Задача A. Контроль скорости

Как известно, пройденное расстояние равно произведению времени на скорость. Поэтому для того, чтобы найти расстояние S между началом и концом трассы, необходимо сложить произведения $v_i \cdot t_i$. Суммарное время на прохождение всей трассы будет равно сумме всех t_i, обозначим её за T. Далее надо проверить, что средняя скорость S/T не превосходит v_{max}; так как $T > 0$, то это эквивалентно тому, что $S \leqslant v_{max} \cdot T$, что позволяет решить задачу в целых числах, не переходя к вещественным.

Задача B. Вася и признаки делимости

Докажем, что для всех оснований, кратных трём, признак делимости на 3 — `last`, для всех чисел, имеющих остаток 1 при делении на 3, признак делимости — `sum`, и для всех чисел, имеющих остаток 2 при делении на 3, признак делимости — `diff`.

Пусть у нас $a_n \cdot b^n + \ldots + a_1 \cdot b + a_0$ — запись в системе счисления по основанию b.

Если b делится на 3, то сумма $a_n \cdot b^n + ... + a_1 \cdot b$ делится на b и тем самым делится на 3, поэтому число делится на 3 тогда и только тогда, когда оставшееся слагаемое a_0 делится на 3.

Если $b = 3k + 1$, то для всех n $b^n = 3x + 1$, где x_n — некоторое целое число. Этот факт можно доказать через раскрытие скобок в $(3k + 1)^n$, а можно просто доказать по индукции.

$n = 0$: $(3k + 1)^0 = 1 = 3 \cdot 0 + 1$ - верно

Пусть верно для $n \leqslant i$, то есть $(3k + 1)^i = 3x_i + 1$. Тогда $(3k + 1)^{i+1} = (3x_i + 1) \cdot (3k + 1) == (3 \cdot (3x_i \cdot k + k + x_i) + 1)$. Переход доказан.

Тогда $a_i \cdot b^i = a_i \cdot (3k+1) = a_i \cdot 3k + a_i$ даёт тот же остаток при делении на 3, что и a_i. Тем самым число имеет тот же остаток, что и сумма его цифр.

Если $b = 3k + 2$, то докажем, что для чётных i b^i имеет остаток 1 при делении на 3, а для нечётных - остаток 2.

Заметим, что $b^2 = (3k + 2)^2 = 9k^2 + 6k + 3 + 1 = 3K + 1$, то есть для чётных степеней по предыдущему свойству остаток равен 1. В случае нечётной степени получаем $b^{2n+1} = (b^2)^n \cdot b = (3K + 1) \cdot (3k + 2) = 9Kk + 6K + 3k + 2 = 3(3Kk + 2K + k) + 2$, то есть остаток равен 2.

С учётом того, что $3k + 2$ можно представить как $3(k + 1) - 1 = 3k' - 1$, остаток исходного выражения при делении на 3 равен остатку от деления на 3 суммы a_{2k} с плюсом и a_{2k+1} с минусом. то есть свойство `diff`.

Таким образом, требуется вывести или само число B, если остаток совпадает, или $B + 1$, или $B - 1$. Исключение составляет случай $B = 2$ и `sum`, где надо вывести $B = 4$, так как системы счисления с основанием 1 не бывает.

Задача C. Углы между прямыми

Данную задачу можно решить как с помощью векторной геометрии, так и без неё.

В случае с векторной геометрией будем считать, что вершина 1 расмоложена на положительном направлении оси x. Тогда вершина i имеет полярный угол $-(i - 1) \cdot 2\pi/n$, соответственно, если радиус положить равным 1, то имеет координату x, равную $cos(-2\pi(i-1)/n)$, и координату y, равную $sin(-2\pi(i - 1)/n)$. Соответственно, строим векторы A_1A_2 и B_1B_2 и находим угол между векторами (для того, чтобы ответ получился более точным, рекомендуется использовать функцию atan2(y,x), где первый аргумент — косое (псевдоскалярное) произведение векторов, а второй аргумент — скалярное произведение векторов), после чего приводим его к нужному диапазону.

