Задача A. Дневник дождя

Автор задачи: Михаил Иванов, разработчик: Рита Саблина

Между текущей записью и предыдущей записью 14 дней, запись за прошлое воскресенье должна была быть сделана через 7 дней после записи, оставленной две недели назад, и за 7 дней до сегодняшней записи. Если $n \geqslant 8$, то прошлое воскресенье было в текущем месяце, пропущенная запись должна быть подписана числом $n - 7$. Если $n \leqslant 7$, то прошлое воскресенье было в прошлом месяце, пропущенная запись должна быть подписана числом $m + 7$.

Задача B. Магнитные игры

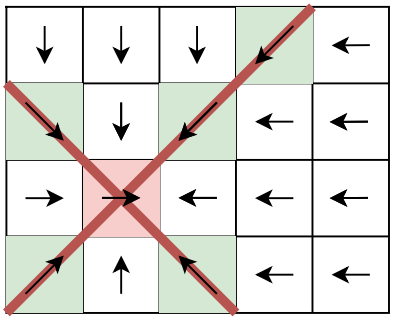
Автор задачи: Семен Чебыкин, разработчик: Павел Ральников

Решение за $O(n^2m^2)$:

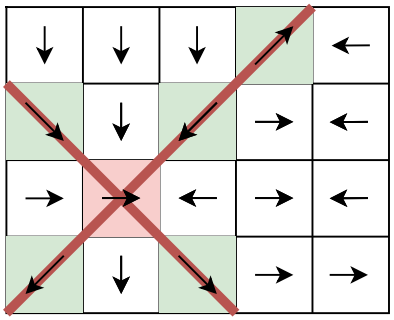
Для решения задачи переберем возможное расположение магнита. Зафиксировав его, мы можем однозначно построить таблицу направлений стрелок. Теперь проверим, совпадают ли направления стрелок в получившейся таблице с советующими им направлениями стрелок в исходной таблице. Должна быть ровно одна строка и ровно один столбец, в котором направления отличаются, причем в этой строке и этом столбце все направления должны быть инвертированы исходным. Это можно посчитать проходом по таблице. Если это так, то мы нашли ответ.

Быстрое решение:

Рассмотрим клетки таблицы, в которых стрелки компасов направлены по диагонали. Заметим, что даже если аномалия изменила направление такой стрелки, то оно все еще осталось диагональным. Также понятно, что все такие клетки образуют на поле две диагонали - одна будет идти сверху вниз и слева направо, вторая — сверху вниз и справа налево, а на их пересечении будет лежать клетка с магнитом (стрелка компаса в клетке с магнитом не обязательно направлена по диагонали).



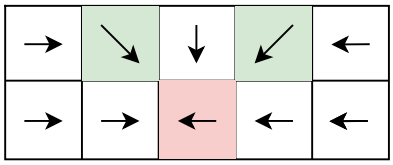
Здесь и далее красным цветом в таблице будет отмечена клетка с магнитом, зеленым — клетки, в которых направление стрелок диагональное. Также на данном рисунке красными линиями отмечены диагонали.



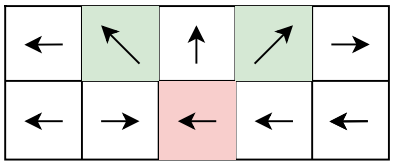
Этот же пример, но с аномалией, инвертирующей показания компасов на четвертой строке и в четвертом столбце.

Пересечем эти диагонали и за $O(nm)$ найдем номер строки и столбца, в которых показания стрелок инвертированы.

Для этого решения существуют проблемные случаи, когда диагонали могут не выявляться. При $n = 2$ или $m = 2$:



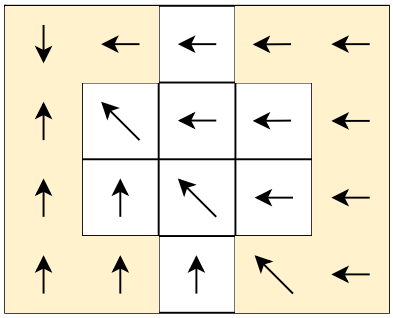
Все стрелки смотрят на магнит



Этот же пример, но с аномалией, инвертирующей показания компасов на первой строке и в первом столбце.

