Случай задачи об опасной близости, сводящийся к одномерному интервальному поиску

Е. А. Снегова

В работе исследуется задача о поиске движущихся внутри квадрата объектов, которые могут столкнуться с движущимся объектом-запросом, где под столкновением понимается нахождение в опасной близости. Для всех законов движения объектов и запросов, удовлетворяющих условию, что скорость любого объекта не превышает скорость любого запроса в каждый момент времени, представлен алгоритм имеющий логарифмическую относительно общего числа объектов сложность поиска, вставки и удаления, и линейный объем памяти.

1. Введение

Внутри квадрата $[0,1] \times [0,1]$ движутся точечные объекты. Они появляются на двух смежных сторонах и совершают свое прямолинейное движение по заранее известному закону в направлении, перпендикулярном той стороне, на которой они появились.

Объекты, двигающиеся от одной из сторон, будем называть объектами-данными (или просто объектами), а объекты, двигающиеся от другой стороны — объектами-запросами (или просто запросами).

Считаем, что имеется счетное множество объектов-данных, движущихся на плоскости так, что их координаты в зависимости от времени задаются парой $(f(t-t_i), y_i)$, где $i \in \mathbb{N}$, $y_i \in [0,1]$ — координата появления на оси Y, а моменты появления t_i образуют строго возрастающую последовательность положительных чисел. Аналогично, движение объекта-запроса задается парой $(x, f_q(t-t_q))$, где $x \in [0,1]$

координата появления на оси X, а момент появления $t_q \geqslant 0$. Библиотекой в момент t_q назовем множество $V(t_q)$ объектов i, таких что $(f(t_q - t_i), y_i) \in [0,1]^2$.

В задаче требуется для произвольного запроса перечислить все объекты из библиотеки, с которыми он в процессе своего движения будет находиться на расстоянии меньшем, чем ρ по Манхэттену.

Основными характеристиками алгоритма решения этой задачи являются сложность поиска, измеряемая в элементарных арифметических операциях и операциях сравнения, сложности вставки объектов в базу данных и удаления объектов из нее, а также объем памяти, необходимый для хранения структур данных.

Предположим, что алгоритм может выдавать в качестве ответа не только решение задачи, но и некоторое количество «лишних» объектов, то есть решать задачу с погрешностью. Такое предположение может привести к снижению сложности и объема алгоритма. В общем случае, без использования погрешности задачу возможно свести лишь к двумерной задаче интервального поиска, но, как известно [1], решение этой задачи требует больших затрат памяти.

В данной работе рассматривается задача об опасной близости в ситуации, когда скорость объектов не превышает скорости запросов для любого момента времени, то есть выражение $f'(\tau + \tau') - f'_q(\tau) \leqslant 0$ для любого $\tau \in [0, \tau^q_{\max}]$ и любого τ', такого, что $\tau + \tau' \in [0, \min\{\tau_{\max}, \tau^q_{\max}\}]$, где $\tau_{\max}$ и $\tau^q_{\max}$ есть время движения внутри квадрата $[0,1] \times [0,1]$ для объектов и запросов соответственно, а через f и f_q обозначаются функции, называемые законами движения объектов и запросов, задающие соответственно абсциссу объекта и ординату запроса в зависимости от времени. В работе представлен алгоритм решающий данную задачу путем сведения к задаче одномерного интервального поиска (динамический случай), в которой базой данных является конечное множество точек на прямой, а запросом — отрезок, и требуется для каждого запроса перечислить все точки из библиотеки, ему принадлежащие. Алгоритм работает с небольшой погрешностью, относительно общего числа объектов внутри квадрата, за счет которой достигается логарифмическая сложность поиска, вставки и удаления, и линейный объем памяти.

Кроме того, в работе показано, что для того, чтобы алгоритм работал без погрешности, достаточно расширить область наблюдения за объектами на величину ρ по оси X.

В [2] рассматривалась задача об опасной близости для случая фиксированных скоростей объектов, то есть $f(\tau) = v\tau$, а $f_q(\tau) = v_q\tau$.

Автор выражает благодарность профессору Э. Э. Гасанову за постановку задачи и помощь в работе.

2. Постановка задачи и результаты

Пусть $\tau_{\max}$ и $\tau^q_{\max}$ есть время движения внутри квадрата $[0,1] \times [0,1]$ для объектов и запросов соответственно, и пусть $\Delta \geqslant 0$, $a \geqslant 0$. Законами движения объектов и запросов назовем соответственно функции $f : [0, \tau_{\max} + \Delta] \to [0, 1 + a]$ и $f_q : [0, \tau^q_{\max}] \to [0, 1]$, такие, что они являются возрастающими и непрерывными, имеют обе односторонние производные в каждой внутренней точке и $f(0) = f_q(0) = 0$, $f(\tau_{\max}) = f_q(\tau^q_{\max}) = 1$, $f(\tau_{\max}+\Delta) = 1+a$. Область $[0,1]\times[0,1]$ будем называть областью решения, а область $[0, 1 + a] \times [0, 1]$ — областью наблюдения.

Положение любого объекта и любого запроса задается моментом появления, начальной координатой и соответствующим законом движения. Будем считать, что законы движения объектов и запросов f и f_q фиксированы и являются параметрами задачи, поэтому *объектами* назовем пары чисел вида $o_i = (t_i, y_i)$, а *запросами* — пары вида $q = (t_q, x)$.

Обозначим через $Q(t_q) = \{(t_q, x) | x \in [0, 1]\}$ множество всех возможных значений запроса в момент t_q. Для запроса момент появления t_q всегда совпадает с текущим моментом времени, поэтому в дальнейшем вместо $Q(t_q)$ будем иногда писать просто Q.

Множество

$$V(t_q) = \{(t_i, y_i) : (f(t_q - t_i), y_i) \in [0, 1]^2\}$$

назовем библиотекой в момент t_q.

Аналогично изменим обозначение библиотеки: вместо $V(t_q)$ будем писать просто V, понимая под этим множество всех объектов, находящихся в текущий момент времени внутри квадрата. Через $|V|$ обозначим число объектов в библиотеке.

Пусть $\rho \in (0, 1/2)$ число, называемое расстоянием опасной близости. Скажем, что объект (t_i, y_i) и запрос (t_q, x) находятся в опасной близости в момент времени t, если в этот момент

расстояние по Манхэттену между ними не больше ρ, то есть $|f(t-t_i)-x|+|y_i-f_q(t-t_q)|\leqslant\rho$.

