Разбор задачи «Lock Puzzle»

Ответ «`NO`» только в том случае, когда мультимножества букв в s и в t не совпадают. В любом другом случае любую строку можно получить из любой.

Рассмотрим решение, использующее не более $4n$ операций. Заметим, что пара операций «`shift` $n-k$», «`shift` n» разворачивает префикс длины k, оставляя остальную строку без изменений. Используя операцию «разворот префикса», можно привести любую строку к любой, ставя символы один за одним в конец строки. Чтобы поставить определенный символ в конец строки, нужно сначала развернуть префикс так, чтобы этот символ оказался на первой позиции в строке, а затем еще раз развернуть префикс, поставив этот элемент на нужное место. Таким образом, используя два разворота префикса на символ, можно получить любую строку, что дает $4(n-1)$ операцию.

Для того, чтобы получить решение, делающее $\frac{5}{2}n$ операций нужно, используя пять операций «добавить» два символа к текущей последовательности. Это можно сделать, например, следующим способом (подчеркнута строка, которая выбирается в качестве β):

1. `...x`<u>`...abc`</u> → `cba......x`
2. <u>`cba......x`</u> → `x......abc`
3. `x......`<u>`abc`</u> → `cbax......`
4. `cbax...`<u>`y..`</u> → `..ycbax...`
5. `..ycbax`<u>`...`</u> → `.....ycbax`

Таким образом, из последовательности `abc` на конце строки, мы получили последовательность `ycbax`, на два символа длиннее. Выбрав нужным образом `x` и `y`, можно поддерживать инвариант, что на конце текущей строки всегда монотонно убывающая или возрастающая последовательность. После того, как эта последовательность имеет длину n, наша строка является либо циклическим сдвигом нужной, либо развернутым циклическим сдвигом нужной. Циклический сдвиг на k можно выполнить последовательностью операций «`shift` $n-k$», «`shift` k», «`shift` n». Таким образом получается решение за $\frac{5}{2}n + O(1)$ операций, получающее 100 баллов.

Разбор задачи «Cooking»

Заметим, что в процессе готовки есть повторяющиеся отрезки — это отрезки между последовательными включениями плиты. Назовём такой отрезок *периодом*. Рассмотрим два случая:

- Если $k \leqslant d$, Юля всегда приходит в момент, когда плита выключена, то есть $period = d$.
- В случае $k > d$ от одного включения до другого p раз Юля приходит, когда плита еще включена, и не совершает никаких действий. В данном случае p — это такое число, что $p \cdot d < k$ — плита еще включена, $(p+1) \cdot d \geqslant k$ — плита уже выключена. Тогда длина периода — это $(p+1) \cdot d$, а число p можно найти, как $\lceil k/d \rceil - 1$. Тогда $period = \lceil k/d \rceil \cdot d$.

Далее для удобства будем считать, что курица состоит из $2t$ частей и на включённой плите за единицу времени приготовится две части, а на выключенной — одна.

Вычислим, на сколько частей приготовится курица в течение одного периода: $cooking = 2k + (period - k) = k + period$. Узнаем, сколько нам нужно полных периодов: $num = \lfloor 2t/cooking \rfloor$. После этого числа периодов готовки останется приготовить еще $carry = 2t - num \cdot cooking$ частей.

Рассмотрим этот остаток:

