ЛАБОРАТОРНАЯ РАБОТА № 1
ПОСТРОЕНИЕ ХЕШ - ТАБЛИЦЫ

1 Цель работы

Целью работы является построение хеш-таблицы, содержащей заданную последовательность элементов (ключей).

2 Основные теоретические положения

Хеширование является самым быстродействующим из известных методов программного поиска. Это его качество особенно проявляется при работе с наборами данных большого размера.

Данный метод весьма удобен тем, что не требует ни какого упорядочивания, ни сортировки ключевых слов.

Высокая скорость выполнения операций хеширования обусловлена тем, что элементы данных запоминаются, и впоследствии выбираются из ***ячеек памяти, адреса которых являются простыми арифметическими функциями содержимого соответствующих ключевых слов.***

Адреса, получаемые из ключевых слов методом хеширования, называются ***хеш-адресами***.

Таким образом, идея хеширования состоит в том, чтобы взять некоторые характеристики ключа и использовать полученную частичную информацию в качестве основы поиска.

Мы вычисляем ***хеш-функцию*** f(k), т.е. производим над ключом k некоторое арифметическое вычисление и получаем ***хеш-адрес***, указывающий индекс в таблице, где хранится элемент k и ассоциированная с ним информация.

С хеш-функцией связана так называемая ***хеш-таблица***, ячейки которой пронумерованы и хранят сами данные или ссылки на данные.

Часто отображение, осуществляемое хеш-функцией, является отображением "многие к одному" и приводит к ***коллизиям***.

При возникновении коллизии два или более ключа ассоциируются с одной и той же ячейкой хеш-таблицы.

Поскольку два ключа не могут занимать одну и ту же ячейку в таблице, мы должны разработать стратегию разрешения коллизий, т.е. использовать какой-нибудь метод, указывающий альтернативное местоположение.

Следовательно, чтобы использовать хеш-таблицу, программист должен выбрать хеш-функцию f(k) и метод разрешения коллизий.

Хеш-функция, не порождающая коллизий, называется ***совершенной*** хеш-функцией. Нет смысла искать совершенную хеш-функцию, если множество ключей не является постоянным. Однако для таблицы символов компилятора (содержащую зарезервированные слова while, template, class и т.д.) совершенная хеш-функция крайне желательна. Тогда для определения того, является ли некоторый идентификатор зарезервированным словом, потребуется лишь одна проба.

Найти совершенную хеш-функцию даже для конкретного набора ключей очень сложно. Кроме того, если данный набор ключей пополнится новыми ключами, совершенная хеш-функция, как правило, перестает быть совершенной.

Хорошая хеш-функция должна удовлетворять следующим требованиям:

1. Хеш-функция должна отображать ключ в целое число.
2. При этом количество коллизий должно быть ограниченным.
3. Вычисление самой хеш-функции - очень быстрым.

Рассмотрим методы, которые удовлетворяют этим требованиям.

Наиболее часто используется ***метод деления***, требующий двух шагов. Сперва ключ должен быть преобразован в целое число, а затем полученное значение вписывается в диапазон 0...n-1 с помощью оператора получения остатка. На практике метод деления используется в большинстве приложений с хешированием.

Метод умножения обладает одним важным качеством. В начале с его помощью получается нормализованный хеш-адрес, расположенный в интервале [0, 1). Этот адрес далее масштабируется в соответствии с истинным размером таблицы. Нормализованным хеш-адресом назовем величину

h(k) = ck mod t, где c - некоторая константа из интервала [0, 1).

Итоговый хеш-адрес вычисляется как f(k) = t h(k), где t - размер таблицы хеширования. Качество хеширования по методу умножения зависит от выбора константы c. Хорошие результаты дает c=0.618034.

Метод середины квадрата предусматривает преобразование ключа в целое число, возведение его в квадрат и возвращение в качестве значения функции последовательность цифр, извлеченных из середины этого квадрата.

Метод алгебраических преобразований над ключом позволяет получать хеш-адреса путем выполнения алгебраических действий над цифрами, из которых состоит ключ. Например, сложение по модулю пары соседних цифр ключа, сложение первой и последней цифр ключа, получение суммы всех цифр ключа.

