ОЛІМПІАДА З ІНФОРМАТИКИ У МІСТІ КИЄВІ У 2012–2013 НАВЧАЛЬНОМУ РОЦІ

Знов'як Юрій Володимирович,
інженер з програмного забезпечення Google New York Center,
Мисак Данило Петрович,
керівник гуртка СШ №52 м. Києва,
Рибак Олександр Владиславович,
аспірант Інституту математики НАН України,
Рудик Олександр Борисович,
кандидат фізико-математичних наук, доцент кафедри методики природничо-математичної освіти і технологій Інституту післядипломної педагогічної освіти Київського університету імені Бориса Грінченка.

Стаття містить умови орієнтовних завдань II (районного) і завдань III(міського) етапу олімпіади з основ інформатики й обчислювальної техніки у місті Києві у 2012–2013 навчальному році та авторські розв'язання цих завдань. Публікацію адресовано учням класів з поглибленим вивченням математики, учасникам олімпіад з інформатики, студентам математичних спеціальностей, учителям і викладачам вищих навчальних закладів.

Цього навчального року орієнтовні завдання II етапу (упорядник — Данило Мисак), на відміну від попередніх років, не містили завдань відбірково-тренувальних зборів і були краще узгоджені з рівнем підготовки переважної кількості учасників олімпіади. Розглянемо їх детально.

ЗАВДАННЯ II ЕТАПУ

1. Петрик П'яточкін рахує склади
Назва програми: syllable.*

Умова. Допитливий київський школяр Петрик П'яточкін якось зацікавився мовознавством — наукою про мову. Для важливого дослідження Петрику потрібно з'ясувати, скільки назви різних натуральних чисел мають складів, тобто скільки містять голосних літер. Допоможіть хлопцю, написавши програму, яка дає відповідь на це питання.

Вхідні дані. У вхідному файлі вказано натуральне число n, кількість складів у якому необхідно порахувати. Число n не перевищує 100.

Вихідні дані. У вихідний файл виведіть кількість складів, які у своїй назві українською мовою має число n.

Приклади

№	Вхідний файл syllable.in	Вихідний файл syllable.out
1	1	2
2	25	3

Пояснення до прикладів

Слово «один» має два склади (містить дві голосні літери). Назва числа 25 — «двадцять п'ять» — має три склади (містить три голосні літери).

Ідея розв'язання

Можна вручну підрахувати кількість складів у кожному з чисел від 1 до 100 і створити програму з розгалуженнями типу IF THEN ELSE або CASE / SWITCH, які залежно від числа на вході повертають потрібний результат. Але цей підхід досить трудомісткий і скоріше за все учасник, який пішов таким шляхом, зробить принаймні декілька помилок у підрахунку або під час занесення результатів у програму.

Натомість можна помітити, що назви всіх двоцифрових чисел, починаючи з числа 20, утворюються шляхом сполучення слова, що позначає кількість десятків («двадцять», «сорок», «дев'яносто») і слова, що позначає кількість одиниць («один», «вісім»). При цьому у назвах чисел, кратних 10, компонент одиниць відсутній. Тому слід окремо внести у програму (наприклад, як масив) кількості складів у назвах чисел 1, 2, 3, ..., 18, 19, 20, 30, 40, ..., 90, 100, а решту відповідей давати як суму двох значень, занесених для числа, що позначає кількість десятків, і числа, що позначає кількість одиниць заданого у вхідному файлі значення n.

2. Петрик П'яточкін купує книги
Назва програми: books.*

Умова. Щоб краще підготуватися до різноманітних олімпіад, у яких бере участь Петрик П'яточкін, хлопець замовляє книги у зарубіжному інтернет-магазині. На жаль, Петрику доводиться платити не лише за самі книги, але також і за їх доставку в Україну: незалежно від кількості книг, доставлених хлопцю за один раз, Петрик платить за одну доставку d гривень. Крім того, українська митниця за один раз дозволяє безплатно провозити через кордон щонайбільше m книг, а якщо кількість книг перевищує m, митниця бере за кожну понаднормово завезену книгу t гривень податку. Перед олімпіадою з інформатики Петрик хоче купити в інтернет-магазині n книг. Він може замовити доставку всіх книг разом або як завгодно розподілити книги на довільну кількість окремих доставок. Допоможіть хлопцю визначити, яку найменшу загальну суму грошей він повинен витратити на доставки й на мито, щоб перевезти усі n книг в Україну.