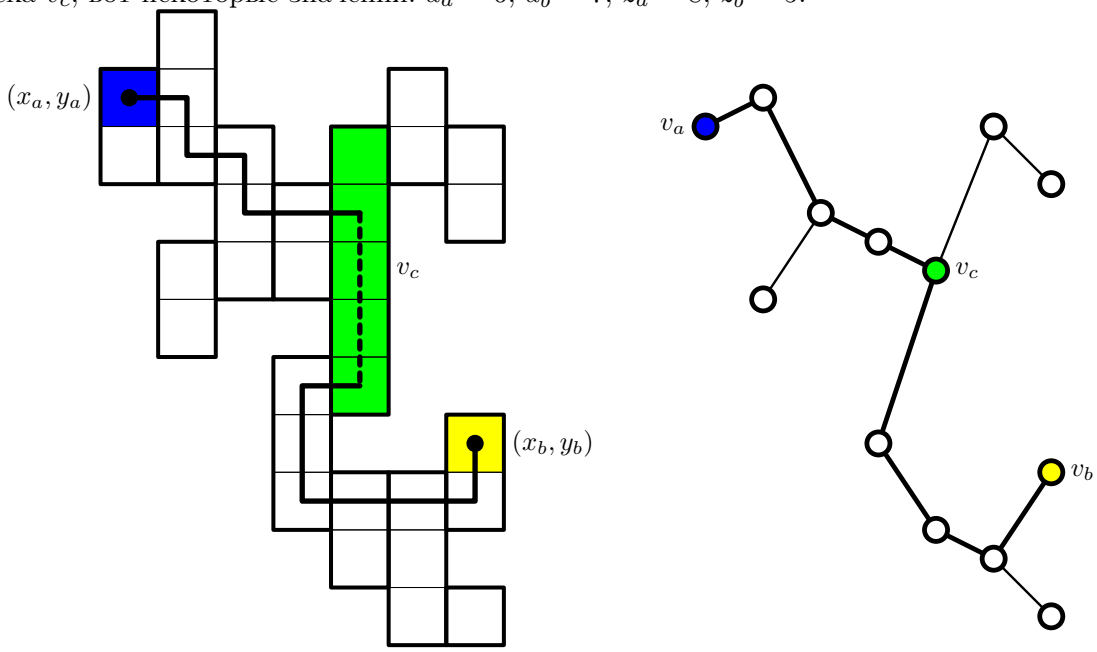
- Если $carry > 2k$, то курица успеет приготовиться на $2k$ частей за следующие k минут на включённой плите и на $carry - 2k$ частей за следующие $carry - 2k$ минут. Тогда ответ: $ans = num \cdot period + carry - k$
- Если $carry \leqslant 2k$, оставшиеся части курицы приготовятся на включённой плите за следующие $carry/2$ минут. В этом случае ответ: $ans = num \cdot period + carry/2$

Разбор задачи «Iqea»

Разобьем фигуру на полоски подряд идущих клеток, находящихся в одном столбце. Будем рассматривать полоски как вершины. Соединим две вершины ребром, если полоски, которым они соответствуют имеют общую границу. Несложно заметить, что получившийся граф будет деревом. Рассмотрим две клетки (x_a, y_a) и (x_b, y_b). Пусть клетка (x_a, y_a) лежит в полоске с индексом v_a, а клетка (x_b, y_b) — в полоске с индексом v_b. Тогда любой кратчайший путь между клетками (x_a, y_a) и (x_b, y_b) проходит только по полоскам, соответствующим вершинам, лежащим на пути между v_a и v_b. При этом, он имеет непустое пересечение с каждой из таких полосок. Поэтому, если построить центроидную декомпозицию дерева полосок, кратчайший путь между двумя клетками обязательно проходит через центроид, разделяющий вершины, в которых лежат эти клетки.

Таким образом, можно решать стандартную задачу на дереве про покраску вершин и поиск ближайшей покрашенной с помощью центроидной декомпозиции. Осталось научиться объединять расстояние до клеток в центроиде. Пусть требуется найти кратчайший путь из (x_a, y_a) в (x_b, y_b), проходящий через полоску v_c. Пусть d_a — кратчайшее расстояние от клетки (x_a, y_a) до полоски v_c, а z_a — y координата ближайшей к (x_a, y_a) клетки на полоске v_c (x координата у всех клеток полоски одинаковая, поэтому она нас не интересует). Аналогично определим d_b и z_b для клетки (x_b, y_b). Тогда искомое расстояние равно $d_a + d_b + |z_a - z_b|$. Поэтому, для объединения двух расстояний до клеток в вершине нужно хранить h значений, и применять к ним операцию $val_i = \min(val_i, d_a + |z_i - z_a|)$, где h — количество клеток в полоске, z_i — y координата i-й клетки в полоске. Это можно сделать, например, с помощью дерева отрезков.

