Лабораторная работа № 4 по курсу дискретного анализа: Поиск образца в строке

Выполнил студент группы 08-207 МАИ *Днепров Иван.*

Условие

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

1. Необходимо найти вхождение одного слова в текст, количество строк текста, их длинна, размер алфавита и соответственно длинна искомого образца. Слова считать элементом алфавита.

2. 3. Поиск одного образца при помощи алгоритма Апостолико-Джанкарло. 1. Слова не более 16 знаков латинского алфавита (регистронезависимые).

Метод решения

В начале программа зачитывает паттерн и по нему строит алфавит в дереве PATRICIA, в котором каждому слову соответствует целочисленное значение (индекс), далее при работе с паттерном и поиске его в тексте мы будем оперировать этими индексами. Выполняем пропроцессинг паттерна (получаем массивы значений функций N, R, L' и l'). Далее вызываем функцию AGSearchAndPrint, которая работает с буфером текста размер которого равен 10 паттернам. В нем хранятся индексы соответствующих слов. Мы пробегаем по буферу, выводи результаты поиска и зачитываем очередную порцию в буфер. При достижении конца потока ввода мы обрабатываем буфер, освобождаем память и завершаем программу.

Описание программы

В main я только инициализирую необходимые структуры, запускаю функции зачитывания паттерна, наполнения словаря, препроцессинга паттерна, поиска входжений паттерна в тексте и освобождение памяти. В agalgorithm у меня находятся функции препроцессинга паттерна и его поиска в тексте. В ptread находятся две цункции: считывания паттерна и считывания части текста в буфер. В TPatricia хранятся функции работы с деревом, на котором реализован алфавит. А в wordabc функции работы со словарем.

Дневник отладки

Я не использовал буфер для текста, а считывал его целеком, в итоге на 14 тесте я не проходил по ограничениям по памяти. Я не верно считал сдвиги для правила хорошего суффикса при несовпадении. Я добавли работу с функцией l'(i), которая находит

максимальную длинну префикса для подслова 1..i паттерна, благодаря которой при не нахождении хорошего суффикса слова мы не теряем вхождение в случае, если значение функции l'(i) в данной позиции не нулевое. Я хранил соварь в линейном списке, за счет чего не проходил по времени на 17 тесте. В итоге я взял дерево из 2 лабораторной и скорость поиска в словаре для больших словарей стала значительно выше. Так же было множество недочетов вроде опечаток в функциях, не закомментированного отладочного вывода и путаници с индексами.

Выводы

Алгоритм Апостолико-Джанкарло – не тривиален в реализации алгоритм, но с точки зрения скорости работы это отличный алгоритм поиска, который, при совмещении с PATRICIA в качестве словаря будет работать очень быстро для поиска в большом тексте паттерна с большим словарем. Данный алгоритм применим для поиска точных совпадений паттерна в тексте, в том случае, если паттерн мы получаем либо раньше, либо одновременно с текстом, так как данный алгоритм не производит предобработку текста. А предобработка паттерна окупается почти всегда, за исключением случаем, с очень небольшим текстом для поиска.