Хеш-память

Хеширование является самым быстродействующим из известных методов программного поиска. Это его качество особенно проявляется при работе с наборами данных большого размера. Данный метод удобен тем, что не требует ни какого-либо упорядочивания, ни сортировки ключевых слов. Высокая скорость выполнения операции хеширования обусловлена тем, что элементы данных запоминаются, а впоследствии выбираются из ячеек памяти, адреса которых являются простыми арифметическими функциями содержимого соответствующих ключевых слов.

Адреса, получаемых из ключевых слов методом хеширования называются хеш-адресами. Так как хеширование проецирует более большое множество значений в более малое, то существует вероятность того, что значение хеш-адреса от нескольких ключевых слов будут одинаковыми, данное явление получило название «коллизия».

Преобразование ключевых слов в допустимые адреса памяти в общем случае выполняется с помощью некоторой функции хеширования

**Метод цепочек.** Технология сцепления элементов состоит в том, что элементы множества, которым соответствует одно и то же хеш-значение, связываются в цепочку-список. Операции поиска или удаления данных требуют просмотра всех элементов соответствующей ему цепочки, чтобы найти в ней элемент с заданным ключом. Для добавления данных нужно добавить элемент в конец или начало соответствующего списка, и, в случае если коэффициент заполнения станет слишком велик, увеличить размер массива и перестроить таблицу.

При предположении, что каждый элемент может попасть в любую позицию таблицы с равной вероятностью и независимо от того, куда попал любой другой элемент, среднее время работы операции поиска элемента составляет O(1+k), где k – коэффициент заполнения таблицы.

**Метод открытой адресации**. В отличие от хеширования с цепочками, при открытой адресации никаких списков нет, а все записи хранятся в самой хеш-таблице. Каждая ячейка таблицы содержит либо элемент динамического множества, либо NULL. В этом случае, если ячейка с вычисленным индексом занята, то можно просто просматривать следующие записи таблицы по порядку до тех пор, пока не будет найден ключ K или пустая позиция в таблице.

При любом методе разрешения коллизий необходимо ограничить длину поиска элемента. Если для поиска элемента необходимо более 3 – 4 сравнений, то эффективность использования такой хеш-таблицы пропадает и ее следует реструктуризировать (т.е. найти другую хеш-функцию), чтобы минимизировать количество сравнений для поиска элемента

**Таблица Прямого Доступа.** Простейшей организацией таблицы, обеспечивающей идеально быстрый поиск, является таблица прямого доступа. В такой таблице ключ является адресом записи в таблице или может быть преобразован в адрес, причем таким образом, что никакие два разных ключа не преобразуются в один и тот же адрес. При создании таблицы выделяется память для хранения всей таблицы и заполняется пустыми записями. Затем записи вносятся в таблицу – каждая на свое место, определяемое ее ключом. При поиске ключ используется как адрес и по этому адресу выбирается запись. Если выбранная запись пустая, то записи с таким ключом вообще нет в таблице. Таблицы прямого доступа очень эффективны в использовании, но, к сожалению, область их применения весьма ограничена.

*Разница состоит в размере в пространстве ключей. При хешировании пространство ключей меньше пространства записей и, соответственно, возможны (и они случаются) коллизии, чего нет в таблиах прямого доступа.*

*При хешировании мы используем некую функцию от ключа, которая возвращает нам адрес (числовое значение) в таблице. В таблицах прямого доступа у нас сам ключ интерпретируется как число. Пример с фамилией это демонстрирует: строка длиной в 10 символов интерпретируется как 10-байтовое число*

**Остатки от деления.** Простейшей хеш-функцией является деление по модулю числового значения ключа Key на размер пространства записи HashTableSize. Результат интерпретируется как адрес записи. Следует иметь в виду, что такая функция хорошо соответствует первому, но плохо – последним трем требованиям к хеш-функции и сама по себе может быть применена лишь в очень ограниченном диапазоне реальных задач. Однако операция деления по модулю обычно применяется как последний шаг в более сложных функциях хеширования, обеспечивая приведение результата к размеру пространства записей.

**Метод свертки.** Цифровое представление ключа разбивается на части, каждая из которых имеет длину, равную длине требуемого адреса. Над частями производятся определенные арифметические или поразрядные логические операции, результат которых интерпретируется как адрес. Например, для сравнительно небольших таблиц с ключами – символьными строками неплохие результаты дает функция хеширования, в которой адрес записи получается в результате сложения кодов символов, составляющих строку-ключ.

В качестве хеш-функции также применяют функцию преобразования системы счисления. Ключ, записанный как число в некоторой системе счисления P, интерпретируется как число в системе счисления Q>P. Обычно выбирают Q=P+1. Это число переводится из системы Q обратно в систему P, приводится к размеру пространства записей и интерпретируется как адрес.

**Закрытое хеширование.** В отличие от открытого хеширования закрытое не требует каких-либо дополнительных структур данных. В ячейках таблицы хранятся не указатели, а элементы исходного массива, доступ к каждому из которых осуществляется по хеш-значению ключа, при этом одна ячейка может содержать только один элемент. Применим простую методику, которая называется линейным хешированием. При линейном хешировании hi(x) = (h(x) + і) mod В.

**Открытое хеширование.** Основная идея метода заключается в том, что множество данных (возможно, очень большое) разбивается на конечное число классов. Для В классов, пронумерованных от 0 до В-1, строится хеш-функция h такая, что для любого элемента x исходного множества функция h(x) принимает целочисленное значение из интервала 0, …, В-1, которое соответствует классу, которому принадлежит элемент x. Элемент x называют ключом, h(x) – хеш-значением х, а классы – сегментами.