Задача A. Прыгающий аппарат

Автор задачи: Степан Филиппов, разработчик: Арсений Кириллов

Посмотрим на любой путь и на любой вертикальный отрезок в нём. Нам не важно, в каком порядке шли предыдущие отрезки, если они приведут в ту же точку. Поэтому можно считать, что мы сначала использовали горизонтальные отрезки, а потом вертикальные. Аналогично, нужно рассмотреть пути, в которых сначала вертикальные отрезки, а потом горизонтальные.

Посчитаем с помощью задачи о рюкзаке все возможные длины первых частей пути, которые мы можем получить. Если S — сумма сил всех пружин, а l — это сумма некоторых из них, то надо в ответ добавить все точки (l, x) и (x, l), где $0 \leqslant x \leqslant S - l$. Для каждой суммы l, что мы можем набрать, её вертикальный столбец имеет $S - l + 1$ клетку. Между вертикальными столбцами для длин $l_1 < l_2$ мы хотим понять, сколько горизонтальных полос есть. Заметим, что это длины этих полос будут должны быт хотя бы l_2, а значит их ровно столько, сколько можно получить сумм, больше или равных l_2.

Задача B. Треугольники и окружность

Автор задачи: Даниил Орешников, разработчик: Ильдар Гайнуллин

Когда треугольник ABC не содержит центр внутри себя? Легко видеть, что это может быть только когда есть диаметр, который не пересекает треугольник.

Если расстояние AB на окружности хотя бы $\frac{L}{2}$, тогда можно провести отрезок AB, а затем подвинуть его, чтобы он не содержал точек A или B и был диаметром.

Однако, легко видеть, что если каждое из расстояний AB, BC, AC на окружности достаточно маленькое, то такого диаметра не найдется.

Отсюда наша задача: посчитать число треугольников на данных точках, что каждое из расстояний AB, BC, AC строго меньше, чем $\frac{L}{2}$. Назовем такие треугольники хорошими, а остальные плохими.

Заметим, что каждый треугольник содержит не больше чем одной плохой пары точек, соответственно если мы посчитаем для каждой упорядоченный пары точек на расстоянии $\geqslant \frac{L}{2}$ число треугольников, которые содержат эти две точки подряд в таком порядке, и просуммируем, мы получим число плохих треугольников.

Отсортируем координаты точек $x_1 < x_2 < \ldots < x_n$, далее рассмотрим два случая:

- Плохая пара точек это пара x_i, x_j где $i < j$.

  В таком случае должно выполнятся $x_j - x_i \geqslant \frac{L}{2}$, и в качестве третьей точки треугольника можно выбрать любую точку среди $1, 2, \ldots, i-1$ или $j+1, j+2, \ldots, n$.

  С помощью метода двух указателей легко поддерживать множество подходящих j, перебирая i, а затем простой формулой суммы арифметической прогрессии просуммировать ответ,

- Плохая пара точек это пара x_i, x_j где $i > j$.

  В таком случае должно выполняться $L-(x_i-x_j) \geqslant \frac{L}{2}$, и в качестве третьей точки треугольника можно выбрать любую точку среди $j+1, j+2, \ldots, i-1$.

  Этот случай аналогично можно учесть с помощью двух указателей.

Итого, после сортировки мы можем решить задачу за $\mathcal{O}(n)$, и итоговое время работы есть $\mathcal{O}(n \log n)$.

Задача C. Игра

Автор задачи: Андрей Станкевич, разработчик: Ильдар Загретдинов

Заведем счетчик для количества очков и раундов. Изначально сыгранных раундов 0, количество очков — n. Количество очков При очередном броске выясняем, если счет больше, чем выдал генератор, то уменьшаем счет в текущем раунде. Если меньше — игнорируем этот бросок. Если же

счет равен сгенерированному числу, то этот раунд заканчиваем, увеличиваем счетчик раундов и перед следующим броском делаем счет равным счету в начале раунда. Если же последний бросок заканчивает раунд, то счет в последнем раунде будет в таком случае равен нулю, и этот раунд будет закончен, но новый не начинается, что нам и нужно.

