Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №4 по курсу «Дискретный анализ»

Студент: А. О. Дубинин
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б

Дата:
Оценка:
Подпись:

Москва, 2019

Лабораторная работа №4

Задача: Необходимо реализовать один из стандартных алгоритмов поиска образцов для указанного алфавита.

Вариант алфавита: Числа в диапазоне от 0 до $2^{32} - 1$.

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

1 Описание

Согласно [1] Алгоритм Кнута — Морриса — Пратта (КМП-алгоритм) — эффективный алгоритм, осуществляющий поиск подстроки в строке. Время работы алгоритма линейно зависит от объёма входных данных, то есть разработать асимптотически более эффективный алгоритм невозможно.

Алгоритм был разработан Д. Кнутом и В. Праттом и, независимо от них, Д. Моррисом. Результаты своей работы они опубликовали совместно в 1977 году.

Идея состоит в том, что за основу мы берем наивный алгоритм, только отличие будет, в том, что при несовпадении наивный алгоритм сдвигает образец на одну позицию, когда КМП сдвигает на больше позиций.

Для того, чтобы сделать больший сдвиг, КМП вводит 2 функции: префикс функция и функция неудачи.

Префикс функция, согласно[2]:
Дана строка $s[0 \dots n-1]$. Требуется вычислить для неё префикс-функцию, т.е. массив чисел $\pi[0 \dots n-1]$, где $\pi[i]$ определяется следующим образом: это такая наибольшая длина наибольшего собственного суффикса подстроки $s[0 \dots i]$, совпадающего с её префиксом (собственный суффикс — значит не совпадающий со всей строкой). В частности, значение $\pi[0]$ полагается равным нулю.

Функция неудач, согласно[4]:
Для каждого места i от 1 до $n+1$ определим функцию неудачи $F(i)$ как $\pi(i-1)+1$, причем $\pi(01)$ приняты равными 0.

Мы используем указатель p на место в P и один указатель c (от "current"— текущий символ) на место в T. После несовпадения в позиции i + 1 > 1 строки P алгоритм Кнута-Морриса-Пратта "сдвигает"P так, что следующими будут сравниваться символы в позиции c строки T и в позиции $\pi(i)+1$ строки P. Но $\pi(i)+1 = F(i+1)$, так что общий "сдвиг"можно выполнить за константное время, просто полагая p равным F(i + 1). Осталось два особых случая. Когда несовпадение нашлось в позиции 1 из P, то p полагается равным F(1) = 1, а c увеличивается на 1. Когда находится вхождение P, то P сдвигается вправо на n - $\pi(n)$ мест. Этот сдвиг реализуется тем, что F(n + 1) полагается равным $\pi(n)+1$.

2 Исходный код

Искомый образец задается на первой строке входного файла.

Затем следует текст, состоящий из чисел, в котором нужно найти заданный образец.

lab4.cpp	
int64_t ReadNumber(bool &EOF_status, int32_t &column, int32_t &row, bool &newline);	Функция, считывающая одно число.
void ZFunc(std::vector<int32_t> &z, const std::vector<int64_t> &string);	Функция которая строит вектор Z-блоков vector<int32_t> &z, по входящему вектору образца.
void PrefixFunc(std::vector<int32_t> &sp, const std::vector<int32_t> &z);	Функция, вычисляющая префикс функцию, с помощью Z-блоков.
void FailureFunc(std::vector<int32_t> &F, const std::vector<int32_t> &sp);	Функция неудач, вычисляемая с помощью префикс вектора.
void Kmp(const std::vector<int64_t> &string, const std::vector<int32_t> &F);	Функция, считывающая текст, вычисляющая и выводящая вхождения образцов в этот текст.

3 Производительность

Реализованный алгоритм КМП сравнивается с наивным поиском образца в тексте. Мной был реализован наивный поиск:

```
1  void NaiveSearch(std::vector<int32_t> &pattern, vector<pair<pair<int32_t, int32_t>,
       int32_t >> &text) {
2
3      for (int k = 0; k < text.size() - pattern.size(); ++k) {
4          int i = k;
5          int j = 0;
6          while (text[i].second == pattern[j] && j < pattern.size()) {
7              i++;
8              j++;
9          }
10         if (j == pattern.size()) {
11             cout << text[k].first.first << ", " << text[k].first.second << endl;
12         }
13     }
14
15 }
```

На тесте состоящем из 2 500 000 слов, образец длиной 50, алфавит 20000.

```
art@mars:~/study/semester_3/DA/labs/my_lab_4/test_code$ bash wrapper.sh
50210,50211c50210,50211
<Search: Naive
<Time of working 1547 ms.
---
>Search: KMP
>Time of working 665 ms.
Failed
```

Поиск наивным алгоритмом выполняется за достойное время. Согласно [3] наивный алгоритм работает за $O(m * (n - m))$, где m – длина образца, а n – длина текста. Однако, если длина образца мала по сравнению с текстом, то сравнения происходят не так долго.

В алгоритме КМП суммарное время работы алгоритма будет равно Θ (m+n), где n — длина текста T.

4 Выводы

Выполнив вторую лабораторную работу по курсу «Дискретный анализ», я изучил различные виды поиска по тексту, реализовал алгоритм Кнута-Морриса-Пратта. Узнал историю этого алгоритма, что первым этот алгоритм опубликовал Матиясевич в 1969 году, но переведен на английский был позднее, чем вышел алгоритм КМП, поэтому собственно за их фамилиями и закрепился этот алгоритм. Так же посмотрел на разные реализации этого алгоритма: с использованием Z-функции и без. Отдельно хочется отметить, что с определенными изменениями можно реализовать алгоритм для поиска подстроки в реальном времени.

Список литературы

[1] *Алгоритм Кнута — Морриса — Пратта*
URL: https://ru.wikipedia.org/wiki/Алгоритм_Кнута_-_Морриса_-_Пратта.

[2] *MAXimal :: algo :: Префикс-функция. Алгоритм Кнута-Морриса-Пратта*
URL: http://emaxx.ru/algo/prefix_function.

[3] *Наивный алгоритм поиска подстроки в строке*
URL: https://neerc.ifmo.ru/wiki/index.php?title=Наивный_алгоритм_поиска_подстроки_в_строке.

[4] Дэн Гасфилд *Строки, деревья и последовательности в алгоритмах. Информатика и вычислительная биология* — Издательство « СПб.: Невский диалект», 2003.. — 656 с. (ISBN 5-7940-0103-8)