ВКОШП 2010:
Разбор задач

21 ноября 2010 года

Задача «Арифметическая прогрессия»

$$17, 239, 461, \ldots$$

Над задачей работали

- Идея задачи: Андрей Станкевич
- Текст условия: Сергей Мельников
- Тесты, проверяющая программа и др.: Антон Ахи
- Решения: Владимир Ульянцев, Сергей Мельников
- Текст разбора: Сергей Мельников

Формулировка задачи

- Дано $2n$ чисел, из них надо выбрать n чисел, образующих арифметическую прогрессию.
- Гарантируется, что это можно сделать.

Идея решения

- Если выбрать случайный элемент исходной последовательности, то с вероятностью $\frac{1}{2}$ он будет принадлежать арифметической прогрессии.
- Если выбрать два случайных элемента a и b, то оба будут принадлежать арифметической прогрессии с вероятностью $\frac{1}{4}$.
- Разность арифметической будет делителем $a - b$.
- Переберем все делители числа $a - b$ за $O(\sqrt{n})$.

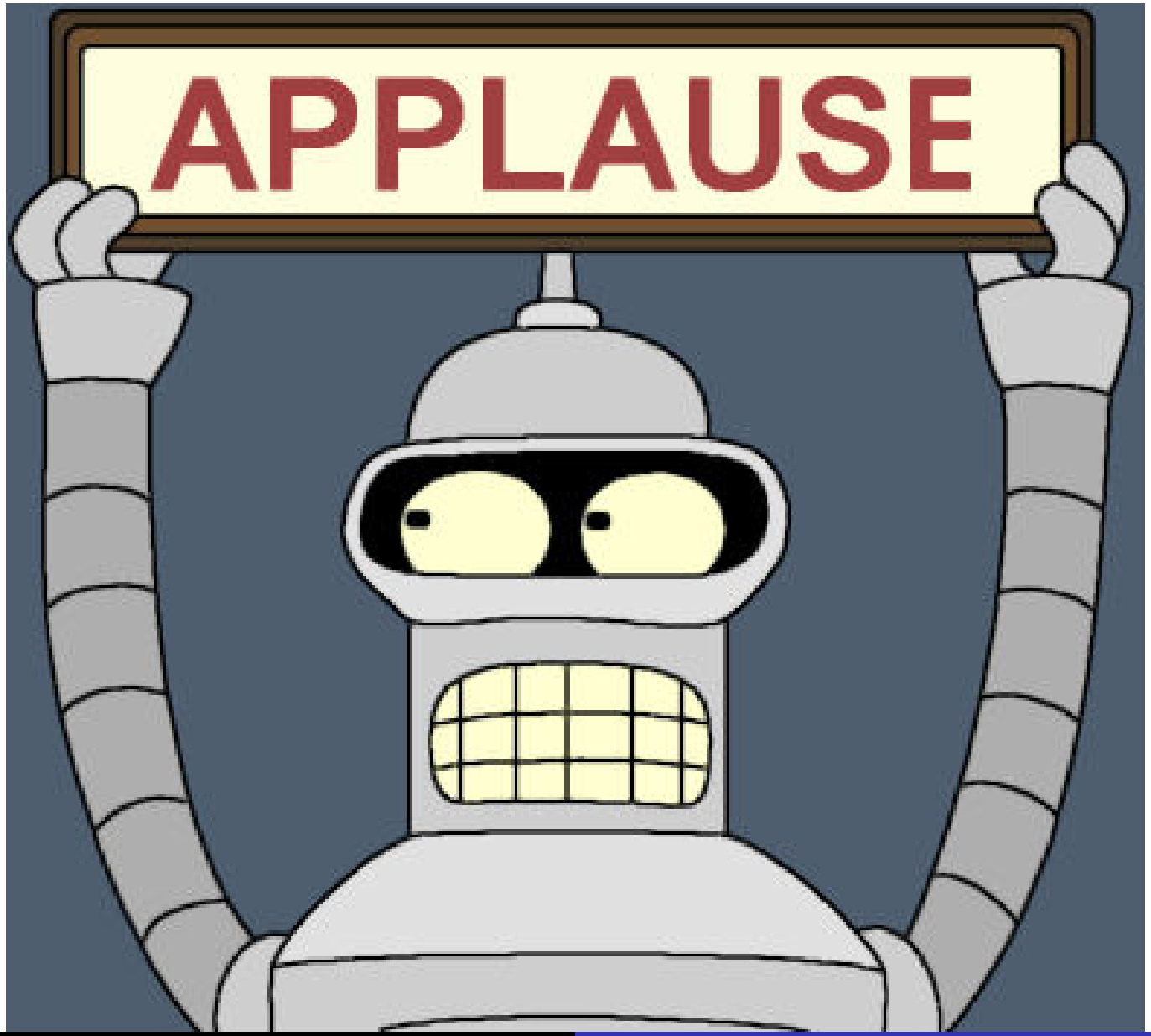
Идея решения

- Знаем элемент a и разность d, проверим какая максимальная длина арифметической прогрессии может быть.
- Для этого отсортируем исходный массив.
- Можно проверять, принадлежит ли элемент исходной последовательности, двоичным поиском.
- Будем уменьшать число a на d, пока оно будет принадлежать исходной последовательности.
- Будем увеличивать a на d, пока оно будет принадлежать исходной последовательности.
- Если есть не менее n элементов, то это ответ.

