Задача A. Тщательное планирование

Автор и разработчик задачи: Мария Жогова

Давайте представим разложение числа a_i на сумму разрядных слагаемых: $a_i = a_{i,9} \cdot 10^9 + a_{i,8} \cdot 10^8 + \ldots + a_{i,1} \cdot 10 + a_{i,0}$, где $a_{i,j}$ — значение j-й цифры числа a_i при нумерации с нуля от разряда единиц.

Наша задача — найти такую функцию f, чтобы максимизировать сумму $S = \sum\limits_{i=1}^{n} f(a_{i,9}) \cdot 10^9 + f(a_{i,8}) \cdot 10^8 + \ldots + f(a_{i,1}) \cdot 10 + f(a_{i,0})$. Иными словами, мы просто переписали сумму упомянутых выше выражений по i от 1 до n, если переназначение цифр обозначить за f.

Перепишем эту сумму, сгруппировав слагаемые по степеням десятки, а не по числу из набора. Для этого давайте для каждого числа a_i найдем

$$b_{i,k} = \sum_{j=0}^{9} \left(10^j \cdot \begin{cases} 1 & \text{если } a_{i,j} = k \\ 0 & \text{иначе} \end{cases} \right)$$

Такое выражение будет равно «вкладу» цифры k в значение i-го навыка. Ясно, что $a_i = \sum\limits_{k=0}^{9} k \cdot b_{i,k}$. Пусть $c_k = \sum\limits_{i=1}^{n} b_{i,k}$, назовем это число полным вкладом цифры k. Тогда

$$S = \sum_{i=1}^{n} \left(\sum_{j=0}^{9} f(a_{i,j}) \cdot 10^j \right) = \sum_{i=1}^{n} \left(\sum_{k=0}^{9} f(k) \cdot b_{i,k} \right) = \sum_{k=0}^{9} \left(f(k) \cdot \sum_{i=1}^{n} b_{i,k} \right) = \sum_{k=0}^{9} f(k) \cdot c_k$$

Для того, чтобы получить максимально возможное значение S, функция f должна отображать l и m, так, чтобы $f(l) > f(m) \iff c_l \geqslant c_m$. Понятно, что если f назначает цифре с не-максимальным вкладом новое значение 9, то ее можно поменять с цифрой, вклад которой максимален, и сумма увеличится. Значит нужно распределить новые значения цифр так, чтобы наибольшие новые значения были даны цифрам, имеющим максимальный вклад.

Приведенное выше распределение не учитывает то, что после переназначения в записи навыков не должно быть ведущих нулей. В каком случае это может произойти? В том случае, когда в записи всех n навыков используются 10 цифр, цифра k имеет минимальный вклад и является первой в записи хотя бы одного навыка. Чтобы этого избежать, давайте отдельно рассмотрим все цифры, с которых не начинается никакой навык. Среди всех таких цифр выберем ту, которая имеет минимальный вклад, тогда функция f должна заменять ее на 0. Теперь уберем из рассмотрения выбранную цифру, а значения f для остальных цифр выберем так же, как в общем случае. Ясно, что такое распределение дает максимальную сумму S.

Все решение выглядит так:

1. Для каждой цифры посчитаем ее вклад в общую сумму, является ли она первой в записи какого-либо навыка и общее количество различных использованных цифр в записи навыков;

2. Если в записи всех навыков были использованы все 10 цифр, найдем ту, которая имеет минимальный вклад и с которой не начинается ни один навык. Так как мы ее заменим на 0, то в итоговой сумме она никак себя не проявит. Теперь будем считать, что она имеет вклад 0 в итоговую сумму, то есть $c[q] = 0$;

3. Построим массив g, где $g[k] = (k, c[k])$. Отсортируем этот массив по не возрастанию вклада в итоговую сумму, то есть по второй компоненте каждого элемента;

4. Ответ будем считать жадным образом. Так как мы заменяем i-ю цифру в порядке не возрастания вклада на цифру $10 - i$, то $ans = \sum\limits_{k=1}^{|g|} (10 - k) \cdot g[k][0]$.

Рассмотрение цифр каждого из навыков занимает константное количество действий, а всего навыков n, значит время работы решения — $\mathcal{O}(n)$.

Задача B. Активная подготовка к битве

Автор задачи и разработчик: Владимир Рябчун

Отсортируем людей по возрастанию p_i и перенумеруем (теперь люди с меньшими номерами — это люди с меньшими значениями p_i).

