МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УЖГОРОДСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ЛІТНЯ ШКОЛА З ПРОГРАМУВАННЯ

Матеріали лекцій, умови та розбір задач 2017-2019 рр.

Ужгород • Рік-У • 2020

УДК 519.85(023)
Л11

Л11 **Літня школа з програмування** : Матеріали лекцій, умови та розбір задач 2017-2019 рр. / За ред. Сергія Вапнічного, Олександра Міци, Сергія Оришича. – Ужгород : Рік-У, 2020. – 336 с.

ISBN 978-617-7692-91-0

У книзі опубліковано вибрані матеріали лекцій, умови та розбір задач Літньої школи з програмування, яка проходила в Ужгороді в 2017-2019 рр.

Збірник буде корисний для всіх, хто цікавиться програмуванням, готується до олімпіад, хоче вдосконалити знання зі складання алгоритмів і розв'язування задач.

УДК 519.85(023)

ISBN 978-617-7692-91-0

© Василь Білецький, Віталій Герасимів, Матвій Асландуков, Павло Іржавський, Євген Задорожній, Ігор Баренблат, Олександр Рибак, Роман Білий, Тарас Савицький, Андрій Селіванов, Даніїл Смельський, Адальберт Макарович, 2020
© Сергій Вапнічний, Олександр Міца, Сергій Оришич (упорядкування, редагування), 2020
© ТОВ «РІК-У», 2020

ПЕРЕДМОВА

Літня школа в Ужгороді стала традиційним етапом підготовки до основних змагань зі спортивного програмування. Саме спортивне програмування стало перспективним інтелектуальним видом спорту. З кожним роком школярів і студентів, які цікавляться олімпіадами з програмування, стає все більше. Олімпіади з програмування самі собою є дуже цікавими змаганнями через їх регламент, зокрема й завдяки автоматизації та абсолютній прозорості ранжування учасників. Участь в олімпіадах, крім освоєння важливих тем з алгоритміки, розвиває такі якості, як спритність і нестандартність мислення. Навіть найлегше завдання або найпростіший алгоритм двоє програмістів реалізують абсолютно по-різному. Реалізація придуманих ідей – це своєрідний спосіб їх викладення за допомогою однієї з дозволених на олімпіаді мов програмування, яка є тільки інструментом.

Як і в іншій сфері діяльності, у спортивному програмуванні для досягнення успіху потрібно дуже багато працювати. Уміння придумувати розв'язки, генерувати ідеї – такі ж, як грати у футбол, бігати марафони, малювати картини… І що найголовніше – усі ці навички можна тренувати й розвивати. Так, талант або схильність до певного виду діяльності відіграють не останню роль у становленні людини як професіонала. Ці якості однозначно полегшують формування і розвиток спеціальних здібностей,

необхідних в обраній сфері діяльності. Але не менший вплив на результат мають такі людські якості, як дисциплінованість, витримка і стійкість. Напевно, усе ж більшість тих, хто читає, погодяться, що останні якості можна успішно розвивати. Що й роблять школярі та студенти, готуючись до наступних олімпіад. По суті, перемога на змаганні не така важлива й цікава сама собою. Для учасника це тільки показник свого особистісного розвитку – чи зміг він побороти свою лінь, організувати свій вільний час, зрештою, чи став він кращим щодо попередньої версії себе.

Багато колишніх олімпійців організовують успішні проєкти, пов'язані не тільки з програмуванням та ІТ-технологіями. Завдяки участі в олімпіадах, їм вдалося виробити стійкість до складних психологічних навантажень. Провівши тисячі годин за тренуваннями, намагаючись не витратити навіть одної з них даремно, вони навчилися оцінювати вірогідність перемоги і поразки. Опанували наявні й виробили власні методи боротьби зі стресовими ситуаціями, із сумнівами й занепокоєннями, що їх тією чи іншою мірою відчувають олімпійці під час змагань. Це вироблена здатність приймати важливі рішення й нести відповідальність за них.

У цій книзі зібрано матеріали лекцій, які проходили на Літній школі в Ужгороді у 2017-2019 роках. Вони дозволять вам краще підготуватись до змагань зі спортивного програмування.

І пам'ятаймо: успіх приходить до тих, хто йде до нього!