Итого

- Выбираем случайные элементы, с вероятностью $\frac{1}{4}$ мы выбираем элементы из арифметической прогрессии.
- Перебор всех делителей $O(\sqrt{n})$
- У чисел до 10^9 не более нескольких сотен делителей.
- Проверка ответа $O(n \log n)$
- Вопросы?

Задача «Бендер»



Над задачей работали

- Идея задачи: Антон Банных
- Текст условия: Антон Ахи
- Тесты, проверяющая программа и др.: Владимир Ульянцев
- Решения: Антон Банных, Михаил Майоров
- Текст разбора: Антон Ахи

Формулировка задачи

- Робот Бендер строит последовательность x_i из n элементов
- $x_1 = c$; $x_i = a \cdot x_{i-1} + b$ для $i > 1$
- На i-ом шаге меняются местами стаканчики на позициях $(x_i \mod k) + 1$ и $(x_{i+1} \mod k) + 1$
- Найти a, b и c такие, чтобы шарик под стаканчиком на позиции r после n дейтсвий оказался под стаканчиком на позиции l

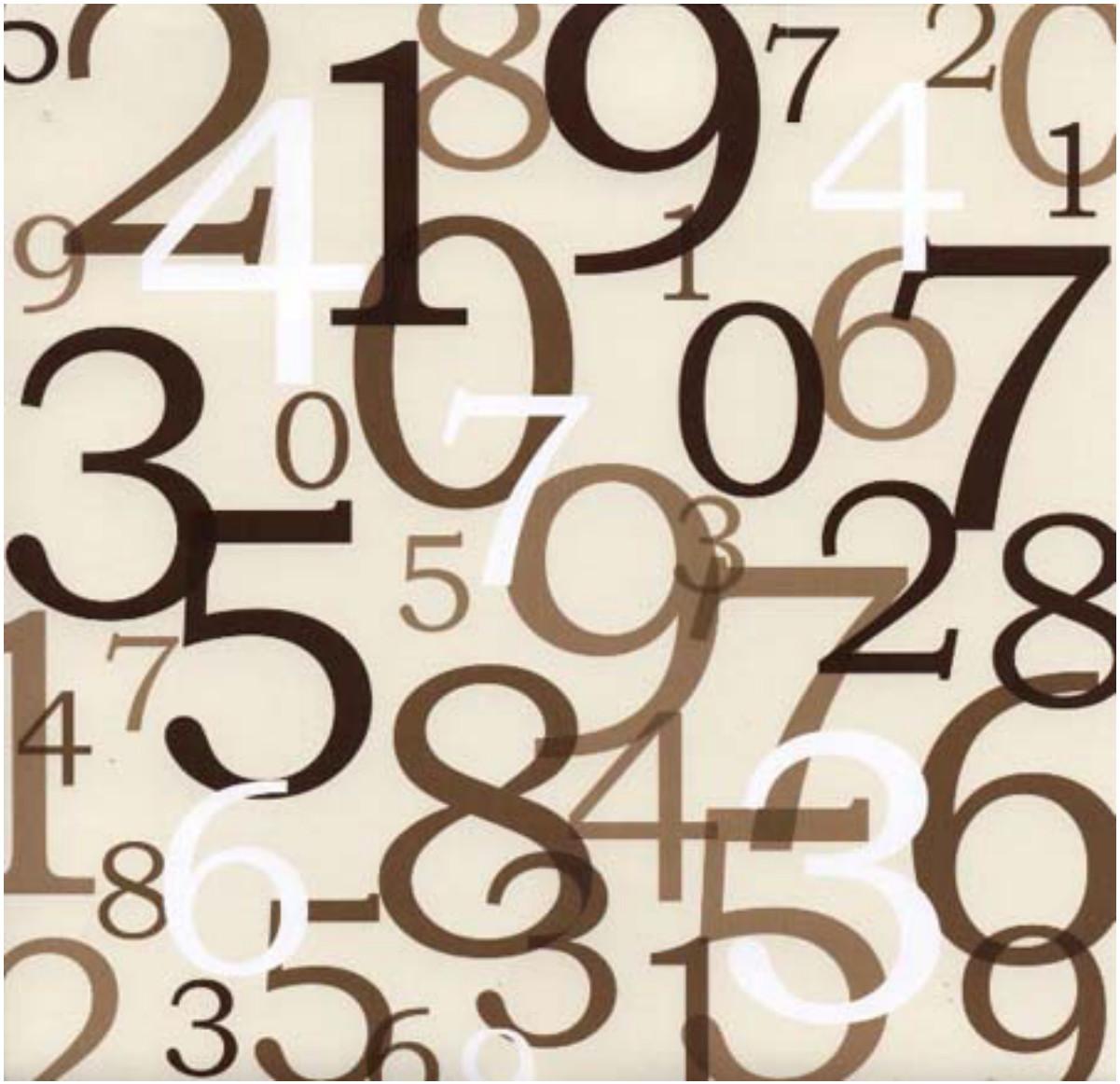
Идея решения

- Так как k невелико и важно лишь значение $x_i \mod k$, то можно перебрать все возможные a, b и c от 0 до $k - 1$ и проверить их
- Проверку осуществлять за $O(n)$, совершая все обмены
- Если ответ найдет, вывести его, иначе вывести «`Impossible`»

Итого

- Решение совершает $O(n \cdot k^3)$ операций
- Вопросы?