Может оказаться так, что первых $t > 0$ людей Карнаж может съесть, не привлекая внимания полиции. В таком случае он может это сделать, выбирая людей в порядке возрастания p_i: действительно, если он, не привлекая внимания полиции, съедает человека с $p_i > p_j$ раньше, то $P \geqslant p_i$ и $P + p_i \geqslant p_j$, но из первого же неравенства следует, что $P \geqslant p_j$ и $P + p_j \geqslant p_i$, а значит их можно было поменять местами. Пройдем по началу списка людей циклом `while`, добавляя силу рассмотренных людей к P, пока она не превосходит текущего значения P. В тот момент, когда цикл остановится, у всех оставшихся людей сила будет больше текущей силы Карнажа.

Заметим, что, съев некоторого человека, Карнаж может сразу же после этого съесть и всех людей с меньшей силой (так как его собственная сила теперь превзошла силу съеденного человека). Поэтому, если Карнаж может, добившись настороженности полиции C, съесть i-го человека, он может при той же настороженности получить силу $P_{\text{нач}} + \sum\limits_{j=1}^{i} p_j$. Если посмотреть на это с другой стороны, то, чтобы съесть очередного i-го человека, если $p_i > P_{\text{текущей}}$, необходимо и достаточно $\min\limits_{j=i}^{n} c_j$ внимания полиции на то, чтобы съесть любого человека $j \geqslant i$, а затем всех предыдущих.

Таким образом, все решение заключается в следующем: будем обрабатывать людей по очереди, каждый раз поддерживая (P, C) — текущие оптимальные силу Карнажа и настороженность полиции, достигаемые при съедении всех обработанных людей. Если p_i нового человека не превосходит $P_{\text{текущей}}$, то новая пара получается увеличением P на p_i без изменения C. Затем, как только встречается человек номер i, которого невозможно съесть незаметно, Карнажу следует выбрать j, на котором достигается $\min\limits_{j=i}^{n} c_j$, после чего запомнить следующую пару $\left(P = \sum\limits_{k=1}^{j} c_k, C = c_j\right)$, съесть всех людей до j «бесплатно» и перейти к рассмотрению следующих.

Можно заметить, что первый элемент в таких парах всегда увеличивается, а второй — не уменьшается. Таким образом, для ответа на каждый вопрос можно применить двоичный поиск по запомненным парам, либо отсортировать запросы и пройтись двумя указателями.

Задача C. В поисках Венома

Автор задачи и разработчик: Владислав Власов

В каждый момент времени мы имеем прямоугольник со сторонами x и y, и на каждом шаге из большей стороны вычитается меньшая, то есть если $x > y$, то совершается переход от пары строн оставшейся области (x, y) к паре $(x - y, y)$. Сам же процесс заканчивается, когда $x = y$ и последним действием сканируется вся оставшаяся область.

Если просто просимулировать этот процесс, как описано в условии, решение, ожидаемо, не пройдёт по времени. Однако можно заметить, что процесс очень похож на поиск НОД двух чисел, то есть *алгоритм Евклида*.

Быстрый способ провести целиком алгоритм Евклида — заменить вычитание на деление с остатком. В данном случае можно было переходить от пары (x, y) к паре $(y, x \bmod y)$, увеличивая при таком действии ответ на $\left\lfloor \frac{x}{y} \right\rfloor$, так как именно столько вычетаний заменяется одним делением.

Как и *алгоритм Евклида*, такое решение работает $\mathcal{O}(\log(a + b))$.

Задача D. Спрятать заложницу

Автор задачи: Ильгиз Якупов, разработчик: Владислав Власов

Нам дан полный граф, в котором необходимо выделить как можно больше реберно-непересекающихся остовных деревьев.

Сначала решим задачу для четного количества вершин. Всегда можно сделать $\frac{n}{2}$ деревьев. Больше нельзя, потому что в графе только $\frac{n(n-1)}{2}$ ребер, а каждое остовное дерево состоит в точности из $n-1$ ребра. Докажем, что ровно столько выбрать можно.

Возьмем как базу граф двух вершин и единственное остовное дерево в нем, и научимся переходить от $n-2$ вершин к n, добавив в ответ вершины u и v.