Несмотря на то, что два или более ключей могут хешироваться одинаково, они не могут занимать в хеш-таблице одну и ту же ячейку.

Нам остаются два пути: либо найти для нового ключа другую позицию в таблице, либо создать для каждого значения хеш-функции отдельный список, в котором будут все ключи, отображающиеся при хешировании в это значение. Оба варианта представляют собой две классические стратегии разрешения коллизий - ***открытую адресацию с линейным или квадратичным опробованием*** и ***метод цепочек.***

Открытая адресация с линейным или квадратичным опробованием.

Эта методика предполагает, что каждая ячейка таблицы помечена как незанятая.

Поэтому при добавлении нового ключа всегда можно определить, занята ли данная ячейка таблицы или нет.

Если да, алгоритм осуществляет "опробование" по кругу, пока не встретится "открытый адрес" (свободное место).

Отсюда и название метода.

Если размер таблицы велик относительно числа хранимых там ключей, метод работает хорошо, поскольку хеш-функция будет равномерно рассеивать ключи по всему диапазону и число коллизий будет минимальным. По мере того как коэффициент заполнения таблицы приближается к 1, эффективность процесса заметно падает.

В линейном пробинге к хеш-адресу прибавляется (или вычитается) по единице до тех пор, пока не обнаружится незанятая ячейка.

$a_0 = f(k)$ - *первичный адрес,*

$a_i = (a_0 + i) \bmod t$, $(a_i = (a_0 + i) \% t)$,

где i – *номер пробы,*

t – *размер хеш- таблицы*,

при открытой адресации $t \geq 1.5\ N$, где

N - *общее количество элементов*.

Недостаток линейных проб заключается в том, что вычисленные адреса имеют тенденцию группироваться вокруг первичных адресов.

Первичное группирование ячеек можно устранить, применяя нелинейные методы, в частности квадратичные пробы, при которых последовательность резервных адресов строится с использованием некоторой квадратичной функции.

$a_0 = f(k)$ - *первичный адрес,*

$a_i = (a_0 + i^2) \bmod t$, $(a_i = (a_0 + i^2) \% t)$,

где i – *номер пробы,*

t *– размер хеш- таблицы*,

при открытой адресации **t ≥ 1.5 N**, где

N *- общее количество элементов*.

Небольшой недостаток квадратичных проб заключается в том, что при поиске пробуются не все строки таблицы, т.е. при включении элемента может не найтись свободного места, хотя на самом деле оно есть.

Если размер t - простое число, то при квадратичных пробах просматривается по крайней мере половина таблицы.

Стратегия разрешения коллизий: ***Метод цепочек***.

При этом подходе к хешированию таблица рассматривается как массив связанных списков.

Каждая такая цепочка называется блоком и содержит записи, отображаемые хеш-функцией в один и тот же табличный адрес.

Если таблица является массивом связанных списков, то элемент данных просто вставляется в соответствующий список в качестве нового узла.

Чтобы обнаружить элемент данных, нужно применить хеш-функцию для определения нужного связанного списка и выполнить там последовательный поиск.

Размер хеш-таблицы t при использовании метода цепочек берется из интервала ***t = [1/3*N…1/2*N],*** где ***N*** - количество элементов.

В общем случае метод цепочек быстрее открытой адресации, так как просматривает только те ключи, которые попадают в один и тот же табличный адрес.

Кроме того, открытая адресация предполагает наличие таблицы фиксированного размера, в то время как в методе цепочек элементы таблицы создаются динамически, а длина списка ограничена лишь количеством памяти. Основным недостатком метода цепочек являются дополнительные затраты памяти на поля указателей. В общем случае динамическая структура метода цепочек более предпочтительна для хеширования.

3 Последовательность выполнения работы

1. Сгенерировать m неповторяющихся элементов размерностью n с использованием датчика случайных чисел.
2. Построить хеш-таблицу, используя заданную хеш-функцию и заданный способ разрешения коллизий.
3. При разрешении коллизий с использованием открытой адресации с квадратичными пробами при размещении и поиске элементов проверяются не все строки таблицы, т.е. при включении элемента может не найтись свободного места.