ЗМІСТ

ЧАСТИНА 1. 2017 РІК

• КОНТЕСТ ВАСИЛЯ БІЛЕЦЬКОГО

Основні досягнення Василя Білецького:

- Золота медаль в фіналі Міжнародної студентської олімпіади з програмування, 2008 рік, Банфф (Канада).
- II місце на змаганні “Google Code Jam 2013”, Лондон (Великобританія).
- Тренер команди LNU_Penguins.

ЗАДАЧІ ТА ІДЕЇ ЩОДО РОЗВ’ЯЗАННЯ

ЗАДАЧА A. Ужгородський рейтинг

Цього року організатори літньої школи в Ужгороді вирішили ввести рейтинг для **N** команд, що приїхали на тренувальні збори. Рейтинг команди — це ціле додатне число. Нехай рейтинги команд на зборах — $R_1, R_2, \ldots, R_N$. Відомо, що максимальний рейтинг рівний **Max**, мінімальний — **Min**, а середнє арифметичне — **Mean**.

$$Max = \max(R_1, R_2, ..., R_N)$$
$$Min = \min(R_1, R_2, ..., R_N)$$
$$Mean = \frac{R_1, +R_2 + ... + R_N}{N}$$

Вам необхідно знайти рейтинги команд, що приїхали на ужгородську літню школу з програмування.

Формат вхідних даних

Чотири цілих числа **N, Max, Min** та **Mean** через пробіл.

Формат вихідних даних

Виведіть **N** цілих чисел $R_1, R_2, ..., R_N$ через пробіл. Якщо існує більше одного розв'язку — виведіть будь-який. Якщо розв'язку не існує — виведіть "**Impossible**"(без лапок).

Обмеження

$1 \le N \le 100,$
$1 \le Min \le Mean \le Max \le 10000.$

Приклади

stdin	stdout
4 2000 1000 1400	1600 1000 2000 1000
4 2000 1000 1800	Impossible

Примітка

У першому прикладі є багато розв'язків, наприклад

- 1000 1275 1325 2000
- 2000 1000 1599 1001

Розв'язання задачі «A. Ужгородський рейтинг»

Легко побачити, що сума всіх рейтингів рівна **Mean** · **N**. Також очевидно, що сума рейтингів не може бути меншою

ніж **(N–1)** · **Min+Max** та не може бути більшою ніж **(N–1)** · **Max+Min**. Отже, розв'язок задачі існує тоді і тільки тоді, коли

(N–1)· Min + Max ⩽ Mean· N ⩽ (N–1)· Max + Min.

Якщо розв'язок існує, то побудувати його можна багатьма способами. Наприклад, нехай усі рейтинги рівні **Min**, окрім одного, який рівний **Max**. Далі поки сума рейтингів менша, ніж **Mean*N,** виберемо довільний рейтинг, менший, ніж **Max,** та збільшимо його на одиницю.

ЗАДАЧА B. Ужгородські дороги

Дороги в Ужгороді відомі своєю високою якістю та неймовірною міцністю. Єдина проблема — далеко не всюди в місті вони є.

Ужгород — це мегаполіс, що складається з кварталів. Кожен квартал задається своїми координатами **(r, c)**, де **r, c** — цілі числа. Два квартали **(r1,c1)** та **(r2,c2)** називаються сусідніми, якщо **|r1 –r2|+|c1 –c2| = 1**. Якщо у двох сусідніх кварталах є дороги, то з одного можна доїхати на авто в інший за одну хвилину.

Перед виборами ужгородська влада пообіцяла жителям зробити таке: для кожної пари кварталів (не обов'язково сусідніх), де є дороги **(r1,c1)** та **(r2,c2)**, з одного можна буде доїхати на авто в інший за **|r1 – r2| + |c1 – c2|** хвилин (можливо транзитом через інші квартали, де є дороги). Для цього адміністрація міста планує побудувати дороги у кварталах, де їх поки немає. Проте можновладці, як завжди, хочуть заощадити кошти платників податків, тому після побудови нових доріг ё(виконання передвиборчої обіцянки) кількість кварталів з дорогами має бути мінімально

можливою. Зауважте, що обіцянка буде стосуватися і тих кварталів, де побудують дороги.

Вас нещодавно прийняли на стажування в ужгородську міську раду, і, звісно ж, це непросте завдання делегували саме Вам.

Формат вхідних даних

Перший рядок містить ціле число **N** — кількість кварталів міста, де зараз є дороги. Кожен з наступних **N** рядків містить два числа **Rj** та **Cj** — координати відповідного кварталу з дорогами.

Формат вихідних даних

Мінімальна кількість кварталів з дорогами.