Пятерку $I=(\rho,f,f_q,Q,V)$ будем называть *задачей об опасной близости*. Содержательно она состоит в перечислении по произвольному запросу $(t_q,x)\in Q$ всех тех и только тех объектов из библиотеки V, которые в какой-то момент движения в области наблюдения оказываются в опасной близости с запросом.

Ответом задачи об опасной близости $I=(\rho,f,f_q,Q,V)$ на запрос $q=(t_q,x)$ назовем множество $J(I,q)$ объектов из V, которые будут с этим объектом-запросом в опасной близости в области наблюдения, то есть

$$J(I,q)=\{o_i=(t_i,y_i)\in V|\ \forall i\ \ \exists t\in[t_q,\min\{t_i+\tau_{\max}+\Delta,t_q+\tau^q_{\max}+\Delta^q\}]:\ |f(t-t_i)-x|+|y_i-f_q(t-t_q)|\leqslant\rho\}.$$

Поскольку в дальнейшем будет представлен способ неточного решения данной задачи (в ответ будут попадать «лишние» объекты), то для оценки точности решения необходимо предположить, что координаты появления объектов и запросов, а также моменты появления объектов имеют некоторые наперед заданные распределения.

Последовательность $\bar V_\lambda=\{(t_i,y_i),i=\overline{1,\infty}\}$, такую что координаты появления y_i распределены равномерно и независимо на отрезке $[0,1]$, а моменты появления t_i образуют пуассоновский поток с параметром λ, то есть $t_{i+1}=t_i+\Delta_i$, где Δ_i для любого i имеет показательное распределение с параметром λ, назовем последовательностью объектов.

Множество всех последовательностей $\bar V_\lambda$ будем обозначать $\mathbb{V}_\lambda$.

При фиксации последовательности $\bar V_\lambda$ *библиотекой* $V_\lambda(t_q)$ назовем множество объектов, находящихся в текущий момент времени t_q внутри квадрата, то есть:

$$V_\lambda(t_q)=\{(t_i,y_i)\in\bar V_\lambda:t_q\in[t_i,t_i+\tau_{\max}]\}.$$

Как правило, вместо $V_\lambda(t_q)$ будем писать просто V.

Множество всех библиотек в текущий момент времени определим как

$$\mathbb{V}_\lambda(t_q)=\{V_\lambda(t_q):\bar V_\lambda\in\mathbb{V}_\lambda\}.$$

Через $\tilde{Q}(t_q)$ или просто $\tilde{Q}$ обозначим множество всех возможных значений запроса в текущий момент времени t_q, при условии, что координата появления x равномерное и независимое распределение на отрезке $[0, 1]$, то есть:

$$\tilde{Q}(t_q) = \{(t_q, x) | x \sim U[0, 1]\}.$$

Тогда шестерку $I = (\rho, f, f_q, \tilde{Q}, V, \lambda)$ будем также называть задачей об опасной близости. От задачи (ρ, f, f_q, Q, V) она отличается только тем, что в ней введены предположения о характере распределения параметров, задающих положение объектов и запросов.

В качестве модели базы данных будем использовать *информационные графы (ИГ)* [1] с возможностью поиска, вставки и удаления в них.

Структура ИГ.

Пусть F — множество одноместных предикатов, заданных на множестве Q. Пусть G — множество функций, определенных на Q, области значений которых являются начальными отрезками натурального ряда. Назовем их *переключателями*.

Пару $\mathcal{F} = (F, G)$ будем называть *базовым множеством*.

Пусть нам дан произвольный ориентированный граф. Выделим в нем одну вершину и назовем ее *корнем* ИГ. Также выделим произвольное множество вершин, отличных от корня, и назовем их *листьями*. Вершины графа, не являющиеся листьями, будем называть *внутренними вершинами*.

Сопоставим каждому листу графа некоторую запись из множества V.

Выделим некоторое подмножество вершин, которые назовем *точками переключения*. Каждой точке переключения v поставим в соответствие переключатель из G. Занумеруем все дуги, исходящие из v последовательными числами, начиная с 1. Такие дуги назовем *переключательными*. Вершины, не являющиеся переключательными, назовем *предикатными*. Дуги, исходящие из предикатных вершин также назовем *предикатными*. Каждой предикатной дуге сопоставим предикат из F.

Полученный граф назовем *информационным графом* над базовым множеством $\mathcal{F}$ с библиотекой V.

Поиск в ИГ.

Каждому ИГ U с базовым множеством $\mathcal{F}$ и библиотекой V можно сопоставить процедуру поиска. Предполагается, что эта процедура хранит в своей внешней памяти структуру U. Входными данными является запрос $q \in Q$. Выходными данными является подмножество библиотеки V.

Пусть на вход процедуры поступил запрос $q \in Q$. Введем понятие активного множества вершин и внесем в него в начальный момент корень ИГ U и помечаем его. Далее по очереди просматриваем вершины из активного множества и для каждой из них проделываем следующее:

- если рассматриваемая вершина — лист, то запись, приписанную вершине включаем в ответ,
- если рассматриваемая вершина — точка переключения, то вычисляем на запросе q переключатель, соответствующий данной вершине, и если дуга, нагрузка которой равна значению переключателя существует, и ее конец — не помеченная вершина, то помечаем конец дуги и включаем его в множество активных вершин,
- если рассматриваемая вершина — предикатная, то просматриваем по очереди исходящие из нее дуги и вычисляем значения предикатов, приписанных этим дугам на запросе q. Концы дуг, которым соответствуют предикаты со значением, равным 1, если они непомеченные, помечаем и включаем в множество активных вершин,
- исключаем рассматриваемую вершину из активного множества.

Процедура завершает свою работу в тот момент, когда активное множество становится пустым.

Множество, полученное на выходе процедуры, обозначим $J(U, q)$ и будем называть *ответом* ИГ U на запрос q.

Скажем, что ИГ U *решает задачу поиска* для задачи об опасной близости $I = (\rho, f, f_q, \tilde{Q}, V, \lambda)$ (или $I = (\rho, f, f_q, Q, V)$), если ответ, полученный на выходе процедуры поиска, совпадает с ответом на запрос, то есть

$$J(U, q) = J(I, q).$$

Скажем, что ИГ U *решает задачу поиска с погрешностью* для задачи об опасной близости $I = (\rho, f, f_q, \tilde{Q}, V, \lambda)$ (или $I = (\rho, f, f_q, Q, V)$), если ответ на запрос является подмножеством ответа, полученного на выходе процедуры поиска, то есть

$$J(U, q) \supset J(I, q).$$

Сложность поиска в ИГ.