Задача «Различные числа»



Над задачей работали

- Идея задачи: Виталий Аксенов
- Текст условия: Антон Банных
- Тесты, проверяющая программа и др: Сергей Мельников
- Решения: Сергей Мельников, Сергей Поромов, Андрей Комаров
- Текст разбора: Павел Кротков

Формулировка задачи

- Даны два целых числа: a и b
- Требуется определить количество различных чисел среди следующих:
  - $a + b$, $b + a$
  - $a - b$, $b - a$
  - $a * b$, $b * a$
  - $\frac{a}{b}$, $\frac{b}{a}$
  - a^b, b^a

Идея решения

- Проверим все эти числа и посчитаем различные
- Если какое-то число не считается — не будем его вычислять, мы сможем сравнить это число с другими и не зная его точного значения

Что значит не вычисляется?

- Модуль числа слишком большой
- Модуль числа слишком маленький

Что делать с очень маленьким числом?

- Очень маленьким(меньше 10^{-18}) может получиться только число a^b или b^a
- Если a^b или b^a очень мало и одно из чисел a и b меньше нуля, а другое больше, то a^b и b^a различны
- Если a^b или b^a очень мало и числа a и b меньше нуля, то $a^b = b^a$ только в случае равенства a и b

Что делать с очень большим числом?

- Очень большим(больше 10^{18}) может получиться только число a^b или b^a
- Если a^b или b^a очень велико, то $a^b = b^a$ только в случае равенства a и b

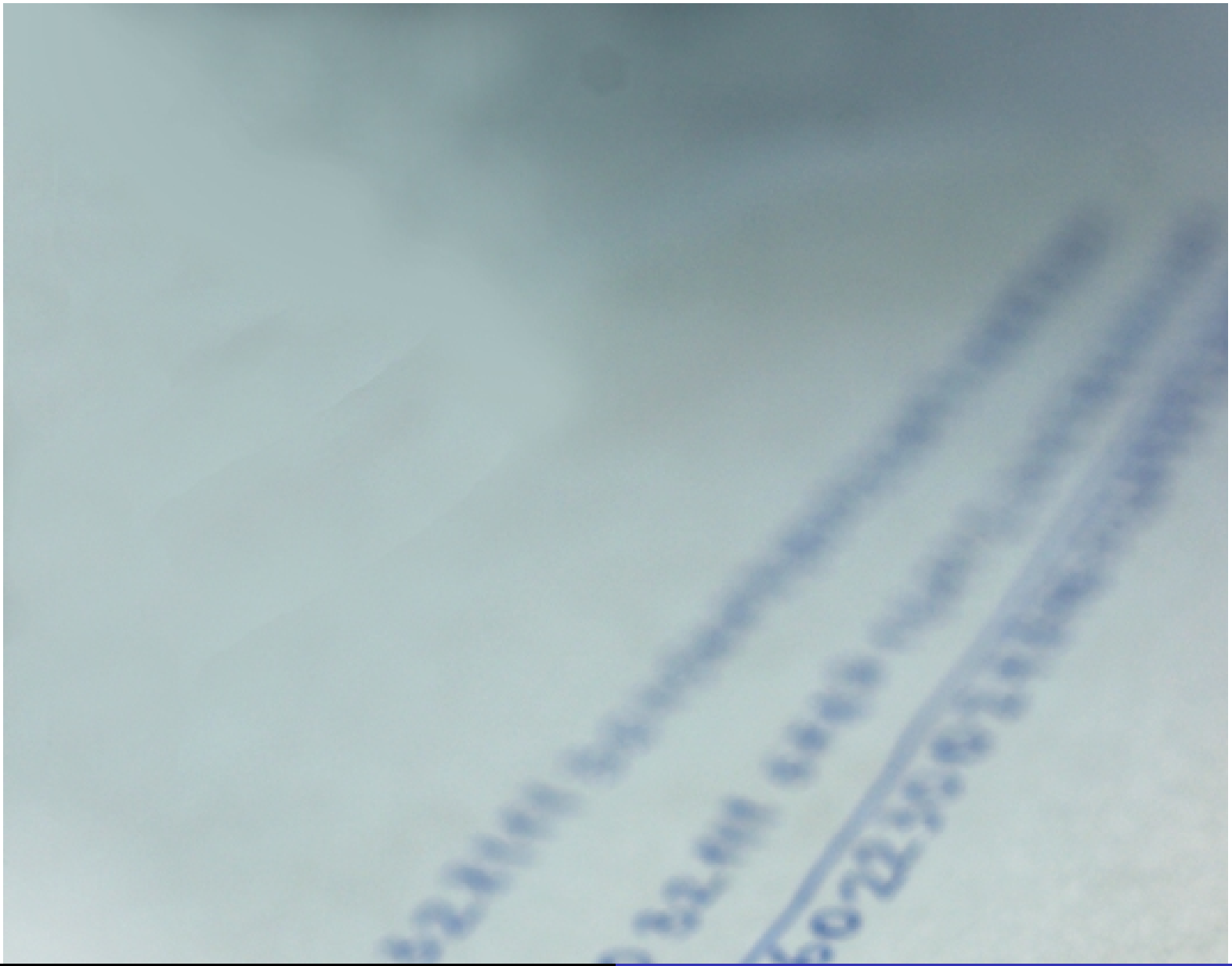
Итого

- Считаем значения, которые мы можем посчитать, и ищем количество различных.
- Если мы получили очень большие или маленькие числа, то скажем, сколько их
- Складываем ответы

Спасибо за внимание

- Вопросы?