Задача D. Забор

Автор задачи: Георгий Корнеев, разработчик: Дмитрий Саютин

Для начала заметим, что область, ограниченная забором, должна быть выпуклой, иначе забор точно можно укоротить:

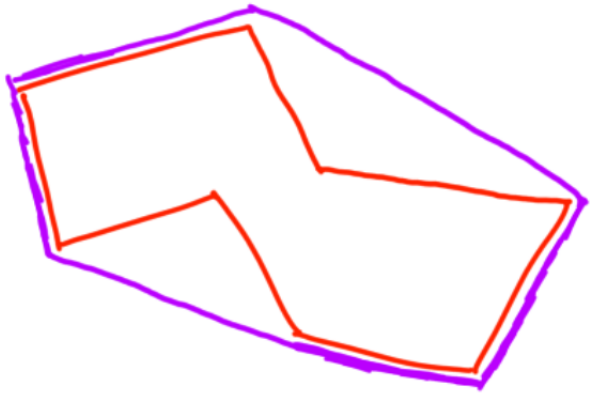

Рис. 1: Невыпуклый забор (красный) не может быть оптимальным: его выпуклая оболочка (показана розовым) всегда будет короче

Также заметим, что все точки $(x_i \pm l, y_i)$ и точки $(x_i, y_i \pm l)$ (для всех исходных точек (x_i, y_i)) должны лежать внутри упомянутого выпуклого множества. Также, из-за того что все стены вертикальные или горизонтальные, оказывается что этого условия достаточно. Так что получается, что нужно просто найти выпуклую оболочку множества точек $(x_i \pm l, y_i)$ и $(x_i, y_i \pm l)$.

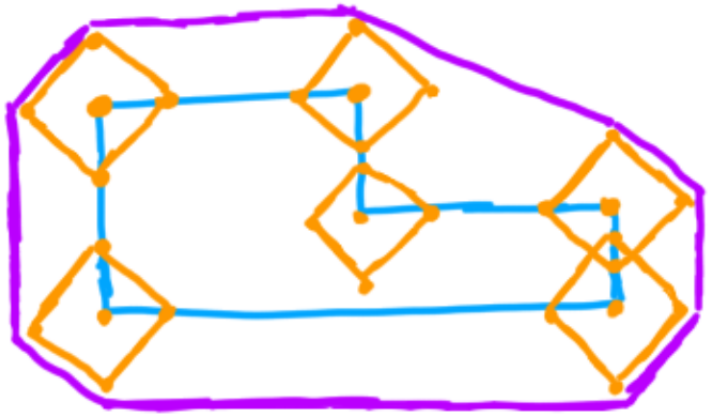

Рис. 2: На картинке изображен исходный многоугольник (синеньким), рассматриваемые точки (оранжевым, объединены в квадраты для наглядности) и оптимальный забор (розового цвета), который можно найти как выпуклую оболочку точек

.

Задача E. Флаг со звёздами

Автор задачи: Дмитрий Штукенберг, разработчик: Виктор Шарепо

Введем несколько обозначений:

- $first$ — число звезд в первом ряду,
- $difference$ — разность числа звезд в первом и втором ряду ($difference \in \{-1, 0, 1\}$),
- max_row — максимальное число звезд в одном ряду,
- min_row — минимальное число звезд в одном ряду,
- $rows$ — число рядов.
- Напоминание: n — число звезд.

Заметим, что n не меньше чем $min\_row \cdot rows$, а значит либо min_row, либо $rows$ не больше $\sqrt{n}$. К тому же, пара $(first, difference)$ или пара $(rows, difference)$ однозначно задает всю фигуру, а значит всего различных фигур асимптотически не более $\sqrt{n}$ и их все можно перебрать за время $\mathcal{O}(\sqrt{n})$.

Переберем все пары $(first, difference)$, а затем все пары $(rows, difference)$, для каждой такой пары определим, существует ли фигура из n звезд с такими параметрами. Если да, то посчитаем $rows$, max_row и обновим ответ: $answer := \min(answer, |rows - max\_row|)$.

