Глава 1

Хэширование

Поиск с помощью дерева бинарного поиска в лучшем случае занимает время $O(N \log N)$, в худшем — $O(N)$. Но само построение сбалансированного дерева бинарного поиска достаточно сложный процесс и не всегда удобный.

Рассмотрим другой способ построения базы для поиска — хэширование. Вообще, хэширование достаточно распространено. Например, во многих базах для аутентификации используется не пароль, а хэш этого пароля. Т. е., Вы вводите пароль, вычисляется хэш этого пароля и проверяется с хранящимися данными.

В данном разделе не будут рассмотрены сложные функции хэширования. Рассмотрим только простейшие варианты.

Основная цель хэш-таблиц — расположить элементы в таблице по K ячейкам в соответствии со значением хэш-функции $h(x)$. Для каждого элемента x строится хэш-функция, такая, что значение $h(x)$ находится в интервале $[0, \ldots, B-1]$. Потом элемент x ставится в ячейку, соответствующую значению хэш-функции.

Элемент x часто называют ключом, $h(x)$ — хэш-значением. Значение хэш-функции должно быть обязательно целочисленным. В дальнейшем будет предполагать, что элементы располагаются по ячейкам хэш-функции равномерно и независимо.

Для простых хэш-функций достаточно часто бывают ситуации, когда несколько ключей имеют одинаковые значения хэш-функции. Такие ситуации называют *коллизиями*.

Предполагаем, что ключ всегда является целочисленным значением. Если, например, ключом является строка, то можно представить ее в виде целого числа, написанного в определенной системе счисления.

Например, строка `Hy.` В таблице ASCII кодов это слово представляется как $(72, 121)$. Основание системы счисления— 128. Следовательно, строку можно представить как число $X = 72 \times 128 + 121 = 9337$.

Рассмотрим два метода разрешения коллизий: открытое хэширование (метод цепочек) и закрытое хэширование (метод открытой адресации).

1.1 Открытое хэширование

Представление данных в таком случае напоминает поразрядную сортировку (можно считать хэш-таблицей, где хэш-функция — цифра в определенном разряде).

Пусть есть N данных. Размер таблицы — M, следовательно, хэш-функция принимает значения в диапазоне $[0, \ldots, M-1]$.

Хэш-таблица представляет собой массив списков. Список выбран как структура, позволяющая удалять данные за константное время.

При хорошей хэш-функции в каждой ячейке таблицы будет находится в среднем $\alpha = \dfrac{N}{M}$. Назовем α *коэффициентом заполнения* хэш-таблицы.

Алгоритм 1: Создание хэш-таблицы

Вход: A — массив размерности N, M — размерность хэш-таблицы
Выход: Хэш-таблица
начало алгоритма
  цикл пока *не дошли до конца массива* **выполнять**
    · Определяем значение хэш-функции $k = h(A[i])$;
    · Добавляем элемент массива в k-ый список хэш-таблицы;
конец алгоритма

Например, для $N = 200$, $M = 50$ коэффициент заполнения $\alpha = 4$. Следовательно, поиск и удаление элемента занимает время $O(1 + \alpha)$.

Алгоритм 2: Поиск или удаление элемента хэш-таблицы

Вход: A — хэш-функция размерности M, X — элемент для поиска или удаления
Выход: Измененная хэш-таблица (при удалении) или указатель на найденный элемент (при поиске)
начало алгоритма
  · Определяем значение хэш-функции для элемента X;
  цикл пока *не дошли до конца списка соответствующей ячейки хэш-таблицы* **выполнять**
    · Ищем необходимый элемент;
    · Удаляем найденный элемент;
конец алгоритма

Основные достоинства открытого хэширования:

1. Неограниченный размер хэш-таблицы (элементы в списки можно добавлять без ограничений)
2. Поиск и удаление за время $O(1 + \alpha)$, что меньше поиска в сбалансированном дереве бинарного поиска.

Рассмотрим простейшие хэш-функции.

1.1.1 Метод деления

Самая простая хэш-функция — остаток от деления на M :
$h(x) = x \mod M$.

Хэширование достаточно быстрое. Если правильно подобрать размер таблицы, то хэш-функция достаточно эффективна.

Нельзя выбирать в качестве M степень двойки. Неудачным является и выбор $M = 2^P - 1$.

Самым удачным является выбор в качестве M простого числа, достаточно далекого от степени двойки.