Вхідні дані. У єдиному рядку вхідного файлу записані чотири натуральні числа n, d, m, t — відповідно кількість книг, які треба перевезти; вартість однієї доставки; найбільша кількість книг, яку можна перевезти за одну доставку без сплати мита; розмір мита за кожну понаднормово завезену книгу. Відомо, що $n(d+t)<2\cdot10^9$ і $m<2\cdot10^9$.

Вихідні дані. У вихідний файл виведіть єдине натуральне число — найменшу сумарну кількість грошей, яку має сплатити Петрик за доставку і як мито, щоб отримати замовлені в інтернет-магазині книги.

Приклади

№	Вхідний файл books.in	Вихідний файл books.out
1	5 4 3 6	8
2	3 2 2 1	3

Пояснення до прикладів

У першому прикладі потрібно перевезти 5 книг. Якщо замовити їх однією доставкою, доведеться сплатити 4 грн за доставку і за 5−3=2 понаднормово завезені книги по 6 грн, усього 16 грн. Якщо ж розподілити книги на дві доставки, наприклад 3 книги на першу і 2 книги на другу, треба буде сплатити 2·4=8 грн за дві доставки, зате під час жодної з доставок платити за понаднормово завезені книги не доведеться. Три чи більше доставок будуть коштувати явно більше за 8 грн, тому 8 грн — найменша сума, яку доведеться витратити.

У другому прикладі потрібно перевезти 3 книги. Якщо замовити їх однією доставкою, треба буде сплатити 2+(3−2)·1=3 грн. А за дві чи більше доставок доведеться заплатити вже не менше ніж 2·2=4 грн.

Ідея розв'язання

Якщо Петрик замовить k окремих доставок, то за умови оптимального розподілу книг між доставками хлопцю доведеться сплатити грн: kd грн за k доставок, а також $(n-km)t$ грн за $n-km$. понаднормово завезених книг, якщо $n-km>0$. Запровадьмо таке позначення:

$$f(k) = kd + \max\{0,\ n - km\} \cdot t,\quad k \in \mathbb{N},$$

і розгляньмо різницю $f(k+1)-f(k)$:

$$f(k+1) - f(k) = \begin{cases} d, & n - km < 0, \\ d - (n - km)t, & 0 \leqslant n - km < m, \\ d - mt, & n - km \geqslant m. \end{cases}$$

Запишімо це трохи інакше, позначивши через $n \bmod m$ остачу від ділення n на m:

$$f(k+1) - f(k) = \begin{cases} d, & k > \dfrac{n}{m}, \\ d - (n \bmod m)t, & \dfrac{n}{m} - 1 < k \leqslant \dfrac{n}{m}, \\ d - mt, & k \leqslant \dfrac{n}{m} - 1. \end{cases}$$

Зрозуміло, що якщо $n/m<1$, тобто $n<m$, то замовляти більше ніж одну доставку Петрику сенсу немає. Тому відповіддю в цьому випадку є значення $f(1)$. Далі вважатимемо, що $n/m \geqslant 1$.

Якщо $d-mt \geqslant 0$, то $f(k+1) \geqslant f(k)$ незалежно від величини k. Тому мінімальне значення функція f набуває за найменшого k, тобто при $k=1$, а шукана сума грошей — це f(1).

Якщо $d-mt<0$, але $d-(n \bmod m)t>0$, то $f(k+1)<f(k)$ при $k \leqslant [n/m]-1$ та $f(k+1)>f(k)$ при $k \geqslant [n/m]$. Тому мінімальну величину функція f набуває при $k=[n/m]$, а шукана сума грошей — це $f([n/m])$.