1. Перебирая $first$ и $difference$, мы легко определяем число звезд во втором ряду: $second = first + difference$, а далее имеем три случая:
   (a) n делится нацело на $(first + second)$.
   Тогда фигура существует (и у нее четное число рядов), $rows = \frac{n}{first+second} \cdot 2$, $max\_row = \max(first, second)$
   (b) n не делится нацело на $(first + second)$, но $(n - first)$ делится.
   Тогда фигура существует (и у нее нечетное число рядов), $rows = 1 + \frac{n-first}{first+second} \cdot 2$, $max\_row = \begin{cases} first, & \text{если } rows = 1, \\ \max(first, second) & \text{иначе.} \end{cases}$.
   (c) Фигура не существует.
2. Перебирая $rows$ и $difference$, мы можем определить $first$ *(если оно существует)*: $first = \begin{cases} (\frac{n}{\frac{rows}{2}} - difference) \cdot \frac{1}{2}, & \text{если } rows \text{ четное,} \\ \frac{n - difference \cdot \frac{rows-1}{2}}{rows} & \text{иначе.} \end{cases}$. Если полученное число оказалось целым, продолжаем с парой $(first, difference)$ как в пункте 1.

Задача F. Стринг-арт

Автор задачи: Павел Маврин, разработчик: Александра Дроздова

Сформулируем нашу задачу на языке теории графов, дан неориентированный связный граф, требуется построить по нему дерево, у каждой вершины которого есть цвет, такое, что после склеивания вершин одинакового цвета получится исходный граф. Рассмотрим любое остовное дерево исходного графа. Заметим, что дерево подходит под условия на результирующий граф, и в нем ничего менять не нужно. Достроим это дерево так, чтобы после сжатия оно превратилось в исходный граф. Для этого для каждого оставшегося ребра можно добавить вершину того же цвета, что любой из концов ребра, и провести из новой вершины ребро в другой конец рассматриваемого ребра. Граф после этого остается деревом. После склеивания новой вершины и ее соответствующего конца,

как раз получится нужное ребро не из остовного дерева. Найти остовное дерево можно, например, поиском в глубину. Асимптотика: $O(n + m)$

Задача G. Мороженое

Автор и разработчик задачи: Иван Волков

Для начала отметим, что поскольку мы едим постоянно и с одинаковой скоростью, то за t секунд мы всегда съедим ровно $u \cdot t$ грамм мороженого, поэтому можно минимизировать не массу съеденного мороженного, а время, в течении которого мы его ели.

Далее, ключевая идея состоит в том, что верен следующий жадный алгоритм: в каждый момент времени надо есть то мороженое, которого осталось больше всего. При этом в какой-то момент у нас появятся два стаканчика с одинаковым количеством мороженого. Тогда мы должны будем есть мороженое из них 'одновременно' (по-настоящему одновременно мы есть не умеем, но мы можем каждый раз есть по чуть-чуть и затем переключаться с одного стаканчика на другой). Далее, мы будем есть 'одновременно' из трех стаканчиков с самым большим количеством мороженого, и так далее, пока все мороженое не растает.

Почему это оптимально? Пусть нашим алгоритмом мы съели все мороженое за t секунд. Заметим, что если в какой-то момент мы начали есть мороженое из некоторого стаканчика, то мы и далее будем из него есть, до самого конца. Поэтому для каждого стаканчика мы либо не начинали из него есть, и тогда мороженое в нем растаяло целиком, либо мороженое в нем оставалось до самого конца. То есть из i-го стаканчика растаяло ровно $\min\{a_i, v \cdot t\}$ грамм мороженого. Тогда при любом $t' < t$, вне зависимости от наших действий, суммарная масса растаявшего мороженого не может оказаться больше. Но и масса съеденного мороженого тоже будет меньше: $u \cdot t'$ грамм вместо $u \cdot t$. Поэтому в момент t' обязательно останется нерастаявшее мороженое, то есть мы не можем успеть съесть все за меньшее, чем t время.

