Python Task. Отчёт о выполнении работы

Постановка задачи:

Реализовать несколько алгоритмов поиска подстроки в строке (минимум 4) и сравнить их по производительности, использованию памяти.

Характеристики вычислительной машины:

Процессор: Apple Silicon M1, 3.2 ГГц, 8 Core (4+4)

Оперативная память: 8.0 Гб

Операционная система: 64 – разрядная Mac OS

План тестирования:

1. Написание алгоритмов
2. Генерация текстов, тестов
3. Измерения
4. Тестирование
5. Анализ результата
6. Отчет

Асимптотическая сложность алгоритмов:

σ - размер алфавита t - длина текста

p - размер паттерна a – размер

ответа m – суммарная длина всех паттернов

1. Наивный (Brute – Force) Среднее:

$O(p(t - p))$

Худшее: $O(t^2)$

Память: $O(1)$

2. Алгоритм Рабина-Карпа Среднее:

$O(p + t)$

Худшее: $O(pt)$

Память: $O(1)$

3. Алгоритм Кнута-Морриса-Пратта

Среднее: $O(p + t)$

Худшее: $O(p + t)$

Память: $O(p)$

4. Алгоритм Ахо-Корасик Среднее:

$O(t)$

Худшее: $O(pt)$

Память: $O(m\sigma)$

5. Алгоритм Бойера – Мура

Среднее: $O(m + t + a)$ Худшее: $O(t)$

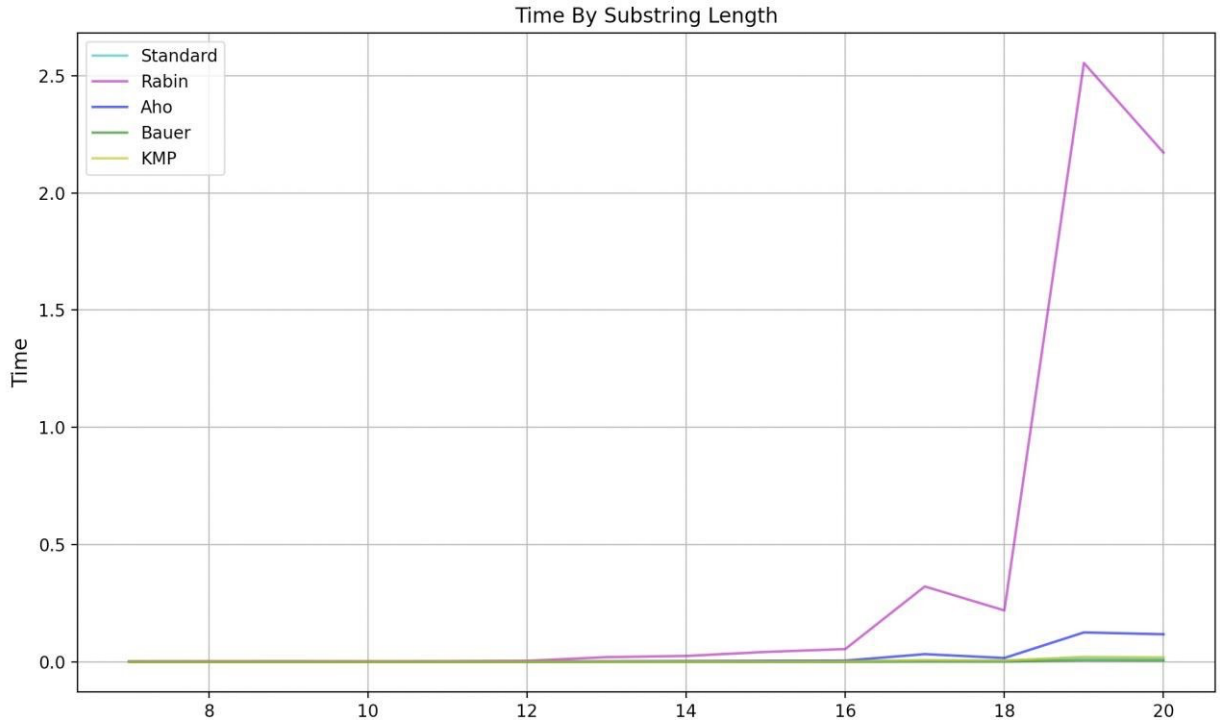
Память: $O(p + \sigma)$

Входные данные для тестирования:

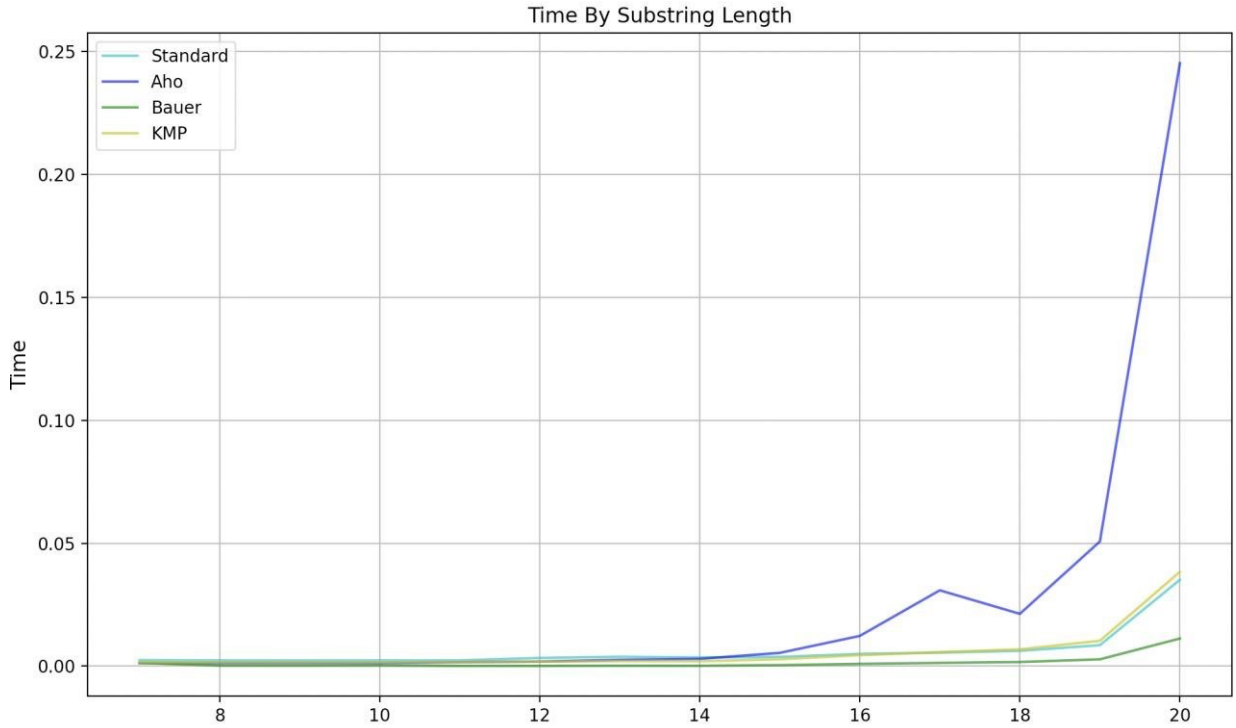
Данные генерируются таким образом, что подстрока всегда начнется с какого-то конкретного места (на конце).

Лучшие и худшие данные для каждого конкретного алгоритма рассматриваться не будут, так как в этом нет никакого смысла. Мы хотим узнать, как алгоритмы работают при реальном использовании, на полностью случайных данных, а не на математически смоделированных. Заметим, что в методе подсчета памяти использован встроенный интерфейс сборщика мусора gc. Он собирает и уничтожает лишние данные как до, так и после прогона алгоритма.