Пусть нам дан информационный граф U над базовым множеством $\mathcal{F}$. Каждому символу $g \in G$ ($f \in F$) поставим в соответствие неотрицательное число $t(g)$ ($t(f)$), обозначающее сложность переключателя g (предиката f).

Сложностью поиска по запросу q в ИГ U назовем сумму сложностей всех переключателей и предикатов, вычисленных в ходе процедуры поиска. Обозначим эту сложность $T(U, q)$.

Вставка и удаление записи в ИГ.

Назовем *вставкой* объекта o в ИГ U с библиотекой V такое преобразование ИГ U в ИГ U', при котором библиотека V преобразуется в $V \cup \{o\}$ и U' остается ИГ, решающим задачу поиска для библиотеки $V \cup \{o\}$.

Удалением объекта o из ИГ U с библиотекой V назовем такое преобразование ИГ Uв ИГ U', при котором библиотека V преобразуется в $V \backslash \{o\}$ и U' остается ИГ, решающим задачу поиска для библиотеки $V \backslash \{o\}$.

Вставка/удаление представляет собой первоначальный поиск места вставки/удаления и последующую последовательность преобразований вершин, дуг графа, их пометок, которые мы договорились считать элементарными, то есть на выполнение которых тратиться небольшое и константное время. Множество элементарных операций обозначим H. Оно будет состоять из склейки и переброски вершин. Чтобы описать эти операции введем несколько понятий из теории графов.

Если в ориентированном графе есть дуга, ведущая из вершины v в вершину w, то вершину v назовем *отцом* вершины w, а вершину w *сыном* вершины v. Два сына одного отца будут называться *братьями*.

Операция *переброски* может быть применена к вершине ИГ, имеющей четырех сыновей. Пусть вершина v имеет четырех сыновей. Для операции переброски образуем новую вершину w. Два сына v переделаем в сыновей w (то есть из w проведем дуги в эти вершины, а дуги, ведущие в эти вершины из v, уберем), а два оставшихся сына оставим сыновьями v. Затем сделаем w братом v, сделав его сыном отца v, то есть проведем дугу из отца v в вершину w. Если у v не было отца, то образуем новую вершину и проводим из нее дуги в вершины v и w.

Операция *склейки* может быть применена к двум вершинам и заключается в простом объединении (склеивании) этих двух вершин. При этом одна из вершин удаляется, а все дуги, ведущие в удаляемую вершину и исходящие из удаляемой вершины, после операции склейки будут вести во вторую вершину и исходить из второй вершины соответственно.

Сложность вставки/удаления в ИГ.

Сопоставим каждой элементарной операции $h \in H$ величину $t(h)$ и назовем ее сложностью операции h.

Тогда *сложность вставки* объекта o в ИГ U равна сумме сложностей всех вычисленных предикатов и переключателей в ходе поиска места вставки и сумме сложностей всех элементарных операций из H, произведенных в ходе выполнения процедуры вставки. Обозначим ее $S(U, o)$.

Аналогично, *сложность удаления* объекта o из ИГ U определим как сумму сложностей соответствующих элементарных операций поиска и преобразования и обозначим как $R(U, o)$.

Понятие алгоритма.

Алгоритмом A решения задачи об опасной близости $I = (\rho, f, f_q, \tilde{Q}, V, \lambda)$ (или $I = (\rho, f, f_q, Q, V)$) будем называть ИГ U с определенными на нем процедурами поиска, вставки и удаления, такой, что ИГ U решает (с погрешностью или без) задачу поиска для задачи об опасной близости I.

Пусть A — алгоритм, решающий задачу об опасной близости, а U_A — ИГ, соответствующий этому алгоритму. Тогда *сложность поиска* по алгоритму A на запросе q для задачи об опасной близости I обозначим как $T_A(I, q)$ и будем считать равной $T(U_A, q)$.

Сложность вставки по алгоритму A объекта o в библиотеку V для задачи об опасной близости $I = (\rho, f, f_q, \tilde{Q}, V, \lambda)$ (или $I = (\rho, f, f_q, Q, V)$) обозначим $S_A(I, o)$ и будем считать равной $S(U_A, o)$.

Сложность удаления по алгоритму A объекта o из библиотеки V для задачи об опасной близости $I = (\rho, f, f_q, \tilde{Q}, V, \lambda)$ (или $I = (\rho, f, f_q, Q, V)$) обозначим $R_A(I, o)$ и будем считать равной $R(U_A, o)$.

Объемом алгоритма A для задачи об опасной близости I назовем число ребер в ИГ U_A и обозначим $Q_A(I)$.

Ответ на запрос $q \in Q$ для задачи об опасной близости I, полученный в результате работы алгоритма A, будем обозначать $J_A(I, q)$ и считать равным $J(U_A, q)$.

Понятие погрешности алгоритма.

Погрешностью алгоритма A, решающего для любого V задачу $I = (\rho, f, f_q, \tilde{Q}, V, \lambda)$, назовем число $f_A(\rho, \lambda)$, равное математическому ожиданию от числа объектов, не находящихся в опасной близости с запросом, но попавших в ответ, точнее

$$f_A(\rho, \lambda) = M_q M_V |J_A(I, q) \backslash J(I, q)|,$$

где математическое ожидание берется по всем V из $\mathbb{V}_\lambda$ и всем q из $\tilde{Q}$.

Будем писать, что $f(n) = O(g(n))$, если существуют такие константы $C_1 > 0, C_2 > 0$, и константы C_3,C_4 такие, что для каждого n выполнено:

$$C_1 g(n) + C_3 \leqslant f(n) \leqslant C_2 g(n) + C_4.$$

Теорема 1. *Пусть выражение* $f'(\tau + \tau') - f'_q(\tau) \leqslant 0$ *выполняется для любого* $\tau \in [0, \tau^q_{\max}]$ *и любого* τ', *такого, что* $\tau + \tau' \in [0, \min\{\tau_{\max}, \tau^q_{\max}\}]$, *и область наблюдения совпадает с областью решения. Тогда существует алгоритм* A, *решающий задачу об опасной близости* $I = (\rho, f, f_q, \tilde{Q}, V, \lambda)$ *с погрешностью, такой что для сложности поиска ответа на любой запрос* q, *вставки любого объекта-данного* o, *удаления любого объекта-данного* $\tilde{o}$ *по алгоритму* A, *и для объема алгоритма* A *верны соответственно следующие оценки*

$$T_A(I, q) = O(\log_2 |V|),\ S_A(I, o) = O(\log_2 |V|), \tag{1}$$

$$R_A(I, o) = O(\log_2 |V|),\ Q_A(I) = O(|V|), \tag{2}$$

при этом погрешность алгоритма оценивается как

$$f_A(\rho,\lambda) \leqslant C\lambda(\frac{\rho^2}{2}-\frac{\rho^3}{3}),\ \ C=\frac{1}{\min\limits_{\xi\in[0,\tau^q_{\max}]} f'_q(\xi)}-\frac{1}{\max\limits_{\xi\in[0,\tau^q_{\max}]} f'_q(\xi)}.$$