Обмеження

$1 \le N \le 1000$,
$1 \le Rj, Cj \le 1000000000\ (10^9)$,
усі координати кварталів (**Rj, Cj**) попарно різні.

Приклади

stdin	stdout
4 1 3 3 6 6 4 4 1	12

Примітка

Для оптимального виконання обіцянок необхідно додатково побудувати дороги ще у **8** кварталах: **(2,3) (3,3) (3,4) (3,5) (4,2) (4,3) (4,4)** та **(5,4).**

Розв'язання задачі «B. Ужгородські дороги»

Розіб'ємо всі задані квартали на групи за першою їхньою координатою `R` та впорядкуємо групи за зростанням `R`.

Очевидно, що для першої групи необхідно побудувати дороги у всіх кварталах між кварталами з найменшим та найбільшим значенням координати `C`. Далі будемо підтримувати поточний відрізок `X` по координаті `C`, у якому потрібно будувати дороги. При переході до наступної групи, розглянемо всі квартали, що мають більшу координату `R`, ніж попередня група, та знайдемо найменше та найбільше значення координат `C` (отримаємо відрізок `Y`). Також розглянемо всі задані квартали наступної групи та побудуємо відрізок `Z`, узявши відповідні найменше та найбільше значення координати `C`. Якщо відрізки `X` та `Y` перетинаються, то очевидно, що необхідно побудувати дороги у всіх кварталах, що знаходяться в прямокутнику, заданому перетином відрізків `X` та `Y` і проміжком між попередньою та наступною групою.

Якщо ж `X` та `Y` не перетинаються, то з попередньої до наступної групи можна перейти, побудувавши перпендикулярний до `X` відрізок, що сполучає один з кінців `X` (той, що ближчий до `Y`) з прямою, що містить `Y`, та відповідно збільшивши `Y`. Потім за поточний відрізок `X` приймаємо об'єднання відрізка переходу між групами (перетин `X` та `Y`, або найближча до `Y` точка `X`) та відрізка `Z` і переходимо до наступної групи.

Не складно довести, що описаний алгоритм побудує дороги в мінімально можливій кількості кварталів.

ЗАДАЧА C. Ужгородські хіміки

Нещодавно місцеві послідовники Менделєєва відкрили нові чотири хімічні елементи — мадярій (**H**), полякій (**P**), румуній (**R**) та словакій (**S**). Для того щоб утворити одну нову молекулу, потрібно взяти два атоми різних елементів. Відповідно ужгородські вчені записали до своїх здобутків шість нових молекул:

- **HP** — мадяро-полякій
- **HR** — мадяро-руміній
- **HS** — мадяро-словакій
- **PR** — поляко-румуній
- **PS** — поляко-словакій
- **RS** — румуно-словакій

Ужгородські науковці мають **H** атомів мадярію, **P** атомів полякію, **R** атомів румунію та **S** атомів словакію. Допоможіть їм утворити з цих атомів максимально можливу кількість молекул.

Формат вхідних даних

Чотири цілих числа **H, P, R** та **S** через пробіл.

Формат вихідних даних

Виведіть шість цілих чисел через пробіл — кількість молекул **HP, HR, HS, PR, PS** та **RS**, які можуть утворити хіміки, відповідно. Якщо існує більше одного розв'язку, то виведіть той, де кількість молекул **HP** є найбільшою. Якщо ж і далі існує більше одного розв'язку, то оберіть той, де найбільше молекул **HR**, далі — **HS**, далі — **PR**, далі — **PS**.

Обмеження

$0 \le H,P,R,S \le 1000000000 (10^9)$.

Приклади

stdin	stdout
1 2 3 4	1 0 0 0 1 3
0 4 7 47	0 0 0 0 4 7

Примітка

У першому прикладі можна утворити **5** молекул, використавши всі атоми. Одночасно утворювати молекули **HP** та **PR** не можна, бо тоді залишаться **2** атоми **R** та **4** атоми **S**, а загальна кількість молекул буде **4**. У другому прикладі використати всі атоми неможливо.

Розв'язання задачі «C. Ужгородські хіміки»

Маючи набір атомів, легко визначити максимальну кількість молекул, які можна з них утворити. Нехай **T** — загальна кількість атомів, тоді якщо існує елемент, кількість атомів якого X більша ніж $\frac{T}{2}$, тоді максимальна кількість молекул рівна **T – X**, інакше вона **рівна** $\left[\frac{T}{2}\right]$.