По нашему предположению есть $\frac{n-2}{2}$ остовов на исходных $n-2$ вершинах (без u и v). Разобьем эти $n-2$ вершины на группы X и Y, по $\frac{n-2}{2}$ вершин в каждой. Обозначим за X_i и Y_i (пронумерованные в любом порядке) i-ю вершину в X и Y соответственно. Дополним i-й остов ребрами из X_i в u и из Y_i в v, как раз количество остовов совпадает с размерами групп $\frac{n-2}{2}$.

Теперь нам лишь нужно добавить одно новое дерево к нашему текущему ответу. Проведем все ребра $Y_i \to u$ для всех i, все ребра из X_i в v для всех i, и ровно еще одно ребро $u \to v$. Ни одно из этих ребер мы не использовали в предыдущих остовах, так что остовы по-прежнему реберно не пересекаются.

Интересно, что таким образом мы использовали все ребра дерева из четного количества вершин. Для нечетного же случая построим деревья на $n-1$ вершинах как описали выше, после чего добавим еще одну вершину. При построении мы использовали все ребра, кроме тех $n-1$, которые соединяют последнюю вершину с остальными. Ровно $\frac{n-1}{2}$ из них мы проведем для дополнения всех построенных остовов, после чего у нас не останется ребер на новый.

Так как количество ребер в полном графе равно $\frac{n(n-1)}{2}$, и нам нужно провести и вывести практически каждое из них, асимптотика решения $\mathcal{O}(n^2)$.

Задача E. Защищенная тюрьма

Автор задачи и разработчик: Даниил Орешников

Посмотрим, сколько надо заплатить, чтобы построить i-ю комнату внутри, а j-ю снаружи. Если $a_j \geqslant a_i$ и $b_j \geqslant a_j$, то платить не придется ничего. Если $a_j < a_i$ и $b_j < b_i$, то придется заплатить полную цену $a_i + b_i - a_j - b_j$. Иначе же, придется заплатить либо $a_i - a_j$, либо $b_i - b_j$ в зависимости от того, длины какой стороны не хватает.

Рассмотрим комнату (a_i, b_i). Если ограничиться в выборе внешней комнаты только типами с не меньшими значениями b_j, ответом будет $\max\left(0, \min\limits_{j:\, b_j \geqslant b_i} a_i - a_j\right)$. Аналогичное верно при рассмотрении комнат, у которых значения a_j не меньше, чем a_i — ответ будет $\max\left(0, \min\limits_{j:\, a_j \geqslant a_i} b_i - b_j\right)$. Таким образом, найти минимальное количество денег, которое надо заплатить, если выбирать комнаты, в которые i-я уже хотя бы по одному направлению помещается, можно следующим образом. Отсортируем все типы комнат независимо по a и по b, на каждом из полученных отсортированных массивов насчитаем *максимумы на суффиксах* значений b и a соответственно. Запомнив для каждой комнаты ее позицию в таких сортировках, можно за $\mathcal{O}(1)$ найти ответ. Более подробное описание этих двух случаев можно найти в разборе «базовой» версии этой задачи.

Осталось рассмотреть случай, когда выгодно «расширить» комнату, изначально меньшую i-й по обоим направлениям. Для этого достаточно перебирать все комнаты в порядке возрастания b, и на всех уже рассмотренных комнатах поддерживать *декартово дерево* по явному ключу a. Для комнаты (a_i, b_i) достаточно разделить это дерево по ключу a_i и найти в левой части максимум величины $a_j + b_j$, которую можно специально поддерживать на поддеревьях. Время работы такого решения в сумме — $\mathcal{O}(n \log n)$ на сортировку и декартово дерево.

Задача F. Размещение симбиотов

Автор задачи и разработчик: Даниил Орешников

Будем характеризовать *состояние* носителей i-й пары парой (x, y), где x — суммарная опасность симбиотов, расположенных в $2i-1$-м носителе, а y — в $2i$-м носителе.

Рассмотрим возможные расположения двух симбиотов из i-й пары в носителях i-й пары, и какие состояния носителей им соответствуют. Если оба симбиота расположились в предыдущей паре, состояние будет равно $(0, 0)$ (случай 1). Если один из симбиотов расположился в предыдущей паре,

то состояние может быть равно $(0, a_{2i-1})$, $(0, a_{2i})$ или двум симметричным им (случаи 2 и 3). Аналогично, случай 4 соответствует состоянию (a_{2i-1}, a_{2i}) и симметричному ему (которых поровну). И случай 5 возможен, если $a_{2i-1} + a_{2i} \leqslant B$, и соответствует состояниям $(a_{2i-1} + a_{2i}, 0)$ и $(0, a_{2i-1} + a_{2i})$ (количество способов достичь которых, аналогично, одинаково в силу симметрии).