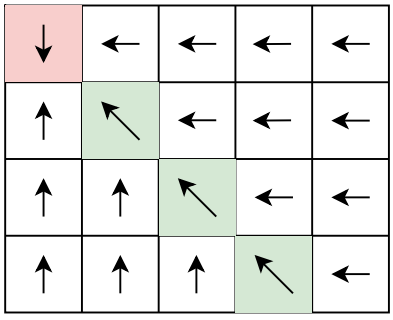
Для этого случая запустим решение за $O(n^2m^2)$, и так как либо n, либо m — это 2, то оно будет работать быстро.

Другой случай соответствует положению магнита или в одной из четырех угловых клеток таблицы, или в смежных с ними. При таком положении одна из диагоналей не выявляется или выявляется сложно:

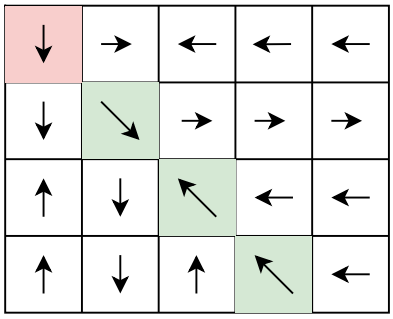


Клетки, которые описывались выше, отмечены желтым

Пример для случая, когда магнит лежит в угловой клетке:



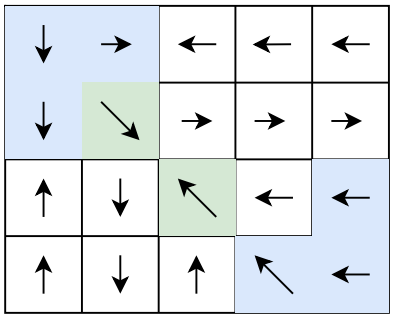
Все стрелки смотрят на магнит



Этот же пример, но с аномалией, инвертирующей показания компасов на второй строке и во втором столбце.

В данном же случае посмотрим на образовавшуюся диагональ.

Если она идет сверху вниз и слева направо (как в примере выше), то магнит может лежать только в шести точках - в самой левой верхней клетке таблицы и двумя смежными с ней, а также в самой правой нижней и двумя смежными с ней. Проверим эти точки, среди них точно будет ответ.



Этот же пример с аномалией. Синим цветом отмечены клетки, которые нужно проверить

Если же диагональ идет сверху-вниз и справа-налево, то магнит может лежать также только в шести точках — в самой правой верхней клетке таблицы и двумя смежными с ней, а также в самой левой нижней и двумя смежными с ней. По аналогии проверим эти точки и точно найдем ответ.

Задача C. Загадка у костра

Автор задачи: Дмитрий Клёпов, разработчик: Михаил Первеев

Основное решение.

Формализуем условие задачи: требуется построить граф на n вершинах, в котором две вершины соединены ребром тогда и только тогда, когда их степени равны, минимизировав количество ребер. Для начала заметим, что все вершины одинаковой степени будут попарно соединены ребрами, поэтому они образуют полный подграф. Также, так как никакие две вершины разных степеней не соединены ребром, можно понять, что каждая компонента связности графа — клика, а степени вершин в разных компонентах связности попарно различны.

Тогда нам нужно выбрать такие размеры компонент связности $s_1, s_2, \ldots, s_k$, что выполнены следующие условия:

1. Все размеры компонент связности попарно различны: $s_1 < s_2 < \ldots < s_k$.
2. Суммарное количество вершин равно n: $s_1 + s_2 + \ldots + s_k = n$.
3. Суммарное количество ребер $\frac{s_1(s_1-1)}{2} + \frac{s_2(s_2-1)}{2} + \ldots + \frac{s_k(s_k-1)}{2}$ минимально.

Решим данную задачу при помощи динамического программирования. Пусть $dp[n][max]$ — минимально возможно количество ребер в графе на n вершинах при условии, что размер максимальной компоненты связности не превосходит max. База динамики: $dp[0][i] = 0$ для всех $i = 0 \ldots n$. Переход: $dp[n][max] = \min\left(dp[i][max-1],\ dp[i-max][max-1] + \frac{max(max-1)}{2}\right)$. Ответ находится в $dp[n][n]$.

