Практическая работа №6
«АЛГОРИТМЫ ПОИСКА»
Часть 6.1. Быстрый доступ к данным с помощью хеш-таблиц

Цель работы: освоить приёмы хеширования и эффективного поиска элементов множества.

1. Хеширование для достижения константного времени доступа к записи в таблице

Хеширование как преобразование исходных данных в выходную битовую строку находит применение в таких сферах, как контроль целостности при передаче данных (контрольные суммы), информационная безопасность (защита паролей, ЭЦП) и некоторые другие.

В том числе хеширование может быть использовано и для организации эффективного (с константным временем O(1)) поиска (также вставки и удаления) элементов данных в ***динамическом множестве***.

Хеш-функция при этом создаёт отображение множества ключевых значений во множество индексов соответствующих записей данных в массиве в виде вспомогательной ***хеш-таблицы*** (рис. 1).

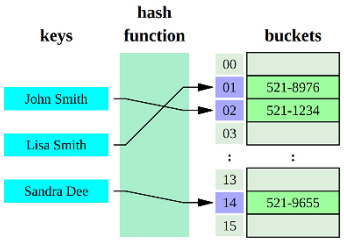

Рис. 1. Индексы элементов динамического множества данных как результат хеширования значения ключевых полей элементов полезных данных

В этом случае при вводе ключа поиска программа вычислит хеш и затем по хеш-таблице определит индекс искомой записи в массиве полезных данных, что открывает к ней прямой доступ.

Алгоритм хеш-функции может быть основан на делении (модальная арифметика, полиномиальный хеш), умножении (хеширование Фибоначчи), на подходе под названием «универсальное хеширование», а также некоторых других.

Например, для алгоритма, основанного на делении, хеш-функция может быть реализована на основе модальной арифметики:

$$h = K \bmod Q, \qquad (1)$$

где K – ключевое значение, Q – наибольшее необходимое количество различных значений хеш-функции (и, как следствие, допустимое количество записей в динамическом множестве).

Если K – составное значение (например, строка символов), то его можно представить в виде полинома.

Примечание: в рамках данной практической работы целесообразно использовать алгоритмы, основанные на делении.

Одним из свойств хеш-функции является необязательность уникальности значений хеша для различных входных наборов данных. Это объясняет ненулевую вероятность возникновения ***коллизии*** – ситуации, когда по разным ключевым значениям может быть вычислено одинаковое хеш-значение. Таким образом, двум или более наборам данных может быть сопоставлен одинаковый индекс в массиве — а это недопустимо.

Для устранения (разрешения, преодоления) коллизии можно использовать методы ***цепного хеширования*** и ***хеш с открытой адресацией***.

Цепным хешированием называется способ разрешения коллизий, когда динамическое множество полезных данных организуется в виде массива ***линейных списков***, состоящих из элементов с одинаковыми хеш-значениями, т.е. индексами в массиве (рис. 2).

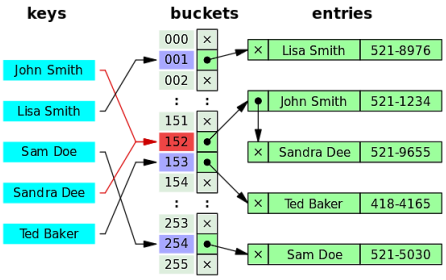

Рис. 2. Схема организации цепного хеширования

При этом в хеш-таблице ключам сопоставляются индексы головных элементов этих списков в массиве.

Массив списков может стать на некотором этапе работы программы неоднородным – несколько длинных списков и множество пустых элементов массива. С одной стороны, массив, даже пустой, занимает память. С другой стороны, время доступа к данным в списке линейное, а не константное, т.е. налицо снижение эффективности поиска.

На практике создают сначала небольшой массив, а по мере заполнения элементами перестраивают его, т.е. увеличивают размер с ***рехешированием*** (пересчетом хешей с новым значением Q).

Критерием необходимости перестройки массива является соотношение n/m – ***коэффициент нагрузки***, где n – это количество уже имеющихся записей, m – длина массива. При достижении значения этого коэффициента 0,75+, следует увеличить длину массива вдвое. Это гарантирует, что длины списков будут относительно небольшими.

