ПРЕЗЕНТАЦИЯ НА ТЕМУ: «ХЕШ -ТАБЛИЦЫ»

Подготовила:

Студентка группы БАС-091

Гальченко Н. В.

СОВЕРШЕННЫЙ ХЕШ

- **Хеш-функцией** на множестве K возможных ключей называется функция h, которая отображает K в некоторый целочисленный интервал a...b.Другими словами, для любого key K функция дает значение i = h(key), такое, что a i b.
- На практике обычно интервал задается в форме 0...Z – 1 для некоторого целого Z. Хеш-функция h(key) задается в форме f(key) Mod(Z) (по модулю емкости контейнера), где функция f возвращает целочисленное значение, приводимое к нужному интервалу взятием по модулю. Массив, применяемый для хранения данных, имеет размерность Z.

Пример:

-ключ-символьная строка C++

```
// хеш-функция для символьной строки.
// Возвращает значение в диапазоне от 0 до 100
int HF(char *key)
{
  int len = strlen(key), hashf = 0;

  // если длина ключа равна 0 или 1, возвратить key[0].
  // иначе сложить первый и последний символ
  if (len <= 1)
    hashf = key[0];
  else
    hashf = key[0] + key[len-1];

  return hashf % 101;
}
```

-**метод деления** (division method)

```
int HF(int key)
{
return key % 100; // метод деления на 100
}
```

-метод **середины квадрата** (midsquare technique)

```
// возвратить средние 10 бит произведения key*key
int HF(int key);
{
  key *= key;          // возвести ключ в квадрат
  key >>= 11;          // отбросить 11 младших бит
  return key % 1024    // возвратить 10 младших бит
}
```

Принято считать, что хорошей, с точки зрения практического применения, является такая хеш-функция, которая удовлетворяет следующим условиям:

- функция должна быть простой с вычислительной точки зрения;
- функция должна распределять ключи в хеш-таблице наиболее равномерно;
- функция не должна отображать какую-либо связь между значениями ключей в связь между значениями адресов;
- функция должна минимизировать число коллизий – то есть ситуаций, когда разным ключам соответствует одно значение хеш-функции (ключи в этом случае называются *синонимами*).

Время выполнения хеш-функции:

Хеш-функция зависит только от ключей, а не от числа элементов, так что если count – это размерность нашей задачи, то время, затрачиваемое на вычисление функции, есть O(1) или O(l), если учитывается длина ключа – l, но можно предположить, что хеш-функция использует только первые K символов ключа, где K – константа.

- **Хеш-таблица** – это структура данных, реализующая интерфейс ассоциативного массива, то есть она позволяет хранить пары вида "ключ- значение" и выполнять три операции: операцию добавления новой пары, операцию поиска и операцию удаления пары по ключу. Хеш-таблица является массивом, формируемым в определенном порядке хеш-функцией.

Свойства хеш-таблицы:

- Выполнение операции в хеш-таблице начинается с вычисления хеш-функции от ключа. Получающееся хеш-значение является индексом в исходном массиве.
- Количество хранимых элементов массива, деленное на число возможных значений хеш-функции, называется *коэффициентом заполнения хеш-таблицы* (*load factor*) и является важным параметром, от которого зависит среднее время выполнения операций.
- Операции поиска, вставки и удаления должны выполняться в среднем за время O(1). Однако при такой оценке не учитываются возможные аппаратные затраты на перестройку индекса хеш-таблицы, связанную с увеличением значения размера массива и добавлением в хеш-таблицу новой пары.
- Механизм разрешения коллизий является важной

- Предположение, что в нашем примере все имена различаются по первой букве, приводит к тому, что хеш-функция для различных имен дает различные значения. В общем случае хеш-функция называется **совершенной**, если для разных значений ключа она вырабатывает разные значения. Для совершенной хеш-функции вставка и поиск требуют O(1) времени.

КОЛЛИЗИЙ

Для несовершенных функций встречаются *коллизии*, когда разные ключи дают одно и то же значение функции.

Методы разрешения коллизий

- метод цепочек (внешнее или открытое хеширование);
- метод открытой адресации (закрытое хеширование).

Метод цепочек

Метод открытой адресации

Последовательность, в которой просматриваются ячейки хеш-таблицы, называется **последовательностью проб**. В общем случае, она зависит только от ключа элемента, то есть это последовательность $h_0(x), h_1(x), ..., h_{n-1}(x)$, где x — ключ элемента, а $h_i(x)$ — произвольные функции, сопоставляющие каждому ключу ячейку в хеш-таблице. Первый элемент в последовательности, как правило, равен значению некоторой хеш-функции от ключа, а остальные считаются от него одним из приведённых ниже способов.

Последовательности проб:

- Линейное пробирование: ячейки хеш-таблицы последовательно просматриваются с некоторым фиксированным интервалом k между ячейками (обычно, $k = 1$), то есть i-й элемент последовательности проб — это ячейка с номером $(\text{hash}(x) + ik) \bmod N$. Для того, чтобы все ячейки оказались просмотренными по одному разу, необходимо, чтобы k было взаимно-простым с размером хеш-таблицы.
- *В общем,линейное опробование* сводится к последовательному перебору сегментов таблицы с некоторым фиксированным шагом:
- **адрес=h(x)+ci,**
- где i – номер попытки разрешить коллизию;
- k – константа, определяющая шаг перебора.

Квадратичное пробирование: интервал между ячейками с каждым шагом увеличивается на константу. Если размер хеш-таблицы равен степени двойки ($N = 2^p$), то одним из примеров последовательности, при которой каждый элемент будет просмотрен по одному разу, является:

hash(x) mod N, (hash(x) + 1) mod N, (hash(x) + 3) mod N, (hash(x) + 6) mod N, ...

- Проще говоря шаг перебора сегментов нелинейно зависит от номера попытки найти свободный сегмент:
- **адрес=h(x)+ci+di^2,**
- где i – номер попытки разрешить коллизию,
- c и d – константы.

Двойное хеширование: интервал между ячейками фиксирован, как при линейном пробировании, но, в отличие от него, размер интервала вычисляется второй, вспомогательной хеш-функцией, а значит может быть различным для разных ключей. Значения этой хеш-функции должны быть ненулевыми и взаимно-простыми с размером хеш-таблицы, что проще всего достичь, взяв простое число в качестве размера, и потребовав, чтобы вспомогательная хеш-функция принимала значения от 1 до $N - 1$.

- Т.е. *двойное хеширование*, основано на нелинейной адресации, достигаемой за счет суммирования значений основной и дополнительной хеш-функций:

Пример:

Закрытое хеширование, применяемое в классе HASH_TABLE библиотеки EiffelBase, не использует связных списков, а работает с массивом ARRAY[G]. В любой момент времени некоторые его позиции заняты, а некоторые – свободны:

Если при вставке хеш-функция вырабатывает уже занятую позицию, например, i, как показано на следующем рисунке, то применяемый механизм последовательно будет испытывать другие позиции – i1, i2, i3, пока не найдет свободную ячейку:

- Общий прием состоит в следующем: если хеш-функция вырабатывает позицию для первого кандидата i = f(key) Mod(Z), то последующие позиции определяются как i + increment, i +2 * increment, i +3 * incrementи так далее, все по модулю Z.
Величина increment вычисляется как f(key) Mod (Z -1). Такой алгоритм используется в классе HASH_TABLE библиотеки EiffelBase