Теорема 2. *Пусть выражение* $f'(\tau+\tau')-f'_q(\tau)\leqslant 0$ *выполняется для любого* $\tau\in[0,\tau^q_{\max}]$ *и любого* τ'*, такого, что* $\tau+\tau'\in[0,\min\{f^{-1}(1+\rho),\tau^q_{\max}\}]$*, и область наблюдения имеет вид* $[0,1+\rho]\times[0,1]$*, то алгоритм* A*, упоминаемый в теореме 1, будет решать задачу* $I=(\rho,f,f_q,Q,V)$ *без погрешности и оценки сложности и объема будут удовлетворять неравенствам (1)–(2).*

Последняя теорема не зависит от распределений моментов и координат появления объектов и распределений координат появления запросов.

3. Вспомогательные утверждения

Лемма 1. *Пусть* $I=(\rho,f,f_q,Q,V)$ *— задача об опасной близости, область наблюдения имеет вид* $[0,1]^2$ *и для любого* $\tau\in[0,\tau^q_{\max}]$ *и любого* τ'*, такого, что* $\tau+\tau'\in[0,\min\{\tau_{\max},\tau^q_{\max}\}]$ *верно, что*

$$f'(\tau+\tau')-f'_q(\tau)\leqslant 0. \tag{3}$$

Тогда объект $o_i=(t_i,y_i)$ *из библиотеки* V *принадлежит ответу на запрос* $q=(t_q,x)$ *(то есть* $o_i\in J(I,q)$*) тогда и только тогда, когда выполняются условия:*

$$\left[\begin{array}{l}
\left\{\begin{array}{l} x<\rho \\ t_i-f_q^{-1}(y_i)\geqslant t_q-f^{-1}(x+\rho)\end{array}\right. \\
\left\{\begin{array}{l} x\in[\rho,1-\rho] \\ t_i-f_q^{-1}(y_i)\in[t_q-f^{-1}(x+\rho),t_q-f^{-1}(x-\rho)]\end{array}\right. \\
\left\{\begin{array}{l} x>1-\rho \\ y_i\leqslant x+\rho-1 \\ t_i-f_q^{-1}(y_i)\leqslant t_q-f^{-1}(x-\rho)\end{array}\right. \\
\left\{\begin{array}{l} x>1-\rho \\ y_i>x+\rho-1 \\ t_q-\tau_{\max}+f_q^{-1}(1-x+y_i-\rho)-f_q^{-1}(y_i)\leqslant \\ \qquad\qquad\qquad\qquad\leqslant t_i-f_q^{-1}(y_i)\leqslant t_q-f^{-1}(x-\rho).\end{array}\right.
\end{array}\right. \tag{4}$$

Доказательство. Из определения $J(I,q)$ следует, что для того, чтобы объект $o_i = (t_i, y_i)$ из библиотеки V принадлежал ответу на запрос $q = (t_q, x)$ необходимо и достаточно, чтобы существовало t, удовлетворяющее условиям

$$\begin{cases} |f(t-t_i) - x| + |y_i - f_q(t-t_q)| \leqslant \rho \\ t \in [t_q, \min\{t_i + \tau_{\max}, t_q + \tau^q_{\max}\}]. \end{cases} \tag{5}$$

Обозначим $g(t) = |f(t-t_i) - x| + |f_q(t-t_q) - y_i|$. Будем искать минимум функции $g(t)$ на отрезке $[t_q, \min\{t_i + \tau_{\max}, t_q + \tau^q_{\max}\}]$ и записывать первое условие из (5) для этой точки. Рассмотрим несколько случаев.

1) $t_q + f_q^{-1}(y_i) \leqslant t_i + f^{-1}(x)$. Тогда минимум $g(t)$ нужно искать на отрезке $[t_q + f_q^{-1}(y_i), \min\{t_i + f^{-1}(x), t_q + \tau^q_{\max}\}]$, на котором $g(t) = f_q(t-t_q) - y_i + x - f(t-t_i)$. Заметим, что условие (3) равносильно условию $f'(t-t_i) \leqslant f'_q(t-t_q)$. Значит, $g(t)$ на данном отрезке является неубывающей функцией и достигает своего минимального значения в точке $t_q + f_q^{-1}(y_i)$. Тогда первое условие из (5) выглядит как

$$g(t_q + f_q^{-1}(y_i)) = x - f(t_q + f_q^{-1}(y_i) - t_i) \leqslant \rho \Leftrightarrow$$
$$\Leftrightarrow f(t_q + f_q^{-1}(y_i) - t_i) \geqslant x - \rho,$$

что эквивалентно отсутствию ограничений для случая $x < \rho$ и условию

$$t_i - f_q^{-1}(y_i) \leqslant t_q - f^{-1}(x - \rho)$$

для случая $x \geqslant \rho$.
В конечном итоге, первый случай эквивалентен совокупности

$$\left[\begin{cases} x \geqslant \rho \\ t_i - f_q^{-1}(y_i) \in [t_q - f^{-1}(x), t_q - f^{-1}(x-\rho)], \\ x < \rho \\ t_i - f_q^{-1}(y_i) \geqslant t_q - f^{-1}(x). \end{cases} \right.$$

2) $t_q + f_q^{-1}(y_i) > t_i + f^{-1}(x)$. Тогда минимум $g(t)$ нужно искать на отрезке $[\max\{t_i + f^{-1}(x), t_q\}, \min\{t_q + f_q^{-1}(y_i), t_i + \tau_{\max}\}]$, на котором $g(t) = f(t-t_i) - x - f_q(t-t_q) + y_i$. Поскольку $f'(t-t_i) \leqslant$

$f'_q(t-t_q)$, то $g(t)$ на данном отрезке является невозрастающей функцией и достигает своего минимального значения в правом конце.

