XVI командная олимпиада школьников по программированию

13 декабря 2015 года

Задача Y. Фальшивая монета

Задача Y. Фальшивая монета

- Идея задачи — Фольклор
- Подготовка тестов — Николай Ведерников
- Разбор задачи — Дмитрий Якутов, Николай Будин

Постановка задачи

- Есть 12 монет, среди них одна фальшивая, не равная по весу остальным.
- Нужно не более, чем за 3 взвешивания на чашечных весах без гирь, определить фальшивую.

Решение задачи

- Пронумеруем монеты числами от 1 до 12.
- Взвесим монеты $\{1, 2, 3, 4\}$ с монетами $\{5, 6, 7, 8\}$.
- Рассмотрим 2 случая: они оказались равны или не равны.

Решение задачи. Случай 1

- Тогда среди них нет фальшивой, то фальшивая — одна из монет $\{9, 10, 11, 12\}$.
- Взвесим 9 и 10.
- Если они оказались равны, фальшивая монета среди $\{11, 12\}$. Взвесим 11 с настоящей, например, 9, если они равны, фальшивая — 12, иначе — 11.
- Если 9 и 10 не равны, фальшивая среди них. Взвесив 9 с любой настоящей, мы поймем, какая из 9 и 10 фальшивая.

Решение задачи. Случай 2

- Предположим, $\{1, 2, 3, 4\}$ — тяжелее, второй случай разбирается аналогично. Тогда, либо одна из монет $\{1, 2, 3, 4\}$ — фальшивая и тяжелее настоящей, либо одна из монет $\{5, 6, 7, 8\}$ — фальшивая и легче настоящей.
- Взвесим $\{1, 2, 5\}$ и $\{3, 4, 6\}$.
- Если они равны, среди них нет фальшивой, и фальшивая среди 7 и 8. Оставшимся взвешиванием определим её.

Решение задачи. Случай 2

- Если $\{1, 2, 5\}$ — тяжелее, то 5 не может быть фальшивой, так как в этом случае эта кучка была бы легче второй кучки. Аналогично 3 и 4 не могут быть фальшивыми.
- Взвесим 1 и 2. Если они не равны, то более тяжелая из них — фальшивая. Иначе, фальшивая — 6.
- Аналогично разберем случай, когда $\{1, 2, 5\}$ легче чем $\{3, 4, 6\}$.

Задача Z. Угадай строку

Задача Z. Угадай строку

- Идея задачи — Дмитрий Филиппов
- Подготовка тестов — Дмитрий Филиппов
- Разбор задачи — Николай Будин

Постановка задачи

- Жюри загадало битовую строку длины n ($1 \leqslant n \leqslant 1000$).
- Программа может делать запросы вида: «Присутствует ли битовая строка s как подстрока в загаданной?»
- Нужно отгадать строку не более чем за 1024 запроса.

Решение задачи

- Двоичным поиском найдем максимальную по длине строку из нулей, входящую в загаданную. Обозначим за k ее длину.
- Теперь будем дописывать к ней справа единицы и задавать вопрос про новую строку.
- Если такая строка присутствует, допишем в конец новую единицу.
- Если такой строки нет, заменим последнюю единицу на ноль, предполагая, что такая подстрока в строке есть, и допишем единицу.

Решение задачи

- Заметим, что мы не проверяем принадлежность строки после замены последней единицы на ноль. После того, как получим суффикс загаданной строки, мы начнем всегда дописывать нули в конец.
- Когда в конце нашей строки нулей станет больше k, наша строка гарантированно не будет являться подстрокой загаданной.
- Двоичным поиском найдем префикс нашей строки, который является суффиксом загаданной.
- Теперь будем дописывать слева единицы таким же образом, пока длина строки не станет равна n.

Задача A. Агроном-любитель

Задача A. Агроном-любитель

- Идея задачи — Сергей Копелиович
- Подготовка тестов — Михаил Дворкин
- Разбор задачи — Михаил Дворкин

Постановка задачи

Входные данные:

- Загадана последовательность чисел $a_1, \ldots, a_n$.
- Найти самый длинный подотрезок, в котором нет трех одинаковых чисел подряд.

