УДК 004.41

Семенюк А. М., студент 2 курсу спеціальності 122 «Комп'ютерні науки»
Науковий керівник: Вєтров О.С., ст. викладач кафедри прикладної математики та кібербезпеки

РЕАЛІЗАЦІЯ ФУНКЦІЇ ХЕШ-ТАБЛИЦЬ МОВОЮ ПРОГРАМУВАННЯ PYTHON

Донецький національний університет імені Василя Стуса, м. Вінниця

В мові програмування **PYTHON**, як і в багатьох сучасних мовах програмування високого рівня існують засоби для доступу до елементів пам'яті за ключем, а не за індексом. Для PYTHON таким асоціативним масивом є "**словники**".
Їх використання дає ряд переваг:

- з допомогою словників можна розробляти ефективні структури даних;
- для словників не потрібно писати вручну алгоритми пошуку даних, оскільки ці операції вже є реалізовані;
- словники можуть містити об'єднані дані у вигляді записів;
- словники ефективні при представленні розріджених структур даних.

Але, незважаючи на колосальні властивості, як і кожній вбудованій функції, користувачу завжди не вистачає чогось "дуже потрібного", "родзинки". В таких випадках рішення просте – зроби свій клас.
Що ж представляє собою такий тип даних.
Асоціативний масив ще застосовуються терміни: ґеш, мапа (карта),… – абстрактний тип даних (інтерфейс до сховища даних), що дозволяє зберігати дані у вигляді набору пар **ключ-значення** та доступом до значень за їх ключем.
Для операцій над такими масивами відносяться наступні:

- вставити – параметри (ключ, значення);
- замінити – параметри (ключ, значення);
- шукати – параметри (ключ);
- вилучити – параметри (ключ).

По замовчуванню передбачено, що дві пари з однаковими ключами не можуть знаходитись в межах одної мапи. У парі **[k(ey), v(ar)]** значення **v** називається значенням, що асоціюється з ключем **k**. Це забезпечує дуже швидке вставляння та пошук незалежно від кількості елементів. Ці операції (а іноді й видалення) виконуються за час, близький до константи – O(1).
По суті мапа, це звичайний масив, де місце розташування елемента залежить від значення самого елемента. В ньому поєднуються переваги інших концепцій пам'яті – **лінійної та бінарної**. Лінійна пам'ять дуже повільна в доступі, але може містити різнотипні дані – пошук здійснюється за лінійний час. Бінарна

пам’ять є швидкою – пошук відбувається за логарифмічна час – проте, може виконуватися лише для впорядкованих списків.
З погляду інтерфейсу асоціативний масив зручно розглядати як звичайний масив, де в якості індексів використовуються не лише цілі числа, але і рядки, і значення інших типів. Але так як доступ до комірки пам’яті на первинному рівні відбувається за цифровими адресами (навіть при послідовному доступі), то виникає задача перекодування ключа в деякий адрес. Ця операція – називається **ґешування** – операція перетворення вхідного масиву даних довільної довжини у вихідний бітовий рядок фіксованої довжини ($\boldsymbol{M}$) за допомогою деякого визначеного алгоритму (рис. 1).