а) Пусть $t_i+\tau_{\max}\geqslant t_q+f_q^{-1}(y_i)$. Тогда искомая точка минимума — $t_q+f_q^{-1}(y_i)$ и

$$g(t_q+f_q^{-1}(y_i))=-x+f(t_q-f_q^{-1}(y_i)+t_i)\leqslant\rho\Leftrightarrow$$
$$\Leftrightarrow f(t_q+f_q^{-1}(y_i)-t_i)\leqslant x+\rho,$$

что эквивалентно совокупности

$$\left[\begin{array}{l}\left\{\begin{array}{l}x\leqslant 1-\rho\\ t_i-f_q^{-1}(y_i)\geqslant t_q-f^{-1}(x+\rho)\end{array}\right.\\ \left\{\begin{array}{l}x>1-\rho\\ t_i-f_q^{-1}(y_i)\geqslant t_q-\tau_{\max}.\end{array}\right.\end{array}\right.$$

Таким образом, результатом случая 2(а) является

$$t_i-f_q^{-1}(y_i)\in[t_q-f^{-1}(x+\rho),t_q-f^{-1}(x))$$

для $x\leqslant 1-\rho$ и

$$t_i-f_q^{-1}(y_i)<t_q-f^{-1}(x)$$

для $x>1-\rho$.

б) Теперь пусть $t_i+\tau_{\max}<t_q+f_q^{-1}(y_i)$. Тогда $g(t)$ достигает своего минимального значения в точке $t_i+\tau_{\max}$ и

$$\begin{array}{rcl}g(t_i+\tau_{\max}) & = & -x+f(\tau_{\max})+y_i-f_q(t_i+\tau_{\max}-t_q)\leqslant\\ \leqslant\rho & \Leftrightarrow & f_q(t_i+\tau_{\max}-t_q)\geqslant 1-x+y_i-\rho,\end{array}\qquad(6)$$

что верно всегда в случае $y_i<x+\rho-1$ (а условие $y_i\leqslant x+\rho-1$ может выполняться только если $x>1-\rho$).
Для $y_i\geqslant x+\rho-1$ должна быть верна система

$$\left\{\begin{array}{l}1-x+y_i-\rho\leqslant 1\\ t_i\geqslant t_q-\tau_{\max}+f_q^{-1}(1-x+y_i-\rho).\end{array}\right.$$

Заметим, что с учетом условия этого пункта: $t_i < t_q + f_q^{-1}(y_i) - \tau_{\max}$ первое уравнение в системе необходимо заменить на более строгое:

$$1 - x + y_i - \rho < y_i \Leftrightarrow x > 1 - \rho.$$

В итоге, случай 2(b) эквивалентен:

$$\left[\begin{array}{l} \left\{\begin{array}{l} x > 1-\rho \\ y_i < x + \rho - 1 \\ t_i - f_q^{-1}(y_i) < t_q - \tau_{\max} \end{array}\right. \\ \left\{\begin{array}{l} x > 1-\rho \\ y_i \geqslant x + \rho - 1 \\ t_i - f_q^{-1}(y_i) \in [t_q - \tau_{\max} + f_q^{-1}(1 - x + y_i - \rho) - f_q^{-1}(y_i), t_q - \tau_{\max}]. \end{array}\right. \end{array}\right.$$

Объединение результатов пунктов 1, 2(a) и 2(b), с учетом того, что $\rho < \frac{1}{2}$, дает утверждение леммы.

Тем самым лемма 1 доказана.

Лемма 2. *Пусть в условиях леммы 1 область наблюдения имеет вид* $[0, 1+\rho] \times [0, 1]$. *Тогда объект* $o_i = (t_i, y_i)$ *из библиотеки* V *принадлежит ответу на запрос* $q = (t_q, x)$ *тогда и только тогда, когда выполняются условия:*

$$\left[\begin{array}{l} \left\{\begin{array}{l} x < \rho \\ t_i - f_q^{-1}(y_i) \geqslant t_q - f^{-1}(x + \rho) \end{array}\right. \\ \left\{\begin{array}{l} x \geqslant \rho \\ t_i - f_q^{-1}(y_i) \in [t_q - f^{-1}(x + \rho), t_q - f^{-1}(x - \rho)] \end{array}\right. \end{array}\right. \tag{7}$$

Доказательство. По условию леммы область определения функции f расширена до $[0, 1+\rho]$ и $\tau_{\max} = f^{-1}(1+\rho)$. Следовательно, выкладки леммы 1 меняются: результатом пункта 2(a) становится

$$t_i - f_q^{-1}(y_i) \in [t_q - f^{-1}(x + \rho), t_q - f^{-1}(x)),$$

а пункт 2(b) вообще никогда не реализуется, поскольку, с одной стороны, по условию этого пункта

$$t_i + \tau_{\max} - t_q < f^{-1}(y_i),$$

а, с другой стороны, условие (6) можно теперь переписать как

$$\begin{aligned} g(t_i+\tau_{\max}) &= -x+1+\rho+y_i-f_q(t_i+\tau_{\max}-t_q)\leqslant \\ \leqslant \rho &\Leftrightarrow t_i+\tau_{\max}-t_q\geqslant f_q^{-1}(1-x+y_i), \end{aligned}$$

то есть необходимым условием для выполнения 2(b) является $x>1$, чего быть не может. Из пунктов 1 и 2(а) леммы 1 с учетом сделанных замечания получается утверждение этой леммы.

Тем самым лемма 2 доказана.

Пусть $B\subset[0,\infty)\times[0,1]$ и пусть $\xi_V(B)$ — случайная величина, обозначающая **число** объектов данных (t_i,y_i) из V, таких что $(t_i,y_i)\in B$, где случайные величины t_i образуют пуассоновский поток с параметром λ, то есть $t_{i+1}=t_i+\Delta_i$, где Δ_i для любого i распределено по показательному закону с параметром λ, а случайная величина y_i для любого i равномерно распределена на отрезке $[0,1]$.

Рассмотрим множество

$$B_0=\{(t,y): y\in[0,a], t\in[t_1(y),t_2(y)]\},$$

где $t_1(y),t_2(y)$ произвольные непрерывные функции из множества $[0,1]$ в множество $[0,\infty)$, такие что $t_1(y)\leqslant t_2(y)$.

Пример множества B_0 изображен на рисунке 1.

Лемма 3.

$$M_V\xi_V(B_0)=\lambda\int\limits_0^a(t_2(y)-t_1(y))dy.$$

Доказательство данной леммы приведено в [2].