Пример построения дерева. Синим и желтым отмечены клетки (x_a, y_a) и (x_b, y_b), зеленым отмечена полоска v_c, вот некоторые значения: $d_a = 6$, $d_b = 7$, $z_a = 8$, $z_b = 5$.



Значения d_a и z_a можно вычислить для всех пар клетка и вершина во время построения центроидной декомпозиции.

Итоговая асимптотика решения — $O(n \cdot log(n) + q \cdot log^2(n))$ времени и $O(n \cdot log(n))$ памяти.

Разбор задачи «Sleepy game»

Заметим, что ответ — последовательность смежных вершин чётной длины такая, что из последней вершины нет переходов.

Построим следующий граф состояний:

Состояние — пара $(v, parity)$, где v — вершина исходного графа, а $parity$ — чётность количества вершин на пути из s в v. Для каждого ребра uv исходного графа добавим в граф состояний рёбра $(u, 0) \to (v, 1)$ и $(u, 1) \to (v, 0)$. Тогда в графе состояний есть путь из $(s, 1)$ в $(v, parity)$ тогда и только тогда, когда в исходном графе есть путь из s в v, чётность которого $parity$.

Найдём все достижимые из $(s, 1)$ состояния в графе состояний обходом в ширину или в глубину.

Если среди них есть состояние $(v, 0)$ такое, что в исходном графе из v нет рёбер, то Петя может выиграть: достаточно ходить по вершинам пути из $(s, 1)$ в $(v, 0)$ в графе состояний.

Иначе же нужно проверить, можно ли сыграть 10^6 ходов для того, чтобы была ничья. Если мы добьёмся ничьи, то через какую-то вершину мы пройдём дважды, так как $n < 10^6$. А тогда достаточно проверить, есть ли в исходном графе цикл, достижимый из s. В этом случае мы доходим до этого цикла, после чего можем ходить по нему, пока не наберём 10^6 ходов.

Разбор задачи «World of Tank»

1. **Первая подгруппа:**

   Воспользуемся идеей динамического программирования: $dp[i][j]$ — можно ли попасть в клетку в i-й строке j-м столбце. Это значение легко обновляется через значения в клеточка $(i, j-1)$ и $(3-i, j)$ (строки нумеруются с 1). Сначала необходимо обновить и $(1, j)$ и $(2, j)$ через предыдущий столбец, и только после этого обновлять их друг через друга.

   Чтобы восстановить ответ достаточно запомнить для каждой ячейки динамики — откуда ее обновили. Начнем с любой клетки на n+1-м столбце и будем переходить в ту клетку, откуда произошло обновление динамики, до тех пор, пока не окажемся в 0-м столбце.

   Асимптотика решения: $O(n)$.

2. **Вторая подгруппа:**

   Поскольку $\lceil \frac{n}{2} \rceil \leqslant t \leqslant n$, то за весь путь танк мог произвести не более одного выстрела. Для начала просто запустим динамику из предыдущей группы, если она найдет путь, то задача решена,иначе переберем препятствие в которое будет стрелять танк. Удалим его. Снова посчитаем динамику. Пусть теперь путь появился, иначе перейдем к следующему препятствию, предварительно вернув удаленное. Теперь необходимо проверить, что танк вообще мог взорвать это препятствие:

   (a) Препятствие должно находиться хотя бы в столбце с номером $t + 1$, иначе пушка танка не успеет зарядиться.
   (b) Танк должен суметь добраться до точки, из которой выстрел попадет в нужное препятствие, а после этого дойти до конца.

   Пусть препятствие в клеточке с координатами (i, j). Поскольку до удаления препятствия путь не существовал, а после появился, то он обязательно пролегает через соответствующую клеточку.

