D. Гра «Вгадай»

Назва задачі	Guessing Game
Обмеження часу	4 с
Обмеження використання пам'яті	1024 МБайт

У старому місті Лунда є вулиця з N будинків, які пронумеровані від 0 до $N-1$. Каріна живе в одному з цих будинків, і її друзі Оля та Уляна хочуть вгадати, в якому саме будинку вона проживає. Замість того, щоб просто розповісти своїм друзям, де вона живе, Каріна вирішує зіграти з ними в гру. Перед початком гри Оля та Уляна знають лише кількість будинків на вулиці. На цьому етапі Оля і Уляна можуть вибрати додатнє ціле число K і домовитися про стратегію. Після цього будь-яке подальше спілкування заборонено.

Гра складається з двох фаз. У першій фазі Каріна обирає порядок відвідування будинків таким чином, щоб її будинок був останнім. Потім вона приводить Олю по будинках один за одним у цьому порядку, не розкриваючи заздалегідь Олі порядку відвідування. Для кожного будинку, крім будинку Каріни, Оля має право написати на дверях цього будинку одне ціле число від 1 до K крейдою. Для останнього будинку, який вони відвідують - будинку Каріни, Каріна сама записує ціле число від 1 до K на дверях.

У другій фазі гри Уляна йде по вулиці від будинку 0 до будинку $N-1$ і читає всі числа, які написані на дверях Каріною та Олею. Тепер вона хоче вгадати, у якому будинку живе Каріна. У неї є дві спроби, щоб вгадати правильно, і тоді вони разом з Олею виграють гру, якщо вона успішно вгадає. В іншому випадку Каріна перемагає в грі.

Чи можете ви розробити стратегію, за якої Оля та Уляна гарантовано виграють гру? Ваша стратегія оцінюватиметься на основі значення K (що менше, то краще).

Імплементація

Це задача з багаторазовим виконанням, що означає, що ваша програма буде виконуватись кілька разів. При першому виконанні буде використовуватись стратегія Олі. Після цього буде використовуватись стратегія Уляни.

Перший рядок вхідних даних міститиме два цілих числа P і N, де P дорівнює 1 або 2 (перша або друга фаза), N — кількість будинків. **За винятком прикладу із умови (не використовується для**

оцінки), N **завжди дорівнюватиме** $100\,000$.

Вхідні дані залежать від фази:

Фаза 1

Ваша програма повинна почати роботу з виведення числа K в один рядок $(1 \leq K \leq 1\,000\,000)$. Потім, $N - 1$ разів, він повинен прочитати рядок, що містить індекс i $(0 \leq i < N)$, і вивести рядок із цілим числом A_i $(1 \leq A_i \leq K)$, де A_i — число, яке Оля пише на дверях будинку i. Кожен індекс i, крім індексу будинку Каріни, з'явиться рівно один раз у певному порядку, визначеному оцінювачем.

Фаза 2

Програма повинна зчитати рядок з N цілих чисел $A_0, A_1, \ldots, A_{N-1}$, де A_i — це число, записане на дверях будинку i.

Потім вона повинна вивести рядок з двох цілих чисел s_1 і s_2 $(0 \leq s_i < N)$ — індекси, що вгадуються. Значення s_1 і s_2 можуть бути рівними.

Деталі імплементації

Зауважте, що під час виконання вашої програми на фазі 2 програма перезапускається. Це означає, що ви не можете зберігати інформацію в деяких змінних між запусками.

Після кожного виведення рядка переконайтеся, що очистили буфер виводу стандартного потоку, інакше ваша програма отримає Time Limit Exceeded.

У Python функція `print()` автоматично очищує буфер виводу. На C++ використовуйте `cout << endl;`, що не тільки виводить символ нового рядка, але й очищає буфер; якщо використовуєте printf, то використовуйте `fflush(stdout)` для очищення буфера виводу.

Оцінювач для цієї задачі може бути **адаптивним**, що означає, що він може змінювати свою поведінку в залежності від виводу вашої програми, щоб запобігти проходженню евристичних розв'язків. Він може провести пробний запуск першої фази, переглянути ваш вивід, а потім запустити першу фазу насправді, використовуючи інформацію, отриману з попереднього запуску.

