Московский Авиационный Институт
(Национальный Исследовательский Университет)
Институт № 8 информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работ №2 по курсу
«Дискретный анализ»

Поразрядная сортировка

Студент: Пермяков Никита Александрович
Группа: М80 – 208Б-19
Вариант: 6-1
Преподаватель: Кухтичев Антон Алексеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2020

Содержание

Постановка задачи.

Условие: Необходимо реализовать алгоритм поиска образцов для указанного алфавита.

Вариант 1 - 1:

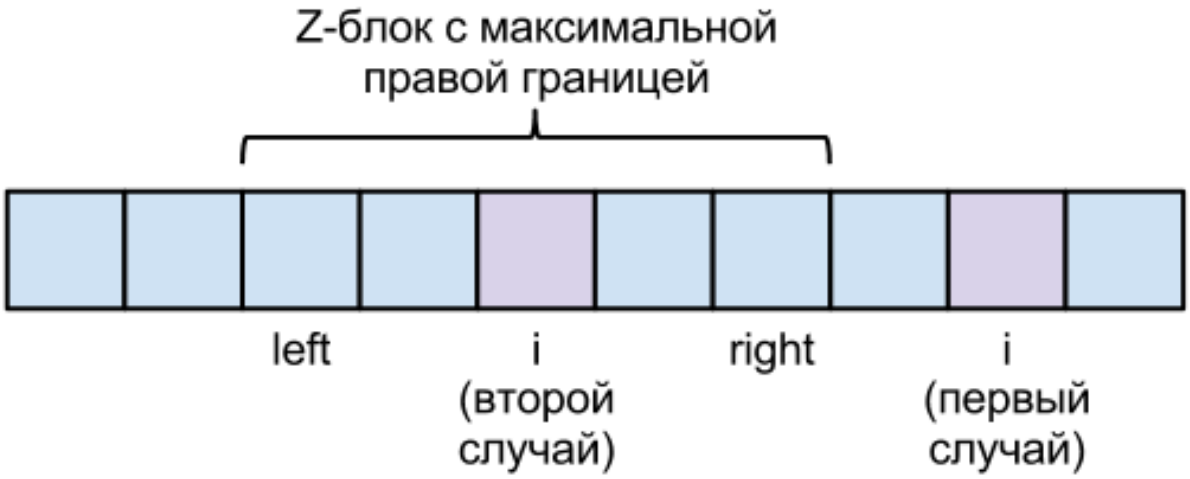
Алгоритм: Поиск одного образца при помощи алгоритма Кнута-Морриса-Пратта.

Алфавит: Слова не более 16 знаков латинского алфавита (регистронезависимые)

Метод и алгоритм решения:

Для решения задачи был создан класс вектор, где хранились входные данные в структуре. Алгоритм поиска подстроки в строке должен быть реализован за линейное время.

Сначала найдем зет функцию, которая будет для каждого элемента паттерна ставить в соответствие число - длина максимального префикса подстроки, начинающейся с позиции x в строке S, который одновременно является и префиксом всей строки S.



Этот алгоритм работает за O(|S|), так как каждая позиция пробегается не более двух раз: при попадании в диапазон от left до right и при высчитывании Z-функции простым циклом.

Функция при КМП обходе будет сдвигать строку патерна на заданное число элементов.

Пусть файл любой длины, значит нужно научить программу обрабатывть поток частями, при этом не теряя и не перебирая несколько раз возможные вхождения.

Общие сведения о программе

Сниппет зет-функции, принимает на вход паттерн, позвращает значение зет функции. Массив z изначально заполняется нулями. Текущий самый правый отрезок совпадения полагается равным [0; 0], т.е. заведомо маленький отрезок, в который не попадёт ни одно i. Внутри цикла по i = n-1 сначала определяем начальное значение z[i] — оно либо останется нулём, либо вычислится на основе приведённой формулы.

После этого выполняется алгоритм, который пытается увеличить значение z[i] настолько, насколько это возможно. В конце выполняется обновление текущего самого правого отрезка совпадения [l;r], если, конечно, это обновление требуется — т.е. если i + z[i] - 1 > r.