Задача «Игра»



Над задачей работали

- Идея задачи: Виктор Матюхин
- Текст условия: Антон Банных
- Тесты, проверяющая программа и др.: Олег Давыдов
- Решения: Сергей Мельников
- Текст разбора: Нияз Нигматуллин

Формулировка задачи

- Дано одно целое число из N цифр
- В игру играют два игрока
- Игроки по очереди выписывают цифры в конец второго числа
- Выигрывает тот игрок, после хода которого второе число станет больше первого

Идея решения

- Если после хода какого-нибудь игрока цифр во втором числе стало больше чем в первом, то этот игрок выиграл
- Рассмотрим два случая: N четное и нечетное

N четное

- Если игра дойдет до $N + 1$ цифры, то выиграет первый
- Первый игрок хочет обеспечить игру до $N + 1$ цифры: первые N цифр второго числа меньше чем первое число
- Второй игрок хочет выиграть на N-том ходе: полученное число больше заданного
- Первый хочет минимизировать число, а второй — максимизировать

N четное

- На первом ходу первый игрок ставит цифру 1
- На каждом своем следующем ходу, первый ставит цифру 0, а второй — цифру 9
- Получаем число $19090\ldots09$
- Если полученное число из N цифр больше чем заданное число, то выиграл второй игрок, иначе — первый

N нечетное

- Если игра дойдет до $N+1$ цифры, то выиграет второй
- Второй игрок хочет обеспечить игру до $N+1$ цифры: первые N цифр полученного числа меньше чем заданное число
- Первый игрок хочет выиграть на N-ом ходе: полученное число больше заданного
- Первый хочет максимизировать число, а второй — минимизировать

N нечетное

- На каждом своем следующем ходу, первый ставит цифру 9, а второй — цифру 0
- Получаем число $9090\ldots09$
- Если полученное число из N цифр больше чем заданное число, то выиграл первый игрок, иначе — второй

Итого

Вопросы?

Задача «Огромная парковка»

Над задачей работали

- Идея задачи: Георгий Корнеев
- Текст условия: Антон Ахи
- Тесты, проверяющая программа и др.: Сергей Поромов
- Решения: Сергей Мельников, Антон Банных, Нияз Нигматуллин, Сергей Поромов
- Текст разбора: Антон Ахи

Формулировка задачи

- На огромной парковке много автомобилей и столбов, а также одно свободное место
- Хочется вывести один из автомобилей с парковки
- Разрешается сдвигать любую машину на соседнее место, если оно свободно

Идея решения

- Обход в ширину по состояниям
- Состояние — позиция свободного места и машины, которую хочется вывести
- Число состояний $O(n^2 \cdot m^2)$

Как переходить между состояниями

- Можно «переместить» свободное место на соседнюю позицию, если там не столб
- Если эта позиция была занята автомобилем, который необходимо вывести, то необходимо изменить его позицию
- Переход между состояниями совершается за $O(1)$

Итого

- Алгоритм совершает $O(m^2 \cdot n^2)$ операций и требует $O(m^2 \cdot n^2)$ памяти
- Вопросы?

Задача «День рождения»

Над задачей работали

- Идея задачи: Глеб Евстропов
- Текст условия: Олег Давыдов
- Тесты, проверяющая программа и др.: Сергей Мельников
- Решения: Сергей Мельников, Антон Банных, Олег Давыдов
- Текст разбора: Олег Давыдов

Формулировка задачи

- Дано n друзей, из них надо выбрать некоторое подмножество
- У каждого из друзей есть ограничения в обе стороны на размер доли, то есть, на количество выбранных друзей
  - Если i-й друг может заплатить от a_i до b_i, то он может быть в группе приглашённых размером от $\lceil \frac{S}{b_i} \rceil - 1$ до $\lfloor \frac{S}{a_i} \rfloor - 1$ человек
- Нужно максимизировать суммарное веселье

Идея решения

- Если известно число k (количество пришлашённых друзей), то задача решается очень просто: из всех друзей, которых мы можем взять в группу размера k, надо выбрать k друзей с максимальным весельем
- Можно перебрать все различные k
- Но ограничения не позволяют перебрать всех друзей для каждого k, так что надо научиться быстро узнавать сумму k максимальных по веселью допустимых друзей

Как это сделать

- Отсортируем всех друзей в порядке уменьшения веселья: теперь всегда надо брать просто k первых допустимых друзей
- Будем перебирать k в порядке увеличения и следить за множеством допустимых друзей
  - Тогда каждый друг будет один раз добавлен в наше множество и один раз удалён из него
  - Чтобы это сделать, достаточно выписать для каждого друга два события: его появление и его уход; упорядочим эти события по их «времени», то есть по граничному размеру группы
  - Такой подход (выписать события и их отсортировать) часто встречается в задачах по программированию
- Осталось научиться брать сумму k первых друзей из допустимого множества. Для этого можно применить какую-нибудь структуру данных

Структура данных

- Такая структура данных — любое сбалансированное дерево (например, декартово). Ключом в дереве будет номер друга
- Каждая вершина дерева должна хранить сумму веселья и количество вершин в поддереве. Эти значения нужно будет пересчитывать в операциях, изменяющих дерево
- Помимо обычных операций (добавление и удаление элемента) понадобится ещё одна — подсчитать сумму веселья на первых k вершинах. Именно для этого мы стали хранить сумму на поддереве и его размер.