Асимптотика: $\mathcal{O}(n^2)$.

Альтернативное решение.

Проанализировав, как выглядят оптимальный ответ для малых значений n, можно заметить следующую закономерность. Пусть max — минимальное целое число, такое, что $1+2+\ldots+max \geqslant n$. Теперь рассмотрим два случая:

1. Если $1 + 2 + \ldots + max = n$, то в оптимальном ответе будет ровно $\sum\limits_{i=1}^{max} \frac{i(i-1)}{2}$ ребер.
2. В противном случае, пусть $t = (1+2+\ldots+max) - n$. Тогда в оптимальном ответе будет ровно $\left(\sum\limits_{i=1}^{max} \frac{i(i-1)}{2}\right) - \frac{t(t-1)}{2}$ ребер.

Асимптотика: $\mathcal{O}(\sqrt{n})$, $\mathcal{O}(\log n)$ или $\mathcal{O}(1)$ в зависимости от способа нахождения величины max и вычисления суммы квадратов последовательных натуральных чисел.

Бонусное решение.

Пусть $f(x) = \left\lfloor \sqrt{2x + \frac{1}{4}} + \frac{1}{2} \right\rfloor$ и $s = f(n)$. Оказывается, что ответ можно вычислить по формуле:

$$\frac{s(s-1)(s-2)}{6} + \frac{\left(n - \frac{s(s-1)}{2}\right) \cdot \left(s^2 + 3s - 2(n+1)\right)}{4}$$

Для подробностей можно обратиться к следующей последовательности в OEIS: `https://oeis.org/A121924`.

Асимптотика: $\mathcal{O}(\log n)$ или $\mathcal{O}(1)$ в зависимости от способа вычисления квадратного корня.

Задача D. Дано дерево

Автор задачи: Андрей Заварин, разработчик: Виктор Романенко

Обозначим за S — сумму длин путей до всех вершин запроса. Будем хранить дерево лениво, поддерживая только те вершины, которые участвовали в запросах. Сделать это можно, к примеру, используя структуру на указателях, и создавать новую вершину, когда заходим в неё при запросе.

Теперь научимся отвечать на запросы.

При запросе первого типа спустимся в вершину запроса и проставим два значения: *timer* — номер текущего запроса и *color* — цвет, в который вершина была покрашена, причём оба эти значения инициализируем -1.

При запросе второго типа поддерживаем локально два значения *timer* и *color* — аналогично, номер последнего встреченного запроса изменения цвета на пути от корня и цвет, в который должна быть покрашена текущая вершина. Если в текущей вершине v её значение *timer* больше чем то, которое мы поддерживаем, то изменяем значения текущие значения *timer* и *color* на значения, записанные в этой вершине. При спуске изменяем значения *color* по правилам из условия задачи. Когда мы доходим в вершину запроса, ответ — это значение переменной *color*.

Решение работает за $O(S)$ памяти и времени.

Задача E. Совсем как огуречик

Автор задачи: Алексей Лучинин, разработчик: Николай Будин

Нужно представить x в виде $\sum_{i=1}^{k} s_i \cdot 2^{p_i}$, где $s_i \in \{-1, 1\}$, $0 \leqslant p_i$, а k — минимально.

Сделаем несколько наблюдений:

- В оптимальной сумме нет двух одинаковых p_i.
  - Если $p_i = p_j$ и $s_i \neq s_j$, то эти два слагаемых в сумме дают 0, их можно выкинуть.
  - Если $p_i = p_j$ и $s_i = s_j$, то эти два слагаемых можно заменить на одно $s_i \cdot 2^{p_i+1}$.
- Существует оптимальный ответ, в котором нет двух слагаемых, для которых верно $p_i = q$, $p_j = q + 1$.
  - Если $p_i = q$, $p_j = q + 1$ и $s_i \neq s_j$, то вместо этих двух слагаемых можно добавить одно $s_j \cdot 2^{p_i}$.
  - Если $p_i = q$, $p_j = q + 1$ и $s_i = s_j$. Выберем среди всех таких пар ту, в которой значение q — минимально. И возьмем из всех оптимальных ответов такой, чтобы это минимальное значение q было максимально. Тогда заменим пару $s_i \cdot 2^{p_i}$ и $s_j \cdot 2^{p_j}$ на пару $-s_i \cdot 2^{p_i}$ и $s_j \cdot 2^{p_j+1}$. Сумма не изменилась, количество слагаемых не изменилось, но минимальное q, такое, что существует пара $p_{i'} = q$, $p_{j'} = q + 1$, увеличилось. А мы брали такой оптимальный ответ, что это q максимально. Получили противоречие.

- Ответ для x равен ответу для $|x|$, поэтому сделаем такую замену и будем решать для неотрицательных x.
- Ответ для $x = 0$ равен 0.
- Ответ для $x \bmod 2 = 0$ равен ответу для $\frac{x}{2}$. Из ответа для $\frac{x}{2}$ можно получить ответ для x — нужно просто увеличить все p_i на 1. Осталось доказать, что ответ для x не может быть меньше, чем ответ для $\frac{x}{2}$. Заметим, что в ответе для x нет слагаемого с $p_i = 0$, иначе бы сумма получилась нечетной, поэтому во всех слагаемых $1 \leqslant p_i$. Уменьшим все p_i на 1, получим ответ для $\frac{x}{2}$.
- В ответ для $x \bmod 4 = 1$ ($x = \overline{\ldots 01}_2$) входит слагаемое $+2^0$. В ответ для такого x должно входить слагаемое с $p_i = 0$, иначе бы сумма получилась четной. Пусть в ответ входит слагаемое -2^0. Тогда все остальные слагаемые в сумме должны давать $x + 1$. $x + 1 = \overline{\ldots 10}_2$, поэтому среди слагаемых, дающих в сумме $x + 1$, обязательно будет слагаемое 2^1 или -2^1. Получили, что у нас есть два слагаемых с p_i, отличающимся на 1. Противоречие.
- В ответ для $x \bmod 4 = 3$ ($x = \overline{\ldots 11}_2$) входит слагаемое -2^0. Аналогичными рассуждениями можно получить, что если мы вместо -2^0 возьмем слагаемое 2^0, то получим два слагаемых с p_i, отличающимся на 1.

В итоге, вычисляем ответ пользуясь следующей рекурсивной формулой:

$$f(x) = \begin{cases} 0 & \text{если } x = 0 \\ f(\frac{x}{2}) & \text{если } x \bmod 2 = 0 \\ 1 + f(\frac{x-1}{4}) & \text{если } x \bmod 4 = 1 \\ 1 + f(\frac{x+1}{4}) & \text{если } x \bmod 4 = 3 \end{cases}$$

Несложно заметить, что время вычисления ответа для x — $O(\log x)$.

Задача F. Ленивое призерство

Автор задачи: Алексей Лучинин, разработчик: Григорий Шовкопляс

Пусть при отптимальной стратегии Алексей будет решать задачи с номерами от l до r включительно, пропустив задачу с номером k.

Заметим, что если среди чисел $a_l, a_{l+1}, \ldots a_r$ есть число меньше a_k, то Алексей может пропустить эту задачу вместо k-й, при этом сумма набранных баллов увеличится. Таким образом, получаем, что для фиксированных l и r для получения оптимальной суммы баллов можно пропускать минимальную по стоимости задачу на отрезке.

Тогда переформулируем исходную задачу. Требуется найти в массиве, отрезок чисел $a_l, a_{l+1}, \ldots a_r$ такой, что $\sum_{i=l}^{r} a_i - \min(a_l, a_{l+1}, \ldots a_r) \geqslant \lceil \sum_{i=1}^{n} a_i/2 \rceil$.

Данную задачу можно решать методом двух указателей и поддержки минимума в окне. Поддерживаем отрезок, такой, что разность его суммы и минимума не меньше $\lceil \sum_{i=1}^{n} a_i/2 \rceil$. При этом минимум на отрезке можно вычислять через Sparse Table, либо поддерживать очередь с минимумом, либо использовать мультисет.