Поэтому в программе ограничить количество проб этим способом, например, до 30 попыток и продолжить разрешение коллизии с помощью линейных проб.

4. Произвести анализ следующих параметров полученной хеш-таблицы:

a) коэффициент заполнения таблицы α, который равен отношению занятой памяти ко всей имеющейся или, другими словами, отношению количества заполненных ячеек к размеру таблицы;

b) среднее число шагов, необходимых для размещения элементов в таблицу.

4 Формы отчетности

1. На экране дисплея должны отображаться: последовательность сгенерированных ключей, хеш-таблица, рассчитанные параметры.
2. Электронный вариант программы.
3. Отчет о проделанной лабораторной работе, содержащий следующие пункты:

1) задание на лабораторную работу, содержащее используемые хеш-функцию, метод устранения коллизий;
2) входные и выходные данные;
3) порядок построения хеш-таблицы;
4) формулы расчета требуемых параметров хеш-таблицы.

5 Контрольные вопросы

1) Что такое хеш-адрес, хеш-функция?
2) Что называется коллизией?
3) Какие методы устранения коллизий вы знаете?
4) Как рассчитывается размер хеш-таблицы при каждом из методов устранения коллизий?

ВАРИАНТЫ ЗАДАНИЙ

1. Построить хеш - таблицу, содержащую последовательность из m = 53 неповторяющихся элементов размерности n = 5. Элементы генерируются с помощью датчика случайных чисел.

Хеш - функция - сумма всех цифр элемента.

Метод разрешения конфликта - квадратичные пробы.

2. Построить хеш - таблицу, содержащую последовательность из m = 49 неповторяющихся элементов размерности n = 2. Элементы генерируются с помощью датчика случайных чисел.

Хеш - функция - средняя часть квадрата ключа.

Метод разрешения конфликта - квадратичные пробы.

3. Построить хеш - таблицу, содержащую последовательность из m = 49 неповторяющихся элементов размерности n = 3. Элементы генерируются с помощью датчика случайных чисел.

Хеш - функция – две последние цифры квадрата ключа.

Метод разрешения конфликта - квадратичные пробы.

4. Построить хеш - таблицу, содержащую последовательность из m = 48 неповторяющихся элементов размерности n = 4. Элементы генерируются с помощью датчика случайных чисел.

Хеш - функция - f(k) = (k -7)%t, где t-размер хеш-таблицы.

Метод разрешения конфликта - квадратичные пробы.

5. Построить хеш - таблицу, содержащую последовательность из m = 53 неповторяющихся элементов размерности n = 5. Элементы генерируются с помощью датчика случайных чисел.

Хеш - функция - три последние цифры ключа.

Метод разрешения конфликта - квадратичные пробы.

6. Построить хеш – таблицу, содержащую последовательность из m = 45 неповторяющихся элементов размерности n = 4. Элементы генерируются с помощью датчика случайных чисел.

Хеш – функция – сумма последних трех цифр ключа.

Метод разрешения конфликта – квадратичные пробы.

7. Построить хеш – таблицу, содержащую последовательность из m = 53 неповторяющихся элементов размерности n = 4. Элементы генерируются с помощью датчика случайных чисел.

Хеш-функция – сумма первой и последней цифр элемента.

Метод разрешения конфликта – квадратичные пробы.

8. Построить хеш – таблицу, содержащую последовательность из m=53 трехзначных неповторяющихся чисел (элементов).

Хеш-функция – f(k) = (k+29) % t, где k-элемент, t-размер хеш-таблицы.

Метод разрешения конфликта – квадратичные пробы.

9. Построить хеш - таблицу, содержащую последовательность из m = 48 неповторяющихся элементов размерности n = 4. Элементы генерируются с помощью датчика случайных чисел.

Хеш - функция - f(k) = (k -3) % t, t – размер хеш-таблицы.

Метод разрешения конфликта - квадратичные пробы.

10. Построить хеш - таблицу, содержащую последовательность из m = 46 неповторяющихся элементов размерности n = 8. Элементы генерируются с помощью датчика случайных чисел.

Хеш - функция - сложение всех цифр ключа.

Метод разрешения конфликта - квадратичные пробы.