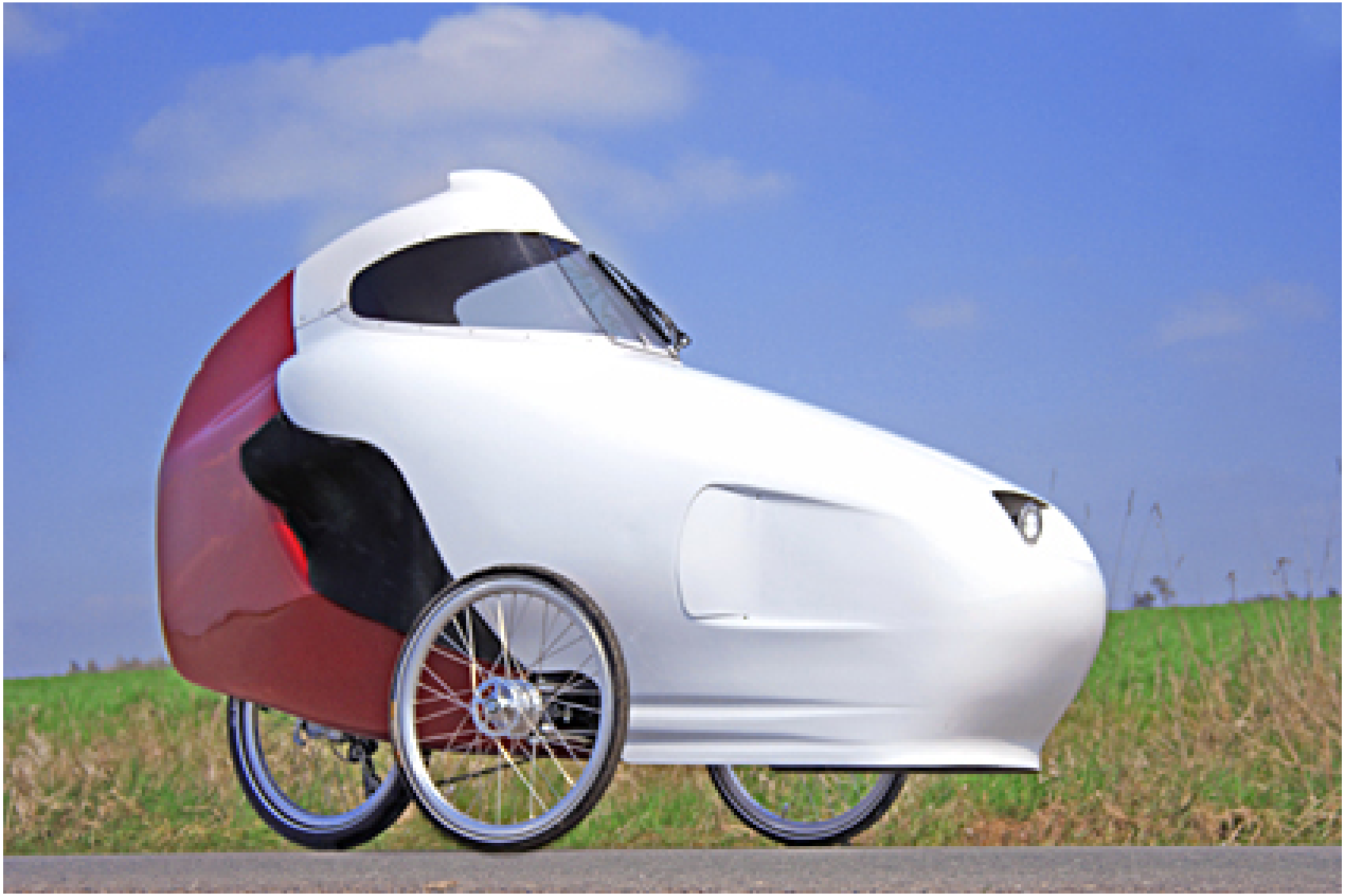
Альтернативный вариант

- Такая структура данных — два дерева отрезков
- Оба дерева отрезков будут считать суммы на массиве из n элементов
- В первом дереве отрезков будем хранить 1 или 0 для каждого друга, в зависимости от того, есть ли друг в множестве
- Во втором дереве для друга из множеста будем хранить его веселье f_i, а для друга не из множества — ноль.
- Сумма веселья первых k допустимых друзей считается в два этапа:
  - С помощью первого дерева подсчитаем номер k-го друга из множества
  - Зная нужный индекс, возьмём сумму на отрезке во втором дереве

Итого

- Мы получили решение, работающее за $O(n \log n)$ и использующее $O(n)$ памяти
- Различные решения жюри занимают от трёх до шести килобайт, что довольно много для олимпиадной задачи
- Домашнее задание: придумать решение за $O(n\sqrt{n})$
- Вопросы?