Итоговая сложность решения $O(n)$ или $O(n \log n)$ в зависимости от способа вычисления минимума.

Задача G. Длина последовательности

Автор и разработчик задачи: Мария Жогова

Предположим, существует два отрезка длины S такие, что первый начинается с l_i, а второй с l_2, и $l_1 < l_2$. Заметим, что количество элементов в первом отрезке не меньше, чем во втором. Это верно, потому что все числа на втором отрезке имеют не меньшую длину, следовательно, нам потребуется меньше чисел, чтобы набрать ту же сумму.

Наберем отрезок $[0...right]$, такой, что его длина в десятичной записи, обозначим ее $curS$, равна хотя бы S, и при этом $right$ минимален. Быстро это можно сделать при помощи предподсчета. Для этого воспользуемся фактом, что k-значных чисел (для $k > 1$) $10^k - 10^{k-1}$, они дают вклад в десятичную запись, равный $k \cdot \left(10^k - 10^{k-1}\right)$, а для $k = 1$ вклад в сумму 10.

Далее будем называть числа на отрезке одинаковой длины единым блоком чисел длины k, где k — длина чисел в их десятичной записи.

Воспользуемся методом двух указателей, чтобы найти ответ на задачу. Указатели будут указывать на границы отрезка, который мы рассматриваем. Сейчас указатели установлены на $left = 0$ и $right$. Будем двигать вправо левый указатель пока длина записи больше, чем S, иначе двигать правый. Таким образом мы найдем самый левый подходящий отрезок, который и является ответом на задачу.

Докажем, что такой метод найдет подходящий ответ не более чем за 10^5 итераций.

Заметим, что если $curS - S \leqslant 10$, то можно выкидывать из набранного ответа чисела из блока один, пока не получим нужную сумму, очевидно, что это самый левый из возможных ответов. Поймем, что $curS - S < len(S)$ всегда, если это не так, то мы не должны были взять последнее число в отрезке. Значит, если $right < 10^{12}$, то по утверждению выше ответ найдется за 10 сдвигов указателя.

Рассмотрим остальные случаи. Заметим, что $right$ не больше, чем 10^{17}, так как $\sum\limits_{k=1}^{17} k \cdot \left(10^k - 10^{k-1}\right) > 10^{18}$. А из утверждений выше нам интересны только числа в блоках с 12 по 17.

Пусть мы запустили алгоритм и левый указатель пришел в начало блока k_{left}, а правый находится в k_{right}. Пусть правый указатель не переходил в следующий блок. Поскольку $12 \leqslant k_{right} \leqslant 17$, то для всех k_{right} существует $k_{left} \in [2:5]$, которое с ним взаимно просто. Нам надо найти количество движений левого указателя (x) и правого указателя (y), что $k_{left} \cdot x + k_{right} \cdot y = curS - S \pmod{k_{right}}$. Такое уравнение всегда имеет решения, одно из которых имеет $x + y \leqslant 2 \cdot k_{right}$, то есть не больше, чем за количество чисел в блоке k_{left} мы найдем нужный отрезок.

Если же мы перейдем правой границей в другой блок, то из утверждения выше нам все еще хватит чисел в блоке, на котором стоит левый указатель. Заметим, что такой переход будет не более одного раза.

Таким образом, указатели пройдут не больше чем за количество чисел в блоках с первого по пятый, что и требовалось доказать.

Задача H. Сосны

Автор и разработчик задачи: Александр Гордеев

Нетрудно заметить, что нам важны только индексы ламп типа «A». Оптимальный ответ будет если половина из ламп типа «A» будет светиться красным светом, а другая половина - синим.

Мы хотим получить любую из оптимальных перестановок, поэтому найдём такую, при которой сначала все лампы будут синими или белыми, а потом, начиная с некоторого момента, все лампы будут красными или белыми.

