**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

Курсовая РАБОТА

**по дисциплине «Алгоритмы и структуры данных»**

Тема: Хеш-таблица (двойное хеширование) vs Хеш-таблица (открытая адресация). Исследование

**Вариант 6**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 1381 |  | Рымарь М.И. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2022

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| Студентка Рымарь Мария Игоревна | | |
| Группа 1381  Вариант 6 | | |
| Тема работы: Хеш-таблица (двойное хеширование) vs Хеш-таблица (открытая адресация). Исследование | | |
| Задание:  "Исследование" - реализация требуемых структур данных/алгоритмов; генерация входных данных (вид входных данных определяется студентом); использование входных данных для измерения количественных характеристик структур данных, алгоритмов, действий; сравнение экспериментальных результатов с теоретическими. Вывод промежуточных данных не является строго обязательным, но должна быть возможность убедиться в корректности алгоритмов. | | |
|  | | |
| Предполагаемый объем пояснительной записки:  Не менее 15 страниц. | | |
| Дата выдачи задания: 25.10.2022 | | |
| Дата сдачи реферата: 24.12.2022 | | |
| Дата защиты реферата: 24.12.2022 | | |
| Студентка |  | Рымарь М.И. |
| Преподаватель |  | Иванов Д.В. |

**Аннотация**

В ходе выполнения курсовой работы была создана программа на языке программирования Python, в которой реализованы таких структуры данных, как Хеш-таблица с открытой адресацией и Хеш-таблица с двойным хешированием. Структуры данных сравниваются по теоретической сложности базовых операций, также сравниваются полученные результаты с экспериментальными значениями.

**Summary**

During the course work, a program was created in the Python programming language, which implements such data structures as a hash table with open addressing and a hash table with double hashing. The data structures are compared according to the theoretical complexity of the basic operations, and the results obtained are also compared with experimental values.**содержание**

|  |  |  |
| --- | --- | --- |
| 1. | Введение | 5 |
| 2. | Теоретические сведения | 6 |
| 3. | Ход выполнения работы | 10 |
| 3.1. | Реализация структур данных | 8 |
| 3.1.1.  3.1.2.  3.2.  3.3. | Хеш-таблица с открытой адресацией  Хеш-таблица с двойным хешированием  Теоретическая оценка сложности базовых операций  Сравнение с экспериментальными значениями | 8  8  9  10 |
| 4. | Тестирование | 12 |
| 5. | Заключение | 16 |
| 6. | Список использованных источников | 17 |
| 7. | Приложение А. Исходный код | 18 |
|  |  |  |
|  |  |  |
|  |  |  |

**введение**

Кратко описать цель работы, основные задачи и методы их решения. Требуется написать программу, которая производит сравнение двух структур данных.

**Цель и задачи работы**

Цель: исследовать такие структуры данных, как Хеш-таблица с открытой адресацией и Хеш-таблица с двойной адресацией.

Задачи:

* Реализовать структуры данных;
* Осуществить генерацию входных данных, вид входных данных определить самостоятельно;
* Произвести теоретическую оценку сложности базовых операций;
* Сравнить значения, полученные в ходе работы программ, с теоретическими.

**2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

**2.1. Хеш-таблица**

Хеш-таблицей называется структура данных, обеспечивающая очень быструю вставку и поиск. На первый взгляд звучит слишком хорошо, чтобы быть правдой: независимо от количества элементов данных вставка и поиск (а иногда и удаление) выполняются за время, близкое к постоянному: O(1) в O-синтаксисе. На практике это лишь несколько машинных команд.

Для пользователя хеш-таблицы обращение к данным происходит практически мгновенно. Все делается настолько быстро, что компьютерные программы часто используют хеш-таблицы при необходимости сделать выборку из десятков тысяч элементов менее чем за секунду (как, например, в системах проверки орфографии).

Хеш-таблицы по скорости значительно превосходят деревья, которые выполняют операции за с относительно малое время O(logN). Операции с хеш-таблицами не только быстро выполняются, но и относительно просто программируются.

У хеш-таблиц также имеются свои недостатки. Они реализуются на базе массивов, а массивы трудно расширить после создания. У некоторых разновидностей хеш-таблиц быстродействие катастрофически падает при заполнении таблицы, поэтому программист должен довольно точно представлять, сколько элементов данных будет храниться в таблице (или приготовиться к периодическому перемещению данных в другую хеш-таблицу большего размера — процесс занимает довольно много времени).

Кроме того, при работе с хеш-таблицами не существует удобного способа перебора элементов в определенном порядке (скажем, от меньших к большим). Если необходима такая возможность, стоит выбрать другую структуру данных.