Другой способ преодоления коллизий – хеширование с открытой адресацией (рис. 3). Если в массиве в строке с определённым индексом записи нет, то адрес открыт и в соответствующую строку можно поместить новый элемент. Иначе – адрес закрыт (коллизия) и необходимо по некоему алгоритму осуществить ***последовательность проб*** – сместиться относительно закрытого адреса в поисках открытого. Все базовые операции (поиск, вставка, удаление элемента) так или иначе задействуют пробирование, но у каждой свои нюансы.

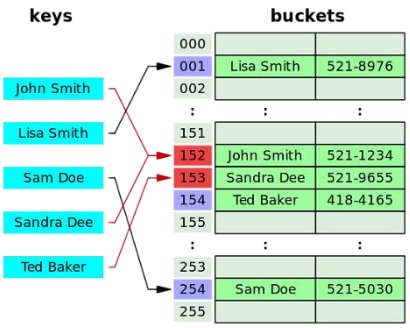

Рис. 3. Пример заполнения массива на основе открытой адресации

Распространённые схемы пробирования: линейное, квадратичное пробирование, двойное хеширование.

В наиболее простой схеме **линейного** пробирования смещение относительно адреса коллизии кратно целочисленной константе (эту константу следует задать так, чтобы они с длиной массива были взаимно просты):

$$адрес=h(x)+ci \quad (2)$$

где i – номер попытки разрешить коллизию; c – константа, определяющая шаг перебора.

В **квадратичной схеме** шаг перебора сегментов нелинейно зависит от номера попытки найти свободный сегмент:

$$адрес=h(x)+ci+di^2 \quad (3)$$

где i – номер попытки разрешить коллизию, c и d – константы.

В схеме **двойного хеширования** смещение относительно закрытого адреса кратно величине второй хеш-функции, схожей, но не эквивалентной основной:

$$адрес=h(x)+ih_2(x) \quad (4)$$

В случае открытой адресации имеет смысл создать массив сразу наибольшей длины. В противном случае при постепенном заполнении массива записями будет всё более длительной процедура поиска открытого адреса. Затраты времени на перестройку этого массива лишь снизят эффективность всей программы.

2. Практическое задание

Разработайте приложение, которое использует ***хеш-таблицу*** (пары «ключ – хеш») для организации прямого доступа к элементам ***динамического множества*** полезных данных. Множество реализуйте на массиве, структура элементов (перечень полей) которого приведена в индивидуальном варианте (п.3).

Приложение должно содержать ***класс*** с ***базовыми операциями***: вставки, удаления, поиска по ключу, вывода. Включите в класс массив полезных данных и хеш-таблицу. Хеш-функцию подберите самостоятельно, используя правила выбора функции.

Реализуйте расширение размера таблицы и ***рехеширование***, когда это требуется, в соответствии с типом разрешения ***коллизий***.

Предусмотрите автоматическое заполнение таблицы 5-7 записями.

Реализуйте текстовый ***командный интерфейс*** пользователя для возможности вызова методов в любой произвольной последовательности, сопроводите вывод достаточными для понимания происходящего сторонним пользователем подсказками.

Проведите полное тестирование программы (все базовые операции, изменение размера и рехеширование), тест-примеры определите самостоятельно. Результаты тестирования включите в отчет по выполненной работе.

Примечание: тесты должны включать в себя случаи коллизий, проверке подлежит правильность вставки, поиска и удаления записей, вызвавших коллизию.

Оформите отчёт с подробным описанием созданного массива и хеш-таблицы, подходов к программной реализации базовых операций, описанием текста исходного кода и проведенного тестирования программы.

В отчёте сделайте вывод о проделанной работе, основанный на полученных результатах.
Содержание отчёта:

1. Титульный лист.
2. Цель работы.
3. Ход работы (по каждому заданию):
    a. Формулировка задачи.

b. Математическая модель решения (описание алгоритма).
c. Код программы с комментариями.
d. Результаты тестирования.

4. Вывод (решены ли задачи, достигнута ли цель).

Для сдачи практической работы потребуется:
- отчёт – оформляется в виде электронного документа в форматах Word или PDF, прикрепляется к соответствующему заданию в СДО;
- программные проекты, реализованные по заданиям;
- доклад по результатам выполнения практической работы (по отчёту).

