Звіт

Лабораторна робота №1

Шевченко Максим, МІ-3

Варіант 1

Поліноміальне хешування. Пошук рядків, що повторюються

***Умова***

Реалізувати структуру даних типу «множина рядків». Рядки являють собою непусті послідовності довжиною ≤15 символів латинських літер. Структура даних повинна підтримувати операції додавання, видалення та перевірки наявності рядків. Проблему колізій вирішити методом ланцюжків. Загальна кількість операцій ≤106. Необхідно знайти рядки, що повторюються, та поділити їх на групи, щоб у кожній групі були однакові рядки.

***Теоретичні відомості***

**Хешування** — перетворення вхідного масиву даних довільної довжини у вихідний бітовий рядок фіксованої довжини. Такі перетворення також називаються хеш-функціями, або функціями згортання, а їхні результати називають хешем, хеш-кодом, хеш-сумою, або дайджестом повідомлення.

**Хеш-функція** — функція, що перетворює вхідні дані будь-якого (як правило великого) розміру в дані фіксованого розміру. Хеш-функція використовується зокрема у структурах даних — хеш-таблицях, широко вживаних у програмному забезпеченні для швидкого пошуку даних.

**Хеш-таблиця** — структура даних, що реалізує інтерфейс асоціативного масиву, а саме, вона дозволяє зберігати пари (ключ, значення) і здійснювати три операції: операцію додавання нової пари, операцію пошуку і операцію видалення за ключем.

**Колізія** – ситуація, коли для різних ключів отримується одне й те саме хеш-значення.

**Метод ланцюжків** – метод розв’язання колізій, в якому кожна комірка масиву є вказівником на зв'язаний список (ланцюжок) пар ключ-значення, відповідних одному і тому самому хеш-значенню ключа. Колізії просто призводять до того, що з'являються ланцюжки довжиною більше одного елемента.

**Поліноміальні хеш-функції** – хеш-функції, у яких хешем рядка є число , де деяке натуральне число, а код -го символа рядка . Кільцевим поліноміальним хешем називається число .

***Аналіз***

За умовою задачі максимальна кількість операцій ≤106, максимальна довжина рядка ≤15, латинських літер 26 + порожній символ. Звідси робимо висновок, що можлива кількість комбінацій між 2615 та 2715, а отже ймовірність знайти однакові рядки у структурі неймовірно мала, що унеможливить виконання поставленого завдання (усі групи однакових слів складатимуться з одного елемента). Цю проблему я вирішив наступним чином: згенерував 105 випадкових слів і випадково обираючи команди записав у вхідний файл розміром 106. Це гарантує наявність слів, які повторюються. Щодо вибору параметрів p та m, то в даній реалізації я обрав p=31 (31>26, підходить) і m=100000321. Звісно, при такому невеликому m імовірність колізії () близька до 1, але колізій уникнути неможливо з огляду на умову задачі. Крім того, це демонструє стійкість розробленої програми до колізій, а також суттєво економить пам’ять програми й особливо час її виконання.

Час обчислення хешу залежить від довжини слово й складає , але довжина слова обмежена, звідси .

При вставці елементу ми просто додаємо його в кінець ланцюга, що займає час .

При видаленні елемента не гарантовано, що він є в таблиці. Тому час видалення в найгіршому випадку відповідає часу пошуку елемента, який ми проаналізуємо далі.

Час пошуку залежить від коефіцієнту заповнення таблиці, який обчислюється як α=n/m, де n – кількість елементів, а m – загальна кількість комірок таблиці. В найгіршому випадку усі n елементів таблиці мають однаковий хеш і усі містяться в одному зв’язному списку, тоді час пошуку буде Θ(n), а також використовується час, що йде на обчислення хеш-функції.

Не враховуючи час О(1), необхідний для обчислення хеш-функції, та час, необхідний для доступу до комірки хеш таблиці, розглянемо математичне очікування кількості елементів, які мають бути перевірені алгоритмом пошуку. Ми повинні розглянути два випадки: коли пошук невдалий і в таблиці немає елементів з заданим ключем, та коли пошук закінчується успішно і в таблиці знаходиться елемент із заданим ключем.

За припущенням поліноміального хешування, будь-який ключ, який ще не знаходиться в таблиці, з рівною ймовірністю може бути поміщений у будь-яку з комірок. Математичне очікування часу невдалого пошуку ключа дорівнює часу пошуку до кінця списку, очікувану довжину якого позначимо як α. Таким чином, при невдалому пошуку математичне очікування кількості елементів, що перевіряється, дорівнює α, а загальний час, що необхідний для пошуку, включає час, необхідний для обчислення хеш-функції, тобто *Θ(1 + α).*

Припустимо, що шуканий елементом з рівною ймовірністю може бути будь-яким елементом, що зберігається в таблиці. Кількість елементів, що перевіряються в процесі успішного пошуку елементу x, на 1 більше, ніж кількість елементів, що зберігаються в списку перед x. Елементи, що знаходяться в списку до x, були вставлені до того, як елемент x був збережений в таблиці, оскільки нові елементи додаються в кінець списку. Щоб знайти математичне очікування кількості елементів, що перевіряються, ми візьмемо середнє по всім n елементам x в таблиці значення, яке дорівнює 1 плюс математичне очікування кількості елементів, що додані в список до шуканого елемента. Нехай позначає -й елемент, вставлений в таблицю () і хай . Визначимо для ключів та індикаторну випадкову величину За припущення поліноміального хешування , тому . Таким чином математичне очікування кількості елементів, що перевіряються, у випадку успішного пошуку дорівнює

Таким чином, час, необхідний для проведення успішного пошуку, включаючи час обчислення хеш функції, складає

***Результати реалізації***

Результатом є чудова програмна реалізація мовою С# структури даних типу «множина рядків». Правильність виконання можна перевірити, переглянувши файл output.txt. Процес виконання записаний у файлі log.txt. Якість хешування й відсутність колізій, крім однакових слів, дає змогу спостерігати файл hist.txt. Загалом, всі завдання лабораторної було виконано.