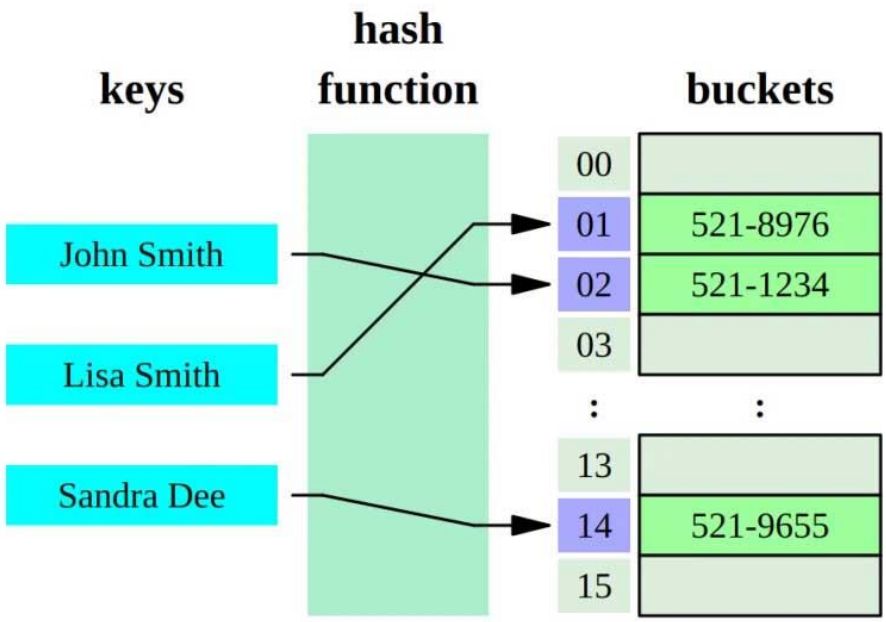

Рис. 1 – Приклад ґешування

Ґеш-функція, яку ще називають **функцією згортки** отримує вхідну частину даних, яку ми називаємо ключем, а на виході вона видає ціле число, відоме як ґеш-значення (або ґеш-код). Потім ґеш-значення прив'язує наш ключ до певного індексу ґеш-таблиці. По закінченню вводу отримаємо впорядкований список. **Ґеш** ($\boldsymbol{H(x)}$) це натуральне число, яке часто записують у шістнадцятьковому вигляді.
Ґеш-функція характеризується рядом умов, серед них:

- На виході ґеш-функції завжди отримуються значення фіксованої довжини;
- Для однакових вхідних даних, ґеш-функція повертає однаковий результат;
- Ґеш-функція має просту алгоритмічну складність, для уникнення зайвого обчислювального навантаження.

В якості операцій перетворення (саме це і є ґеш-функція) використовуються:

- операції на базі ділення – ***H(x) = x %% M***;
- операції на базі множення – ***H(x) = x * M(func)***;
- операції на базі середніх квадратів;
- згортка;
- для рядків.

На базі цих, та інших положень для асоціативних масивів, розробимо власну комірку МАПИ, реалізовану з використанням структури типу "class:

```
# Об'єкт - Таблиця користувача
class Мара:       # Характеристики
    hashUs = -1
    keyUs = ""
    valUs = ""
    nextUs = -1
```

Так як в якості ключа може бути використаний довільний тип, то задаємо для нього зміну **string**. Тому формування ґеш-коду будемо виконувати по асоціативним даним. Якщо це рядок, то

- розбиття його на базові поняття, кожному з яких присвоєний деякий код:
- отримання сукупності довільно розподілених чисел;
- проведення деякої алгебраїчної операції.

Якщо число, то на першому кроці створюємо ґрупи цифр які кодуємо, а далі, по алгоритму – тоді виконуємо алгебраїчні дії.

Оптимально підібраний алгоритм розподіляє дані рівномірно що значно скорочує тривалість виконання базових операцій

Коли виникнуть **колізії** – ситуації, коли для цілком різних даних, результатом функції буде одне значення, чи коли **потужність** ґеш-таблиці досягне критичного значення запускається відповідна процедура вирішення цих проблем, наприклад перерахунок ґеш-таблиці, чи формування "**області переповнення**".

Збільшення елементів, які розташовуються в окремому пакеті, значно понижують значення часу пошуку, тому що, як не дивно, файл переповнення буде мати менший об’єм.

Хороша реалізація ґеш-таблиці вимагає щоб пошук елемента у таблиці та вставка нового елементу до ґеш-таблиці відбувалися за сталий час. Тому, як правило, вставка нового елементу у таблицю здійснюється як додавання його в кінець або у початок ланцюжка (залежно від організації структури самого ланцюжка).

Список літератури:

1. Креневич А.П. Алгоритми і структури даних. Підручник. – К.: ВПЦ "Київський Університет", 2021. – 200 с.

2. Коротєєва Т. О. Алгоритми та структури даних: навч. посібник / Т. О. Коротєєва. – Львів: Львівської політехніки, 2014. – 280 с.

3. The Python Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/tutorial/index.html.