Например, пусть $N = 2000$. Предположим, что в данном случае достаточно, чтобы коэффициент заполнения таблицы был равен трем. Следовательно, $M \approx \dfrac{N}{\alpha} \approx 701$. Тогда для данного случая хэш-функция — $h(x) = x \mod 701$.

Пример 1.1. Дан набор чисел: 12, 17, 25, 41, 23, 11, 24, 21, 26, 44, 33, 10, 20, 19, 29. Построить хэш-таблицу.

$N = 15$.

Выбираем в качестве M простое число, например, 7. Выбор не самый удачный ($7 = 2^3 - 1$,) но для $N = 15$ в любом случае хэш-таблица не будет самой наглядной. Просто для примера.

Результат:

0:	21
1:	29
2:	23 44
3:	17 24 10
4:	25 11
5:	12 26 33 19
6:	41 20

□

1.1.2 Метод умножения

1. Сначала x умножается на коэффициент $0 < A < 1$ и получаем дробную часть полученного выражения.

2. Результат умножается на M и берется целая часть.

Таким образом хэш-функция имеет вид $h(x) = \lfloor M(xA \mod 1) \rfloor$, где $xA \mod 1 = (xA - \lfloor xA \rfloor)$ — получение дробной части, $\lfloor z \rfloor$ — целая часть числа z.

В качестве A выбирается золотое сечение: $A = \dfrac{\sqrt{5}-1}{2} \approx 0.61803\ldots$

В данном случае в качестве M как раз лучше всего выбрать степень двойки для удобства умножения.

Пример 1.2. Дан набор чисел: 12, 17, 25, 41, 23, 11, 24, 21, 26, 44, 33, 10, 20, 19, 29. Построить хэш-таблицу.

$N = 15$.

Выбираем в качестве M степень двойки, например, 8.

Рассмотрим на примере $x = 12$.

$A \times x = 12 * 0.618034 = 7.416408$. Дробная часть — 0.416408. Умножаем на $M = 8$ и берем целую часть. Следовательно, $h(12) = \lfloor 0.416408 \times 8 \rfloor = 3$.

Результат:

0:	26
1:	23 44 10
2:	41 20
3:	12 25 33
4:	17
5:	19
6:	11 24
7:	21 29

□

1.2 Закрытое хэширование

В случае закрытого хэширования все элементы располагаются непосредственно в таблице. Это позволяет избавится от указателей, но накладывает существенные ограничения на размер таблицы.

Каждая ячейка таблицы содержит либо ключ, либо значение NULL. В случае закрытого хэширования удаление элементов вызывает сложности, поэтому не стоит пользоваться этим способом. Также недостатком закрытого хэширования является возможность неудачного подбора хэш-функции, так что для вставки очередного элемента не найдется свободной ячейки.

Для вставки или поиска последовательно исследуются ячейки до тех пор, пока не встретится пустая ячейка.

Хэш-функция имеет зависит от двух параметров: $h'(x, i)$.

Алгоритм 3: Создание хэш-таблицы

Вход: A — массив размерности N, M — размерность хэш-таблицы
Выход: Хэш-таблица `hash`
начало алгоритма
    · Создаем хэш-таблицу и заполняем ее значением `INF`;
    цикл пока *не дошли до конца массива* **выполнять**
        · Определяем значение вспомогательной хэш-функции $k = h(A[i])$;
        · `j = 0`;
        цикл пока *пока не дошли до хэш-таблицы* **выполнять**
            · Определяем значение хэш-функции $p = f(k, j)$;
            если *p ячейка хэш-таблицы не занята* **то**
                · Вставляем $A[i]$ в p-ую ячейку хэш-таблицы;
                · Прекращаем цикл;
            иначе
                · Увеличиваем j;
конец алгоритма

Для поиска элемента определяем значение вспомогательной хэш-функции для этого элемента. И идем по соответствующим ячейкам таблицы до тех пор, пока не встретим искомый элемент или `NULL`.

1.2.1 Линейное хэширование

Пусть есть любая вспомогательная хэш-функция $h'(x)$, рассмотренная в предыдущей главе. Тогда будем рассматривать хэш-функцию вида: $h(x, i) = (h'(x) + i) \mod M$, где i принимает значения в диапазоне $[0, \ldots, M-1]$, M — размер хэш-таблицы ($M \geqslant N$).

Первой возможной ячейкой является та, которую дает вспомогательная хэш-функция, далее последовательно исследуются все ячейки, пока не встретится пустая. Возможно создание длинной последовательности занятых ячеек, ячеек, что удлиняет время поиска.