Задача «Гонка со временем»



Над задачей работали

- Идея задачи: Юрий Петров
- Текст условия: Сергей Поромов
- Тесты, проверяющая программа и др.: Юрий Петров
- Решения: Юрий Петров, Сергей Поромов
- Текст разбора: Юрий Петров

Формулировка задачи

- Школьники
  - Представлены точками на прямой
  - Движутся к точке 0
  - Движутся с постоянными скоростями
- Веломобиль
  - Представлен точкой на прямой
  - Движется с постоянной скоростью
  - Обязательно подвозит школьника до точки 0

Идея решения

- Зафиксируем порядок подвоза школьников
- Если для очередного школьника:
  - успеваем подвезти и
  - это выгодно (быстрее, чем он сам придёт),

  его нужно подвезти
- Перейдём к следующему школьнику

Выбор порядка для двух школьников

- Позиции и скорости школьников (x_1, v_1) и (x_2, v_2)
- Пусть требуется подвезти сначала первого
- Время встречи первого школьника с веломобилем $t_m = \frac{x_1}{v_1+v}$
- Позиция встречи $x_m = x_1 - t_m \cdot v_1 = t_m \cdot v$
- Время возврата будет равно времени до встречи

Выбор порядка для двух школьников

- Аналогично, окончательное время для обоих школьников будет

$$t_x = 2\frac{x_2 - 2t_m v_2}{v_2 + v} + 2t_m = 2\frac{x_2 - 2t_m v_2 + t_m v_2 + t_m v}{v_2 + v} =$$

$$2\frac{x_2 - t_m v_2 + t_m v}{v_2 + v} = \frac{2}{(v_1 + v)(v_2 + v)}(x_2 v_1 + v x_2 - x_1 v_2 + x_1 v)$$

- Заметим, что если заменить порядок на обратный, получится аналогичная формула
- Теперь очевидно, что первый порядок лучше второго тогда и только, когда $\frac{x_1}{v_1} < \frac{x_2}{v_2}$

Оставшаяся часть решения

- Единственна перестановка, которую нельзя улучшить — отсортированная по $\frac{x}{v}$
- Теперь каждого очередного школьника, который
  - медленнее веломобиля и
  - ещё не пришёл в 0,

  нужно подвезти

Итого

- Время работы есть $O(n \log n)$
- Вопросы?

Задача «Скоростной диаметр для кольцевой дороги»

Над задачей работали

- Идея задачи: Владимир Ульянцев
- Текст условия: Владимир Ульянцев
- Тесты, проверяющая программа и др.: Сергей Поромов
- Решения: Сергей Поромов, Сергей Мельников
- Текст разбора: Сергей Поромов

Формулировка задачи

- Дан монотонный многоугольник
- Необходимо построить вертикальную магистраль длиной ровно d

Идея решения

- Каждая вертикальная прямая пересекает многоугольник по не более, чем одному отрезку
- Интересные точки — всевозможные x_i
- На каждом отрезке между двумя интересными точками расстояние между верхней и нижней цепью изменяется линейно и потому либо не более одного подходящего места, либо бесконечно много

