Лекция 9.

Задача одномерного интервального поиска.
Оценки сложности одномерного интервального
поиска при вариации базового множества.

1 Одномерный интервальный поиск

Интервальный поиск имеет очевидное приложение к системам баз данных. В базе данных, содержащей записи о служащих некоторой компании, каждая запись имеет несколько атрибутов, таких, как возраст, жалование и т. д., и может рассматриваться как точка в n-мерном пространстве, в котором каждый атрибут соответствует измерению, а число измерений пространства n равно числу атрибутов. Типичная задача интервального поиска для двух измерений заключается в выявлении всех служащих, чьи возраст и жалованье находятся в заданных интервалах. Другой пример можно найти у Д. Кнута. Он рассматривал базу данных городов США с координатами в виде широты и долготы. К такой базе естественен вопрос о перечислении всех городов, попадающих в некоторой прямоугольник-запрос. Это типичная двумерная задача интервального поиска.

Исследованию задачи интервального поиска (в другом переводе с английского — регионального поиска) посвящено большое количество работ.

В данном разделе рассматривается следующая задача. Дано конечное множество точек из отрезка $[0, 1]$. Запрос на поиск задает некий отрезок $[a, b] \subseteq [0, 1]$. Надо перечислить все точки из множества, которые попадают в отрезок $[a, b]$. Это известная геометрическая задача поиска, называемая одномерной задачей интервального поиска. В данном разделе исследуется вопрос, какие алгоритмы возникают, если ограничивать набор доступных средств, или, более формально, при различных базовых множествах. Получены также некоторые нижние оценки, с помощью

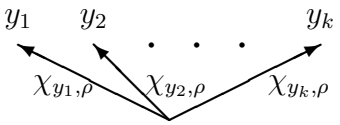

Рис. 1: Информационный граф переборного алгоритма

которых показывается, что соответствующие полученные алгоритмы не могут быть существенно улучшены при данных ограничениях на набор доступных средств.

Итак, в одномерной задаче интервального поиска множество записей Y_{int} есть отрезок $[0,1]$, множество запросов X_{int} есть множество отрезков $[u,v] \subseteq [0,1]$, или множество пар точек (u,v), таких, что $0 \le u \le v \le 1$, то есть $X_{int} = \{x = (u,v) : 0 \le u \le v \le 1\}$.

На X_{int} задано вероятностное пространство $\langle X_{int}, \sigma, \mathbf{P} \rangle$, где σ — алгебра подмножеств множества X_{int}, содержащая все прямоугольники со сторонами параллельными осям координат и прямоугольные равнобедренные треугольники с катетами также параллельными осям координат, $\mathbf{P}$ — вероятностная мера на σ. Будем считать, что мера $\mathbf{P}$ определяется функцией плотности распределения вероятностей $p(u,v)$, то есть для любого $B \in \sigma$

$$\mathbf{P}(B) = \int_B p(u,v)\,du\,dv.$$

Для удобства будем считать, что $p(u,v)$ определена на всем квадрате $[0,1] \times [0,1]$, но $p(u,v) = 0$ при $(u,v) \notin X_{int}$.

Отношение поиска, которое будем обозначать через ρ_{int}, определяется соотношением

$$(u,v)\rho_{int}y \iff u \le y \le v,$$

где $(u,v) \in X_{int},\ y \in Y_{int}$.

Тип $S_{int} = \langle X_{int}, Y_{int}, \rho_{int}, \sigma, \mathbf{P} \rangle$ будем называть типом одномерного интервального поиска.

В следующих пунктах мы исследуем задачи типа S_{int} при различных базовых множествах.

2 Случай базового множества характеристических функций

Пусть

$$F_1 = \{\chi_{a,\rho_{int}} : a \in Y_{int}\}, \tag{1}$$
$$\mathcal{F}_1 = \langle F_1, \emptyset \rangle.$$

Отметим, что $N_f \in \sigma$ для любого предиката $f \in F_1$. Кроме того, понятно, что $\mathcal{F}_1$ полно для типа S_{int}, так как для произвольной библиотеки $V = \{y_1, \ldots, y_k\}$ дерево, изображенное на рисунке 1 и соответствующее переборному алгоритму, решает ЗИП $I = \langle X_{int}, V, \rho_{int} \rangle$.