Спробуємо максимізувати кількість молекул HP. Якщо ми можемо утворити одну таку молекулу, подивимось на максимальні загальні кількості молекул, що ми можемо утворити з атомів до та після утворення однієї молекули **HP**. Якщо ці кількості відрізняються більше ніж на одиницю, то ми не можемо утворювати **HP**, оскільки це призведе до зменшення загальної кількості молекул, які ми можемо утворити. Інакше ми можемо продовжувати утворювати молекули **HP**, допоки це не впливає на загальну кількість молекул. Після цього аналогічно утворюємо молекули **HR**, потім — **HS**, потім — **PR**, потім — **PS**, потім — **RS**.

Описаний алгоритм дає правильні відповіді, але працює надто повільно. Аби пришвидшити його, замінимо лінійний пошук максимальної кількості молекул на бінарний.

ЗАДАЧА D. Ужгородські щасливі числа

Відомо, що щасливим числом є таке додатне ціле число, десятковий запис якого містить тільки четвірки та сімки. Наприклад, щасливими є числа **4**, **7**, **47**, **7777** та **4744474**. Ужгородське щасливе число – це таке число, що може бути записане у вигляді суми не більше ніж **K** щасливих чисел. Наприклад, якщо **K = 3**, то **4, 7, 8 (4+4), 11 (7+4), 12 (4+4+4)** та **121 (44 + 77)** є щасливими ужгородськими числами, а **3, 5, 16** та **28** такими не є. Вам необхідно знайти кількість щасливих ужгородських чисел, не менших ніж **A** та не більших ніж **B**.

Формат вхідних даних

Перший рядок містить ціле число **A**. Другий рядок містит ь ціле число **B**. Третій рядок містить ціле число **K**.

Формат вихідних даних

Кількість щасливих ужгородських чисел.

Обмеження

$1 \le A \le B \le 1000000000000000000$ (10^{18}),
$1 \le K \le 1000000000000000000$ (10^{18}).

Приклади

stdin	stdout
1 12 2	4
1 12 3	5
4777778 7444443 1	0

Примітка

У першому прикладі є чотири щасливі ужгородські числа: **4**, **7**, **8** (**4 + 4**) та **11** (**4 + 7**). У другому прикладі, оскільки **K = 3**, **12** теж є щасливим ужгородським числом.

Розв'язання задачі «D. Ужгородські щасливі числа»

Деякі додатні цілі числа (наприклад, **3** та **13**) взагалі не можна представити у вигляді суми щасливих чисел. Проте не складно помітити, що для всіх чисел починаючи з **28** (**28 = 4 · 7**), таке представлення завжди існує. Далі можна довести, що для всіх чисел, починаючи з **28**, існує представлення у вигляді суми щасливих чисел, що містить не більше **9** доданків. Тому якщо **K ≥ 9**, то всі числа починаючи з **28** є щасливими ужгородськими числами.

Якщо **K ≤ 8**, то ми будемо знаходити кількість ужгородських щасливих чисел, менших ніж **X** (**A** або **B+1**), використовуючи динамічне програмування з кількома вимірами (будемо будувати числа починаючи від менш значущих цифр):

N — кількість уже доданих цифр;

E — результат порівняння з першими (молодшими) **N** цифрами числа **X** (**<**, **>** або **=**);

C — клас еквівалентності (опис наведено нижче).

Значення для кожного стану динамічного програмування — кількість щасливих ужгородських чисел.

Розглянемо всі можливі суфікси з **N** молодших цифр чисел, що можуть бути утворені з **K** або менше доданків (щасливих чисел). Для кожного можливого переносу на наступний (**N+1**)–*ий* розряд (перенос може бути від **0** до **6** включно) нас цікавить максимальна кількість щасливих чисел у сумі, що мають принаймні **N** цифр та формують

заданий суфікс. Множина таких максимальних кількостей для кожного з переносів описує клас еквівалентності `C`. Маючи клас еквівалентності, ми можемо перебрати всі можливі переноси та комбінації четвірок і сімок на (`N+1`)-*их* позиціях щасливих чисел та сформувати переходи між станами динамічного програмування. Всі отримані переходи необхідно групувати за отриманою цифрою у наступному розряді.

ЗАДАЧА E. Ужгородські плиточники

Ужгородські плиточники відомі своєю майстерністю у всьому світі. Вони можуть викласти будь-яку плитку у найнедоступнішому місці.

