Постановка задачи:

Данная работа должна показать относительную скорость 4-ёх алгоритмов, нахождения

подстроки в строке и то,на каких данных определённые алгоритмы работают лучше.

Параметры вычислительного узла:

Процессор --- 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 Ghz

Размер ОЗУ --- 8,00 ГБ

ОС --- Windows 11 версия 22H2

Краткое описание алгоритмов:

n - длина строки

m - длина подстроки

SimpleAlgorithm

Наивный алгоритм поиска подстроки в строке. Имеет сложность O(m\*n).

Обычно нигде не используется,так как есть алгоритмы лучше и несильно сложнее

HashAlgorithm

Немного улучшенный наивный алгоритм . Тоже Имеет сложность O(m\*n).

Однако имеет преимущество за счёт использования хэша. Имеет сложность O(m+n),если колизии никогда не просходят и с учётом предварительной обработки.

KnuthMorrisPrathAlgorithm

Очень эффективен в случае, если в строке и подстроке содержатся повторяющиеся последовательности.

Например,применяется для поиска для анализа геномов живых существ.

Геном шифруется 4 символам часто встречаются повторяющиеся последовательности. Сложность O(n+m) с учётом предварительной обработки

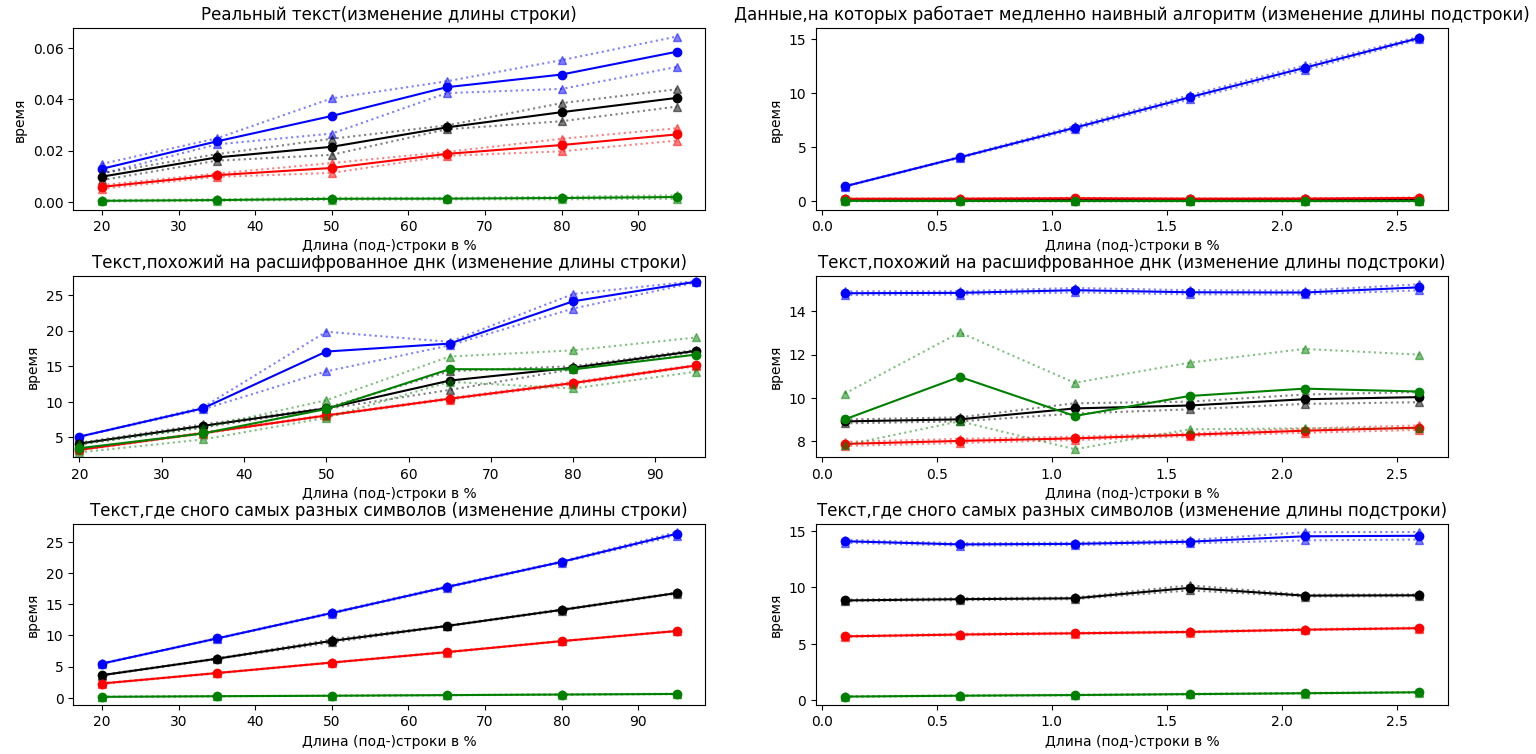
BoeraMurraHorspulaAlgorithm

Используется для анализа теста,где встречается большое кол-во разных букв и

нет часто повторяющихся последовательстей. Сложность в неудачных случаях O(n\*m).

В некоторых удачных O(n/m). В среднем O(n+m).

Результаты измерения:



синий - наивный алгоритм

черный - хэш алгоритм

красный - Кнута Морриса Прата алгоритм

зелёный - Боера Мура Хорспула алгоритм

Точки на графике соединены линиями, т.к на некоторых графиках точки сливаются в одну.