Теорема 1. *Для любой ЗИП I типа S_{int} $T(I, \mathcal{F}_1) = k$.*

Доказательство. Нетрудно убедиться, что для любого $a \in Y_{int}$ точка $(a, a) \in N_{\chi_{a,\rho_{int}}}$ и для любого $a' \in Y_{int}$, такого, что $a' \neq a$ выполняется $(a, a) \notin N_{\chi_{a',\rho_{int}}}$, то есть мы находимся в условиях теоремы об оптимальности перебора, откуда следует, что $T(I, \mathcal{F}_1) = k$.

Тем самым теорема доказана.

3 Базовое множество логарифмического поиска

Пусть

$$F_1 = \{\chi_{a,\rho_{int}} : a \in Y_{int}\}, \tag{2}$$
$$G_1 = \{g_{\leq,a}(u, v) = \begin{cases} 1, & \text{если } u \leq a \\ 2, & \text{если } u > a \end{cases} \; : \; a \in [0, 1]\}, \tag{3}$$
$$\mathcal{F}_1 = \langle F_1, G_1 \rangle. \tag{4}$$

Отметим, что $N_f \in \sigma$ для любого предиката $f \in F_1 \cup \widehat{G}_1$, т.е. базовое множество $\mathcal{F}_1$ измеримо. Поскольку для любой записи $y \in Y_{int}$ $\chi_{y,\rho_{int}} \in F_1$, то $\mathcal{F}_1$ полно для типа S_{int}.

Справедлива следующая теорема.

Теорема 1. *Пусть $I = \langle X_{int}, V, \rho_{int} \rangle$ — ЗИП типа S_{int}, где $V = \{y_1, \ldots, y_k\}$, причем $0 \leq y_1 \leq \ldots \leq y_k \leq 1$; $\mathcal{F}_1$ — базовое множество, определяемое соотношениями (2) – (4). Тогда*

$$T(I, \mathcal{F}_1) \leq \sum_{i=1}^{k-1} \mathbf{P}(O(y_i, \rho_{int}))+] \log_2 k[+1.$$

Доказательство. Построим первое дерево бинарного поиска для библиотеки V, но только нагрузку переключательных вершин мы будем брать из множества G_2, а вместо предикатов $f_{=,a}$ использовать предикаты $\chi_{a,\rho_{int}} \in F_1$. Обозначим это дерево через D.

Для обозначения V_β будет использован тот же смысл, что и для бинарного поиска, т.е. V_β множество записей, соответствующих листьям, схемно достижимым из β.

Для удобства дальнейшего изложения назовем множество переключательных ребер дерева D переключательной частью дерева D, а множество предикатных ребер — предикатной частью дерева D.

Обозначим лист, которому соответствует запись y_i, через α_i ($i = \overline{1,k}$). Из каждого листа α_i ($i = \overline{1,k-1}$) выпустим ребро, ведущее в лист α_{i+1}, и припишем ему предикат $\chi_{y_{i+1},\rho_{int}} \in F_1$.

Это множество из $(k-1)$-го ребра назовем правосторонней концевой цепью.

Полученный граф обозначим через U_V.

Отметим одно достаточно очевидное свойство графа U_V. Пусть β — вершина графа U_V, не являющаяся листом. Пусть $V_\beta = \{y_i, y_{i+1}, \ldots, y_j\}$. Тогда

- если $i = 1$ и $j = k$, то $N_{\varphi_\beta} = X_{int}$;
- если $i = 1$ и $j \neq k$, то$N_{\varphi_\beta} = \{(u, v) \in X_{int} : 0 \leq u \leq y_j\}$;
- если $i \neq 1$ и $j = k$, то $N_{\varphi_\beta} = \{(u, v) \in X_{int} : y_{i-1} < u \leq 1\}$;
- если $i \neq 1$ и $j \neq k$, то $N_{\varphi_\beta} = \{(u, v) \in X_{int} : y_{i-1} < u \leq y_j\}$.

Покажем, что U_V решает ЗИП I. По критерию допустимости информационных графов, для этого достаточно показать, что $\varphi_{\alpha_i} = \chi_{y_i,\rho_{int}}$ для любого $i \in \{\overline{1,k}\}$.

Доказывать это будем индукцией по i.

Базис индукции. $i = 1$.