Цього разу ужгородські майстри мають замостити плиткою підлогу в кімнаті розміру `R` метрів на `C` метрів. Підлога кімнати поділена на квадрати розміром метр на метр, для кожного з яких відомо, чи потрібно викладати там плитку. Плиточники мають у своєму запасі необмежену кількість квадратних плиток цілих розмірів — метр на метр, два метри на два метри, три метри на три метри і так далі. Ужгородці — великі віртуози та естети, саме тому вони хочуть використати мінімальну кількість плиток для замощення цієї підлоги. Плитки не можуть перекривати одна одну та мають бути розміщені тільки там, де потрібно згідно з планом.

Вам потрібно допомогти майстрам знайти мінімальну необхідну кількість плиток.

Формат вхідних даних

Перший рядок містить два цілих числа `R` та `C` через пробіл. Наступні `R` рядків містять опис плану підлоги. Кожен

з `R` рядків містить рівно `C` символів '`.`' або '`#`'. Символ '`.`' позначає квадратний метр кімнати, який потрібно замостити плиткою, а символ '`#`' — місце, де не потрібно її викладати.

Формат вихідних даних

Мінімальна кількість плиток.

Обмеження

`1 ≤ R ≤ 10, 1 ≤ C ≤ 10.`

Приклади

stdin	stdout
3 3	1
3 4 .... ..#.	8

Примітка

У першому прикладі можна використати одну плитку розміру три метри на три метри. У другому прикладі знадобиться одна плитка розміру два метри на два метри та сім плиток розміру метр на метр.

Розв'язання задачі «E. Ужгородські плиточники»

Розглянемо рекурсивний розв'язок повного перебору з поверненням. Будемо розглядати квадратні метри кімнати за рядками, а потім за стовпцями. Кожного разу, коли ми зустрічаємо вільний квадрат, ми маємо обрати розмір плитки, яку

можна розмістити так, щоб поточний квадрат був покритий її верхнім лівим кутом. Будемо підставляти всі можливі варіанти по черзі. Серед усіх варіантів повністю заповненої підлоги виберемо той, який містить мінімальну кількість викладеної нами плитки.

Описаний підхід працює добре для малих тестів, але не вкладається у відведений час для великих. Для його оптимізації використаємо підхід динамічного програмування із запам'ятовуванням. Основна ідея полягає в тому, що якщо для деякого набору квадратів кімнати існує багато різних способів часткового їх викладання плиткою, то нас цікавить тільки один з них (той, що використовує мінімальну кількість плиток). Стан динаміки задає поточний квадратний метр (позиція) та набір покритих квадратних метрів (клітинок), а значення — мінімальна кількість використаних плиток. Будемо розглядати всі стани у порядку збільшення поточної позиції. Через те, що усі види плиток мають квадратну форму, кількість досяжних таких станів буде відносно малою та дозволить вкластися у часові обмеження.

ЗАДАЧА F. Ужгородське поле чудес

Сьогодні в гостях ужгородської версії популярної телевізійної гри *«Поле чудес»* пан Бийло з села Велика Бийгань, що на Берегівщині.

За правилами гри сектори ігрового барабану пронумеровані від **1** до **N** за годинниковою стрілкою і в кожному з них записана деяка буква. Ведучий гри зав'язує Бийлові очі міцною ганчіркою і просить його двічі покрутити барабан.

Рухаючись за годинниковою стрілкою від сектора, де перший раз зупинилася стрілка барабану, до сектора, де вона зупинилася вдруге, ведучий записує на дошці слово, що утворене з букв відповідних секторів. Далі він просить свого гостя вгадувати це слово.

Бийло не бачить слова, проте знає розміщення букв на барабані. За один хід він може поставити ведучому запитання, на яке той має відповісти '**так**' або '**ні**'. Бийло — чемпіон минулого сезону цієї із захопливої гри, тому він хоче мінімізувати очікувану кількість запитань, необхідну для того, щоб вгадати слово. Вам необхідно обчислити цю кількість.

Будемо вважати, що кожен раз, коли Бийло крутить барабан, стрілка зупиняється в кожному з секторів рівноймовірно. Зауважте, що Бийлові потрібно вгадати записане на дошці слово, а не сектори, на які вказувала стрілка барабану.

Формат вхідних даних

Перший рядок містить ціле число N. Другий рядок містить N символів $S_1, S_2, \ldots, S_N$, що записані у відповідних секторах ігрового барабану.

