Задача о поиске ближайшего бит-вектора в большой базе бит-векторов

В поисковых системах и некоторых специализированных базах данных бывает важно хранить объекты некоторого типа (строки, числа, наборы чисел, графические изображения и др.) таким образом, чтобы для любого нового объекта быстро находить какой-нибудь ближайший к нему объект из базы. Конечно, можно считать расстояние от нового объекта до каждого из старых объектов и находить наименьшее расстояние, но при большом объёме базы этот способ слишком медленный.

Поиск ближайшего целого числа

В качестве расстояния $d(x, y)$ между целыми числами x и y берут $|x - y|$. Считаем, что величины целых чисел заранее ограничены (например, это числа типа `int`), так что операция сравнения двух целых чисел производится за постоянное время.

Если дано n целых чисел, то за время порядка $n \log n$ можно составить из них упорядоченный массив a размера n. После этого для произвольного поступившего числа x можно за время порядка $\log n$, используя двоичный поиск, найти такой индекс i, что $a_i \leqslant x \leqslant a_{i+1}$. Ближайшим объектом к x будет либо a_i, либо a_{i+1}.

Другой способ: за время порядка $n \log n$ поместить начальные n чисел в «красно-чёрное дерево». Элементы этого дерева отсортированы, а высота не превосходит $\lceil \log_2 n \rceil + 1$. Тогда для произвольного поступившего числа x можно за время порядка $\log n$ найти подходящее место для x в этом дереве, и ближайшим объектом к x будет один из соседних узлов.

Красно-чёрное дерево лучше, чем упорядоченный массив, так как позволяет быстро добавлять и удалять элементы.

Поиск ближайшей пары целых чисел

Расстояние между парами целых чисел можно определять разными способами. Во многих случаях удобно «L_1-расстояние»:

$$d((a_1, a_2), (b_1, b_2)) = |a_1 - b_1| + |a_2 - b_2|.$$

Говорят, будто для пар целых чисел тоже найдена структура (специальный вид деревьев), которая позволяет быстро находить ближайший элемент. Можно поискать информацию по этому поводу в книгах Д. Кнута и Р. Сэджвика.

Скорее всего, этот подход обобщается на векторы любой наперёд заданной длины (скажем, длины 3 или длины 4), но затраты быстро растут с увеличением длины вектора.

Поиск ближайшего бит-вектора

Дано положительное целое число L и n двоичных векторов длины L (*двоичный вектор* или *бит-вектор* есть вектор, координаты которого могут принимать только два значения: 0 или 1).

Таким образом, суммарная длина исходных векторов равна nL. Это число может быть очень большим (например, 10^{12}).

В качестве расстояния между бит-векторами a и b берётся *расстояние Хэмминга*, т. е. число различных битов:

$$d(a, b) = \sum_{k=1}^{L} |a_i - b_i|.$$

Нужно придумать такой способ хранения исходных бит-векторов, чтобы для любого нового бит-вектора длины L можно было достаточно быстро (например, за $L \log n$ операций) найти какой-нибудь ближайший к нему из начальных бит-векторов.

Близкая задача: поиск кратчайшего расстояния

Формулировка предыдущей задачи довольно туманная («найти такой способ хранения ...»). Вот близкая задача с более простой формулировкой.

Дано n двоичных векторов длины L: $a_1, \ldots, a_n$. Вычислить минимум среди чисел $d(a_i, a_j)$, где $1 \leqslant i, j \leqslant n$, $i \neq j$.

Тривиальный алгоритм считает этот минимум за время порядка Ln^2.

Нужно найти алгоритм, который работает гораздо быстрее. Например, за время порядка $Ln \log n$.

Отметим, что положительное решение предыдущей задачи («правильный способ хранения» битвекторов) сразу давало бы решение этой задачи.