Например, пусть вспомогательной функцией является $h'(x) = x \mod M$. и $M = 20$. Рассмотрим пример из предыдущей главы:

Пример 1.3. Дан набор чисел: 12, 17, 25, 41, 23, 11, 24, 21, 26, 44, 33, 10, 20, 19, 29. Построить хэш-таблицу.

$N = 15$.

12: $h'(12) = 12$. Ячейка с индексом 12 пустая, можно заполнять.

17: $h'(17) = 17$. Ячейка с индексом 17 пустая, можно заполнять.

25: $h'(25) = 5$. Ячейка с индексом 5 пустая, можно заполнять.

41: $h'(41) = 1$. Ячейка с индексом 1 пустая, можно заполнять.

23: $h'(23) = 3$. Ячейка с индексом 3 пустая, можно заполнять.

11: $h'(11) = 11$. Ячейка с индексом 11 пустая, можно заполнять.

24: $h'(24) = 4$. Ячейка с индексом 4 пустая, можно заполнять.

21: $h'(21) = 1$. Ячейка с индексом 1 занята, увеличиваем индекс. Ячейка с индексом 2 пустая, можно заполнять.

26: $h'(26) = 6$. Ячейка с индексом 6 пустая, можно заполнять.

44: $h'(44) = 4$. Ячейка с индексом 4 занята, увеличиваем индекс. Ячейка с индексом 5 занята, увеличиваем индекс. Ячейка с индексом 6 занята, увеличиваем индекс. Ячейка с индексом 7 пустая, можно заполнять.

33: $h'(33) = 13$. Ячейка с индексом 13 пустая, можно заполнять.

10: $h'(10) = 10$. Ячейка с индексом 10 пустая, можно заполнять.

20: $h'(20) = 0$. Ячейка с индексом 0 пустая, можно заполнять.

19: $h'(19) = 19$. Ячейка с индексом 19 пустая, можно заполнять.

29: $h'(29) = 9$. Ячейка с индексом 9 пустая, можно заполнять.

Итого хэш-таблица имеет следующий вид:

0	20
1	41
2	21
3	23
4	24
5	25
6	26
7	44
8	NULL
9	29
10	10
11	11
12	12
13	33
14	NULL
15	NULL
16	NULL
17	17
18	NULL
19	19

□

1.2.2 Квадратичное хэширование

Выбираем хэш-функцию вида: $h(x,i) = (h'(x) + c_1 i + c_2 i^2) \mod M$.

Каждая следующая ячейка смещена относительно нулевой ячейки (значение $h'(x)$) на величину, характеризующуюся квадратичной зависимостью, что лучше линейной. На при неудачном выборе параметров c_1, c_2, m, может возникнуть ситуация, когда для элемента не окажется свободной ячейки, удовлетворяющей заданной хэш-функции.

Например, пусть вспомогательной функцией является $h'(x) = x \mod M$. и $M = 20$. Рассмотрим пример из предыдущей главы:

Пример 1.4. Дан набор чисел: 12, 17, 25, 41, 23, 11, 24, 21, 26, 44, 33, 10, 20, 19, 29. Построить хэш-таблицу.

$N = 15$. Пусть $M = 20$, $c_1 = 1$, $c_2 = 3$.

12: $h'(12) = 12$. Ячейка с индексом 12 пустая, можно заполнять.

17: $h'(17) = 17$. Ячейка с индексом 17 пустая, можно заполнять.

25: $h'(25) = 5$. Ячейка с индексом 5 пустая, можно заполнять.

41: $h'(41) = 1$. Ячейка с индексом 1 пустая, можно заполнять.

23: $h'(23) = 3$. Ячейка с индексом 3 пустая, можно заполнять.

11: $h'(11) = 11$. Ячейка с индексом 11 пустая, можно заполнять.

24: $h'(24) = 4$. Ячейка с индексом 4 пустая, можно заполнять.

21: $h'(21) = 1$. Ячейка с индексом 1 занята, увеличиваем индекс: $1 + 1 * 1 + 3 * 1 = 5$. Ячейка с индексом 1 занята, увеличиваем индекс: $1 + 1 * 2 + 3 * 4 = 15$. Ячейка с индексом 15 пустая, можно заполнять.

26: $h'(26) = 6$. Ячейка с индексом 6 пустая, можно заполнять.