Проще всего найти t с помощью вещественного двоичного поиска по ответу: при фиксированном t из i-го стаканчика нам потребуется съесть $\max\{0, a_i - v \cdot t\}$ грамм, и если суммарная масса мороженого, которое придется съесть, больше чем $u \cdot t$ грамм, то мы взяли t слишком маленьким, иначе - слишком большим.

Другой способ: просто смоделировать алгоритм. Для этого отсортируем все стаканчики по убыванию a_i и будем добавлять их в множество тех, которые мы в данный момент едим, в таком порядке. (Деталь реализации: вместо того чтобы для каждого стаканчика хранить, сколько мороженого в нем в текущий момент осталось, удобнее хранить исходную массу мороженого и прошедшее время).

Итоговое время работы: $\mathcal{O}(n \log n)$ или $\mathcal{O}(n \log C)$ в зависимости от решения.

Бонус: если мы уже нашли оптимальное время t, то корректным способом будет просто поочередно съесть из каждого стаканчика $\max\{0, a_i - v \cdot t\}$ грамм мороженого, тогда нам придется переключаться с одного стаканчика на другой не более $n - 1$ раза (а не делать это постоянно, как было в изначальном решении).

Задача H. Хоровод разнообразия

Автор задачи: Ильдар Гайнуллин, разработчик: Даниил Орешников

Для простоты объяснения будем дальше отождествлять «костюмы/наряды» и «цвета». Когда речь идет о цветах, имеются в виду номера костюмов (одноцветные — то же самое, что и с одинаковыми номерами костюмов).

Давайте обратим внимание на следующее свойство: если нет пар, в которых номера нарядов одинаковые, то всегда можно расположить все пары в хороводе. Это можно доказать по индукции, взяв две пары и поставив их рядом так, чтобы наряды крайних детей все еще отличались, после чего можно рассматривать эти две пары как одну.

Теперь докажем более сильное утверждение: k пар можно расставить в круг тогда и только тогда, когда нет наряда, который надет более, чем на k детях. Действительно, если есть хотя бы $k+1$ наряд одного вида, то нет возможностей расставить $2k$ людей, чтобы никакие два из тех $k+1$-го не находились рядом. И наоборот, если нет наряда в более, чем k экземплярах, то есть способ расставить пары. Далее приведено конструктивное доказательство как это сделать.

Давайте рассмотрим все пары с одинаковыми нарядами x в них и попробуем каждой из них сопоставить либо пару с другими одинаковыми нарядами y, либо пару с нарядами (y, z). Если поставить две такие пары рядом, мы сможем получить «пару» (x, y) с двумя «спрятанными внутри» людьми, и свести решение задачи к случаю отсутствия одинаковых пар.

1. Если существует такой x, что пары (x, x) занимают больше половины одноцветных пар, то пусть всего одноцветных пар a, а пар (x, x) — ровно b, тогда каждой из $a - b$ оставшихся пар можно поставить в соответствие пару (x, x). В конечном итоге у нас появятся $a - b$ разноцветных пар со «спрятанными» людьми и останутся $2b - a$ лишних пар (x, x).

   Заметим, что из условия, что количество x среди всех пар не больше k, следует, что среди изначальных разноцветных пар у нас не более $k - 2b$ пар, в которых есть x. Соответственно, из $k - a$ разноцветных пар хотя бы $2b - a$ не содержат x вообще. Сопоставим их оставшимся парам (x, x).

2. Если такого x не существует, то гарантированно можно сопоставить все пары одноцветных в соответствие друг другу, однако останется одна лишняя, если их количество нечетно. Опять же, обозначим за a общее число одноцветных пар. Давайте выпишем все одноцветные пары так, чтобы одинаковые цвета шли подряд. Тогда можно взять вместе пары номер i и $i + \left\lfloor \frac{a}{2} \right\rfloor$ и заметить, что цвета в них отличаются (если цвет совпал, то он занимает больше половины a). Тогда можно поставить вместе, образуя ту самую «пару со спрятанными внутри людьми».