Ваша програма *повинна* бути детермінованою, тобто поводитись однаково, якщо її двічі запускати з одним і тим же входом. Якщо ви хочете використовувати випадковість у своїй програмі, переконайтеся, що використовуєте фіксоване випадкове початкове число. Це можна зробити, передавши сталу константу в `srand` (у C++) або `random.seed` (у Python), або, якщо використовуються генератори випадкових чисел C++11, вказавши початкове значення при побудові генератора випадкових чисел. Зокрема, ви не можете використовувати

`srand(time(NULL))` в C++. Якщо оцінювач виявить, що ваша програма не є детермінованою, вона отримає вердикт «Неправильна відповідь».

Якщо *сума* часів виконання (до 3) окремих запусків вашої програми перевищує ліміт часу, ваша програма отримає Time Limit Exceeded.

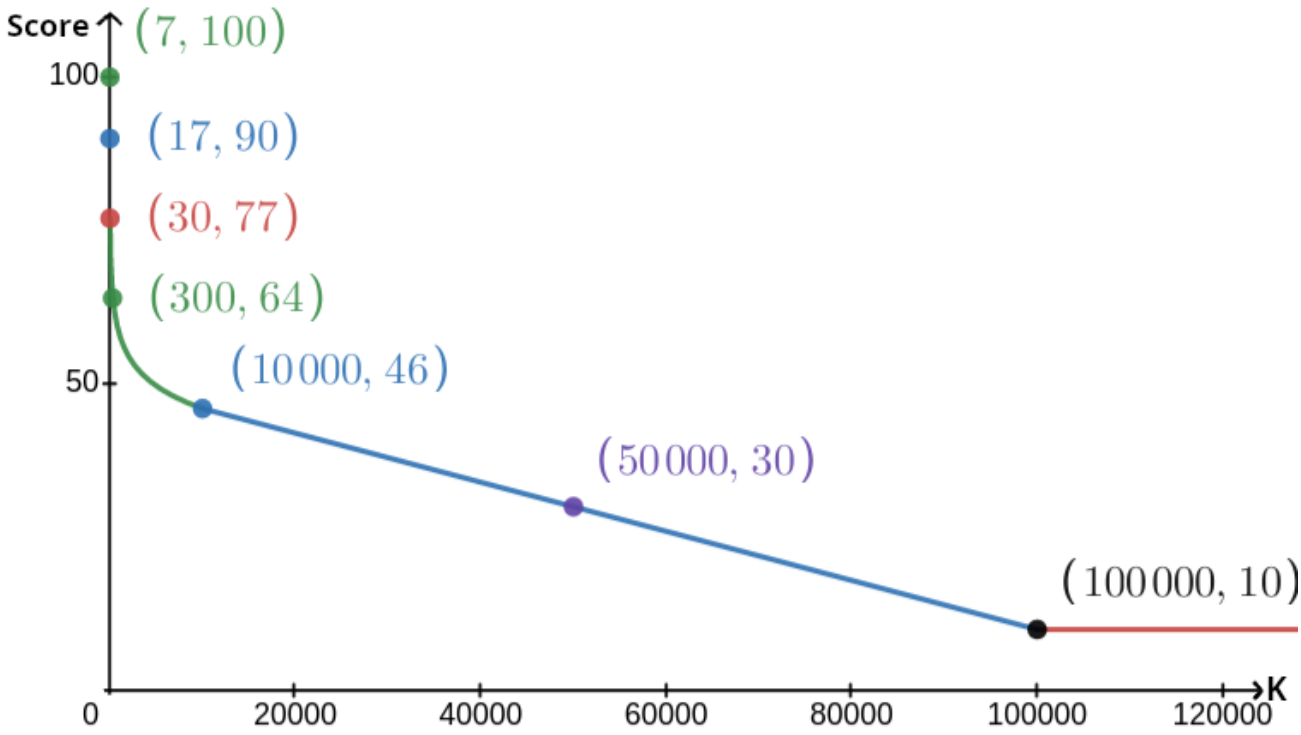
Оцінювання

Ваш розв'язок буде перевірений на певній кількості тестів. Якщо ваш розв'язок невірно працює на будь-якому з цих тестів (наприклад, надає неправильні відповіді (Wrong Answer), викликає помилку виконання (Run-Time Error), перевищує обмеження часу (Time Limit Exceeded) і т. д.), ви отримаєте 0 балів та відповідний вердикт.