44: $h'(44) = 4$. Ячейка с индексом 4 занята, увеличиваем индекс:$4 + 1 * 1 + 3 * 1 = 8$. Ячейка с индексом 8 пустая, можно заполнять.

33: $h'(33) = 13$. Ячейка с индексом 13 пустая, можно заполнять.

10: $h'(10) = 10$. Ячейка с индексом 10 пустая, можно заполнять.

20: $h'(20) = 0$. Ячейка с индексом 0 пустая, можно заполнять.

19: $h'(19) = 19$. Ячейка с индексом 19 пустая, можно заполнять.

29: $h'(29) = 9$. Ячейка с индексом 9 пустая, можно заполнять.

Итого хэш-таблица имеет следующий вид:

0	20
1	41
2	NULL
3	23
4	24
5	25
6	26
7	NULL
8	44
9	29
10	10
11	11
12	12
13	33
14	NULL
15	21
16	NULL
17	17
18	NULL
19	19

□

1.2.3 Двойное хэширование

В качестве хэш-функции выбираем функцию вида: $h(x, i) = (h_1(x) + ih_2(x)) \mod M$, где h_1 и h_2 — вспомогательные хэш-функции.

Начальная ячейка — это значение $h_1(x)$, а смещение — значение $h_2(x)$.

Для того, чтобы хэш-функция могла охватить всю таблицу, значение h_2 должно быть взаимно простым с размером хэш-таблицы. Вариантов выбора несколько: либо выбрать M степенью двойки, а h_2 сконструировать таким образом, чтобы она возвращала только нечетные значения; либо выбрать $h_1(x) = x \mod M$, а $h_2(x) = 1 + (x \mod M')$, где M — простое число, а $M' = M - 1$.

Данная функция реально зависит от двух параметров, поэтому является достаточно хорошей и содержит малое число цепочек занятых ячеек.

Рассмотрим пример из предыдущей главы, в качестве M выбираем простое число, например, 19:

Пример 1.5. Дан набор чисел: 12, 17, 25, 41, 23, 11, 24, 21, 26, 44, 33, 10, 20, 19, 29. Построить хэш-таблицу.

$N = 15$. Пусть $M = 19$, $M' = 18$.

12: $h_1(12) = 12$. Ячейка с индексом 12 пустая, можно заполнять.

17: $h_1(17) = 17$. Ячейка с индексом 17 пустая, можно заполнять.

25: $h_1(25) = 6$. Ячейка с индексом 6 пустая, можно заполнять.

41: $h_1(41) = 3$. Ячейка с индексом 3 пустая, можно заполнять.

23: $h'(23) = 4$. Ячейка с индексом 4 пустая, можно заполнять.

11: $h'(11) = 11$. Ячейка с индексом 11 пустая, можно заполнять.

24: $h'(24) = 5$. Ячейка с индексом 5 пустая, можно заполнять.

21: $h'(21) = 2$. Ячейка с индексом 2 пустая, можно заполнять.

26: $h'(26) = 7$. Ячейка с индексом 7 пустая, можно заполнять.

44: $h'(44) = 6$. Ячейка с индексом 6 занята, увеличиваем индекс:$h_2(44) = 9 \to h(44) = 15$. Ячейка с индексом 15 пустая, можно заполнять.

33: $h'(33) = 14$. Ячейка с индексом 14 пустая, можно заполнять.

10: $h'(10) = 10$. Ячейка с индексом 10 пустая, можно заполнять.

20: $h'(20) = 1$. Ячейка с индексом 1 пустая, можно заполнять.

19: $h'(19) = 0$. Ячейка с индексом 0 пустая, можно заполнять.

29: $h'(29) = 10$. Ячейка с индексом 10 занята, увеличиваем индекс:$h_2(29) = 12 \to h(29) = (10 + 12) \mod 19 = 3$. Ячейка с индексом 3 занята, увеличиваем индекс:$h(29) = (10 + 12 * 2) \mod 19 = 15$. Ячейка с индексом 15 занята, увеличиваем индекс:$h(29) = (10 + 12 * 3) \mod 19 = 8$. Ячейка с индексом 8 пустая, можно заполнять.

Итого хэш-таблица имеет следующий вид:

0	19
1	20
2	21
3	41
4	23
5	24
6	25
7	26
8	29
9	NULL
10	10
11	11
12	12
13	NULL
14	33
15	44
16	NULL
17	17
18	NULL

□