Київський національний університет імені Тараса Шевченка

Факультет комп’ютерних наук та кібернетики

Лабораторна робота №1

Виконав: Олексишин Олексій

Група К-28

Кафедра інтелектуальних програмних систем

**2020**

**Предметна область**

Вариант 3

Предметная область Отдел кадров Объекты Подразделения, Сотрудники Примечание Имеется множество подразделений предприятия. В каждом подразделении работает множество сотрудников.

Для того щоб дану структуру даних можна було помістити в дерево я перевантажив operator< (та інші оператори порівняння), для виведення в консоль operator<<.

**Завдання**

Реалізуйте ідеальне хешування для статичного набору даних.

**Алгоритм і теорія**

Ідеальне хешування – хешування, яке не створює коллізій, тобто кожному значенню хеш-функції відповідає єдине значення ключа.

Алгоритм

Нехай задана деяка статична множина з n елементів. Спочатку застосуємо до кожного елемента звичайну хеш функцію, яка відобразить цю множину в множину з m елементів. Для цього я вибрав хеш-фунцкію MurMur2 (опис можна знайти за посиланням в джерелах). Далі розглянемо кожне значення хеш-функції та ключі, які ми в нього розподілили. Нехай ключу j відповідає Sj елементів. Тоді ми побудуємо хеш-таблицю розміром Sj \* Sj, вибравши для цього свою одну хеш-функцію. При правильному виборі хеш функції можна досягти відсутності колізій.

Для такого подходу можлива ситуація, коли ключу j відповідає надто багато елементів і тоді таблиця Sj \* Sj використає надто багато додаткової пам’яті. Проте, завдяки хорошій вибраній хеш-функції MurMur2 використаної додаткової пам’яті буде в середньому O(n).

**Складність алгоритму**

Операції вставки та видалення наш алгоритм не передбачує, оскільки за умовою множина є статичною. Операція пошуку елемента та отримання елемента за ключем працює за O(1), оскільки для цього треба просто двічі послідовно застосувати відповідні хеш-функції, що потребує O(1) часу. Для реалізації даного хешування нам треба O(n) додаткової пам’яті.

**Реалізація (мова С++)**

Програма реалізує клас CperfectHashTable (який приймає параметром масив з елементів прикладної області), функцію MurMurHash2 та додаткові лінійні хеш-функції виду (a\*k + b) mod m, для хешування на другому рівні.

**Основні модулі програми**

У файлі hash.hpp показаний інтерфейс вказаних класів, та реалізовані вказані функції хешування, а також допоміжні функції для побудови ідеальної хеш-таблиці. Файл struct.hpp містить реалізацію класів предметної області. В файлі main.cpp міститься інтерфейс користувача та приклад роботи з реалізованою структурою даних.

Код можна знайти в моєму репозиторії на github за посиланням:

<https://github.com/ooleksyshyn/semester4/tree/master/algorithms/lab1>

**Інтерфейс користувача та приклади**

Програма дозволяє взаємодіяти з класом через простий інтерфейс – пошук елемента за ключем та перевірка чи є такий елемент в хеш-таблиці. В коді є наведений приклад використання програми.

**Літературні джерела**

* <https://ru.wikipedia.org/wiki/MurmurHash2>
* <https://github.com/ooleksyshyn/semester4/tree/master/algorithms/lab1>