А якщо $d-(n \bmod m)t \leqslant 0$, то $f(k+1) \leqslant f(k)$ при $k \leqslant [n/m]$ та $f(k+1)>f(k)$ при $k \geqslant [n/m]+1$. Тому найменшу величину функція f набуває при $k=[n/m]+1$, а шукана сума грошей — це $f([n/m]+1)$.

На практиці досить написати програму, яка порівнює три числа — значення $f(1)$, $f([n/m]+1)$ і, якщо $[n/m] \neq 0$, $f([n/m])$ — та виводить найменше з них. Усі обчислення слід проводити обережно, щоб не вийти за межі стандартного типу даних. Зокрема, число $f([n/m]+1)$ може перевищувати $2 \cdot 10^9$, але в цьому випадку воно точно не буде найменшим із трьох чисел (бо, замовивши, приміром, одну доставку, Петрик заплатить менше ніж $d+nt \leqslant n(d+t) < 2 \cdot 10^9$ грн).

Алгоритм, який не враховує монотонності функції f при значеннях аргументу від 1 до $[n/m]$ та шукає найменше серед усіх чисел $f(1)$, $f(2)$, ..., $f([n/m])$, $f([n/m]+1)$, набирає лише частковий бал: значення $[n/m]$ попри обмеження, накладені в умові задачі, може досягати мільярда.

3. Петрик П'яточкін пише олімпіаду
Назва програми: olympiad.*

Умова. Прочитавши чимало книжок про алгоритми, Петрик П'яточкін прийшов на районну олімпіаду з інформатики. Перед початком олімпіади з'ясувалося, що багато хто з учасників знає одне одного. Тож організатори вирішили убезпечитися й розсадити всіх учасників по двох кабінетах, де проходить олімпіада, у такий спосіб, щоб жодні два знайомі між собою учасники не сиділи в одному кабінеті. Напишіть програму, яка допоможе організаторам зробити це або скаже, що розсадити в такий спосіб учасників неможливо.

Вхідні дані. У першому рядку вхідного файлу вказано два натуральних числа n та m — кількість учасників районної олімпіади і кількість пар знайомих учасників; $2 \leqslant n < 2000$, $1 \leqslant m < 450\,000$. У кожному з наступних m рядків задано по два числа a_i, b_i — номери знайомих між собою учасників, $1 \leqslant a_i < b_i \leqslant n$, $1 \leqslant i \leqslant m$. Жодна пара номерів a_i, b_i у вхідному файлі не повторюється. Крім того, вхідні дані гарантують, що є не більше ніж один спосіб розсадити учасників по двох кабінетах, щоб жодні два знайомі учасники не сиділи в одному кабінеті.

Вихідні дані. У першому рядку вихідного файлу виведіть у порядку зростання номери учасників, які мають сидіти в кабінеті разом з учасником під номером 1 (Петриком П'яточкіним) включно із самим числом 1 (Петриком).

У другому рядку виведіть у порядку зростання номери учасників, які повинні сидіти в іншому кабінеті.

Якщо жодне розташування учасників не задовольняє умову задачі, в обох рядках виведіть по нулю.

Приклади

Вхідний файл olympiad.in	Вихідний файл olympiad.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

Ідея розв'язання

Задачу можна переформулювати в термінах теорії графів. Нехай учасники олімпіади — це вершини графа, а ребро між двома вершинами проведене тоді й лише тоді, коли два відповідні учасники знайомі між собою. 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 у глибину або в ширину. Надавши вершині 1 (Петрику) кабінет №1 і запустивши будь-який із типів пошуку з цієї вершини, будемо кожній новій пройденій вершині графа надавати кабінет №2, якщо ми прийшли у неї з вершини, якій було присвоєно кабінет №1, і навпаки. При цьому, якщо дана вершина сполучена ребром хоча б з однією іншою вершиною, якій раніше надали той самий кабінет, що й даній вершині, то розбити граф на дві частини неможливо.