Пусть у нас m ламп типа «A». Заведём массив индексов a, где a_i — индекс i-й по счёту лампы типа «A» в исходном массиве $(1 \leqslant i \leqslant m, 1 \leqslant a_i \leqslant n)$. Тогда все лампы $a_1, a_2, \ldots, a_{\frac{m+1}{2}}$ должны светиться синим, а лампы $a_{\frac{m+1}{2}+1}, a_{\frac{m+1}{2}+2}, \ldots, a_m$ - красным.

Чтобы все «A»-лампы в начале светились синим, а дальше красным, достаточно чтобы перестановка возрастала вплоть до элемента с индексом $a_{\frac{m+1}{2}+1}$, а после него убывала. Действительно, если цвет лампы зависит от разницы между текущим элементом и следующим, то все первые лампы будут светиться синим, а дальше красным.

Тогда один из ответов на задачу - перестановка типа $1, 2, 3, \ldots, a_{\frac{m+1}{2}}, n, n-1, n-2, \ldots, a_{\frac{m+1}{2}}+1$. Для получения такой перестановки достаточно в монотонно возрастающей перестановке перевернуть отрезок массива $[a_{\frac{m+1}{2}}+1; n]$.

Задача I. Цирковое выступление

Автор и разработчик задачи: Даниил Орешников

Посмотрим на условие того, что тройка акробатов хорошая. Для этого должно выполняться неравенство

$$a_i b_j + a_j b_k + a_k b_i \geqslant a_i b_k + a_j b_i + a_k b_j.$$

Само по себе это неравенство не дает четкого понимания, в каком порядке стоит располагать акробатов, но, если перенести все в одну часть и сгруппировать, получится

$$((a_j - a_i)b_k + a_i b_j) + ((a_k - a_j)b_i - a_j b_i) \geqslant 0.$$

Добавим $-a_j b_j$ к первой скобке и $a_j b_j$ ко второй. Неравенство при этом останется верным, но выражения в скобках тогда можно будет представить как $(a_j - a_i)(b_k - b_j)$ и $(a_k - a_j)(b_i - b_j)$. Если расположить на плоскости точки $P(a_i, b_i)$, $Q(a_j, b_j)$ и $R(a_k, b_k)$, то можно заметить, что полученное неравенство соответствует предикату левого поворота от вектора $\overrightarrow{PQ}$ к вектору $\overrightarrow{QR}$.

Таким образом, одно из возможных решений задачи — изобразить акробатов точками на плоскости с координатами, равными их высоте и весу, после чего упорядочить точки так, чтобы переход к каждой следующей паре точек соответствовал левому повороту.

Для этого можно было воспользоваться алгоритмом Джарвиса поиска выпуклой оболочки. Найдем с помощью него выпуклую оболочку, на ней каждая следующая сторона повернута против часовой стрелки относительно предыдущей. Но могли остаться точки, которые не вошли в выпуклую оболочку. Чтобы включить их в последовательность, не будем останавливать алгоритм, когда он дойдет до стартовой точки, а позволим ему продолжать работу, на каждом шаге выбирая вершину с минимальным углом из всех вершин, еще не включенных в последовательность. В конечном итоге такой алгоритм построит заворачивающуюся внутрь против часовой стрелки «спираль», на которой все точки будут упорядочены нужным образом.

Задача J. Прямоугольный забег

Автор задачи: Рита Саблина, разработчики: Арсений Кириллов, Рита Саблина

Предположим, что фотограф сделает фото, когда у всех бегунов будет первая координата совпадать с координатой фотографа. Тогда у каждого бегуна есть два варианта попасть на эту фотографию: слева или справа от фотографа. Переберём все эти 2^n вариантов.

Для бегуна с номером i мы знаем, с какой стороны от фотографа он должен оказаться. Значит, можно посчитать, какое расстояние d_i ему надо пробежать, чтобы впервые оказаться с нужной стороны. Также посчитаем размер его дорожки — число p_i. Тогда для всех времён t, равных d_i, $d_i + p_i$, $d_i + 2 \cdot p_i$, ..., этот бегун окажется с нужной стороны от фотографа. Иными словами, число t должно давать остаток d_i при делении на p_i. Тогда мы получаем систему уравнений $t \equiv d_i \pmod{p_i}$.