Но если нет необходимости перебирать элементы в определенном порядке, а размер базы данных можно спрогнозировать заранее, хеш-таблицы не имеют себе равных по скорости и удобству.

**2.2. Хеш-таблица с открытой адресацией**

Если элемент данных не удается разместить в ячейке с индексом, вычисленным посредством хеш-функции, метод открытой адресации ищет в массиве другую ячейку. Существует три основных разновидности открытой адресации, различающихся способом поиска следующей свободной ячейки: линейное пробирование, квадратичное пробирование и двойное хеширование.

Линейное пробирование и квадратичное пробирование будет рассмотрено в этом разделе, двойное хеширование в следующем, так как в ходе реализации курсовой работы требуется реализовать и исследовать хеш-таблицу с открытой адресацией и хеш-таблицу с двойным хешированием двумя отдельными структурами данных.

Алгоритм линейного пробирования последовательно ищет пустые ячейки. Если при попытке вставки элемента выясняется, что ячейка 5421 занята, мы переходим к ячейке 5422, затем к ячейке 5423 и т. д. Индекс последовательно увеличивается до тех пор, пока не будет найдена пустая ячейка. Процедура поиска называется «линейным пробированием», потому что она основана на линейной проверке последовательности ячеек.

При открытой адресации с линейным пробированием возникает проблема группировки. Образовавшиеся группы начинают расширяться. Элементы, хешируемые в пределах группы, добавляются в конец группы, в результате чего группа становится еще больше. Чем больше размер группы, тем быстрее она растет.

Отношение количества элементов в таблице к размеру таблицы называется коэффициентом заполнения. Таблица с 10 000 ячеек, содержащая 6667 элементов, имеет коэффициент заполнения 2/3.

коэффициент\_заполнения = количество\_элементов / размер\_массива;

Группы могут образовываться даже при относительно небольшом коэффициенте заполнения. Одни части хеш-таблицы могут быть заполнены большими группами, другие почти не содержать элементов. Группировка снижает быстродействие таблицы. Квадратичное пробирование пытается избежать образования групп. Его идея заключается в том, чтобы проверять ячейки, находящиеся на больших расстояниях (вместо ячеек, находящихся вблизи от исходной позиции хеширования).

**2.3. Хеш-таблица с двойным хешированием**

Для устранения как первичной, так и вторичной группировки применяется алгоритм двойного хеширования. Вторичная группировка возникает из-за того, что алгоритм, генерирующий последовательность смещений для квадратичного пробирования, всегда генерирует одни и те же смещения: 1, 4, 9, 16 и т. д.

В идеале последовательность проб должна генерироваться в зависимости от ключа (вместо использования набора одинаковых смещений для всех ключей). В этом случае числа с разными ключами, хешируемые в один индекс, будут использовать разные последовательности смещений.

Задача решается повторным хешированием ключа с другой хеш-функцией и использованием результата в качестве смещения. Для заданного ключа размер смещения остается постоянным при пробировании, но для разных ключей используются разные размеры. Практический опыт показал, что вторичная хеш-функция должна обладать некоторыми характеристиками:

- Она не должна совпадать с первичной хеш-функцией.

- Ее результат никогда не должен быть равен 0 (в противном случае смещения не будет, все пробы будут приходиться на одну ячейку, а алгоритм войдет в бесконечный цикл).

Эксперты обнаружили, что для решения этой задачи хорошо подходят функции вида смещение = константа - (ключ % константа); где константа — простое число, меньшее размера массива. Пример функции: stepSize = 5 - (key % 5);

Разные ключи могут хешироваться в один индекс, но для них (вероятно) будут сгенерированы разные смещения. При такой хеш-функции размеры смещений лежат в диапазоне от 1 до 5. На рисунке 1 под буквой а представлен успешный поиск элемента, под буквой б представлен безуспешный поиск элемента.

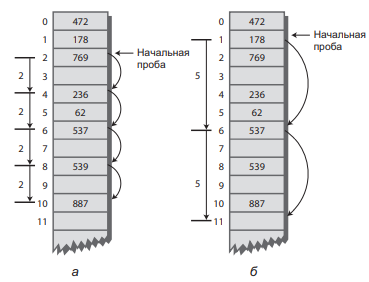
****

Рисунок 1 – Поиск элемента с помощью двойного хеширования

**3. ХОД ВЫПОЛНЕНИЯ РАБОТЫ**

**3.1 Реализация структур данных**

**3.1.1. Хеш-таблица с открытой адресацией**

**3.1.2. Хеш-таблица с двойным хешированием**

**3.2. Теоретическая оценка сложности базовых операций**

**3.3. Сравнение с экспериментальными значениями**

**4. ТЕСТИРОВАНИЕ**

**заключение**

**список использованных источников**

1.

**приложение А**

**Исходный код**