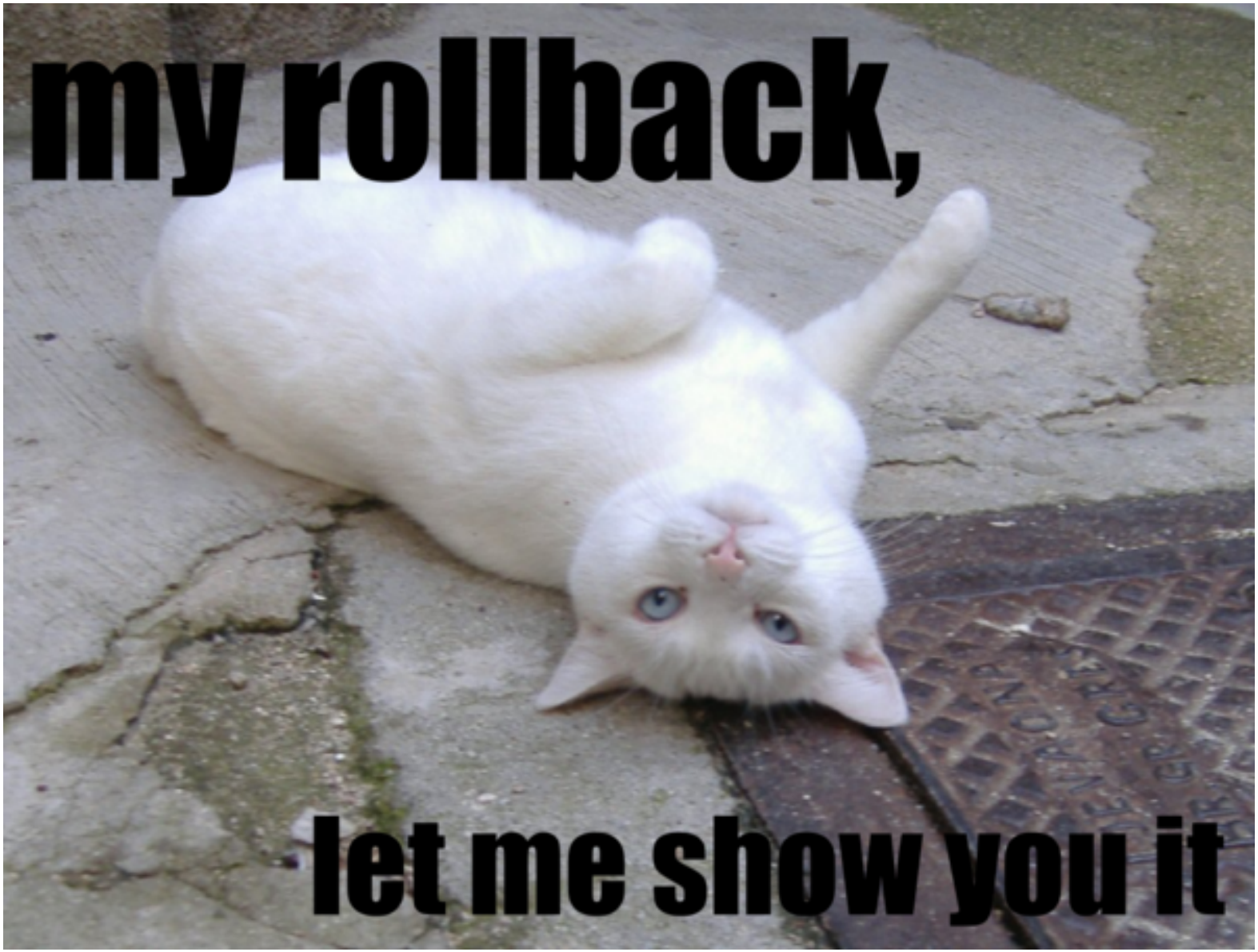
Решение

- Вариант 1 : отсортируем все x_i — решение за $O(n \log n)$
- Вариант 2 : найти самую левую и самую правую точку и бежать двумя указателями по верхней и нижней цепи — решение за $O(n)$
- Для каждого отрезка найти расстояние между верхней и нижней цепью — d_l и d_r
- Если $d_l \leq d < d_r$ или $d_r \leq d < d_l$, то один вариант проведения магистрали
- Если $d_l = d = d_r$, то ответ — бесконечность

Итого

- Время работы есть $O(n \log n)$
- Вопросы?

Задача «Откат»



Над задачей работали

- Идея задачи: Антон Банных
- Текст условия: Антон Банных
- Тесты, проверяющая программа и др.: Антон Банных
- Решения: Антон Банных, Антон Ахи
- Текст разбора: Антон Банных

Формулировка задачи

- Дано n целых чисел $a_1, a_2, \ldots, a_n$ и q запросов
- Запрос — числа l и k
- Ответ на запрос — минимальное число r, такое что $a[l..r]$ содержит k различных чисел
- Онлайн-ответы на запросы

Первый шаг решения

- Упрощение: $l = 1$
- Пусть $b_i = 1$, если число a_i не встречалось ранее
- В противном случае $b_i = 0$
- Ответ на запрос — минимальное r, такое что $\sum_{i=1}^{r} b_i = k$
- Построим дерево отрезков на сумму на массиве b
- Ответ на запрос за $O(\log n)$ при помощи «спуска по дереву»
  - В вершине хранится сумма sum на отрезке $[L..R]$
  - В каждой вершине известно, куда идти (в лебого ребенка, или в правого)

Дальнейшее развитие идеи

- Что делать, если $l = 2$?
- Положим b_1 равным 0
- Пусть j — первое вхождение a_1 в $a[2..n]$
- Положим b_j равным 1
- В массиве b изменилось не более двух элементов
- Легко изменить дерево отрезков соответствующим образом
- Время на изменение $O(\log n)$

Второй шаг

- Умеем отвечать на запрос и переходит к следующему суффиксу за $O(\log n)$
- Предположим, что на запросы можно отвечать в произвольном порядке
- Отсортируем запросы по l
- Будем перебирать суффиксы в порядке уменьшения длины и отвечать на все запросы для данного суффикса
- Время работы $O(n \log n)$
- Используемая память O(n)

Замечания по реализации

- Для эффективного перехода необходимо для i-го символа знать первое вхождение числа a_i в $a[i+1..n]$
- Идем слева направо
- Для каждого числа c помним его последнее вхождение $d[c]$
- При обработке a_i записываем его первое вхождение в $a[i+1..n]$ и обновляем $d[a_i] = i$
- Время работы $O(n)$
- Используемая память $O(n+m)$
- Можно совместить с основной частью, если перебирать суффиксы в порядке увеличения длины

Решение основной задачи

- Основная проблема — ответ на запрос «онлайн»
- Для сохранения всех деревьев отрезков потребуется слишком много памяти
- Наблюдение: при переходе к следующему суффиксу изменяется порядка $2 \log n$ вершин дерева отрезков
- Решение: персистентное дерево отрезков

Персистентное дерево отрезков

- Вместо изменения значения в вершине создаем новую
- Один запрос вида `set` создает $\log n$ вершин по пути от корня к листу.
- Старое дерево отрезков не меняется
- Новый корень соответствует измененному дереву отрезков
- Реализация запроса `get` не меняется
- Вызов *get* от разных корней — все равно, что от разных деревьев

Решение основной задачи

- Переберем все суффиксы
- Для каждого суффикса запомним соответствующий корень дерева отрезков
- Для того, чтобы ответить на запрос вызываем `get` от соответствующего корня
- Время работы $O(n \log n)$
- Используемая память $O(n \log n + m)$

Итого

- Вопросы?