Решение задачи

Идем слева направо, в каждый момент помним:

- текущее количество одинаковых чисел подряд;
- самое левое место старта фотографии, которое может включать текущую позицию.

Решение задачи

- Обновляем текущее количество одинаковых подряд чисел.
- Если их стало три, то продолжающаяся фотография обрывается.
- Если их хотя бы три, то новая фотография начинается так, чтобы задевать две из них.
- Если текущая фотография лучше текущего ответа, то обновляем его.

Задача B. Верёвочный парк

Задача B. Верёвочный парк

- Идея задачи — Фёдор Царёв
- Подготовка тестов — Андрей Комаров
- Разбор задачи — Андрей Комаров

Постановка задачи

- Трасса в верёвочном парке состоит из n платформ
  - платформы соединены $n - 1$ верёвками последовательно.
- Каждая верёвка определяется большим числом параметров:
  - Длина;
  - Минимальное расстояние между людьми;
  - Максимальное число людей на верёвке;
  - Скорости каждого из m посетителей.
- Про каждую платформу известно максимальное число людей на ней одновременно;
- За сколько все посетители пройдут трассу?

Идея решения

Основная идея решения — аккуратно смоделировать всё, что просят в условии:

- Выбираем интервал времени Δt, за который не произойдёт «ничего интересного»;
- Пересчитываем, где окажутся все посетители через Δt секунд.

Осталось понять, что такое «ничего интересного», и как определять Δt.

Идея решения

Основная идея решения — аккуратно смоделировать всё, что просят в условии:

- Выбираем интервал времени Δt, за который не произойдёт «ничего интересного»;
- Пересчитываем, где окажутся все посетители через Δt секунд.

Осталось понять, что такое «ничего интересного», и как определять Δt.

Интересные события

- Каждому посетителю выгодно двигаться по верёвке с максимально возможной скоростью;
- Посетителю, стоящему на платформе, выгодно как можно раньше сойти с неё на верёвку.

Интересные события

- Когда посетитель меняет скорость движения?
  - Если он двигается быстрее того, кто перед ним, и расстояние между ними сократилось до d_i.
- Когда посетитель может сойти с платформы?
  - Если на верёвке перед ним меньше r_i людей и;
  - Последний из людей на верёвке перед отошёл хотя бы на d_i.
- Назовём интервал Δt неинтересным, если в течение этого времени не произойдёт ничего из:
  - Находящийся на верёвке человек поменял скорость.
  - Последний из находящихся на i-й верёвке удалился на d_i.
  - Первый из находящихся на верёвке дошёл до её конца.

Алгоритм

- Для каждого человека храним, на верёвке или на платформе он сейчас и храним его эффективную скорость — то, с какой скоростью он движется.
- Для каждого посетителя считаем минимальное время до интересного события Δt_i с его участием:
  - Подошёл к идущему перед ним на расстояние d_i.
  - Дошёл до конца верёвки.
  - Последний из людей на верёвке и отошёл на d_i от начала.

Алгоритм

- Выбираем в качестве Δt минимум из Δt_i.
- Сдвигаем всех, кто сейчас на верёвке, в соответствие с их эффективными скоростями.
- Пересчитываем для всех новое состояние.
- Повторяем, пока все не окажутся на последней платформе.

Асимптотика

- Выбор очередного Δt происходит за $O(n)$.
- Пересчёт позиций и состояний для данного Δt происходит за $O(n)$.
- Каждый шаг вызван чем-то из:
  - Посетитель дошёл до конца верёвки: $O(nm)$ вариантов.
  - Посетитель дошёл до d_i от начала верёвки: $O(nm)$ вариантов.
  - Посетитель «догнал» идущего перед ним на верёвке и поменял скорость: $O(n^2m)$ вариантов.
- Итого, имеем $O(n^2m)$ шагов, каждый из которых обрабатывается за $O(n)$, поэтому суммарная сложность $O(n^3m)$.

