередній розглянутий відрізок і починаючи з якої, відповідно, потрібно продовжувати рахувати точки. Сумарна знайдена кількість точок і буде відповіддю.

Складність алгоритму — $O(S(\log S+\log N)+N)$ або, якщо скористатися впорядкованістю відрізків і замість двійкового пошуку йти по ланцюгах лінійно, $O(S \log S+N)$.

Альтернативні розв'язання

Розв'язок за $O(S(\log S+N))$ зі знаходженням кількості точок відрізка, що лежать усередині многокутника, за лінійний від кількості вершин многокутника час набирає 60% балів. Якщо перебирати всі точки всередині кругів та наївно перевіряти, чи лежать вони в багатокутнику, можна набрати 30% балів. А якщо перебирати всі точки всередині многокутника та перевіряти, чи лежать вони в межах принаймні одного круга, вдасться заробити лише 20% балів.

(Далі буде)