Однако задача имеет решение и без знания векторов и тригонометрии, с использованием свойств вписанных углов. Если соответствующие хорды A_1A_2 и B_1B_2 пересекаются, то угол между ними, как известно, равен половине суммы соответствующих дуг. Если хорды не пересекаются, то угол между ними равен половине разности соответствующих дуг. Проверить, пересекаются ли хорды, можно по номерам вершин. Далее вычисляется ответ с учётом того, что длина дуги между двумя соседними вершинами равна $2\pi/n$

В этом случае требуется рассматривать большее количество различных случаев (в частности, при проверке пересечения хорд), так что в решении через векторную геометрию куда меньше шансов сделать ошибку.

Задача D. Платные поля

Запустим обход в ширину от каждого из двух полей. Вершины графа - поля; два поля соединены ребром, если с одного поля на другое можно пройти одном ходом коня. Таким образом, расстоянием между двумя полями является минимальное количество ходов коня, требуемых для того, чтобы пройти от первого поля до второго.

Множество вершин, для которых найденное обходом в ширину расстояние равно N, будем называть k-окружностью радиуса N. Множество полей, для которых найденное обходом в ширину расстояние равно N или менее, будем называть k-кругом радиуса N.

Покажем, что решение задачи представляет собой некоторую пару непересекающихся k-кругов с центрами в A и B и радиусами R_A и R_B, в которых существуют две клетки, между которыми можно перейти ходом коня, а затем среди всех таких пар кругов выберем ту, которая содержит наименьшее количество клеток в сумме.

Очевидно, что обе эти клетки будут на k-окружности соответствующего радиуса (иначе бы точка во втором k-круге имела расстояние от первого центра меньше, чем радиус, и входила бы в первый k-круг, что противоречит предположению о непересечении).

Тогда количество ходов коня в минимальном пути равно R_A+R_B+1. Действительно, от A можно добраться до любой клетки на границе круга за R_A ходов, затем за один ход перейти на границу круга R_B и затем за R_B ходов добраться до B.

Пусть существует более короткий путь. Расстояние от старта в нём возрастает на 1 (если это не так, то до клетки с расстоянием N можно добраться за N ходов, то есть существует путь ещё короче). Считаем, что путь начинается в A. Первые R_A ходов принадлежат k-кругу с центром в A. Аналогично, последние R_B ходов принадлежат K-кругу с центорм B. Если существует более короткий путь, чем R_A+R_B+1, то у кругов должна быть общая клетка, что противоречит предположению о непересечении.

Сделать минимальную сумму, которую заплатит купец за поездку между A и B больше количества ходов в минимальном пути не получится: даже если сделать все поля платными, купец выберет один из минимальных путей и заплатит сумму, равную количеству ходов в минимальном пути.

При этом в случае, если платными сделаны все клетки в двух k-кругах с описанным выше свойством, то, во-первых, все поля, по которым проходит кратчайший маршрут, принадлежат одному либо другому k-кругу (по построению), то есть являются платными, и, во-вторых, за любой маршрут из A в B придётся заплатить сумму не меньшую, чем за прохождение по кратчайшему пути.

Действительно, за один ход можно увеличить расстояние от стартового поля (допустим, A) максимум на 1, то есть до первого бесплатного поля можно добраться за не менее чем R_A ходов; в конце маршрута расстояние до поля B будет уменьшаться не более, чем на 1 (иначе существует ход коня между полями с расстоянием, большим 1, что противоречит определению расстояния как минимального количества ходов коня), то есть будет сделан 1 ход, который входит в круг с радиусом R_B (такой ход необходим, так как круги по построению не пересекаются и невозможно одновременно оказаться в обоих кругах), после чего будут сделаны не менее R_B ходов по платной территории, чтобы добраться до B, итого не менее $R_A + R_B + 1$ платных полей, что равно длине минимального пути.