   Для выполнения второго условия достаточно проверить, что танк может добрать до клеточки с координатами $(i, j-1)$, что можно проверить используя динамику из предыдущей группы. Докажем это утверждение: ранее показано, что путь пролегает через взорванное препятствие, значит в него он пришел либо из соседней строки, либо из предыдущего столбца. Если второе, то находясь в $(i, j-1)$ он произведет выстрел и уничтожит препятствие. Если же он пришел из другой строки, то есть из клетки с координатами $(3-i, j)$, в которую танк мог прийти только из точки $(3-i, j-1)$, тогда модифицируем путь: находясь в клетке $(3-i, j-1)$ перейдем в соседнюю строку $(i, j-1)$, выстрелим и перейдем в (i, j), и продолжим путь как было найдено динамикой.

   Таким образом, если путь существует, то его можно модифицировать так, чтобы он пролегал через клетку $(i, j-1)$, в которой танк совершит выстрел и сможет добраться до конца. В обратную сторону — очевидно.

   Чтобы восстановить путь надо взять путь, полученный динамикой (из первой подгруппы) и модифицировать его, как описано выше. Выстрел максимум один и в точке $(i, j-1)$.

   Асимптотика решения: $O(n \cdot (m_1 + m_2))$

   Перед переходом к следующим трем группам установим несколько фактов:

(a) Если танк произвел выстрел, то стоит пойти в клетку, где только что было препятствие, потому что если существует путь, не проходящий через взорвавшуюся клетку,то его легко модифицировать, чтобы он проходил.

(b) Можно накапливать выстрелы танка, пока он едет по одной строке. Иными словами: пусть танк прошел 100 шагов по одной строке без выстрелов и поворотов, тогда скажем, что к этому моменту он накопил 100 шагов. При t равном, например, 3, он мог произвести до 33 выстрелов, пока ехал. Накопление имеет смысл отложенных выстрелов, то есть накапливаем шаги, а когда придется много стрелять, то аккуратно расставим выстрелы на текущей строке, так чтобы расстояние между ними было хотя бы t. При переходе на другую строку запас шагов сбрасывается, потому что мы могли расставить только в той же строке, что и накопили. Но сбрасывается не до нуля, а до минимума из накопленных и t, поскольку до t танк честно накапливает заряд, и если он от предыдущего выстрела прошел t шагов или меньше, то при переходе на другую строку это не потеряется.

3. **Третья подгруппа:**

Для каждого столбца решим — будет ли в нем поворот или нет, поскольку делать больше одного поворота в столбце не имеет смысла. Таким образом сформировали путь. Теперь необходимо проверить — возможно ли пройти таким путем.

Пойдем вдоль пути, накапливая шаги, как было сказано ранее, а так же помним, в какой координате был предыдущий поворот. После каждого поворота проверим, что не перешли в препятствие. Запомним сколько было препятствий от одного поворота до следующего, пусть их s. Теперь необходимо аккуратно расставить выстрелы. Посмотрим, сколько было накоплено шагов сразу после поворота, пусть k, тогда через $t - k$ шагов танк сможет сделать выстрел, но не ранее, потом через t шагов и так далее. Сделаем s выстрелов в описанных выше точках (сначала отступив $t - k$ от предыдущего поворота, а потом с шагом в t). Если длины пути от поворота до поворота не хватит, или какое-то препятствие не будет уничтожено (соответствующий ему выстрел произошел в той же координате, или далее по пути), то путь плохой, и надо перейти к следующему. Если же получилось уничтожить все препятствия, то надо правильным образом посчитать сколько будет накопленных шагов после следующего поворота: надо к этому же числу для предыдущего поворота добавить длину пути между поворотами, вычесть количество препятствий на пути от поворота до поворота, умноженное на t, после чего взять минимум из получившегося значения и t.

Таким образом перебираем пути, пока они не кончатся, либо пока не найдется путь, по которому танк может проехать.

Асимптотика решения: $O(2^n \cdot n)$.