Після завершення пошуку у глибину або в ширину, якщо вдалося надати кабінети всім вершинам, маємо потрібний розподіл учасників у компонента зв'язності графа, який містить вершину 1, а інакше виводимо нулі. Якщо граф складається з єдиного компонента зв'язності, виводимо знайдений розподіл. Якщо ж у графі є два або більше компонентів зв'язності, тобто якщо після завершення пошуку, запущеного з вершини 1, одна чи кілька вершин залишились невідвіданими, то слід також вивести нулі. Справді: якщо потрібний розподіл вершин усіх компонентів зв'язності існує, то він не єдиний, адже розподіли різних компонентів можна як завгодно комбінувати. А згідно з умовою задачі, якби розподіл існував, то він мав би бути єдиним. Отже, у випадку двох і більше компонентів зв'язності умова задачі гарантує відсутність потрібного розподілу учасників.

Насамкінець зауважимо, що обмеження $m<450\,000$ має суто технічний характер і пов'язане з необхідністю обмежити розмір вхідного файлу.

Примітка. Решту матеріалів II етапу олімпіади можна знайти на сайтах «Київські учнівські олімпіади з інформатики» http://kievoi.narod.ru і на сайті олімпіади Солом'янського району http://soi.org.ua. У тексті подано варіант третьої задачі, запропонований у Солом'янському й Печерському районах, він незначно відрізняється від розміщеного на сайті київських олімпіад.

УМОВИ ЗАВДАНЬ III ЕТАПУ

1. Поліклініка

Максимальна оцінка: 100 балів
Обмеження на час: 0,25 сек.
Обмеження на пам'ять: 250 МБ
Вхідний файл: clinic.in
Вихідний файл: clinic.out
Програма: clinic.*

На прийом до лікаря щодня приходить чимало людей. Кожен пацієнт перебуває на прийомі цілу кількість хвилин, але різних пацієнтів лікар може приймати різну кількість часу. Лікар починає прийом у момент часу t_1 хвилин і закінчує прийом у момент часу t_2 хвилин. Це означає, що будь-який пацієнт незалежно від того, скільки часу його прийматиме лікар, може зайти на прийом у моменти t_1, t_1+1, ..., t_2-1. Заходити на прийом до лікаря в інший час або тоді, коли лікар приймає іншого пацієнта, заборонено. Якщо пацієнт приходить у поліклініку в момент t, він чекає на перший момент часу $s=t$ такий, що на цей момент лікар веде прийом, причому вже встиг оглянути всіх пацієнтів, які прийшли у поліклініку раніше, тобто до моменту t. Якщо лікар не встигає оглянути всіх до кінця прийому, решта пацієнтів має прийти наступного дня.

Завдання. Знаючи, у який момент лікар починає і закінчує прийом, те, хто й коли прийде на прийом у конкретний день, а також скільки часу оглядатиме кожного пацієнта лікар, визначте момент часу, у який потрібно прийти на прийом Петрику П'яточкіну, щоб гарантовано потрапити *в цей день* до лікаря, але при тому чекати на прийом якомога менше. У випадку, коли є кілька альтернативних варіантів такого моменту часу, вам потрібно визначити найменший (найбільш ранній) із них.

Вхідні дані. У першому рядку вхідного файлу вказано три числа: кількість охочих потрапити на прийом n, час початку прийому t_1 і час завершення прийому t_2, що більший за t_1. У другому рядку перераховані n чисел $a_1, a_2, ..., a_n$ — час, коли у поліклініку зайшли відповідно перший, другий, ..., n-й охочий потрапити до лікаря. Числа $a_1, a_2, ..., a_n$ попарно різні й розташовані у порядку зростання. У третьому рядку вхідного файлу перераховані n чисел $b_1, b_2, ..., b_n$ — час, необхідний лікарю на огляд відповідно першого, другого, ..., n-го пацієнта.

Усі числа у вхідному файлі натуральні. Кількість пацієнтів n не більша за 10^5, решта чисел не перевищують 10^9.