Якщо ваша програма успішно знаходить індекс будинку Каріни в усіх тестових випадках, ви отримаєте оцінку Accepted, а бали будуть розраховані наступним чином. Позначимо K_{max} максимальне значення K, використане на будь-якому із тестів. Залежно від значення K_{max}:

	Бали
$K_{max} > 99\,998$	10 балів
$10\,000 < K_{max} \leq 99\,998$	$10 + \lfloor 40(1 - K_{max}/10^5) \rfloor$ балів
$30 < K_{max} \leq 10\,000$	$46 + \lfloor 31(4 - \log_{10}(K_{max}))/(4 - \log_{10}(30)) \rfloor$ балів
$7 < K_{max} \leq 30$	$107 - K_{max}$ балів
$K_{max} \leq 7$	100 балів

Функція підрахунку балів зображена на малюнку нижче.



Результати тесту з умови не враховуються при розрахунку оцінки, і ваш розв'язок не обов'язково має працювати для нього.

Інструмент тестування

Для полегшення тестування вашого розв'язку ми надаємо простий інструмент, який ви можете завантажити. Дивіться розділ "Вкладення" унизу сторінки задачі на Kattis. Використання цього інструменту є необов'язковим, і ви можете вносити зміни до нього. Зверніть увагу, що офіційна програма перевірки на Kattis відрізняється від тестового інструменту.

Приклад використання (з $N = 4$, $s = 2$, де s — це число, написане на останньому відвіданому будинку):

Для програм Python, скажімо `solution.py` (зазвичай запускається як `pypy3 solution.py`):

```
python3 testing_tool.py pypy3 solution.py <<<"4 2"
```

Для програм C++ спочатку скомпілюйте його (наприклад, за допомогою `g++ -g -O2 -std=gnu++17 -static solution.cpp -o solution.out`), а потім запустіть:

```
python3 testing_tool.py ./solution.out <<<"4 2"
```

Інструмент тестування буде відвідувати будинки у випадковому порядку. Щоб використовувати певний порядок, змініть інструмент тестування, де написано “MODIFY HERE”.

Приклад взаємодії

Приклад з умови ігнорується для підрахунку балів і ваше рішення не обов'язково повинно працювати для нього.

Припустимо, що $N = 4$ і Каріна живе в будинку 1. Нехай A — список чисел, написаних на будинках. Спочатку $A = [0, 0, 0, 0]$, де 0 означає, що на відповідному будинку не вказано номер.

Під час першого запуску вашого коду:

Дано $N = 4$. Ваше рішення відповідає $K = 3$.

Запитується A_2. Ваше рішення відповідає 3. A тепер $[0, 0, 3, 0]$.

Запитується A_0. Ваше рішення відповідає 1. A тепер дорівнює $[1, 0, 3, 0]$.

Запитується A_3. Ваше рішення відповідає 2. A тепер дорівнює $[1, 0, 3, 2]$.

Нарешті, грейдер встановлює $A_1 = 2$, так що в кінці $A = [1, 2, 3, 2]$. Це означає закінчення першого етапу.

На другому етапі вашого коду ваше рішення передається в список `1 2 3 2`.

Він відповідає `1 3`.

Оскільки одна з відгадок є правильним індексом будинку (1), Оля та Уляна виграють гру.

Вивід інтерактора	Ваш вивід
1 4	
	3
2	
	3
0	
	1
3	
	2

Вивід інтерактора	Ваш вивід
2 4	
1 2 3 2	
	1 3