Сниппет стандартного алгоритма КМП, принимает ранее расчитанную префикс функцию, паттерн, буффер с текстом, позицию, с которой начинать поиск и вектор в который будет записан результат.

Сниппет, выводящий корректный результат на экран. Принимает вектор с индексами вхождения паттернов в буффер, вектор с каких индексов начинаюся новые строки в

буффере, количество ранее считанных линий, размер занимаемой в начале части текста с прошлого вхождения, последняя выведенная позиция в прошлом вхождении.

Буфер можно реализовать как:

- линейный

- динамический

- кольцевой

Важно поймать оптимальное соотношение относительно устройста по параметрам:

1. Эффективность по памяти
2. Эффективность по производительности (минимальные накладные расходы на изменение размера массива; сохранение, по возможности, константного времени доступа на чтение/запись к элементам массива)
3. Совместимость с обычными статическими массивами на низком уровне. Например, необходимым условием для применения динамического массива в вызовах функций API операционной системы может быть обязательное размещение элементов массива в непрерывном блоке физической оперативной памяти в порядке, соответствующем индексации массива. Если такое требование не выполняется, динамические массивы окажется невозможно использовать в сочетании с низкоуровневым кодом.

Последняя позволяет синхронизировать результаты при поиске вхождений строк в текст если сам происк производится одинаковыми кусками которые хранятся в буффере или в целом тексте.

Дневник отладки.

Сначала написал программу с линейным буфером. Работает правильно, но алгоритм не эффективный:

1-7) ошибка компиляции - переделал аргументы для входных данных так, что аргумент стал 1 - строка.

8-13) превышен лимит времени - пробовал рассинхронизацию и scanf (неудачно)..

14) неверный ответ - переполнилась интовая переменная цикла, после изменения её на llu все тесты прошли(9 + 11 с кодстайлом)

15) Изменил на динамический буфер.

Ошибка выполнения - заменял размер массива для входной строки.

30) Изменил на кольцевой буфер

Далее рефакторинг и кодстайл.

Тесты производительности.

скрипт генерации:

```
import numpy as np

import string

from tqdm import tqdm


test_data = ['CAT', 'cat', 'dog', 'Dog', 'CAT', 'Dog', 'T', 'cAT', 'dog', 'Dog', 'Cat', 'cat', 'dog', 'Dog', 'CAT', 'sausage', 'lEna', 'x', 'Cat']


with open('test1.txt', 'w') as f:

    for _ in tqdm(range(1000000)):

        line = ''
```

```
    for _ in range(np.random.randint(10) + 5):
        line += str(np.random.choice(test_data, 1)[0]) + ' '
    f.write(line + '\n')
```

Количество столбцов в файле: [5, 15]

Отдаем программе через pipe

Считаем среднее по трем для каждого набора

Результаты:

Количество рандомных строк	Время обработки
1 000 000	1499,847400 мс
100 000	1349,480374 мс
10 000	990, 86200 мс

Недочеты.

Программа работает правильно, однако имеет в теле main реализацию сразу трех функций алгоритма. Это включение, и вынос структуры с данными за класс с методами позволило пройти 14 тест.

Вывод

При решении задачи поиска всех включений образца в текст наиболее часто применяются алгоритмы Ахо-Корасик и Кнута-Морриса-Пратта.

Алгоритм Ахо-Корасик использует автомат и применяется в случаях — когда образцов для поиска несколько. И решает эту задачу быстрее, чем построение нескольких автоматов Кнута-Морриса-Пратта.

Алгоритм Кнута-Морриса-Пратта решает ту же задачу, однако он не может работать с текстом, вводимым в режиме реального времени, ему нужно заранее знать текст, в котором нужно искать образец. Также, если текст достаточно объемный, алгоритм Кнута-Морриса-Пратта будет использовать гораздо больше памяти, чемалгоритс Ахо-Корасик. Однако, если текст небольшой и дан заранее, алгоритм Кнута-Морриса-Пратта будет работать быстрее.

В данной работе реализовывались все виды буферов. задача была решена с помощью циклического буфера.