Доба на планеті, де мешкає Петрик П'яточкін, триває значно довше, ніж на Землі, тому час початку прийому t_1, час завершення прийому t_2, а також числа $a_1, a_2, ..., a_n$ та $b_1, b_2, ..., b_n$ можуть бути більшими за 1440 — кількість хвилин у земній добі.

Вихідні дані. Вихідний файл повинен містити єдине натуральне число — найменший момент часу, коли Петрик П'яточкін має прийти в поліклініку, щоб гарантовано потрапити до лікаря, але прочекати на прийом якомога менше часу. Якщо Петрик прийде водночас з іншою людиною, його як молодшого пропустять уперед.

Приклади

№	clinic.in	clinic.out
1	3 10 20 7 14 18 5 2 1	17
2	5 10 20 4 9 12 16 22 4 10 10 9 2	9
3	1 10 20 5 15	5
4	1 10 20 15 15	10

Пояснення

В усіх чотирьох прикладах лікар запускає на прийом із 10-ї до 19-ї хвилини включно.

У прикладі 1 пацієнт, що прийшов у момент часу 7 хв, зайде до лікаря першим, тобто о 10-й хвилині, а вийде через 5 хвилин. Відразу після цього — о 15-й хвилині — в кабінет зайде пацієнт, що прийшов у момент часу 14 хв. Він вийде з кабінету через 2 хвилини, відразу після чого ні в кабінеті, ні в черзі нікого не буде. Отже, якщо Петрик П'яточкін прийде в поліклініку о 17-й хвилині, він не чекатиме ні хвилини. Зауважимо: якби Петрик прийшов о 18-й чи о 19-й хвилині, він би також не чекав. Але вивести потрібно 17, бо це перший з усіх моментів часу, коли Петрик може прийти, щоб не чекати на прийом.

У прикладі 2 Петрик повинен чекати не менше ніж 5 хвилин незалежно від того, коли він прийде в поліклініку. Найвигідніший варіант — прийти о 9-й хвилині, щоб зайти в кабінет другим о 14-й хвилині. У цьому випадку одночасно з хлопцем прийде ще один пацієнт. Але згідно з умовою задачі цей пацієнт пропустить Петрика вперед.

У прикладі 3 єдиний спосіб для Петрика потрапити до лікаря у перший день прийому — прийти не пізніше за іншого пацієнта.

У прикладі 4 хлопцю достатньо прийти у момент початку прийому — він одразу потрапить до лікаря.

2. Фарбування

Максимальна оцінка: 100 балів
Обмеження на час: 1 сек.
Обмеження на пам'ять: 32 МБ
Вхідний файл: farben.in
Вихідний файл: farben.out
Програма: farben.*

На заняттях шкільного гуртка Євгенію навчили виготовляти з паперу моделі правильних многогранників (платонових тіл) і напівправильних многогранників (архімедових тіл). Вона запам'ятала такі означення:

1. **Платоновим тілом називають** (опуклий) многогранник, у якому всі грані — правильні однакові многокутники, а многогранні кути при всіх вершинах однакові.

2. **Архімедовим тілом називають** (опуклий) многогранник, у якому всі грані — правильні многокутники (не обов'язково з однаковою кількістю сторін), а многогранні кути при всіх вершинах однакові. При цьому є щонайменше дві різні грані.

Таким чином, в архімедовому тілі грані кожного виду зустрічаються в тій самій кількості й у тій самій послідовності при обході навколо кожної вершини (чи для однакових, чи для протилежних напрямів обходу, якщо «дивитися ззовні»).

Кілька моделей многогранників Євгенія виготовила для оформлення кабінету математики і прихопила додому, щоб вразити своїх рідних. Враження, звичайно, вона справила. Але, вечеряючи, недогледіла, як молодша сестра Марічка взялася розфарбовувати грані: кожну грань лише однією фарбою. При цьому кілька граней могли бути й одного кольору, навіть якщо вони були суміжними, тобто мали спільне ребро. Євгенія задумалась... Її, як майбутнього математика, зацікавило питання: скількома способами можна розфарбувати грані тіла Архімеда, маючи певну кількість фарб, з точністю до *симетрій — рухів простору, що залишають многогранник на тому самому місці*. Інакше кажучи, коли не розрізняють розфарбування, отримані одне з іншого взаємно однозначним відображенням граней при збереженні відношення суміжності.