Задача C. Школьная демократия

Задача C. Школьная демократия

- Идея задачи — Евгений Курпилянский
- Подготовка тестов — Евгений Курпилянский
- Разбор задачи — Евгений Курпилянский

Постановка задачи

- Дана последовательность чисел $a_1, \ldots, a_n$ $(a_i = b_i - g_i)$.
- Требуется разбить её на отрезки длины от l до r так, чтобы **разность** количества отрезков с положительной суммой и количества отрезков с отрицательной суммой была **максимальна**.

Решение задачи за $O(n^2)$

- $sum_i = a_1 + \ldots + a_i$
- res_i — ответ для префикса длины i
- $res_i = \max\limits_{k=l..r} (res_{i-k} + sign(sum_i - sum_{i-k}))$, где

$$sign(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x = 0 \\ -1, & \text{if } x < 0 \end{cases}$$

Оптимизация пересчета значений динамики

$$res_i = \max_{k=l..r} (res_{i-k} + sign(sum_i - sum_{i-k}))$$

- Рассмотрим $X = \max_{k=l..r} res_{i-k}$
- Заметим, что $res_i \geqslant X - 1$
- Если максимум достигается при $k = z$, тогда $res_{i-z} = X$ или $res_{i-z} = X - 1$
- Тогда $res_i = \max(A, B)$, где

$$A = \max_{\substack{k=l..r \\ res_{i-k}=X}} (X + sign(sum_i - sum_{i-k}))$$

$$B = \max_{\substack{k=l..r \\ res_{i-k}=X-1}} (X - 1 + sign(sum_i - sum_{i-k}))$$

Пересчет динамики за $O(1)$ или за $O(\log n)$

- $X = \max\limits_{k=l..r} res_{i-k}$

  $A = \max\limits_{\substack{k=l..r \\ res_{i-k}=X}} (X + sign(sum_i - sum_{i-k}))$

  $B = \max\limits_{\substack{k=l..r \\ res_{i-k}=X-1}} (X - 1 + sign(sum_i - sum_{i-k}))$

  $res_i = \max(A, B)$
- Для нахождения X нужно находить максимум на массиве в движущемся слева направо окне. Это можно делать за $O(\log n)$ с помощью кучи или амортизационно за $O(1)$ с помощью очереди.

Пересчет динамики за $O(1)$ или за $O(\log n)$

- $X = \max\limits_{k=l..r} res_{i-k}$

  $A = \max\limits_{\substack{k=l..r \\ res_{i-k}=X}} (X + sign(sum_i - sum_{i-k}))$

  $B = \max\limits_{\substack{k=l..r \\ res_{i-k}=X-1}} (X - 1 + sign(sum_i - sum_{i-k}))$

  $res_i = \max(A, B)$
- Максимум для A и B достигается там, где достигается минимум для sum_{i-k} при соответствующих ограничениях на k. Это также сводится к нахождению максимума на массиве в движущемся окне.