Формат вихідних даних

Мінімальна очікувана кількість запитань. Відповідь вважатиметься правильною, якщо її абсолютна чи відносна похибка не буде більшою ніж 10^{-7}.

Обмеження

$1 \le N \le 1000$,

$S_1, S_2, \ldots, S_N$ — маленькі букви англійського алфавіту ('a'–'z').

Приклади

stdin	stdout
2 Ab	2.0
3 Xxx	1.66666666667

Примітка

У першому прикладі є 4 можливих слова: ’a’, ’b’, ’ab’, ’ba’. Кожне з них з однаковою ймовірністю може бути записане на дошці. Одна з оптимальних стратегій для Бийла:

• Запитати, чи починається записане на дошці слово з літери ’a’.

• Запитати, чи закінчується записане на дошці слово на літеру ’a’.

Розв’язання задачі «F. Ужгородське поле чудес»

Оскільки Бийло крутить барабан двічі, то ми маємо N2 пар секторів, де зупиниться стрілка барабану вперше та вдруге, кожна з імовірністю $\frac{1}{N^2}$. Кожна пара генерує відповідне слово, що буде записано на дошці. Розглянемо всі такі різні слова. Ймовірність того, що певне слово буде записане на дошці, рівна $\frac{X}{N^2}$, де X — кількість пар, які генерують це слово.

Кожного разу, коли Бийло ставить запитання ведучому, множина можливих слів розбивається на дві. Далі залежно від отриманої відповіді Бийло продовжує гру з однією з цих множин (так або ні). Він продовжує ставити запитання, поки множина містить більше одного слова. Множини слів утворюють бінарне дерево, листки якого є різними словами з відповідними ймовірностями, що генерують пари секторів барабану.

Завдання Бийла — знайти таке дерево, яке мінімізує очікувану відстань (кількість запитань) від кореня до листка. Таке формулювання нагадує задачу оптимального префіксного кодування, яка може бути розв'язана, наприклад, алгоритмом Гофмана.

ЗАДАЧА G. Ужгородський парадокс

Парадокс днів народження оцінює ймовірність того, що у випадково обраній групі збігатимуться дні народження якоїсь пари. У групах кількістю не менше `23` випадково вибраних людей, ймовірність збігу днів народження в якоїсь пари становить більше `50%`. Такий результат суперечить інтуїтивній уяві більшості.

Організатори літньої школи з програмування в Ужгороді вирішили узагальнити парадокс днів народження. Вони хочуть знати, яку мінімальну кількість студентів необхідно запросити на літню школу, щоб ймовірність того, що **`N`** з них народилися в один день, була не меншою ніж **`P`** відсотків.

Для цієї задачі будемо ігнорувати високосний рік. Припускаємо, що всі дні народження однаково ймовірні, ймовірність мати день народження в певний день для конкретного студента становить $\frac{1}{365}$.

Формат вхідних даних
Два цілих числа **`N`** та **`P`** через пробіл.

Формат вихідних даних
Мінімальна кількість студентів.

Обмеження

1 ≤ N ≤ 10,
0 ≤ P ≤ 100.

Приклади

stdin	stdout
2 50	23
4 100	1096

Примітка

У другому прикладі за принципом Діріхле, якщо запросити **1096 = (3*365 + 1)** студентів, то принаймні **4** з них матимуть спільний день народження.

Розв'язання задачі «G. Ужгородський парадокс»

Очевидно, що максимально можлива відповідь:

L = (N – 1) · 365 + 1.

Обчислимо кількість способів обрати дні народження для **K(0≤ K ≤L)** студентів так, щоб жодні **N** з них не народилися в один день. Для цього за допомогою динамічного програмування заповнимо двовимірний масив $A_{d,c}$ — кількість способів обрати дні народження (серед **d** перших днів року) для **c** студентів. Розглядаючи поточний день року, ми перебираємо кількість студентів, що народилися цього дня (щонайбільше **N–1**)

$$A_{d,c} = \sum_{k=0}^{k<N} A_{d-1,c-k} \cdot \binom{c}{k}.$$

Після виконання обчислень нас цікавить таке найменше **c**, що

$$1 - \frac{A_{365,c}}{365^c} \geq \frac{P}{100}.$$

Залежно від реалізації, розв'язок може не вкладатися в обмеження за часом. Проте всі відповіді можуть бути обчислені заздалегідь, оскільки ми маємо тільки `1010` різних варіантів вхідних даних.