3. Варианты индивидуальных заданий

Вариант	Метод хеширования (тип последовательностей проб)	Структура элемента множества. Ключи записей подчеркнуты
1	Цепное хеширование	Читательский абонемент: номер читательского - целое пятизначное число, ФИО, адрес
2	Цепное хеширование	Счет в банке: номер счета целое 7-зн.число, ФИО,адрес
3	Открытая адресация (двойное хеширование)	Студент: номер зачетной книжки, номер группы, ФИО
4	Открытая адресация (квадратичное пробирование)	Регистрация малого предприятия: номер лицензии, название, учредитель
5	Открытая адресация (двойное хеширование)	Товар: код – шестиразрядное число, название, цена
6	Открытая адресация (линейное пробирование)	Специализация вуза: код специальности – (прим.: "09.03.01"), название вуза
7	Открытая адресация (линейное пробирование)	Студент: номер зачетной книжки, номер группы, ФИО
8	Открытая адресация (квадратичное пробирование)	Читательский абонемент: номер читательского - целое пятизначное число, ФИО, адрес
9	Открытая адресация (двойное хеширование)	Читательский абонемент: номер читательского - целое пятизначное число, ФИО, адрес
10	Цепное хеширование	Регистрация малого предприятия: номер лицензии, название, учредитель
11	Открытая адресация (двойное хеширование)	Владелец телефона: номер телефона – последовательность 10 **символов**, адрес
12	Открытая адресация (линейное пробирование)	Страховой полис: номер, компания, фамилия владельца
13	Открытая адресация (двойное хеширование)	Счет в банке: номер счета целое семизначное число, ФИО, адрес
14	Открытая адресация (двойное хеширование)	Регистрация малого предприятия: номер лицензии, название, учредитель
15	Открытая адресация (линейное пробирование)	Счет в банке: номер счета целое семизначное число, ФИО, адрес

16	Открытая адресация (двойное хеширование)	Страховой полис: номер, компания, фамилия владельца
17	Цепное хеширование	Специализация вуза: код специальности – (прим.: "09.03.01"), название вуза
18	Открытая адресация (линейное пробирование)	Товар: код – шестиразрядное число, название, цена
19	Цепное хеширование	Книга: ISBN – 12-значное число, автор, название
20	Открытая адресация (линейное пробирование)	Книга: ISBN – двенадцатизначное число, автор, название
21	Открытая адресация (квадратичное пробирование)	Книга: ISBN – двенадцатизначное число, автор, название
22	Открытая адресация (квадратичное пробирование)	Счет в банке: номер счета целое семизначное число, ФИО, адрес
23	Открытая адресация (квадратичное пробирование)	Страховой полис: номер, компания, фамилия владельца
24	Цепное хеширование	Товар: код – шестиразрядное число, название, цена
25	Открытая адресация (линейное пробирование)	Владелец телефона: номер телефона – последовательность 10 **символов**, адрес
26	Открытая адресация (двойное хеширование)	Специализация вуза: код специальности – (прим.: "09.03.01"), название вуза
27	Цепное хеширование	Студент: номер зачетной книжки, номер группы, ФИО
28	Открытая адресация (квадратичное пробирование)	Владелец телефона: номер телефона – последовательность 10 **символов**, адрес
29	Открытая адресация (линейное пробирование)	Читательский абонемент: номер читательского - целое пятизначное число, ФИО, адрес
30	Открытая адресация (квадратичное пробирование)	Товар: код – шестиразрядное число, название, цена

Вопросы для самоподготовки:

1. Что такое хеширование? В каких областях оно применяется?
2. Перечислите основные свойства хеш-функции.
3. На чём обычно основаны алгоритмы хеширования?
4. Что такое константная вычислительная сложность? Как хеширование помогает реализовать поиск с константным временем доступа к данным?
5. Что такое коллизия? Назовите приёмы устранения (разрешения) коллизий.
6. Что такое цепное хеширование? С чем связана основная проблема этого метода?
7. Что такое рехеширование? Назовите критерий необходимости рехеширования.
8. В чём заключается идея хеширования с открытой адресацией?
9. Назовите наиболее распространённые схемы последовательности проб в открытой адресации.

Литература:

1. Бхаргава А. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих, 2017. – С. 100-126.
2. Кормен Т.Х. и др. Алгоритмы. Построение и анализ, 2013. – С. 285-318.
3. Страуструп Б. Программирование. Принципы и практика с использованием C++. 2-е изд., 2016.
4. Документация по языку C++ [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/cpp/cpp/ (дата обращения 01.09.2021).
5. Курс: Структуры и алгоритмы обработки данных. Часть 2 [Электронный ресурс]. URL: https://online-edu.mirea.ru/course/view.php?id=4020 (дата обращения 01.09.2021).