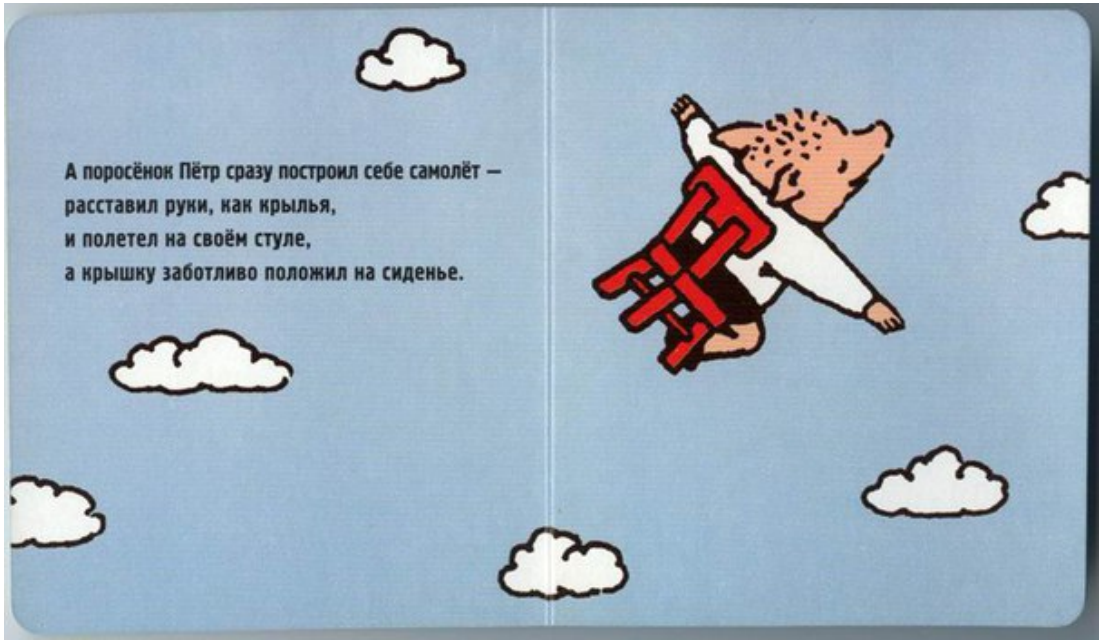
Решение за $O(n)$

Считаем динамику res_i — ответ для префикса длины i. Для пересчета динамики по формулам выше за $O(1)$ нужно поддерживать

- Q — очередь с минимумом, построенную на всем массиве res,
- $\forall i : Q_i$ — очередь с минимумом, построенную на элементах sum_j, где $res_j = i$.

Задача D. Полет мечты

Задача D. Полет мечты



- Идея задачи — Антон Гардер, Антон Евдокимов
- Подготовка тестов — Антон Гардер
- Разбор задачи — Антон Гардер

Постановка задачи

Входные данные:

- Дана точка на поверхности Земли.
- Нужно переместиться из неё на d км. на юг, на запад и на север. Как итог вернуться в начало.
- Нельзя подлетать близко к южному полюсу, $d \geqslant 1$.
- Найти подходящее d.

Решение задачи

- Если старт — северный полюс, то d — любое.
- Иначе, наш путь выглядит следующим образом:
  - Спускаемся на юг на d километров в точку t;
  - Пролетаем d километров вокруг Земли и возвращаемся в точку t;
  - Поднимаемся на север на d километров из точки t обратно на старт.
- Как найти d?

Решение задачи

- Если старт выше экватора, спустимся сначала до экватора.
- Начинаем спускаться вниз.
- Длина пройденного пути вниз увеличивается, а длина окружности, по которой мы будем совершать оборот вокруг Земли, уменьшается.
- Двоичным поиском находим точку, в которой эти длины совпадут.
- Длина пути от старта до этой точки — искомое d.
- Ограничения в задаче заданы так, что данное решение работает без проблем с точностью.

Задача E. Занимательное дежурство

Задача E. Занимательное дежурство

- Идея задачи — Анна Малова
- Подготовка тестов — Григорий Шовкопляс
- Разбор задачи — Григорий Шовкопляс

Постановка задачи

- Дан набор латинских букв.
- Каждый ход:
  - Удаляется две одинаковых буквы.
  - Добавляется любая новая.
- Кто не может сделать ход, проигрывает.
- Нужно узнать, кто выиграет при правильной стратегии.

Общие наблюдения

- Порядок букв не важен.
- Некоторые буквы встречаются в строке чётное количество раз N_{even}, а некоторые нечётное N_{odd}.
- Каждое состояние игры можно охарактеризовать числами N_{even} и N_{odd}.
- Проигрышная позиция — N_{odd} совпадает с числом букв в наборе.
- При удалении двух букв четность не меняется.
- При добавлении одной меняется.
- Игра не содержит циклов.