4. **Четвертая подгруппа:**

Снова воспользуемся идеей динамического программирования: $dp[i][j]$ — максимально количество накопленных шагов, если танк в клеточке (i, j) и если добраться невозможно, то -1.

Переходы точно такие же как и в первой подгруппе, только теперь если в текущей клетке есть препятствие, то можно обновиться через предыдущий столбец, уничтожив препятствие, предварительно проверив, что $dp[i][j-1] \geqslant t$, это значит, что танк успеет накопить нужное число шагов чтобы уничтожить препятствие. Иными словами, чтобы обновиться через предыдущий столбец необходимо проверить, что $dp[i][j-1] \neq -1$, если препятствия в текущей клетке нет, то надо обновить значением $dp[i][j-1]+1$ (просто сделать шаг), иначе проверить, что $dp[i][j-1] \geqslant t$, после чего обновиться значением $dp[i][j-1]-t+1$ (выстрелить и сделать шаг).

При обновлении через клеточку в соседней строке надо проверить, что в текущей клетке нет препятствия и что $dp[3-i][j] \neq -1$, что означает, что туда можно добраться, обновить надо будет значением $min(t, dp[3-i][j])$. При чем надо не забыть сначала $dp[1][j]$ и $dp[2][j]$ обновить через предыдущий столбец, а только после этого обновлять через друг друга, чтобы не потерять информацию.

Восстановление пути как в первой подзадаче, а выбор мест для выстрелов описан в третьей подзадаче.

Асимптотика решения: $O(n)$.

5. **Пятая подгруппа:**

   Для начала докажем, что достаточно поворачивать **в** клетки, находящиеся сразу после препятствий (клетки в той же строке, но в следующем столбце после препятствия).

   Рассмотрим путь. Рассмотрим в нем первый поворот, который не в клетку, которая сразу после препятствия. Тогда рассмотрим ближайшее препятствие, которое находится в той же строке, что и танк после поворота, но имеет меньший номер столбца и не уничтожено:

   (a) Если был поворот в клетку сразу после этого препятствия, то по прямой дойдем до текущей позиции танка, выкидывая ненужные повороты, среди которых будет тот, который был рассмотрен.

   (b) Если поворота не было и препятствие не уничтожено, то это значит, что танк прошел мимо него, тогда можно повернуть в клетку сразу после препятствия и поступить, как в первом пункте.

   Повторяя эти действия путь будет преобразован в путь, где каждый поворот в клетку сразу после препятствия. Таким образом достаточно проверить, что существует такой путь.

   Посчитаем динамику, как в предыдущем решении, только не для всех клеток, а только для тех, которые сразу после препятствий. Как показано выше, этого достаточно.

   Будем перебирать препятствия по увеличению номера столбца. Сначала необходимо обновить через предыдущее препятствие в той же строке. Это делается точно так же, как и в предыдущем решении.

   Теперь обновление через соседнюю строку. Если в текущей клетке есть препятствие, то ее можно обновить только через предыдущее препятствие в этой же строке. Если следующее нерассмотренное препятствие в другой строке имеет тот же номер столбца, что и текущее, тогда сначала надо оба обновить через предыдущие препятствия,а потом через друг друга. Если же имеет больший номер столбца (поскольку перебираем в порядке увеличения номера столбца), то рассмотрим предыдущее препятствие в другой строке, тогда через него легко обновить текущее значение, просто пройдя по строке и повернув. Но может возникнуть проблема, если следующее нерассмотренное препятствие в другой строке имеет номер столбца ровно на 1 больше, чем текущий, то это препятствие будет лежать на пути, тогда надо проверить, что будет накоплено достаточно шагов, чтобы взорвать его.

   Храним для каждой рассмотренной клеточки, откуда ее обновили, что позволит восстановить путь. Походу восстановления пути накапливаем количество взорванных препятствий, чтобы для каждого пройденного отрезка пути без поворотов узнать сколько было выстрелов и восстановить их позиции.

   Асимптотика решения: $O(m_1 + m_2)$.