Санкт-Петербургский Национальный Исследовательский Университет
Информационных Технологий, Механики и Оптики

Факультет инфокоммуникационных технологий и систем связи

Лабораторная работа №1

Вариант №1

Выполнил(и:)

Алексеев Т.

Бабаев Р.

Белисов Г.

Проверил

Мусаев А.А.

Санкт-Петербург,

2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной работы являлось знакомство с различными алгоритмами поиска подстрок: наивным, Рабина-Карпа, Бойера-Мура, Кнута-Морриса-Пратта.

Для достижения данной цели необходимо было выполнить следующие задания:

1. Задание 1: создание строки путём склеивания 500 первых простых чисел. Далее необходимо, используя четыре вышеупомянутых метода, найти самое часто повторяемое двузначное число. Выявить плюсы и минусы каждого алгоритма.
2. Задание 2: для определённого реферата использовать предпочтительный алгоритм поиска подстроки для определения процента плагиата.

Решения данных задач будут находиться на GitHub по ссылке: https://github.com/NorthPole0499/Algoritms_task_6

1. ЗАДАНИЕ 1

Для начала, в данном задании необходимо создать строку содержащую в себе 500 первых простых числе. Это было сделано с помощью функций переборных функций prost_500 и prime_num.

Далее необходимо было реализовать четыре алгоритма: наивный, Рабина-Карпа, Бойера-Мура, Кнута-Морриса-Пратта. Итог работы программы и подтверждение правильности работы можно увидеть на рисунке 1.

Принцип работы наивного алгоритма заключается в последовательном прохождении строки и поочерёдного сравнения всех элементов с требуемым шаблоном. Данный алгоритм очень неэффективен, так как даже, когда этого не требуется, происходит проверка совпадения.

Достоинства: лёгкость в реализации, понятность принципа.

Недостатки: проверяются абсолютно все элементы, как итог, слишком большое число лишних проверок, слишком долгая работа при больших входных данных.

Алгоритм Рабина-Карпа же сложнее за счёт использования хэш-функции. Первоначально производится расчёта хэша для шаблона. Уже после этого происходит проход по строке и расчёт хэша уже для элементов строки, равных длине шаблона. Если хэши совпадают, то происходит поэлементное сравнение строк на совпадение.

Достоинства: за счёт высчитывания хэша происходит ускорение сравнения несовпадающих строк, уменьшение числа поэлементного сравнения.

Недостатки: хэш различных строк может совпадать, из-за этого ошибочные проверки всё равно будут присутствовать, при больших шаблонах подсчитывание хэша может занимать много времени.

Принцип работы алгоритма Бойера-Мура состоит в различном изменении сдвига индекса сравнения. Для начала сравниваем поэлементно

строку с шаблоном, если нет совпадения, то проверяем, есть ли не совпавший элемент из строки в шаблоне. Если есть, то сдвигаем индекс до того момента, пока это совпадение не произойдёт. Если нет, то сдвигаем индекс на длину шаблона.

Достоинства: считается наиболее эффективным алгоритмом для поиска подстроки в общем назначении, поэтому подходит и для заранее неизвестных данных.

Недостатки: на некоторых заранее известных данных и на очень коротких текстах алгоритм будет работать медленнее своих конкурентов.

Алгоритм Кнута-Морриса-Пратта основан на применении префикс-функции. Для начала нам нужно рассчитать её для нашего шаблона, сравнивая префиксы и суффиксы и находя совпадающие из них. Их длины мы записываем в список. Далее сравниваем поэлементно строку и шаблон. При их несовпадении двигаем индекс по определённой формуле: index = длина совпавшего участка – значение префикс функции для не совпавшего элемента + 1.

Достоинства: использование префикс-функции позволяет отбросить много ненужных проверок, не происходит проверок всех элементов.

Недостатки: расчёт префикс-функции может занять много времени, время работы на коротких входных данных и на некоторых наборах может уступать другим.

```
Наивный алгоритм: (31, 73)
Алгоритм Рабина-Карпа: (31, 73)
Алгоритм Бойера-Мура: (31, 73)
Алгоритм Кнута-Морриса-Пратта: (31, 73)

Process finished with exit code 0
```

Приложение 1 – Пример вывода программы для Задания 1

2. ЗАДАНИЕ 2

Вторая задача заключается в том, что нам необходимо определить процент плагиата в реферате (docx-файла) на основе одноимённой статьи в Википедии. Для выгрузки текста из файла и страницы Википедии используются библиотеки python-docx и wikipedia-api соответственно. Плагиатом считается 3 совпадающих слова, идущих подряд, поэтому удобно работать с конкретными словами в списках, которые возвращают функции get_text_from_file и get_text_from_ wiki. Для выведения более точного процента плагиата слова обрабатываются специальным образом, чтобы избавиться от лишних пробелов и знаков препинания.

Для определения плагиата были написаны и рассмотрены 4 алгоритма: Наивный, Рабина-Карпа, Бойера-Мура и Кнута-Морриса-Прата.

Кроме того, с помощью библиотеки timeit было вычислено время работы каждого алгоритма для выявления наиболее эффективного. Как можно увидеть на рисунке 2, таковым оказался алгоритм Рабина-Карпа, который выполняется в худшем случае в 3 раза быстрее остальных в данном случае.

```
Наивный алгоритм: 5.3236673c
Процент плагиата - 56.54%
Алгоритм Рабина-Карпа: 1.531394500000001c
Процент плагиата - 56.54%
Алгоритм Бойера-Мура: 7.333490099999999c
Процент плагиата - 56.54%
Алгоритм Кнута-Морриса-Пратта: 7.021582599999999c
Процент плагиата - 56.54%

Process finished with exit code 0
```

Приложение 2 – Пример вывода программы для Задания 2

ЗАКЛЮЧЕНИЕ

Таким образом, в ходе выполнения данной лабораторной работы были достигнуты все поставленные цели. Произошло знакомство с 4 алгоритмами сортировки: наивным, Рабина-Карпа, Бойера-Мура, Кнута-Морриса-Пратта. Также, были выявлены плюсы и минусы каждого алгоритма, применены на практике.

СПИСОК ЛИТЕРАТУРЫ

1) Алгоритм Кнута-Морриса-Пратта // URL: https://neerc.ifmo.ru/wiki/index.php?title=Алгоритм_Кнута-Морриса-Пратта
(дата обращения: 19.02.2023)