Динамическое программирование

- Стандартная задача динамического программирования для подсчёта выигрышных позиций.
- Состояние — общее число букв в наборе K, а также N_{even} и N_{odd}.
- Переходы:
  - $\langle K, N_{even}, N_{odd} \rangle \longrightarrow \langle K - 1, N_{even} - 1, N_{odd} + 1 \rangle$.
  - $\langle K, N_{even}, N_{odd} \rangle \longrightarrow \langle K - 1, N_{even} + 1, N_{odd} - 1 \rangle$.
- Полезное замечание: $N_{even} = 26 - N_{odd}$. Можно отказаться от третьего параметра.

Альтернативное решение

Формула:

- $N_{odd} = K$: выигрывает Дима.
- N_{odd} — четное: Гриша выиграет только, если $N_{odd} = K - 2$.
- N_{odd} — нечетное: Дима выиграет только, если $N_{odd} < K - 2$.

Доказательство формулы следует из предыдущего решения и оставляется слушателю в качестве упражнения.

Задача F. Рукопожатия

Задача F. Рукопожатия

- Идея задачи — Глеб Евстропов
- Подготовка тестов — Глеб Евстропов
- Разбор задачи — Глеб Евстропов

Постановка задачи

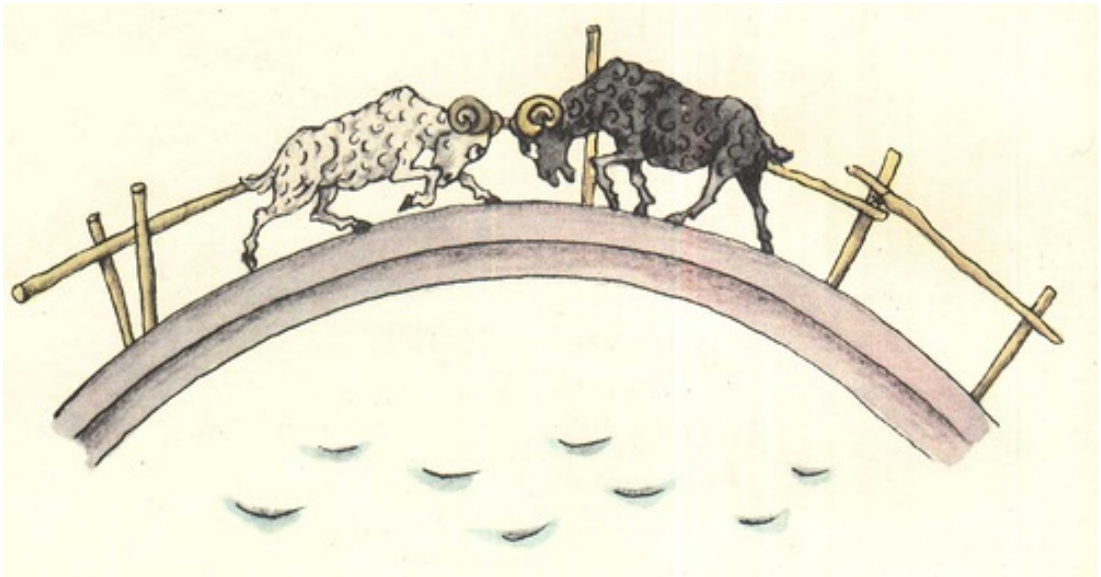
- Загадан некоторый граф $G = (V, E)$.
- Для каждой вершины i дана её степень в подграфе на первых i вершинах.
- Требуется найти максимально возможную $deg(v)$ для подходящего графа.

Решение задачи

- Для каждой вершины i будем узнавать максимально возможную степень в подграфе на вершинах $i, \ldots, n$.
- В выделенном подграфе вершина i может иметь рёбра только с вершинами j, у которых степень в подграфе на вершинах $1, \ldots, j$ больше нуля.
- Так как нас требуют максимизировать степень вершины, мы соединяем вершину i со всеми доступными вершинами.
- Для определения числа таких рёбер мы перебираем вершины с конца, подсчитывая количество вершин с ненулевым числом рёбер в соответствующем подграфе.