Заведем $\texttt{dp}[i][s]$, где $0 \leqslant i \leqslant n$ и $1 \leqslant s \leqslant 5$ — количество способов расположить первые i пар симбиотов в первых i парах носителей, чтобы последняя пара носителей находилась в состоянии, отвечающему случаю s. Пересчет такой динамики заключается в аккуратном разборе случаев расположения симбиотов i-й пары в зависимости от состояния $i-1$-й пары носителей и ожидаемого состояния i-й пары носителей. Можно немного сократить ручной перебор, закодировав случаи состояний и разобрав общий случай перехода.

Ответ в такой динамике будет располагаться в $\sum\limits_{s} \texttt{dp}[n][s]$. При ручном рассмотрении случаев подсчет $\texttt{dp}[i][s]$ требует перебора пяти предыдущих состояний $\texttt{dp}[i-1][s_{old}]$, и дает общую асимптотику времени работы $\mathcal{O}(25n) = \mathcal{O}(n)$.

Задача G. Подозрительные отчеты

Автор задачи и разработчик: Мария Жогова

Рассмотрим классическую динамику для задачи нахождения *наибольшей общей последовательности*. В данном случае мы хотим проверить, что сокращенный отчет целиком «входит» в полный отчет с указанными дополнительными ограничениями. Будем хранить $\texttt{dp}[i][j]$ — могут ли первые i столбцов гистограммы полного отчета содержать первые j столбцов противозаконного отчета, чтобы i-й соответствовал j-му.

При пересчете такой динамики достаточно перебрать k — предыдущий столбец первой гистограммы, соответствующий $j-1$-му столбцу второй. На него накладываются следующие ограничения: $s_k - t_{j-1} = s_i - t_j$ (то, что гистограмма была обрезана по горизонтальной линии, означает, что разности высот столбцов одинаковы) и $s_x \leqslant s_i - t_j$ для всех $k < x < i$ (так как все элементы между выбранными столбцами должны быть не выше линии разреза). Таким образом,

$$\texttt{dp}[i][j] = \bigvee_{\substack{k<i \\ s_k - t_{j-1} = s_i - t_j \\ \max\limits_{k<x<i} s_x < s_i - t_j}} \texttt{dp}[k][j-1]$$

Такой пересчет в наивной реализации работает за $\mathcal{O}(nm^2)$. Попробуем его соптимизировать.

Во-первых, несложно заметить, что нам надо ограничиться только такими k, для которых $s_k = s_i - t_j + t_{j-1}$. А также можно заметить, что достаточно перебрать не все такие k, а только максимальный подходящий. Действительно, чем меньше k, тем больше максимум на отрезке $s_{k+1}, \ldots, s_{i-1}$, а значит если через какой-то меньший k можно обновиться, то и для максимального все условия будут выполнены. А также, любая НОП, заканчивающаяся в меньших k, может быть модифицирована перемещением последнего столбца в больший k без потери корректности. Поэтому достаточно пересчитать $\texttt{dp}[i][j] = \texttt{dp}[k][j-1]$ для максимального $k < i$, для которого $s_k - t_{j-1} = s_i - t_j$ и $\max\limits_{k<x<i} s_x < s_i - t_j$.

Последнее условие либо выполнено для максимального подходящего по разности столбцов k, либо не выполнено вообще ни для какого. Найти максимальный подходящий k можно, запомнив для каждого числа в массиве последнее его вхождение. В данном случае мы знаем, что мы ищем $s_k = s_i - t_j + t_{j-1}$, значит достаточно взять $k = \texttt{last}[s_i - t_j + t_{j-1}]$. Это все можно поддерживать с помощью, например, хэш-таблиц, пока идем слева-направо.