   Здесь следует отметить, что в случае нечетного a надо аккуратно выбрать какая пара останется без соответствующей ей. Выберем такую, для которой найдется подходящая разноцветная. А именно, если все разноцветные пары имеют вид (p, q), где p для всех общий, то мы не сможем найти соответствие паре (p, p).

   Чтобы из нечетного числа a выбрать «лишнюю» одноцветную пару, посмотрим на общие для всех разноцветных пар номера костюмов. Их не более, чем два, а каждая одноцветная пара занимает строго меньше половины всех одноцветных (потому что a нечетно), поэтому среди них есть хотя бы три цвета. Выберем тот, который не является общим для всех разноцветных пар — для него найдется разноцветная пара, в которой нет этого цвета, поставим их рядом.

Таким образом, мы рассмотрели несколько случаев, и для каждого поставили некоторые пары рядом друг с другом, чтобы стоящие по краям таких «пар» люди имели разные номера костюмов. Для такого случая задачу можно решить за $\mathcal{O}(n)$. А еще можно заметить, что все проделанные нами операции можно реализовать за $\mathcal{O}(n)$, потому что нам везде достаточно нескольких проходов по каким-то наборам пар.

Осталось только добиться того, чтобы количество каждого цвета не превосходило количества пар. Заметим, что для этого имеет смысл избавляться только от одноцветных пар. Таким образом, полное решение звучит так — будем удалять одноцветные пары, пока самый часто встречающийся наряд присутствует больше, чем на половине детей, а затем конструктивно объединим пары в разноцветные пары пар, которые затем всегда можно будет расположить по кругу.

Задача I. Три тропинки

Автор задачи: Федор Царев, разработчик: Дмитрий Гнатюк

Воспользуемся динамическим программированием $\texttt{dp}[i][j]$ для подсчета количества способов попасть в j-ю вершину за i ходов. Инициализировать динамику можно значениями $\texttt{dp}[0][0] = 1$, $\texttt{dp}[0][j] = 0$. Также посчитаем массив *degree* — степени вершин

Заметим, что $\texttt{dp}[i][j]$ равно $\sum\limits_{k \to j} \texttt{dp}[i-1][k]$. Ответ почти получен, осталось учесть лишние пути вида $s \to j \to s \to i$, и $s \to j \to j \to i$. Посчитаем их количество. Зафиксируем i, тогда $\texttt{degree}[0]$ — количество путей вида $s \to j \to s \to i$. Аналогично, $\texttt{degree}[i]$ — количество путей вида $s \to i \to j \to i$. Но мы дважды посчитали путь $s \to i \to s \to i$. Значит ответ на задачу будет $\sum\limits_{0 \to j} \texttt{dp}[3][j] - \sum\limits_{0 \to i} \texttt{degree}[0] + \texttt{degree}[i] - 1$

Задача J. Скучная пара

Автор и разработчик задачи: Николай Будин

Даны строки s, t и w_i. Нужно за минимальное количество вставок, удалений и замен символа преобразовать s в t (необходимое количество операций называется редакционным расстоянием). При этом, в процессе нужно получить максимальное число w_i.

Научимся находить редакционное расстояние между строками a и b. Это стандартная задача, решающаяся методом динамического программирования. Состояние $dp[i][j]$ соответствует префиксу строки a длины i и префиксу строки b длины j. Значением в состоянии является редакционное расстояние между соответствующими префиксами a и b. Для пересчета используются следующие переходы:

- $dp[i][j] \leftarrow dp[i-1][j] + 1$
- $dp[i][j] \leftarrow dp[i][j-1] + 1$
- $dp[i][j] \leftarrow dp[i-1][j-1]$, если $a[i-1] = b[j-1]$
- $dp[i][j] \leftarrow dp[i-1][j-1] + 1$, иначе

Используя описанный способ, можно вычислить редакционное растояние между всеми парами строк за $O(L^2)$, где L — суммарная длина всех строк. Обозначим редакционное расстояние между a и b за $dist(a, b)$. Строка w_i может встретиться в процессе преобразования если и только если $dist(s, w_i) + dist(w_i, t) = dist(s, t)$. Строки w_i и w_j могут встретиться в таком порядке в процессе преобразования если и только если $dist(s, w_i) + dist(w_i, w_j) + dist(w_j, t) = dist(s, t)$.

