Доклад про програму на тему пошуку паліндромів за допомогою поліноміального хешування

1. Вступ

Ця програма реалізує пошук паліндромів у множині рядків з використанням поліноміального хешування. Вона дозволяє:

- Вводити рядки вручну або автоматично генерувати випадкові рядки.
- Проводити операції додавання, видалення та перевірки наявності рядків у множині.
- Здійснювати пошук паліндромів у введених або згенерованих рядках.
- Вимірювати час на виконання операцій та пошук паліндромів.

2. Основні компоненти програми

1. **Константи і типи:**
    - MAX_STRINGS – максимальна кількість рядків, яку можна зберігати у множині (1 000 000).
    - MAX_OPERATIONS – максимальна кількість операцій (1 000 000).
    - ull – тип для збереження великих цілих чисел (unsigned long long), використовується для лічильника паліндромів.
2. **Структура PolynomialHash:** Структура відповідає за обчислення поліноміальних хешів для рядків, що спрощує перевірку на паліндромність:
    - **MOD** – велике просте число для зведення хешу по модулю.
    - **BASE** – основа для обчислення хешу.
    - power_mod_base – вектор для збереження степенів основи по модулю.
    - prefix_mod_hash – вектор хешів префіксів рядка.

Конструктор структури обчислює хеші префіксів рядка, а метод get_substring_hash дозволяє швидко отримати хеш для будь-якого підрядка.

3. **Генерація випадкових рядків:** Функція generate_random_string генерує випадковий рядок, що складається з малих латинських літер, довжиною від 1 до 15 символів. Для цього використовуються випадкові числа, що генеруються за допомогою генератора mt19937.

4. **Пошук паліндромів:** Функція find_palindromes здійснює пошук усіх паліндромів у множині рядків:
    - Для кожного рядка обчислюється поліноміальний хеш для оригінального рядка і для його перевернутої версії.
    - Перевіряються всі можливі підрядки, які можуть бути паліндромами.
    - Якщо хеш оригінального і перевернутого підрядка співпадають, рядок вважається паліндромом.
    - Паліндроми додаються до множини palindrome_set, що дозволяє уникати повторень.
5. **Автоматична генерація рядків і вимірювання часу:** Функція generate_strings_and_measure_time:
    - Генерує вказану кількість випадкових рядків.
    - Вимірює час, необхідний для генерації рядків та пошуку паліндромів.
    - Виводить результати на екран.

3. Ручний режим роботи

Програма дозволяє користувачу вводити рядки вручну за допомогою таких операцій:

- **"+"** – додати рядок до множини.
- **"-"** – видалити рядок з множини.
- **"?"** – перевірити, чи є рядок у множині.

Програма перевіряє введені рядки на:

- Максимальну кількість символів (не більше 15).
- Наявність лише малих латинських літер.

Після введення даних, програма може виконати пошук паліндромів у введених рядках і вивести кількість знайдених паліндромів, а також час, витрачений на операції.

4. Автоматичний режим роботи

У цьому режимі програма дозволяє генерувати випадкові рядки:

- Можна обрати кількість рядків для генерації: 1 000 000, 500 000 або 100 000.

- Після генерації рядків програма проводить пошук паліндромів та виводить час, витрачений на кожну операцію.

5. Вимірювання продуктивності

Програма використовує стандартну бібліотеку chrono для вимірювання часу виконання:

- Вимірюється час на генерацію рядків.
- Вимірюється час на пошук паліндромів.

6. Висновки

Ця програма демонструє ефективний підхід до обробки рядків та пошуку паліндромів за допомогою поліноміального хешування. Використання хешування дозволяє швидко порівнювати підрядки, що значно пришвидшує перевірку на паліндромність навіть при великій кількості рядків і операцій.

Програма підходить для роботи з великими наборами даних, оскільки підтримує до 1 000 000 рядків у множині та до 1 000 000 операцій.

Поліноміальне хешування

Що таке поліноміальне хешування?

Поліноміальне хешування — це ефективний метод обчислення числового представлення рядків для їх порівняння. Воно використовує математичну функцію, яка перетворює рядок на унікальне число (хеш), що дозволяє швидко порівнювати рядки або їх підрядки. Поліноміальне хешування широко використовується для задач, де необхідно швидко порівнювати підрядки рядків або знаходити повторення, як у задачі пошуку паліндромів.

Формула поліноміального хешу

Поліноміальний хеш для рядка обчислюється за формулою:

$$H(s) = (s_0 \cdot B^0 + s_1 \cdot B^1 + s_2 \cdot B^2 + \cdots + s_{n-1} \cdot B^{n-1}) \mod M$$

Де:

- s_i — це числове значення i-го символу рядка.
- B — база хешування (в нашій програмі це 111111).
- M — просте число для модуля (у програмі це $10^9 + 9$).
- n — довжина рядка.

Як працює поліноміальне хешування?

Кожному символу рядка відповідає певне число. Поліноміальний хеш комбінує ці числа, перемножуючи їх із степенями бази B. Це дозволяє отримати унікальний хеш для кожного рядка, причому навіть невелика зміна в рядку суттєво змінює його хеш.