Завдання. Знаючи все про суміжність чи несуміжність граней тіла Платона чи Архімеда, визначте кількість розфарбувань цього многогранника для даної кількості кольорів.

Вхідні дані. У першому рядку вхідного файлу вказано *натуральну* кількість фарб m, $m<6$.

Кількість рядків вхідного файлу на 1 перевищує кількість граней многогранника і менша від 24. Усі грані многогранника занумеровано послідовними натуральними числами, починаючи з 1. При $j>1$ j-й рядок містить (невпорядкований) перелік номерів граней, суміжних з гранню $(j-1)$.

Вхідні дані гарантують, що кількість симетрій многогранника не перевищує 120.

Вихідні дані. Вихідний файл має містити кількість розфарбувань граней тіла, виконаних за допомогою m фарб і різних з точністю до симетрій многогранника.

Приклад

farben.in	farben.out
2 2 3 4 1 3 4 1 2 4 1 2 3	5

Пояснення

Різні розфарбування двома фарбами правильного 4-гранника (який є трикутною пірамідою) розрізняють лише за кількістю граней одного кольору.

3. Істотні інверсії

Максимальна оцінка: 100 балів
Обмеження на час: 0,1 сек.
Обмеження на пам'ять: 32 МБ
Вхідний файл: inverses.in
Вихідний файл: inverses.out
Програма: inverses.*

Розглянемо довільну послідовність чисел $x_1, x_2, \ldots, x_n$. Пару індексів (j, k) називають *інверсією (порушенням порядку)*, якщо $j<k$ та $x_j>x_k$. Для довільного невід'ємного числа t пару індексів (j, k) назвемо *t-істотною інверсією*, якщо $j<k$ та $x_j>x_k+t$.

Завдання. Підрахуйте кількість t-істотних інверсій у послідовності $x_1, x_2, \ldots, x_n$.

Вхідні дані. Перший рядок містить записи двох цілих чисел n та t при $1\leqslant n\leqslant 50\ 000$, $0\leqslant t\leqslant 10^9$. У другому рядку записано цілі числа $x_1, x_2, \ldots, x_n$, які за модулем не перевищують 10^9.

Вихідні дані. Єдиний рядок вихідного файлу має містити кількість t-істотних інверсій у послідовності $x_1, x_2, \ldots, x_n$.

Приклади

№	inverses.in	inverses.out
1	6 0 1 2 2 9 5 4	3
2	5 2 1 2 7 3 6	1

Пояснення до прикладів

У прикладі 1 інверсії такі: (4, 5), (4, 6) і (5, 6). Усі ці інверсії — 0-істотні.

У прикладі 2 інверсії такі: (3, 4) і (3, 5). Але $x_3 \leqslant x_5+2$, тобто інверсія (3, 5) не є 2-істотною. Такою є лише пара (3, 4).

4. Фортеця

Максимальна оцінка: 100 балів
Обмеження на час: 0,1 сек.
Обмеження на пам'ять: 32 МБ
Вхідний файл: fortress.in
Вихідний файл: fortress.out
Програма: fortress.*

На полі потрібно збудувати фортецю. План її вигляду згори повинен мати форму *невиродженого опуклого* многокутника, сторони якого зображують вали, а вершини — вежі. Також вежі можна розташовувати на валах. Місцевість, де треба збудувати фортецю, є дуже різноманітною з гео- та гідрологічної точки зору. Тому будувати вежі можна лише у певних точках. На відміну від веж, *прямолінійні* вали можна насипати довільно. Чим більше веж розташовано вздовж огорожі фортеці, тим краще.