В лист α_1 ведет единственное ребро из вершины, которую обозначим через β'. Очевидно, $\varphi_{\alpha_1} = \varphi_{\beta'} \& \chi_{y_1,\rho_{int}}$. Через y_j обозначим максимальную запись в библиотеке $V_{\beta'}$. Примем $c = y_j$, если $j \neq k$, и $c = 1$, если $j = k$. Очевидно, $c \geq y_1$. Тогда

$$\begin{aligned} N_{\varphi_{\alpha_1}} &= \{(u,v) \in N_{\varphi_{\beta'}} : u \leq y_1,\ v \geq y_1\} = \\ &= \{(u,v) \in X_{int} : 0 \leq u \leq c,\ u \leq y_1,\ v \geq y_1\} = \\ &= \{(u,v) \in X_{int} : 0 \leq u \leq y_1,\ v \geq y_1\} = O(y_1, \rho_{int}), \end{aligned}$$

то есть $\varphi_{\alpha_1} = \chi_{y_1,\rho_{int}}$.

Индуктивный переход. Предположим, что для некоторого $i \geq 1$ функция фильтра листа α_i равна $\varphi_{\alpha_i} = \chi_{y_i,\rho_{int}}$.

Рассмотрим лист α_{i+1} $(i + 1 \leq k)$. В него ведут два ребра: из листа α_i и некоторой вершины β, не являющейся листом. Пусть y_l и y_j — минимальная и максимальная записи в библиотеке V_β соответственно. Пусть

$$c_1 = \begin{cases} y_{l-1}, & \text{если } l \neq 1, \\ 0, & \text{если } l = 1, \end{cases}$$

$$c_2 = \begin{cases} y_j, & \text{если } j \neq k, \\ 1, & \text{если } j = k, \end{cases}$$

Очевидно, что $c_1 \leq y_i$ и $c_2 \geq y_{i+1}$. Тогда

$$\varphi_{\alpha_{i+1}} = (\varphi_{\alpha_i} \& \chi_{y_{i+1},\rho_{int}}) \vee (\varphi_\beta \& \chi_{y_{i+1},\rho_{int}}) = (\varphi_{\alpha_i} \vee \varphi_\beta) \& \chi_{y_{i+1},\rho_{int}},$$

или

$$\begin{aligned} N_{\varphi_{\alpha_{i+1}}} &= (\{(u,v) \in X_{int} : u \leq y_i,\ v \geq y_i\} \cup \\ &\quad \cup \{(u,v) \in X_{int} : c_1 < u \leq c_2\}) \cap \\ &\quad \cap \{(u,v) \in X_{int} : u \leq y_{i+1},\ v \geq y_{i+1}\} = \\ &= \{(u,v) \in X_{int} : u \leq y_{i+1},\ v \geq y_{i+1}\} = O(y_{i+1}, \rho_{int}). \end{aligned}$$

Тем самым мы показали, что U_V решает I.

Покажем, что

$$T(U_V) \leq \sum_{i=1}^{k-1} \mathbf{P}(O(y_i, \rho_{int})) +] \log_2 k [+ 1.$$

Рассмотрим произвольный запрос $x \in X_{int}$. Поскольку через переключательную часть дерева D пролегает единственная проводящая цепь, и ее длина не превышает $]\log_2 k[-1$, то на любом запросе будет вычислено не более чем $]\log_2 k[-1$ переключателей. Далее на запросе x вычисляются один или два предиката, соответствующих предикатным ребрам из предикатной части дерева D, растущим из вершины, в которую ведет проводящая цепь. Таким образом, суммарная сложность дерева D не превышает $]\log_2 k[+1$.

Осталось подсчитать сложность правосторонней концевой цепи. Из каждого листа α_i $(i = \overline{1, k-1})$ исходит одно ребро и его сложность равна $\mathbf{P}(O(y_i, \rho_{int}))$.

Следовательно,

$$T(I, \mathcal{F}_2) \leq T(U_V) \leq \sum_{i=1}^{k-1} \mathbf{P}(O(y_i, \rho_{int}))+]\log_2 k[+1.$$

Тем самым теорема 1 доказана. □

Алгоритм поиска, соответствующий графу U_V, построенному в доказательстве теоремы 1, состоит в следующем. В упорядоченной по возрастанию библиотеке с помощью бинарного поиска за $\log_2 k$ шагов находится самая левая запись, находящаяся не левее левого конца запроса. Затем слева направо, начиная с найденной записи, просматриваются записи и сравниваются с правым концом запроса, и если оказывается, что очередная запись не больше правого конца, то эта запись включается в ответ, а если больше, то поиск прекращается, то есть — это традиционный алгоритм решения данной задачи поиска с помощью структуры данных, называемой прошитым двоичным деревом. Считается, что описанный алгоритм оптимален как по времени поиска, так и по памяти, но имеется в виду время в худшем случае, а оптимальность понимается по порядку.