4. Логарифмическое решение задачи об опасной близости

Приведем алгоритм решения задачи об опасной близости $I=(\rho,f,f_q,\tilde{Q},V,\lambda)$ для случая, когда скорость запросов не меньше скорости объектов, то есть для любого $\tau\in[0,\tau^q_{\max}]$ и любого τ', такого, что $\tau+\tau'\in[0,\min\{\tau_{\max},\tau^q_{\max}\}]$ верно, что $f'(\tau+\tau')-f'_q(\tau)\leqslant 0$.

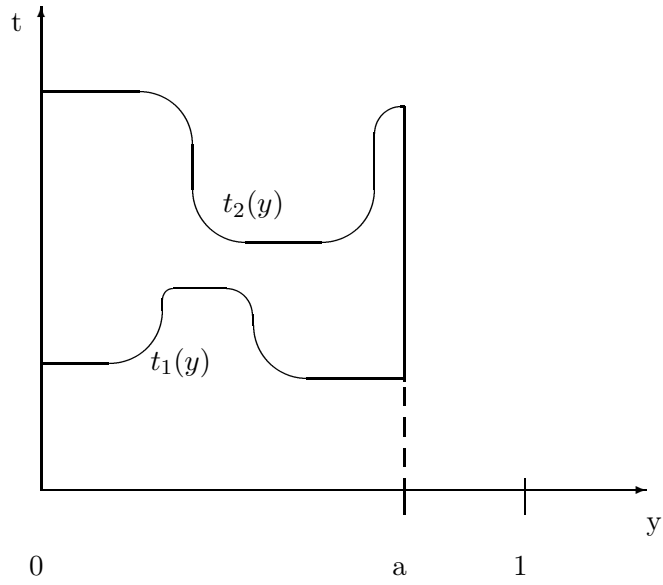

Рис. 1. Область B_0.

Из леммы 1 следует, что объект $o_i = (t_i, y_i)$ из библиотеки V входит в ответ $J(I, q)$ на запрос $q = (t_q, x)$ тогда и только тогда, когда для параметров объекта и запроса выполняется совокупность (4).

Идея алгоритма A заключается в сведении данной задачи к задаче одномерного интервального поиска. Поскольку данные меняются с течением времени (появляются «новые» объекты, исчезают «старые» объекты), то будем использовать ИГ, основанный на динамической структуре данных 2–3 дерева для упорядоченного множества параметров

$$s_i = t_i - f_q^{-1}(y_i).$$

Для каждого запроса $q = (t_q, x)$ алгоритм A будет выдавать в ответ $J_A(I, q)$ объекты $o_i = (t_i, y_i)$ из библиотеки V, удовлетворяющие совокупности

$$\left[\begin{array}{l} \left\{\begin{array}{l} x < \rho \\ t_i - f_q^{-1}(y_i) \geqslant t_q - f^{-1}(x+\rho) \end{array}\right. \\ \left\{\begin{array}{l} x \in [\rho, 1-\rho] \\ t_i - f_q^{-1}(y_i) \in [t_q - f^{-1}(x+\rho), t_q - f^{-1}(x-\rho)] \end{array}\right. \\ \left\{\begin{array}{l} x > 1-\rho \\ t_i - f_q^{-1}(y_i) \in [t_q - \tau_{\max} - C_1(x+\rho-1), t_q - f^{-1}(x-\rho)], \end{array}\right. \end{array}\right. \tag{8}$$

где

$$C_1 = \frac{1}{\min\limits_{\xi \in [0, \tau_{\max}^q]} f_q'(\xi)}.$$

Поскольку при $x > 1-\rho$ и $y_i > x+\rho-1$

$$0 \leqslant f_q^{-1}(y_i) - f_q^{-1}(1-x+y_i-\rho) \leqslant C_1(x+\rho-1),$$

а при $x > 1-\rho$ и $y_i \leqslant x+\rho-1$, учитывая, что $t_q \leqslant t_i + \tau_{\max}$, последняя строчка в совокупности (8) выполняется автоматически, так как

$$t_i - f_q^{-1}(y_i) \geqslant t_q - \tau_{\max} - f_q^{-1}(x+\rho-1) \geqslant t_q - \tau_{\max} - C_1(x+\rho-1),$$

то $J(I,q) \subseteq J_A(I,q)$ для любого q.

Опишем кратко структуру 2–3 дерева. Линейно упорядоченное множество (например, числа $r_1 \leqslant \ldots \leqslant r_n$) можно представить в виде 2–3 дерева следующим образом: элементы множества приписываем листьям слева направо по порядку. В каждой внутренней вершине w хранятся две величины : $L(w)$ максимальный элемент в левом поддереве узла w и $M(w)$ — максимальный элемент в среднем (если две дуги, то в правом) поддереве узла w.

В нашем случае будем использовать «прошитое» 2–3 дерево, то есть 2–3 дерево, в котором от каждого листа идет ребро вправо к соседнему листу.

«Прошитое» 2–3 дерево позволит осуществлять интервальный поиск в упорядоченном множестве $r_1 \leqslant \ldots \leqslant r_n$. Пример «прошитого» 2–3 дерева для $n = 6$ показан на рис. 2.

Свойства 2–3 дерева применительно к задаче поиска идентичных объектов подробно изучена в работе [3]. На основе этой работы опишем операции поиска, вставки и удаления для «прошитого» 2–3 дерева применительно к решаемой задаче, оценим их сложность и объем алгоритма.

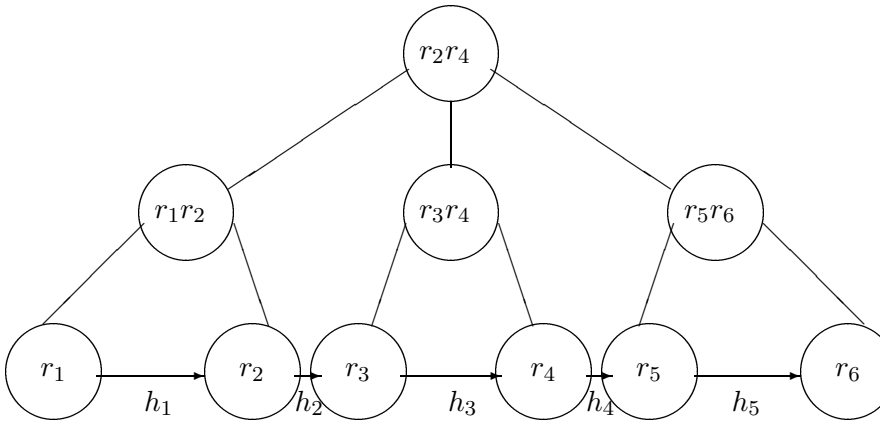

Рис. 2. Пример «прошитого» 2–3 дерева для $r_1 \leqslant r_2 \leqslant \ldots \leqslant r_6$.

