Отчет по заданию “САОД. Задание 6”

Тема

Разработка класса префикс-функции. Переопределение операций, работа с динамической памятью. Оценка сложности алгоритма.

Краткое описание применения префикс-функции для поиска подстроки в тексте

Префикс-функция используется в алгоритме Кнута-Морриса-Пратта (КМП) для эффективного поиска подстроки в тексте. Этот алгоритм позволяет избежать ненужных сравнений, достигая линейной сложности O(N), где N - длина текста.

Принцип работы префикс-функции

Префикс-функция для строки s длиной n вычисляет массив pi, где pi[i] - это длина наибольшего префикса строки s[0...i], который также является суффиксом этой подстроки. Этот массив позволяет быстро сдвигать позицию поиска при несовпадении символов, используя ранее вычисленную информацию о строке.

Алгоритм Кнута-Морриса-Пратта (КМП)

Построение префикс-функции:

Вычисляем префикс-функцию для образца (паттерна) P длиной m.

Используем префикс-функцию для определения максимального сдвига, на который можно сдвинуть образец относительно текста при несовпадении.

Поиск подстроки:

Сравниваем символы текста и образца.

При несовпадении используем значение префикс-функции, чтобы сдвинуть образец максимально далеко, не пропуская потенциальные совпадения.

Продолжаем процесс до тех пор, пока не пройдем весь текст.

Пример

Рассмотрим пример поиска подстроки "abc" в тексте "ababcabcabc".

Вычислим префикс-функцию для подстроки "abc":

Префикс-функция: [0, 0, 0]

Поиск подстроки:

Сравниваем "abc" с текстом, начиная с первой позиции.

Находим совпадение на позиции 2.

При несовпадении используем префикс-функцию для сдвига образца и продолжаем поиск.

Преимущества

Эффективность: КМП алгоритм имеет линейную сложность O(N), что делает его очень эффективным для больших текстов.

Избежание лишних сравнений: Использование префикс-функции позволяет избежать повторных сравнений символов, тем самым улучшая производительность.

Реализация метода Init

Описание реализации метода Init

Метод Init инициализирует массив префикс-функции для заданной строки. Он выполняет следующие действия:

Освобождает ранее выделенную память для массива префикс-функции, если она существует.

Вычисляет длину новой строки и выделяет память для массива префикс-функции.

Инициализирует первый элемент массива префикс-функции значением 0.

Использует два цикла: внешний цикл проходит по символам строки, а внутренний цикл сдвигает индекс префикс-функции при несовпадении символов.

Заполняет массив префикс-функции согласно алгоритму, увеличивая счетчик сравнений при каждом сравнении символов.

Ограничения на повторный вызов метода Init и их устранение

Повторный вызов метода Init может привести к утечке памяти, если ранее выделенная память для массива префикс-функции не будет освобождена. Эта проблема решена следующим образом:

Перед выделением новой памяти проверяется, существует ли уже выделенная память (не равна ли она nullptr). Если память существует, она освобождается с помощью delete[] pv.

Пояснение к генерации строки

Цикл генерации z-строки (префикс-функции) выполняется до N-1, а не до N, так как последний элемент строки уже не имеет последующего символа для сравнения. В префикс-функции сравниваются текущий символ и все предыдущие, поэтому необходимо сравнить все символы строки кроме последнего, чтобы корректно заполнить массив.

Пояснения к подсчету количества сравнений символов

Количество сравнений символов подсчитывается в процессе вычисления префикс-функции. Счетчик увеличивается каждый раз, когда происходит сравнение двух символов строки. Это позволяет экспериментально оценить сложность алгоритма и подтвердить,

что она линейна. В нашем классе для этого используется переменная cmpCount, которая обнуляется в начале метода Init и увеличивается при каждом сравнении символов внутри цикла.

Описание экспериментов

Экспериментальная оценка зависимости количества сравнений от длины строки и размера алфавита

Для проведения экспериментов были сгенерированы строки разной длины и разных размеров алфавита. Для каждой строки вычислялось количество сравнений, выполненных при вычислении префикс-функции.

Соображения насчет копирования объектов класса PF

Копирование объектов класса PF может вызвать проблемы, так как по умолчанию будет копироваться только указатель на массив, а не сам массив. Это может привести к некорректной работе и утечке памяти. Для решения этой проблемы в классе PF необходимо реализовать:

Конструктор копирования: который выполняет глубокое копирование массива префикс-функции.

Оператор присваивания: который также выполняет глубокое копирование, предварительно освобождая ранее выделенную память.

Эти меры обеспечат корректное копирование и присваивание объектов класса PF, избегая утечек памяти и других проблем.