Отчёт по Лабораторной 5

Выполнил Ромашов А.В. студент группы ПМИ-21

В папке release есть .exe файл для каждого задания, в него можно ввести другие тесты, которые необходимо проверить.

**Тесты для task1.(RK)**

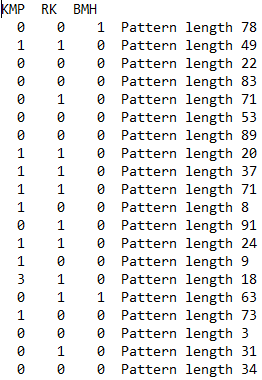
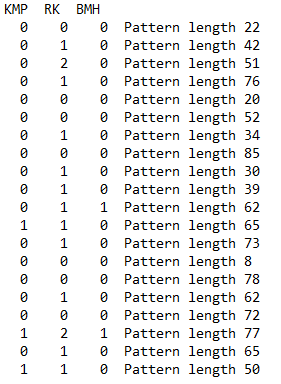
**Тесты для task2.(BMH)**

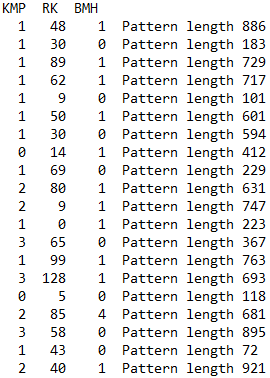
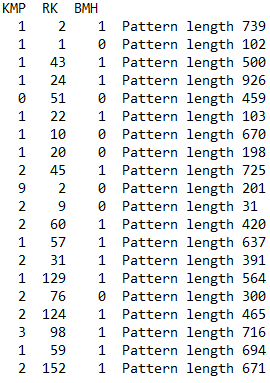
**Тесты для task3.**

//Время в миллисекундах

Для небольшого текста, скорость алгоритмов практически ничем не отличается. (Всего 93 символа)

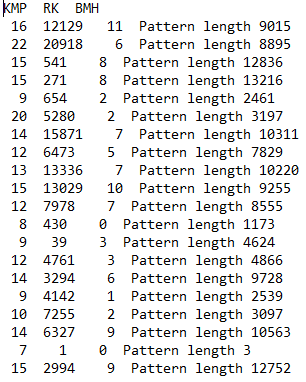
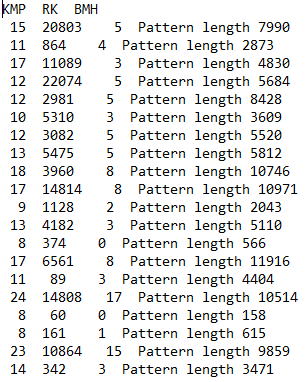


Для среднего текста алгоритм Рабина карпа выполняется значительно дольше. (Всего 1013 символов)



Для большого текста алгоритм KMP выполняется заметно медленнее BMH.

(Всего 13092 слов)



***Сравнительный анализ:***

*n – длина текста. m – длина паттерна*

Наивный алгоритм

Работает медленно при большой длине паттерна.   
Приемлемое время работы на практике и простота реализации. Благодаря этому алгоритм применяется, например, в браузерах и текстовых редакторах (при использовании Ctrl + F), потому что обычно паттерн, который нужно найти, очень короткий по сравнению с самим текстом. Также наивный алгоритм используется в стандартных библиотеках языков высокого уровня (C++, Java), потому что он не требует дополнительной памяти.  
В худшем случае O(n\*m)

Алгоритм Рабина-Карпа.

Можно использовать, когда текст небольшой или для поиска совпадений множественных шаблонов одинаковой длины. Так как чем больше символов, тем больше необходимо найти хэшей, при большом количестве символов, весьма вероятно будут коллизии и поэтому необходима дополнительная проверка строк с совпадающими хэшами, что также замедляет скорость алгоритма.   
В худшем случае эффективность O(n\*m).

Алгоритм Бойера-Мура-Хорспула

Алгоритм Бойера-Мура считается наиболее эффективным алгоритмом поиска шаблонов в стандартных приложениях и командах, таких как Ctrl+F в браузерах и текстовых редакторах. На больших алфавитах (например, Юникод) может занимать много памяти. Нет преимуществ над другими алгоритмами при маленьком алфавите или маленькой длине шаблона.  
В худшем случае эффективность O(n\*m).

Алгоритм Кнутта-Морриса-Пратта

Самый надёжный из рассмотренных алгоритмов, так как не зависит от хэшей, длины паттерна и размера алфавита. Время работы алгоритма линейно зависит от объёма входных данных, то есть разработать асимптотически более эффективный алгоритм невозможно.   
В худшем случае эффективность O(n+m).