Процедура поиска.

Пусть поступил запрос $q = (t_q, x)$.

1) Вычислим границы отрезка поиска: сравним x и $1-\rho$, поставим запросу в соответствие числа

$$A(q) = \begin{cases} t_q - f^{-1}(\min\{x+\rho, 1\}), \text{ если } x \leqslant 1-\rho \\ t_q - \tau_{\max} - C_1(x+\rho-1), \text{ если } x > 1-\rho. \end{cases}$$

$$B(q) = t_q - f^{-1}(\max\{x-\rho, 0\}).$$

Сложность вычисления f^{-1} от любого числа из отрезка $[0,1]$ будем считать равной α_f, сложность операции сложения\вычитания — α_+, сложность операции сравнения двух чисел — $\alpha_\leqslant$, а сложность умножения — α_*. Таким образом, суммарная сложность вычисления $A(q)$ и $B(q)$ варьируется от $(4\alpha_+ + 2\alpha_\leqslant + \alpha_f)$ до $(5\alpha_+ + 3\alpha_\leqslant + 2\alpha_f + \alpha_*)$.

2) Операция поиска листа s_k ближайшего справа к $A(q)$ происходит путем последовательного **прохождения** внутренних вершин, где под прохождением внутренней вершины w будем понимать следующий набор операций сравнения: если $A \leqslant L(w)$, то переходим к левому сыну w, иначе, если у w два сына, или $A \leqslant M(w)$, то переходим к среднему сыну w, иначе к правому сыну w.

Операцию прохождения внутренней вершины внесем в базовое множество $\mathcal{F}$ в качестве переключателя и обозначим ее сложность за γ.

В работе [3] показано, что операция поиска в 2–3 дереве имеет сложность не меньшую, чем $\gamma]\log_3|V|[$ и не большую, чем $\gamma[\log_2|V|]$.

3) Сравним s_k с $B(q)$. Если $s_k \leqslant B(q)$, то выдаем o_k в ответ и переходим по ребру к следующему листу. Предикат сравнения также внесем в базовое множество $\mathcal{F}$ и будем считать, что этот предикат равен 1, если B меньше, чем число s_k приписанное листу.

4) Повторяем предыдущий пункт, пока значение предикатов, вычисляемых на ребрах, равняется 1.

Заметим, что число операций в этом пункте пропорционально длине ответа, а в алгоритме A мы оцениваем сложность без перечисления ответа. Следовательно, для получения сложности без перечисления ответа не нужно учитывать сложность последних двух пунктов.

Получаем, что

$$\gamma]\log_3|V|[+4\alpha_+ + 2\alpha_\leqslant + \alpha_f \leqslant T_A(I,q) \leqslant \\ \leqslant \gamma[\log_2|V|] + 5\alpha_+ + 3\alpha_\leqslant + 2\alpha_f + \alpha_*.$$

Процедура вставки.

Для нового объекта-данного $o_k = (t_k, y_k)$ вычислим $s_k = t_k - f_q^{-1}(y_k)$. Сложность этой операции составляет $\alpha_+ + \alpha_{f_q}$, где α_{f_q} есть сложность вычисления f_q^{-1} от числа из отрезка $[0,1]$.

Чтобы в 2–3 дерево *вставить* новый элемент s_k, нужно найти место для нового листа l, который будет содержать s_k. Для этого ищут элемент s_k в дереве. Если дерево содержит более одного элемента, то поиск s_k окончится в узле f, имеющем двух или трех сыновей, которые являются листами.

Если из узла f выходит только два листа — l_1 и l_2, то делаем l сыном узла f так, чтобы листья остались упорядоченными слева направо. Кроме того, нужно изменить пометки L и M в вершине f, а возможно еще и в нескольких предках f.

Теперь предположим, что у f уже есть три листа — l_1, l_2, l_3. Сделаем l сыном узла f,сохраняя упорядоченность листьев. Чтобы сохранить 2–3 свойство, образуем новый узел g. Два левых сына оставим сыновьями f, а два правых переделаем в сыновей узла g. Затем сделаем g братом узла f, сделав его сыном отца узла f. Эта операция называется *переброской*. Если отец узла f уже имел трех сыновей, то надо рекурсивно повторять переброску до тех пор, пока у всех узлов в дереве останется не более трех сыновей. Если у корня окажется четыре сына, то образуем новый корень с двумя новыми сыновьями, каждый из которых будет иметь в качестве двух своих сыновей двух из четырех сыновей старого корня.

Операция переброски входит в базовое множество элементарных преобразований H. Обозначим ее сложность через γ_1.

Из [3] следует, что сложность вставки нового объекта o не меньше $\gamma]\log_3|V|[$ и не больше $(\gamma+\gamma_1)[\log_2|V|]$. Таким образом,

$$\alpha_+ + \alpha_{f_q} + \gamma]\log_3|V|[\leqslant S_A(I,o) \leqslant \alpha_+ + \alpha_{f_q} + (\gamma+\gamma_1)[\log_2|V|].$$

Процедура удаления.

Для того, чтобы удалить объект o ищем лист, содержащий запись s, соответствующую объекту o и удаляем ее. Для этого начиная от корня ищем s в дереве. Пусть f — вершина, имеющая в качестве сына лист, соответствующий записи s. Удалим этот лист.

Если после удаления у вершины f осталось два сына, то заканчиваем процедуру.

Если же у вершины f остался только один сын, то **«склеим»** f с любым из ее братьев, объединив их листья. После этого, возможно f (точнее новая «склеенная») вершина будет единственным сыном своей родительской вершины. В этом случае повторим операцию «склейки» одним ярусом ближе к корню. Наконец, если после этих операций корень имеет только одного сына, то объявим этого сына новым корнем 2–3 дерева, а старый корень удалим.

Получившееся дерево все еще может не обладать 2–3 свойством, так как при «склейке» вершин количество сыновей может оказаться больше трех. Поэтому необходим еще один поход от листа (брата удаленного листа) до корня, проверяя число сыновей у вершин и по необходимости совершая переброски.

Операция склейки входит в базовое множество элементарных преобразований H. Обозначим ее сложность через γ_2.

Из [3] следует, что сложность удаления любого объекта o удовлетворяет соотношению:

$$\gamma]\log_3|V|[\leqslant R_A(I,o)\leqslant(\gamma+\gamma_1+\gamma_2)[\log_2|V|].$$

Объем $2-3$ дерева с $|V|$ вершинами варьируется от $\frac{3|V|-1}{2}$ до $2|V|-1$, а объем «прошитого» 2–3 дерева больше на $|V|-1$, следовательно,

$$\frac{5|V|-3}{2}\leqslant Q_A(I)\leqslant 3|V|-2.$$

Оценка погрешности.