Жирными точками обозначены средние значения времени работы алгоритмов на конкретных данных

Полупрозрачными точками обозначены максимальные и минимальные возможные значения с учётом погрешности

Максимальное значение = Среднее значение + погрешность

Минимальное значение = Среднее значение - погрешность

(Среднее значение и погрешность вычисленные по формулам в соответствующем файле )

Время работы алгоритмов на самых больших данных **в секундах** :

Реальный текст(изменение длины строки)

SimpleAlgorithm -- 0.058568522334098816 / +- 0.00592452309652095

HashAlgorithm -- 0.040547147393226624 / +- 0.003388300714103021

KnuthMorrisPrathAlgorithm -- 0.02630533277988434 / +- 0.002411428674861216

BoeraMurraHorspulaAlgorithm -- 0.0018752813339233398 / +- 0.0006696823679643401

Текст,похожий на расшифрованное днк (изменение длины строки)

SimpleAlgorithm -- 26.864076256752014 / +- 0.09449497545135868

HashAlgorithm -- 17.15424945950508 / +- 0.13270734483056007

KnuthMorrisPrathAlgorithm -- 15.094659075140953 / +- 0.09748100330652587

BoeraMurraHorspulaAlgorithm -- 16.630280509591103 / +- 2.3979066537092057

Текст,где сного самых разных символов (изменение длины строки)

SimpleAlgorithm -- 26.305947050452232 / +- 0.28988864330274067

HashAlgorithm -- 16.798237696290016 / +- 0.0894775228034449

KnuthMorrisPrathAlgorithm -- 10.704245492815971 / +- 0.07582927853589727

BoeraMurraHorspulaAlgorithm -- 0.6011336296796799 / +- 0.0324461565754641

Данные,на которых работает медленно наивный алгоритм (изменение длины подстроки)

SimpleAlgorithm -- 15.135793685913086 / +- 0.12747532959477492

HashAlgorithm -- 0.1653351386388143 / +- 0.01145274664522238

KnuthMorrisPrathAlgorithm -- 0.30596303939819336 / +- 0.059930620447006755

BoeraMurraHorspulaAlgorithm -- 0.028971036275227863 / +- 0.040447969505512986

Текст,похожий на расшифрованное днк (изменение длины подстроки)

SimpleAlgorithm -- 15.105597779154778 / +- 0.1385728380739316

HashAlgorithm -- 10.037889152765274 / +- 0.2344542590598757

KnuthMorrisPrathAlgorithm -- 8.629228174686432 / +- 0.10438939085585561

BoeraMurraHorspulaAlgorithm -- 10.294344410300255 / +- 1.7030432018581243

Текст,где сного самых разных символов (изменение длины подстроки)

SimpleAlgorithm -- 14.579974070191383 / +- 0.334668838753513

HashAlgorithm -- 9.30153739452362 / +- 0.07985471034958581

KnuthMorrisPrathAlgorithm -- 6.387519896030426 / +- 0.051173060259383514

BoeraMurraHorspulaAlgorithm -- 0.7007870078086853 / +- 0.031186563570722824

Обоснование и анализ результатов:

Реальный текст(изменение длины строки)

Замеры на относительно небольших данных.График отражает зависимость времени работы алгоритма от длинны строки.График должен показать то,как быстро работают алгоритм по сравнению друг с другом

Текст, похожий на расшифрованное днк (изменение длины строки)

Замеры на очень больших данных данных. График отражает зависимость времени работы алгоритма от длинны строки. График должен показать то, что алгоритм Кнута Морриса Прата(КМП) здесь эффективней,чем алгоритм Боера Мурра Хорспула(БМХ), т.к БМХ сокращает сравнение за счёт запоминания букв,которые он сравнил, однако здесь всего четыре буквы и конкретная буква появлятся может часто в тексте,однако в тексте появляются повторяющиеся последовательности из этих букв, за счёт "запоминания" этих последовательностей и сокращается перебор. Хэш алгоритм работает быстро, так как подобраны символы таким образом, что коллизия происходит редко(при имеющейся хэш функции)

Текст, похожий на расшифрованное днк (изменение длины строки)

Замеры на очень больших данных данных. График отражает зависимость времени работы алгоритма от длинны строки. График должен показать то,как быстро работают алгоритм по сравнению друг с другом, а так же то, что здесь БМХ эффективней,чем КМП,ведь встречаются самые разные символы,запоминая которые можно сократить перебор,а вот повторяющихся последовательностей здесь крайне мало,за счёт чего и ускоряет свою работу КМП

Данные, на которых работает медленно наивный алгоритм (изменение длины подстроки)

Замеры на очень больших данных данных. График отражает зависимость времени работы алгоритма от длинны подстроки. График должен достаточно ясно показать,что сложность наивного алгоритма O(n\*m),в то время как у остальных O(n+m)

Текст, похожий на расшифрованное днк (изменение длины подстроки)

Замеры на очень больших данных данных. График отражает зависимость времени работы алгоритма от длинны подстроки на данных с повторяющимися участками. График показывает то,как быстро работают алгоритм по сравнению друг с другом,а ещё, что обычно O(n\*m) - сложность наивного имеет роль в тексте с повторяющимися участками.

Текст, где много самых разных символов (изменение длины подстроки)

Замеры на очень больших данных данных.График отражает зависимость времени работы алгоритма от длинны подстроки на данных не часто встречающимися участками и с различными символами. График показывает то,как быстро работают алгоритм по сравнению друг с другом,а ещё, что обычно O(n\*m) - сложность наивного не имеет роль в тексте с не часто встречающимися участками и с различными символами.

Данные на которых выполнялось тестирование:

Всё находится в директории \Data

В каждой из подпапок было примерно 16 аналогичных файлов имеющий для many\_different\_simbols и recurring\_sequences в 3 000 раз больший размер фалов .