Задача «Рыцарский щит»

Над задачей работали

- Идея задачи: Владимир Ульянцев
- Текст условия: Андрей Комаров
- Тесты, проверяющая программа и др.: Алексей Цыпленков
- Решения: Владимир Ульянцев, Нияз Нигматуллин
- Текст разбора: Алексей Цыпленков

Формулировка задачи

- Дано два треугольника
- Нужно приложить их друг к другу так, чтобы периметр получившейся фигуры был минимальным

Идея решения

- Периметр полученной фигуры — сумма периметров треугольников минус удвоенная длина общей части
- Нужно максимизировать длину общей части

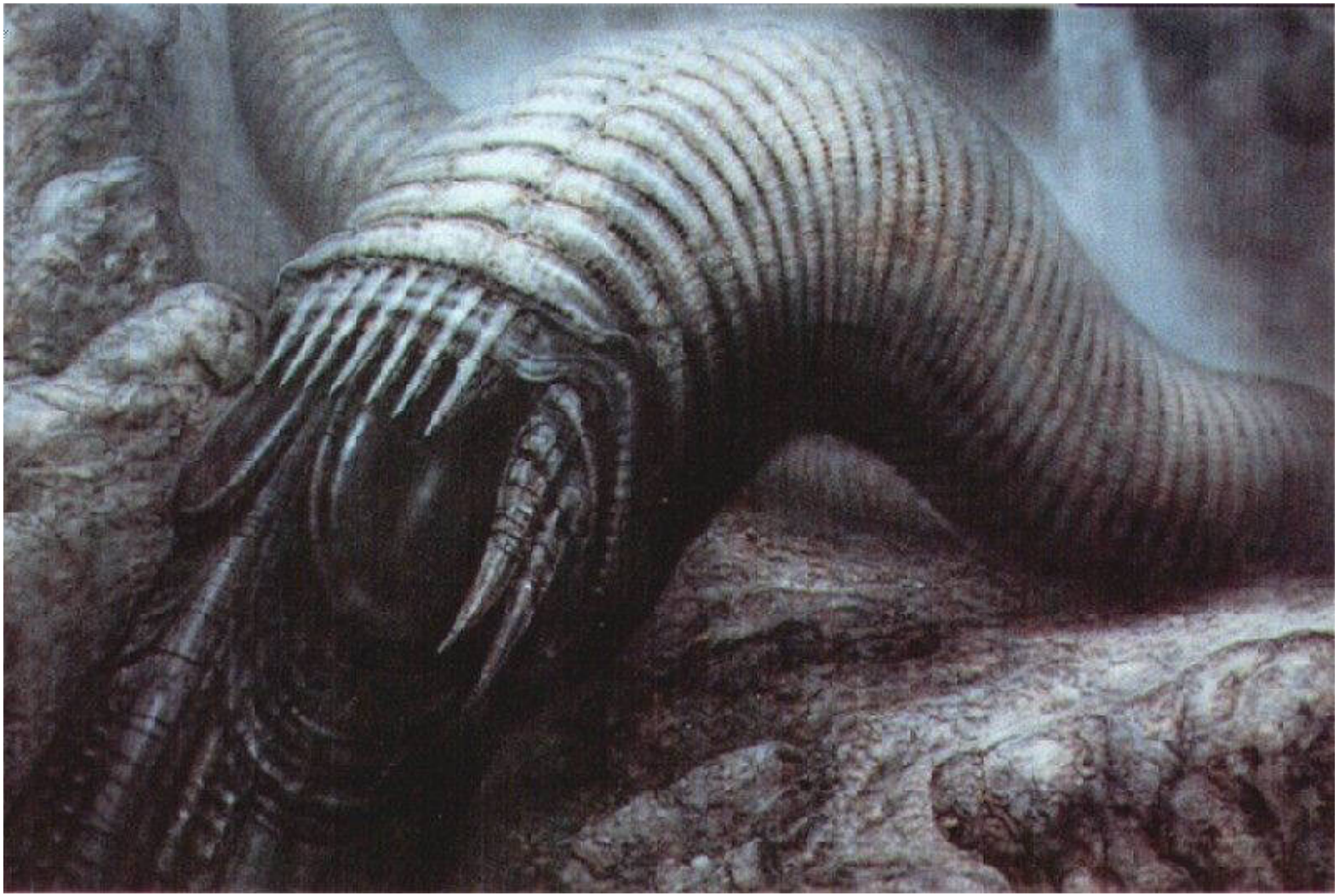
Как это сделать?

- Нужно прикладывать треугольники наибольшими сторонами
- Тогда ответ равен
  $P_1 + P_2 - 2\min(\max(a_1, b_1, c_1),\ \max(a_2, b_2, c_2))$

Итого

Вопросы?

Задача «Змея в метро»



Над задачей работали

- Идея задачи: Юрий Петров
- Текст условия: Юрий Петров
- Тесты, проверяющая программа и др.: Антон Ахи
- Решения: Антон Ахи, Антон Банных
- Текст разбора: Юрий Петров

Формулировка задачи

- Метро — граф
- Каждая вершина лежит не более, чем на одном простом цикле
- Змея лежит вдоль некоторого пути
- Змея не может самопересекаться
- Требуется найти кратчайший путь для змеи

Идея решения

- Заменим каждый цикл на вершину (например, с помощью обхода в глубину)
- Получится дерево
- Возможны два случая:
  - Змея ориентирована в нужную сторону
  - Змея ориентирована неправильно

Случай правильной ориентации

- Требуется найти кратчайший путь любым обходом (в ширину, в глубину)

Случай неправильной ориентации

- Путь состоит из двух частей:
  - Спуск вниз по дереву до цикла, где можно будет развернуться
  - (развернуться можно, если длина цикла не меньше длины змеи)
  - Подъём в обратную сторону

Итого

Вопросы?