Задача G. Фишки

Задача G. Фишки



- Идея задачи — Дмитрий Филиппов
- Подготовка тестов — Дмитрий Филиппов
- Разбор задачи — Дмитрий Филиппов

Постановка задачи

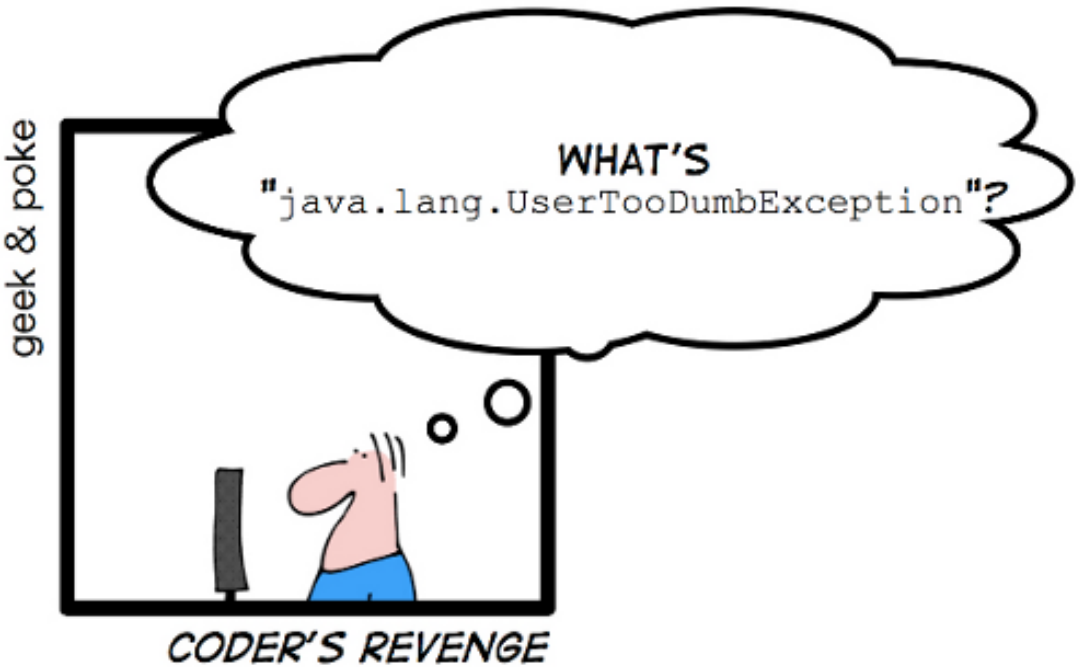
- На поле $n \times n$ находятся две фишки, но неизвестно в каких клетках.
- Каждую из фишек можно попытаться передвинуть в одну из 4 сторон.
- В ответ программа жюри сообщает, удалось ли совершить перемещение.
- Нужно найти начальное положение фишек не более чем за $6n$ ходов.

Решение задачи

- Сдвинем первую фишку до упора вверх, потом вправо.
- После этого первая фишка либо в последней строке, либо в последнем столбце.
- Затем вторую фишку сдвинем до упора вниз, потом влево, потом опять вниз.
- Теперь она точно в клетке $(1, 1)$.
- После этого можно первую фишку сдвинуть до упора вверх, потом вправо, и она окажется в (n, n).
- Зная все передвижения, которые были успешными, можно легко восстановить начальное положение фишек.

Задача H. Отчёт об ошибках

Задача H. Отчёт об ошибках



- Идея задачи — Юрий Петров
- Подготовка тестов — Николай Будин
- Разбор задачи — Николай Будин

Постановка задачи

- Дана распечатка стека вызовов.
- Нужно найти программу с минимальной возможной сложностью, способную сгенерировать такую распечатку.