Покажем, что решения, которые не являются объединением двух непересекающихся k-кругов, не могут быть оптимальными. Пусть какое-то поле в соответствующем k-круге оставлено бесплатным. Тогда купец может выбрать маршрут к границе k-круга, проходящий через это поле и на каждом ходу увеличивающий расстояние от центрального поля круга (например, если поле лежит справа и сверху от центрального поля, включая границы, то два хода "вверх на 1 и вправо на 2"и "вправо на 1 и вверх на 2"увеличивают расстояние, повторяя эти ходы, получаем два различных способа выйти из круга, затратив $R_A - 1$ монет. Затем можно уйти по бесплатным полям "на бесконечность обойти первый круг так, чтобы не пересекаться с ним, зайти во второй круг и затратить $R_B + 1$ ходов, заплатив R_A+R_B. Так как способов не менее 2, то, чтобы это перекрыть, требуется сделать платными минимум два поля вместо одного, или же увеличить радиус второго круга на 1, что потребует явно больше двух полей.

Поэтому остаётся выбрать оптимальный вариант. Так как число ходов удовлетворяет свойствам расстояния, то из топологических соображений видно, что при $R > r$ k-окружность радиуса R

содержит больше полей, чем k-окружность радиуса r; тем самым, если у нас R_A и R_B отличаются больше, чем на 1 (к примеру, $R_B - R_A \leqslant 2$), мы уменьшаем R_B на 1, а R_A увеличиваем на 1, количество платных полей уменьшается на количество полей в k-окружности радиуса R_B и потом увеличивается на количество полей в k-окружности радиуса $R_A + 1$; так как $R_B > R_A + 1$, то общее количество платных полей уменьшится. Поэтому в итоге получится или два k-круга равного радиуса, или два k-круга, радиусы которых отличаются на единицу.

Впрочем, количество полей в k-окружности с радиусом R можно найти экспериментально: для ограничений из задачи можно просто запустить обход в ширину до расстояния 100 (на самом деле хватит и меньшего растояния), после чего посчитать количество элементов с расстоянием i.

Задача E. Алиса и бот

Заметим, что задача нахождения минимального количества операций решается методом динамического программирования.

Пусть dp_i — минимальное количество операций, требуемое для того, чтобы получить единицу из числа i.dp_1, очевидно, равно 0 (ничего делать не надо). Остальные значения dp от 2 до N будем вычислять следующим образом: первоначально присвоим dp_i значение, соответствующее бесконечности (в данной задаче хватает 10^6, так как любая операция уменьшает число как минимум на 1, то количество операций заведомо меньше $N \leqslant M \leqslant 10^6$).

Далее проходим по всем операциям и сначала проверяем, возможно ли эту операцию применить. Для вычитания проверяем, что $i > x_j$, а для деления — что i делится на x_j. Если применение возможно, то обозначим за p_j число $i - x_j$ для вычитания и число i/x_j для деления.

Если значение dp_{p_j} не равно бесконечности, то сравниваем $dp_{p_j} + 1$ и dp_i, Если значение $dp_{p_j} + 1$ меньше, то вариант получения i через p_j лучше текущего, так что присваиваем $dp_i = dp_{p_j} + 1$.

Если в итоге dp_i так и осталось равно бесконечности, то построить i невозможно в принципе, иначе dp_i будет содержать оптимальное количество шагов.

Результат бота будем вычислять в массиве $greed$. $greed_1$ равно 0. Затем в цикле от 2 до N перевычисляем $greed_i$ по следующей формуле: инициализируем $greed_i$ бесконечностью и проверяем применимость всех операций к текущему числу (по аналогии с решением динамикой), для всех применимых операций находим минимальный результат операции min_{prev}, далее $greed_i$ равно бесконечности, если min_{prev} равно бесконечности, в противном случае $greed_i$ равно $min_{prev} + 1$.

После вычисления обоих массивов идём от 1 к N и сравниваем значения dp и $greed$. Для первого же i, для которого dp_i не равно бесконечности, а $greed_i$ равно, выводим это i и «oo». Иначе поддерживаем максимальное значение разности $greed_i - dp_i$ и первое значение i, на котором этот максимум достигается. Если этот максимум равен 0, выводим -1, иначе выводим запомненные значения i и разности.