Систему таких уравнений можно решить с помощью китайской теоремы об остатках. Возможно, полученная система не имеет решения, тогда этот вариант фотографии нельзя получить.

Аналогичные рассуждения можно провести, когда у бегунов будет совпадать вторая координата с фотографом. Из всех полученных решений необходимо выбрать минимальное.

Оценим величину ответа. Полученное решение не будет превосходить наименьшего общего кратного чисел p_i. Ограничение, что $R_R - R_L + C_R - C_L$ делится на 4, гарантирует, что все числа p_i делятся на 8, а тогда НОК этих чисел не превосходит $9 \cdot 10^{18}$. Заметим, что в зависимости от реализации поиска решений китайской теоремы об остатках промежуточные числа могут превосходить это значение, чтобы этого избежать, можно воспользоваться, например, типом `__int128`.

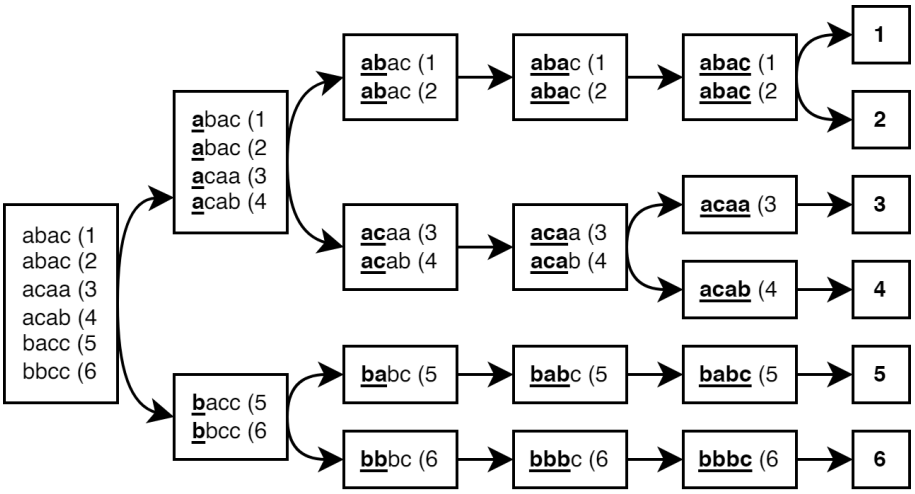
Задача K. Найди пару

Автор и разработчик задачи: Константин Бац

Предположим, ответ на задачу равен k. Рассмотрим префикс длины k в каждом из слов. Разобьём выделенные префиксы на блоки так, чтобы в одном блоке были одинаковые префиксы. Будем говорить, что такие блоки имеют уровень k.

Заметим, что количество слов в каждом из блоков — чётное число. Если это не так, то какому-то слову не найдена пара, что противоречит тому, что k — ответ на задачу.

Рассмотрим деление на блоки для префиксов $k-1$. Блок уровня $k-1$ — это либо блок, который есть на уровне k, либо объединение несколько блоков с уровня k. Из этого следует, что во всех блоках уровня меньше k четное количество слов.



Таким образом, задачу можно сформулировать, как найти такое максимальное k, что блоки уровня k содержат четное количество элементов.

Заметим, что это легко делается, если отсортировать слова в лексикографическом порядке. Будем последовательно рассматривать каждый уровень от 1 до длины слова. Пусть до этого слова разбивались на блоки уровня $k-1$ с четным количество элементов. Рассмотрим уровень k. Выделим группы подряд идущих слов, у которых k–е символы совпадают. Блоки уровня k содержат четное количество элементов тогда и только тогда, когда в каждой из полученных групп четное количество элементов. Действительно, если блоки раньше делись на четное количество элементов, то теперь какие-то блоки разбились на несколько, но в каждом все еще четное количество элементов, а какие-то остались такими же.

Таким образом, в решении необходимо отсортировать слова лексикографически, последовательно проверять уровни до тех пор, пока не найдем максимальный. Поскольку проверка одного уровня делается за линейное время от количества символов, то нужный уровень можно найти за линейное время от суммарной длины строк.

Итого, время работы решения — $\mathcal{O}(nm \cdot \log(nm))$, где m — длина одной строки.