Решение задачи

- Заметим, что функция, стоящая на первом месте в распечатке обязательно должна содержать в себе ошибку.
- Пусть мы зафиксировали вторую функцию, в которой происходит ошибка.
- Тогда посмотрим на все пары последовательных функций, присутствующие в распечатке.
- Если функция, из которой происходит вызов, может упасть из-за ошибки, не будем учитывать эту пару.
- Тогда количество учтённых уникальных пар, является ответом.

Решение задачи

- Для каждой функции предподсчитаем количество различных функций, вызываемых из нее в распечатке.
- Тогда, при паре зафиксированных функций с ошибками, ответом является общее количество уникальных пар минус предподсчитанные значения для функций с ошибками.
- Переберем вторую функцию с ошибкой и выберем ту, при которой ответ минимален.

Задача I. Обмен валюты

Задача I. Обмен валюты

- Идея задачи — Роман Гусарев
- Подготовка тестов — Роман Андреев
- Разбор задачи — Роман Андреев

Постановка задачи

- Дан массив чисел c.
- Нужно найти два таких числа из массива, чтобы $|c_i/c_j - p|$ было минимально.

Решение задачи

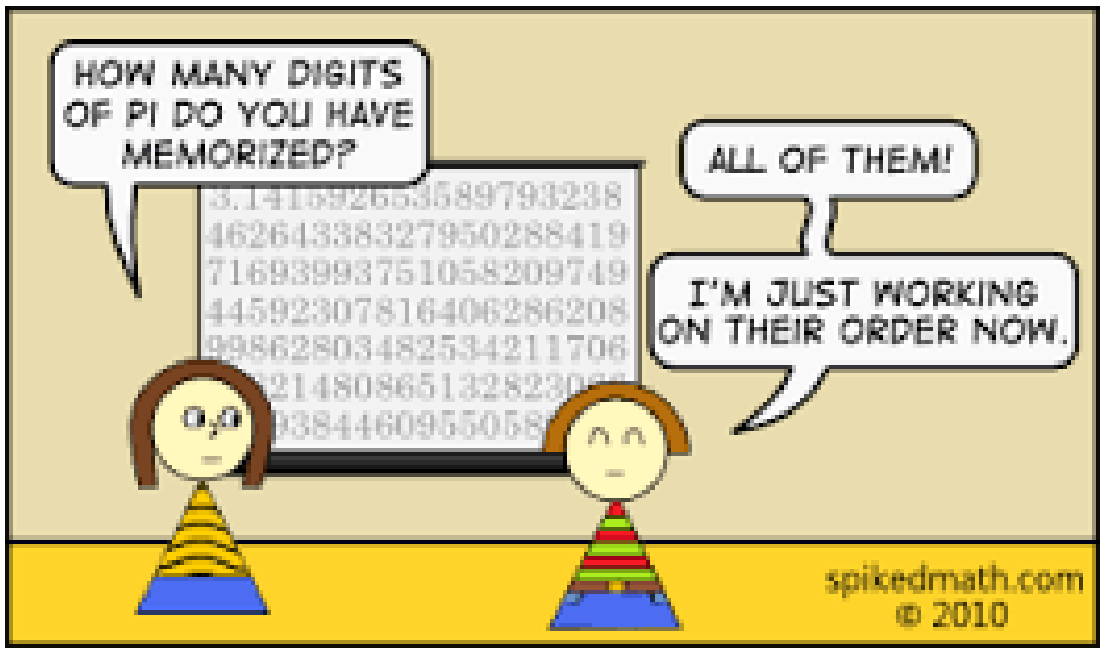
- Отсортируем массив c.
- Переберём j, то есть какое из чисел будет стоять в знаменателе.
- Чтобы найти кандидатов на c_i нужно всего лишь найти значение $c_j \times p$ в нашем массиве (например, бинарным поиском) и попробовать взять первое число, более этого значения и первое число меньшее этого значения.