Из совокупностей (4) и (8) получим условия, которым удовлетворяют объекты из множества $J_A(I,q)\setminus J(I,q)$ для $q=(t_q,x)$, то есть объекты «попадающие в погрешность»:

$$\begin{cases} x>1-\rho \\ y_i>x+\rho-1 \\ t_i-f_q^{-1}(y_i)\in[t_q-\tau_{\max}-C_1(x+\rho-1),t_q-\tau_{\max}+ \\ \qquad\qquad +f_q^{-1}(1-x+y_i-\rho)-f_q^{-1}(y_i)). \end{cases}$$

Заметим, что

$$f_q^{-1}(y_i)-f_q^{-1}(1-x+y_i-\rho)\geqslant C_2(x+\rho-1),$$

где

$$C_2=\frac{1}{\max\limits_{\xi\in[0,\tau^q_{\max}]}f'_q(\xi)}.$$

Тогда, используя лемму 3, оценим погрешность:

$$f_A(\rho,\lambda)=\lambda\int\limits_{1-\rho}^{1}\int\limits_{x+\rho-1}^{1}C_1(x+\rho-1)-(f_q^{-1}(y_i)-f_q^{-1}(1-x+y_i-\rho))dy_idx\leqslant$$

$$\leqslant\lambda\int\limits_{1-\rho}^{1}\int\limits_{x+\rho-1}^{1}(C_1-C_2)(x+\rho-1)dy_idx=\lambda(C_1-C_2)(\frac{\rho^2}{2}-\frac{\rho^3}{3})=$$

$$=\lambda C(\frac{\rho^2}{2}-\frac{\rho^3}{3}).$$

В результате, получаем следующее утверждение.

Утверждение 1. *Пусть область наблюдения совпадает с областью решения и* $f'(\tau+\tau')-f'_q(\tau) \leqslant 0$ *для любого* $\tau \in [0, \tau^q_{\max}]$ *и любого* τ'*, такого, что* $\tau+\tau' \in [0, \min\{\tau_{\max}, \tau^q_{\max}\}]$*. Тогда алгоритм A решает задачу об опасной близости* $I = (\rho, f, f_q, \tilde{Q}, V, \lambda)$*. Для сложности поиска ответа на любой запрос q, вставки любого объекта-данного o, удаления любого объекта-данного o по алгоритму A, для объема и погрешности алгоритма A верны соответственно следующие оценки*

$$\begin{gathered}
\gamma]\log_3|V|[+4\alpha_+ + 2\alpha_{\leqslant} + \alpha_f \leqslant T_A(I,q) \leqslant \\
\leqslant \gamma[\log_2|V|] + 5\alpha_+ + 3\alpha_{\leqslant} + 2\alpha_f + \alpha_*, \\
\alpha_+ + \alpha_{f_q} + \gamma]\log_3|V|[\leqslant S_A(I,o) \leqslant \alpha_+ + \alpha_{f_q} + (\gamma+\gamma_1)[\log_2|V|], \\
\gamma]\log_3|V|[\leqslant R_A(I,o) \leqslant (\gamma+\gamma_1+\gamma_2)[\log_2|V|], \\
\frac{5|V|-3}{2} \leqslant Q_A(I) \leqslant 3|V|-2, \\
f_A(\rho,\lambda) \leqslant \lambda\left(\frac{1}{\min\limits_{\xi\in[0,\tau^q_{\max}]} f'_q(\xi)} - \frac{1}{\max\limits_{\xi\in[0,\tau^q_{\max}]} f'_q(\xi)}\right)\left(\frac{\rho^2}{2} - \frac{\rho^3}{3}\right),
\end{gathered}$$

где γ *есть сложность операции прохождения внутренней вершины,* γ_1 *— сложность операции переброски,* γ_2 *— сложность склейки,* α_f *— сложность вычисления* f^{-1}*,* α_{f_q} *— сложность вычисления* f_q^{-1}*,* α_+ *— сложность операции сложения,* $\alpha_{\leqslant}$ *— сложность операции сравнения,* α_* *— сложность умножения.*

Следствие 1. *Если функция* f_q *линейна, то задача решается без погрешности.*

Доказательство. Если f_q линейна, то погрешность оценивается как $f_A(\rho,\lambda) \leqslant 0$, но поскольку погрешность не может быть отрицательной, то $f_A(\rho,\lambda) = 0$ и задача решается без погрешности.

Следствие 2. *Если в условиях утверждения 1 область наблюдения имеет вид* $[0, 1+\rho] \times [0,1]$*, то задача об опасной близости* $I = (\rho, f, f_q, Q, V)$ *будет решаться без погрешности, с такими же*

сложностями вставки, удаления и таким же объемом. Сложность поиска изменяется на константу и будет оцениваться как

$$\gamma]\log_3|V|[+2\alpha_+ + \alpha_\leqslant + 2\alpha_f \leqslant T_A(I,q) \leqslant \gamma[\log_2|V|] + 2\alpha_+ + \alpha_\leqslant + 2\alpha_f.$$

Доказательство. Для каждого запроса $q = (t_q, x)$ алгоритм A будет выдавать в ответ $J_A(I,q)$ объекты $o_i = (t_i, y_i)$ из библиотеки V, удовлетворяющие совокупности (7) из леммы 2, то есть задача будет решаться без погрешности. Незначительно изменится процедура поиска: границами отрезка поиска теперь будут

$$A(q) = t_q - f^{-1}(x+\rho),$$
$$B(q) = t_q - f^{-1}(\max\{x-\rho, 0\}).$$

Сложность их вычисления равняется $2\alpha_+ + \alpha_\leqslant + 2\alpha_f$.

Теорема 1 является простым следствием утверждения 1, а теорема 2 вытекает из следствия 2.

Список литературы

[1] Гасанов Э. Э., Кудрявцев В. Б. Теория хранения и поиска информации. М.: ФИЗМАТЛИТ, 2002.

[2] Скиба Е. А. Логарифмическое решение задачи об опасной близости // Интеллектуальные системы. 2007. Т. 11, вып. 1–4. С. 693–719.

[3] Лапшов И. С. Динамические базы данных с оптимальной по порядку временной сложностью // Дискретная математика. 2008. Т. 20, вып. 3. С. 89–100.