Для того щоб уникнути переповнення під час обчислень, використовується операція взяття залишку по простому числу M.

Оптимізація за допомогою префіксних хешів

Щоб швидко порівнювати підрядки рядка, використовуються **префіксні хеші**. Це попередньо обчислені хеші для кожного префіксу рядка. Наявність префіксних хешів дозволяє обчислювати хеш будь-якого підрядка за константний час.

Як це працює:

1. Для кожного префікса рядка обчислюється його хеш за поліноміальною формулою.

2. Хеш для підрядка можна отримати за допомогою різниці хешів відповідних префіксів:

$$H(s[l:r]) = (H(s[0:r]) - H(s[0:l]) \cdot B^{r-l}) \mod M$$

Де l — індекс початку підрядка, r — індекс кінця підрядка, а B^{r-l} — заздалегідь обчислені степені бази.

Це дозволяє за лічені операції дізнатися хеш будь-якого підрядка, що суттєво прискорює алгоритм порівняння підрядків у задачах типу пошуку паліндромів.

Використання поліноміальних хешів для пошуку паліндромів

Програма використовує поліноміальні хеші для ефективного пошуку паліндромів у рядках. Паліндром — це рядок, який читається однаково з обох боків. Для перевірки, чи є підрядок паліндромом, можна порівняти його з перевернутим підрядком. Однак пряме порівняння рядків займає багато часу. Тому використовується порівняння хешів підрядків:

- Спочатку обчислюються поліноміальні хеші для вихідного рядка і його перевернутої версії.
- Потім за допомогою префіксних хешів перевіряються всі можливі підрядки, і якщо хеші оригінального і перевернутого підрядків співпадають, то це означає, що підрядок є паліндромом.

Переваги поліноміального хешування

1. **Швидкість**: Поліноміальне хешування дозволяє обчислювати хеш будь-якого підрядка за постійну кількість операцій $O(1)$ після попереднього обчислення префіксних хешів за $O(n)$.
2. **Ефективне порівняння**: Замість того, щоб порівнювати рядки символ за символом, можна порівняти лише їхні хеші, що значно швидше.
3. **Застосовність до різних задач**: Хешування може використовуватися для пошуку паліндромів, пошуку підрядків, перевірки на збіг рядків тощо.

Реалізація в програмі

У програмі реалізована структура `PolynomialHash`, яка:

- Зберігає префіксні хеші рядка.
- Надає метод для швидкого отримання хешу будь-якого підрядка.

Опис:

1. **Константи**:
   - `MOD = 1e9 + 9` — велике просте число, яке використовується для взяття залишку при обчисленні хеша. Це допомагає уникнути переповнення та зменшити ймовірність колізій.
   - `BASE = 111111` — основа поліноміального хешування. Вона визначає, яким чином символи впливають на кінцевий хеш. Чим більше значення основи, тим менша ймовірність, що різні рядки матимуть однаковий хеш.
2. **Статичні поля**:
   - `power_mod_base` — вектор, в якому зберігаються степені основи `BASE` по модулю `MOD`. Ці степені використовуються для обчислення хешів префіксів та підрядків.
3. **Конструктор**:
   - При створенні об'єкта `PolynomialHash` для рядка `input_string`, спочатку ініціалізується масив `prefix_mod_hash`, який зберігає хеші префіксів цього рядка. Префікс — це будь-яка підпослідовність від початку рядка до поточної позиції.
   - Для кожної позиції символу `i` рядка обчислюється його префікс-хеш за наступною формулою:

     $$\text{prefix\_mod\_hash}[i+1] = (\text{prefix\_mod\_hash}[i] + \text{input\_string}[i] \cdot \text{power\_mod\_base}[i]) \% MOD$$

     Тут символ на позиції `i` множиться на відповідний степінь основи `BASE`, і результат додається до хеша попереднього префікса.
4. **Метод** `get_substring_hash`:
   - Цей метод використовується для отримання хеша підрядка рядка. Він працює за допомогою наступної формули:
     $\text{hash\_mod} = (\text{prefix\_mod\_hash}[start + length] - \text{prefix\_mod\_hash}[start] + MOD) \% MOD$
     Ця формула дозволяє швидко обчислити хеш будь-якого підрядка, використовуючи попередньо обчислені префіксні хеші.
5. **Обробка степенів основи**:
   - Якщо рядок довший за вже існуючі степені в масиві `power_mod_base`, додаються нові значення степенів основи за модулем `MOD`:

     $$\text{power\_mod\_base}.push\_back(1LL \cdot \text{power\_mod\_base}.back() \cdot BASE \% MOD);$$

     Це гарантує, що для кожної позиції в рядку можна знайти відповідний степінь основи.

Основна ідея:

Поліноміальне хешування дозволяє швидко порівнювати підрядки або перевіряти їх на еквівалентність за допомогою обчислених хешів. У вашій реалізації це використовується для пошуку паліндромів у рядках: підрядки порівнюються за допомогою їхніх поліноміальних хешів для перевірки симетрії відносно центру.