Технический момент

- Для того, чтобы корректно выбрать минимальный ответ, хватает 64-битного типа.
- Хотим сравнить $|a/b - p|$ и $|c/d - p|$.
- Если $||\lfloor a/b \rfloor - p| - |\lfloor c/d \rfloor - p|| > 2$, то можно сравнить только $|\lfloor a/b \rfloor - p|$ с $|\lfloor c/d \rfloor - p|$.
- Иначе, это означает, что $||a/b - p| - |\lfloor a/b \rfloor - p|| \leq 1$ и $||c/d - p| - |\lfloor a/b \rfloor - p|| \leq 1 + 2$.
- Тогда остается только сравнить $|a - pb| - b \times |\lfloor a/b \rfloor - p|$ с $|c - pd| - d \times |\lfloor a/b \rfloor - p|$, а все эти числа влезают в 64 тип данных.

Задача J. Простая последовательность цифр

Задача J. Простая последовательность цифр



- Идея задачи — Демид Кучеренко
- Подготовка тестов — Демид Кучеренко
- Разбор задачи — Демид Кучеренко

Постановка задачи

- Сгенерировать последовательность цифр по правилу из условия.
- Вычеркнуть k цифр из последовательности, чтобы оставшееся число было максимально возможным.

Решение задачи

- Генерация первых n простых чисел.
- Стандартный алгоритм: решето Эратосфена.

Решение задачи

- Будем хранить позиции, на которых встречается каждая из цифр.
- Например, для простоты, в десяти векторах.
- Для каждой цифры храним номер самой левой позиции этой цифры, которая не обработана.
- Обрабатывать последовательность будем слева направо, а значит указатель сдвигается только в одну сторону.
- Храним количество цифр, которое ещё нужно дописать.

Решение задачи

- Пусть мы хотим дописать новую цифру в ответ.
- Действуем жадно. Пробуем дописать как можно большую цифру, если это возможно.
- Это возможно, если количество цифр после текущей не меньше, чем количество цифр, которое еще нужно дописать.
- Если текущую цифру дописать невозможно, то переходим к меньшей цифре.
- Свигаем указатель, если он указывает на цифру левее, чем последняя дописанная в ответ.
- Таким образом мы никакую цифру не рассматриваем дважды.

Задача K. Робот

Задача K. Робот

- Идея задачи — Евгений Замятин
- Подготовка тестов — Евгений Замятин
- Разбор задачи — Евгений Замятин

Постановка задачи

Входные данные:

- Дано поле $n \times n$. В одной из клеток находится робот. Некоторые клетки выколоты.
- Дана строка s, состоящая из символов «U», «D», «L», «R» — программа робота.
- Требуется посчитать количество подстрок, исполнив которые, робот не пройдет по выколотой клетке и не упрётся в границу поля.

Медленное решение задачи

- Пусть робот изначально находится в клетке (X, Y).
- Будем решать задачу отдельно для каждой выколотой клетки (x, y): ближайшее вхождение выколотой клетки на суффиксе с позиции i.

Ускорение решения задачи

- Предподсчитаем массивы смещений, то есть массив dx_i — смещение робота от стартовой позиции после выполнения i команд по оси X и аналогичный массив dy_i.
- Посмотрим на суффикс команды с позиции i, тогда нужно найти минимальное $t \geq i$, такое что $(X + dx_t - dx_i, Y + dy_t - dy_i) = (x, y)$.
- Это равносильно нахождению $t \geq i$, такое что $(X + dx_t, Y + dy_t) = (x + dx_i, y + dy_i)$.

Ускорение решения задачи

- Отсортируем массив смещений.
- Тогда для фиксированной выколотой клетки и каждого суффикса с позиции i мы можем найти соответствующее t методом двумя указателей по отсортированному массиву смещений.
- Границы поля следует обрабатывать отдельно аналогичным образом только по фиксированной координате.
- Время работы: $O(t \log t + (4 + m) \cdot t)$, где m — количество выколотых клеток.

Вопросы?