Осталось только для полученного k проверять условие на максимум между ним и i. Искать максимум на интервале (k, i) можно с помощью *разреженной таблицы*. Предподсчитав ее на гистограмме s, можно проверять, ограниченность максимума на отрезке за $\mathcal{O}(1)$. Итоговая сложность: $\mathcal{O}(n \log n + nm)$

Задача H. Еще более защищенная тюрьма

Автор задачи и разработчик: Даниил Орешников

После нажатия на сектор номер i вся последовательность чисел — это циклический сдвиг исходной последовательности на a_i против часовой стрелки, из которого затем выкинут элемент a_i.

Посмотрим, как решалась бы задача, если бы элементы не убирались из рассмотрения. Тогда нам бы просто надо было научиться быстро сравнивать два циклических сдвига, после чего найти минимальный из них с данными ограничениями можно как за линию, так и сделав полную сортировку всех циклических сдвигов, используя найденный компаратор. Для этого можно построить *суффиксный массив*, который позволит нам сравнивать циклические сдвиги за $\mathcal{O}(1)$.

Теперь, возвращаясь к исходной задаче, научимся сравнивать два циклических сдвига, из которых вычеркнули по одному элементу. Заметим, что если разрезать оба циклических сдвига по позициям вычеркнутых элементов, каждый циклический сдвиг превратится в три отрезка подряд идущих (если считать последовательность циклической) элементов исходной последовательности. Таким образом, можно свести сравнение двух «неполных» циклических сдвигов к сравнению не более трех пар отрезков одинаковой длины из исходной последовательности.

Сравнивать две произвольные строки одинаковой длины можно, сохранив используемый при построении суффиксного массива массив классов для каждой итерации. После i-й итерации нам известен порядок на подстроках длины 2^i, таким образом для сравнения двух подстрок длины a, достаточно посмотреть на их префиксы и суффиксы длины $2^{\lfloor \log_2 a \rfloor}$ и сравнить их классы после $\lfloor \log_2 a \rfloor$-й итерации.

Сделав константное число таких сравнений, мы можем получить результат сравнения двух ответов, получаемых выбором любых i-го и j-го секторов, а значит, опять же, за $\mathcal{O}(n)$ можно найти минимальный ответ. Суммарное время работы равно $\mathcal{O}(n \log n)$ на построение суффиксного массива.

Задача I. Симбиоты внутри

Автор задачи и разработчик: Константин Бац

Представим схему передачи энергии в виде ориентированного графа. Назовем органы источниками, а симбиотов — потребителями. Источники и потребители будут вершинами, а связи — ребрами. Тогда задача заключается в том, чтобы построить граф, в котором существует путь от любого источника до любого потребителя, не проходящий через другие источники (чтобы каждый из потребителей продолжил бы получать энергию при их отказе). Среди всех таких графов требуется найти граф с минимальным количеством ребер, а среди всех таких — с минимальной суммарной длиной ребер.

В таком графе должно быть хотя бы $n+m-1$ ребро. Действительно, при замене всех ориентированных ребер на неориентированные граф должен стать связным, а в связном графе не может быть меньше ребер. Заметим, что тогда в этом графе нет циклов, а значит если из a достижима b, то из b не достижима a. В таком случае все потребители должны быть достижимы из какого-то одного, с которым связаны все источники. Иначе для первого потребителя в топологической сортировке будет существовать источник, из которого он не достижим.

Поэтому минимальное число ребер достигается, когда все источники поставляют какому-то потребителю энергию, а он раздает ее другим. Схему передачи между потребителями тогда можно представить в виде подвешенного дерева. Суммарная длина связей в таком случае равна сумме евклидова расстояния от всех источников до некоторого потребителя и сумме всех ребер в дереве потребителей. Эти два слагаемых можно оптимизировать независимо, так как от выбора «первого» потребителя зависит лишь ориентация ребер в мин. остове.

Давайте минимизируем суммарную длину связей в дереве потребителей. Эту часть задачи можно решить алгоритмом поиска минимального остовного дерева в полном графе потребителей из m вершин, для чего удобно воспользоваться алгоритмом Прима с асимптотикой $\mathcal{O}(m^2)$. Найти минимальную сумму расстояний от всех источников до некоторого потребителя можно, перебрав всех потребителей и для каждого посчитав расстояние до всех источников за $\mathcal{O}(nm)$.

Итого, строим минимальный остов за $\mathcal{O}(m^2)$, перебираем всех потребителей и выбираем среди них такой, от которого расстояние до всех источников будет минимальным за $\mathcal{O}(nm)$, и выводим ответ. Общее время работы — $\mathcal{O}(m \cdot (n+m))$.