Задача L. Перекладывание дорог

Автор задачи: Федор Царев, разработчик: Григорий Хлытин

Рассмотрим три дороги i, j, k.

Реализуем функцию `check(`i`, `j`, `k`)`, которая будет проверять, можно ли выбрать эти дороги $\{i, j, k\}$ для успешного завершения программы благоустройства. Заметим, что для положительного ответа на этот вопрос необходимо и достаточно выполнения хотя бы одного из следующих условий:

- `dist(`i`, `j`)` $\leqslant$ `length(`k`)` — расстояние между дорогами i и j не больше, чем длина дороги k
- `dist(`i`, `k`)` $\leqslant$ `length(`j`)` — расстояние между дорогами i и k не больше, чем длина дороги j
- `dist(`j`, `k`)` $\leqslant$ `length(`i`)` — расстояние между дорогами j и k не больше, чем длина дороги i

При выполнении хотя бы одного из этих условий мы всегда сможем переложить ровно одну дорогу так, чтобы три получившиеся дороги образовывали связный асфальтированный участок.

Для проверки этих условий реализуем функцию `dist(`s`, `t`)`, которая будет находить расстояние между дорогами s и t. Так как каждая дорога представляет собой отрезок, мы переходим к задаче нахождения расстояния между отрезками на плоскости.

Большинство методов вычислительной геометрии реализуются при помощи нахождения скалярного и псевдоскалярного (косого, векторного) произведения векторов:

- `dot_prod(`$\vec{v}_1$`, `$\vec{v}_2$`)` $= x(\vec{v}_1) \cdot x(\vec{v}_2) + y(\vec{v}_1) \cdot y(\vec{v}_2)$
- `cross_prod(`$\vec{v}_1$`, `$\vec{v}_2$`)` $= x(\vec{v}_1) \cdot y(\vec{v}_2) - y(\vec{v}_1) \cdot x(\vec{v}_2)$
- `length(`$\vec{v}$`)` $= \sqrt{x^2(\vec{v}) + y^2(\vec{v})}$

Вспомним, что по условию каждый из отрезков задан двумя различными точками. Пусть отрезок s задан двумя точками A_s и B_s, а отрезок t задан двумя точками A_t и B_t.

Заметим, что расстояние `dist(`s`, `t`)` между отрезками s и t равно нулю тогда и только тогда, когда отрезки пересекаются, в противном случае расстояние между отрезками равно минимуму из четырех расстояний:

- `dist_point_to_segment(`A_s`, `t`)`
- `dist_point_to_segment(`B_s`, `t`)`
- `dist_point_to_segment(`A_t`, `s`)`
- `dist_point_to_segment(`B_t`, `s`)`

Реализуем функцию `dist_point_to_segment(`P`, `u`)` для вычисления расстояния на плоскости между точкой P и отрезком u (который также задан двумя точками A_u и B_u). Заметим, что если верно условие:

$$\texttt{scalar\_prod}(\vec{v}(A_u, B_u),\ \vec{v}(A_u, P)) \geqslant 0 \ \texttt{and}\ \texttt{dot\_prod}(\vec{v}(B_u, A_u),\ \vec{v}(B_u, P)) \geqslant 0$$

то перпендикуляр из точки P падает на отрезок u, а это значит, что ответ — это расстояние между точкой и прямой:

- `dist_point_to_line(`P`, `u`)` $= |$ `cross_prod(`$\vec{v}(A_u, B_u)$`, `$\vec{v}(A_u, P)$`)` / `length(`$\vec{v}(A_u, B_u)$`)` $|$

Если же это условие неверно, то перпендикуляр из точки P не падает на отрезок u, а это значит, что ответ — это минимум из расстояний:

- `dist_point_to_point(`P`, `A_u`)` $=$ `length(`$\vec{v}(P, A_u)$`)`
- `dist_point_to_point(`P`, `B_u`)` $=$ `length(`$\vec{v}(P, B_u)$`)`

Таким образом, переберем все тройки $\{i, j, k\}$ и для каждой тройки запустим функцию проверки `check(`i`, `j`, `k`)`. Ассимтотика работы решения: $O(m^3)$.