Завдання. Визначте, яку найбільшу кількість веж можна розташувати у перетинах (стиках) валів і вздовж валів фортеці, яка при виді згори має форму *невиродженого опуклого* многокутника, за умови, що в усіх стиках валів (вершинах многокутника) міститимуться вежі.

Вхідні дані. Перший рядок містить запис цілого числа n ($1 \leqslant n \leqslant 100$) — кількості точок, де можна будувати вежі. У кожному з наступних n рядків записано по два цілих числа x_j та y_j — координати точки, де можна будувати вежу ($|x_j| \leqslant 10\,000$, $|y_j| \leqslant 10\,000$). Усі точки (x_1, y_1), (x_2, y_2), …, (x_n, y_n) є різними.

Вихідні дані. Єдиний рядок вихідного файлу має містити запис найбільшої кількості веж фортеці. Якщо побудувати фортецю неможливо, то потрібно записати 0.

Приклади

№	fortress.in	fortress.out
1	3 0 0 1 1 2 2	0
2	4 1 1 10 1 1 10 3 3	3
3	4 −1 0 0 0 1 0 0 1	4

5. Вкладені множини

Максимальна оцінка: 100 балів
Обмеження на час: 5 сек.
Обмеження на пам'ять: 32 МБ
Вхідний файл: nested.in
Вихідний файл: nested.out
Програма: nested.*

Петрик нещодавно дізнався про поняття множини і вкладення множин. Він зацікавився послідовностями $S_0 \supseteq S_1 \supseteq S_2 \supseteq \ldots \supseteq S_N$, у яких кожний наступний елемент — підмножина попереднього, S_0 — певна скінченна множина $\{a_1, a_2, \ldots, a_M\}$.

Петрик зауважив: кожній з вкладених множин можна поставити у відповідність ціле число

$$H(S) = \sum_{a_k \in S} 2^{k-1}.$$

Таким чином, $H(S_0)=2^M-1$, $H(\varnothing)=0$.

Петрик випадковим чином обрав N чисел: $h_1, h_2, \ldots, h_N$. Йому стало цікаво: скільки існує різних наборів вкладених множин $S_1, S_2, \ldots, S_N$ при $H(S_1)=h_1 \pmod{41}$, $H(S_2)=h_2 \pmod{41}$, …, $H(S_N)=h_N \pmod{41}$? На жаль, Петрик збився з рахунку, тому просить вас про допомогу.

Вхідний файл. Перший рядок вхідного файлу містить два цілих числа N і M: $1 \leqslant N \leqslant 5$, $1 \leqslant M \leqslant 20$.

Другий рядок містить N цілих чисел $h_1, h_2, \ldots, h_N$, кожне з яких задовольняє умову: $0 \leqslant h_k \leqslant 40$.

Вихідний файл. Єдиний рядок вихідного файлу повинен містити одне ціле число — шукану кількість вкладених множин. Відомо, що це число не перевищує 10^{18}.

Приклади

№	nested.in	nested.out
1	3 5 7 3 3	1
2	3 5 7 4 5	0
3	1 7 3	4
4	3 10 31 29 17	28

Пояснення до тестів 1–3:

1. Існує лише один варіант:
$S_1=\{a_1, a_2, a_3\}$, $S_2=S_3=\{a_1, a_2\}$.
$H(S_1)=7$, $H(S_2)=H(S_3)=3$.

2. Такої комбінації вкладених множин не існує.

3. $H(\{a_1, a_2\})=2^{1-1}+2^{2-1}=3$,
$H(\{a_3, a_4, a_6\})=2^2+2^3+2^5=4+8+32=44=41+3$,
$H(\{a_1, a_3, a_5, a_7\})=2^0+2^2+2^4+2^6=1+4+16+64=85=2\cdot 41+3$,
$H(\{a_2, a_3, a_4, a_5, a_6, a_7\})=2^1+2^2+2^3+2^4+2^5+2^6=2+4+8+16+32+64=126=3\cdot 41+3$.

(Далі буде)