Построим ориентированный граф. Вершинами этого графа будут являться строки. Проведем ребро из s в t. Проведём ребро из вершины s в вершину w_i и из w_i в t, если w_i может получиться в процессе преобразования s в t. Проведём ребро из w_i в w_j, если эти две строки могут встретиться друг за другом в процессе преобразования. В данном графе нужно найти путь из s в t с максимальным количеством внутренних вершин. Несложно заметить, что граф является ацикличным.

Осталось научиться находить самый длинный путь из s в t в ацикличном графе. Сделаем топологическую сортировку графа и снова воспользуемся методом динамического программирования. Значение в состоянии $dp[v]$ — максимальная длина пути из вершины v в вершину t. Для пересчета используются следующие переходы:

- $dp[t] \leftarrow 0$
- $dp[v] \leftarrow dp[to] + 1$, для всех ребер (v, to)

Задача K. Шашки

Автор задачи: Даниил Орешников, разработчик: Виктор Шарепо

Сразу отбросим случай $b = 0$, при котором ни одной черной полосы получиться не может, а значит ответ равен 0.

Продолжим с $b > 0$. Очевидно, ответ не может быть больше b.

Теперь давайте поймем какое минимальное число белых шашек нужно, чтобы построить башенку с b полосами. Можно начать на маленьких примерах:

1. Если $b = 1$, то белых шашек вообще не нужно.
2. Если $b = 2$, то без белых шашек получится всего одна полоса, а если поместить одну между двумя черными, то получится уже две полосы.
3. Если $b = 3$, то без белых шашек, как обычно, получится всего одна полоса, с одной белой шашкой получится не более двух полос, а значит нужно хотя бы две белые шашки, чтобы построить башенку с тремя полосами в следующем порядке: черная, белая, черная, белая, черная.

Можно заметить, что минимальное необходимое число белых шашек всегда $b-1$, а располагать их нужно чередуя по одной с черными (начиная с черной).

Таким образом:

1. Если $a \geqslant b-1$, то ответ b: нужно построить башенку из b черных и $b-1$ белых шашек как описано выше, а оставшиеся белые шашки поместить, например, на верх башенки.

2. Иначе, если $a < b-1$, то ответ $a+1$: нужно построить башенку из $a+1$ черных и a белых шашек так же чередуя по одной, начиная с черной, а оставшиеся черные шашки поместить, например, на верх башенки. Из рассуждений выше, можно понять, что ни одна другая конструкция не даст ответ больший чем a + 1.

Задача L. Магниты

Авторы и разработчики задачи: Семен Степанов, Максим Кузин

Посмотрим, как изменятся координаты точек $(x_l, y_l), (x_{l+1}, y_{l+1}), \ldots, (x_r, y_r)$, если они образовывали горизонтальный отрезок (то есть $y_l = y_{l+1} = \ldots = y_r$). Тогда точка (x_i, y_i) перейдёт в точку $(x_l, y_l + (x_i - x_l))$. Видно, что при таком повороте все x координаты точек из запроса станут одинаковыми, то есть каждое значение x_i уменьшится на $x_i - x_l$. Так как точки запроса образовывали непрерывный отрезок, то $x_i - x_l = i - l$.

Получается, чтобы перейти от горизонтального отрезка к вертикальному необходимо ко всем $x_l, x_{l+1}, \ldots, x_r$ прибавить арифметическую прогрессию $0, -1, -2, \ldots, -(r-l)$, а ко всем $y_l, y_{l+1}, \ldots, y_r$ прибавить арифметическую прогрессию $0, 1, 2, \ldots, r-l$. Это можно сделать стандартными методами с помощью дерева отрезков.

Необходимо применить такие же рассуждения для преобразования вертикального отрезка в горизонтальный. Тогда точка (x_i, y_i) перейдёт в точку $(x_l + (y_i - y_l), y_l)$.