Задача J. Долгое путешествие

Автор задачи: Фитисов Артем, разработчик: Мария Жогова

Заметим, что масса любого симбиота ограничена 10^9. Поэтому масса i-го симбиота после того, как он пожертвовал собой, может дойти до симбиота с номером j, удаленного от i-го не дальше, чем на $R = \log_2 10^9 \leqslant 30$. При этом масса i-го симбиота может как-то увеличить массу j-го, и эта обновленная масса может дойти до симбиота k на расстоянии также не больше R от j. Таким образом на вес k-го симбиота могут повлиять симбиоты, удаленные от него не дальше, чем на $2 \cdot R$.

Так как у каждого симбиота есть сосед слева и справа, то масса k-го симбиота может меняться в течении первых $4 \cdot R$ лет. Далее все ближайшие симбиоты погибнут, а масса остальных симбиотов уже не будет доходить до k-го ни в каком виде.

Давайте считать, что $4 \cdot R < n - 1$. Для $1 \leqslant t \leqslant 4 \cdot R$ ответ можно предпочитать, а для больших t он очевидно будет равен ответу при $t = 4 \cdot R$.

Давайте для каждого $0 \leqslant i \leqslant n-1$ пронаблюдаем то, как его масса будет влиять на $2 \cdot R$ соседей слева и справа. Пусть $r[i][j]$ — масса i-го симбиота при условии, что j его соседей справа поделились своей массой, а $l[i][j]$ — масса i-го симбиота при условии, что j его соседей слева поделились своей массой. Ясно, что $l[(i+j) \bmod n][j] = \left\lfloor \frac{l[(i+j-1) \bmod n][j-1]}{2} \right\rfloor + a[(i+j) \bmod n]$, а $r[(i+j) \bmod n][j] = \left\lfloor \frac{r[(i+j+1) \bmod n][j-1]}{2} \right\rfloor + a[(i+j) \bmod n]$ для всех $1 \leqslant j \leqslant 2 \cdot R$.

Тогда максимальная достижимая i-м симбиотом за t лет масса $m[i][t] = \max\limits_{0 \leqslant j \leqslant t} l[i][j] + r[i][t-j] - a[i]$. Ответ на задачу равен $ans[t] = \max\limits_{i=0}^{n-1} m[i][t]$. Это верно для любых $0 \leqslant t \leqslant \min(n-1, 4R)$. Для $t > 4 \cdot R$ при условии $4 \cdot R < n - 1$ ответ будет равен $ans[4R]$.

Если $n - 1 \leqslant 4 \cdot R$, то $ans[t]$ для $t < n - 1$ считается так же, как в общем случае. Рассмотрим отдельно случай, когда $t = n - 1$. В этом случае первый симбиот, который пожертвовал собой поделится своим весом с правым и левым соседом. Поэтому $m[i][n-1] = \max\limits_{j=0}^{n} l[i][j] + r[i][n-j] - a[i]$. Следовательно, $ans[n-1] = \max\limits_{i=0}^{n-1} m[i][n-1]$. В остальных случаях ($t > n - 1$) $ans[t] = ans[n-1]$.

Общий план решения задачи.

1. Посчитаем $l[i][j]$ и $r[i][j]$ для $0 \leqslant i \leqslant n$, $0 \leqslant j \leqslant 2 \cdot R$ через рекуррентные соотношения.
2. Для всех t от 1 до $\min(4R, n-1)$ посчитаем ответ $ans[t] = \max\limits_{0 \leqslant i \leqslant n-1} \left(\max\limits_{\max(0, t-2R) \leqslant j \leqslant \min(t, 2R)} l[i][j] + r[i][t-j] - a[i] \right)$.
3. Если $n - 1 < 2 \cdot R$, то $ans[n-1] = \max\limits_{0 \leqslant i \leqslant n-1} \left(\max\limits_{\max(0, n-2R) \leqslant j \leqslant \min(n-1, 2R)} l[i][j] + r[i][n-j] - a[i] \right)$.
4. Выведем выводить ответы на запросы. Для всех t_i либо уже посчитан, либо он равен ответу при $t = min(n-1, 4R)$, который посчитан.

Общее время работы — $\mathcal{O}(R^2 \cdot n + m) = \mathcal{O}(n + m)$.