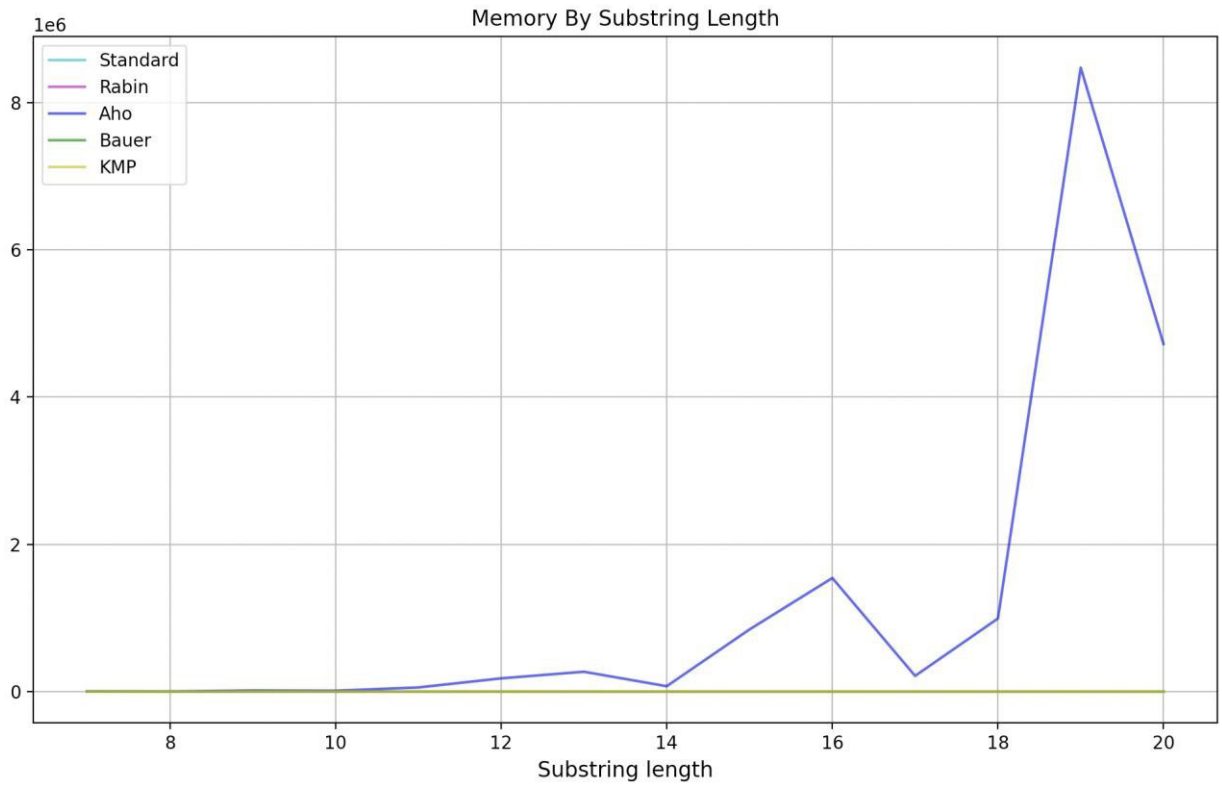
Результаты:



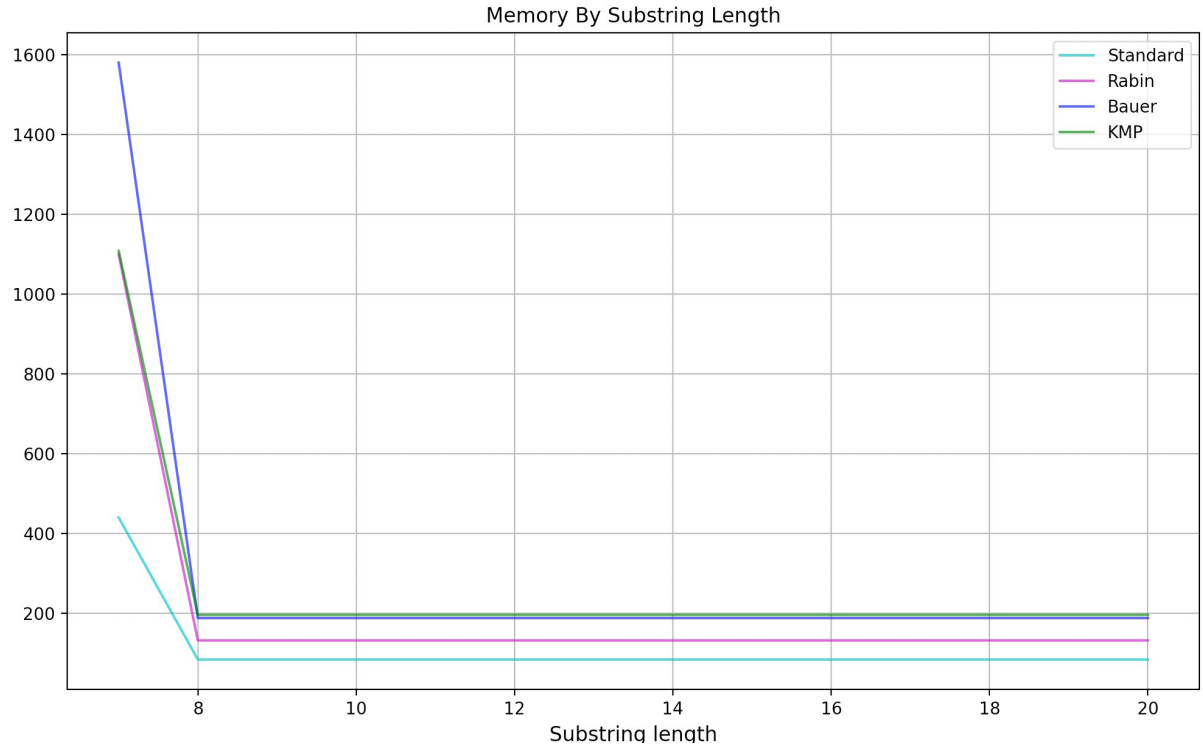
Время с Рабином – Карпом [1]



Время без Рабина – Карпа [2]



Память с Ахо – Корасиком [3]



Память без Ахо – Корасика [4]

Обоснование результатов

Итак, алгоритм Рабина – Карпа самый медленный [1]. Такой вывод удалось сделать из-за того, что почти на каждой его итерации производилась достаточно сложная арифметика (работа хэш - функции), которая тянула за собой время в космос.

По памяти, очевидно, алгоритм Ахо -Корасика самый прожорливый [3]. Это связано с наличием структур, необходимых для построения конечного автомата, а также предобработкой текста. Если не обращать внимание на структуры этого алгоритма, то по памяти все алгоритмы работают примерно одинаково [4].

Самым быстрым оказался алгоритм Бойера – Мура (что логично, так как он считается самым эффективным и используется много где: str.find в python или Ctrl + F)

Чеклист корректности тестирования:

Написание алгоритмов

1.1. В алгоритме нет ненужных вычислений. Функции count_time() и count_memory() вычисляют затраченные время и память соответственно только для работы алгоритма. 1.2. На вычислительном узле завершены все некритичные для тестирования и ресурсоемкие задачи. Тестирование

проводилось при оптимальной загрузке процессора только проектом.

1.3. Выполнен тестовый запуск перед итоговым. Предварительные замеры были сделаны.

1.4. Был составлен план тестирования.

1.5. План тестирования содержит разнообразные размеры данных.

1.6. Не было оценено время тестирования, если слишком мало или велико.

Генерация тестов

2.1. Охарактеризован лучший случай данных алгоритма.

2.1.1. Сгенерированы лучшие данные.

2.2. Охарактеризован худший случай данных для алгоритма.

2.2.1. Сгенерированы худшие данные.

2.3. Сгенерированы случайные данные.

Измерение времени и памяти

3.1. Подготовлено окружение для корректного, с учетом погрешности, измерения времени работы алгоритма.

3.2. Для каждого измерения выполняется несколько холостых запусков. Графики строились не по первому запуску скрипта.

3.3. Автоскрипты для запуска написаны не были.

Тестирование

4.1. Запущено тестирование.

Анализ результатов

5.1. В отчёте на графиках экспериментальные точки не соединены линиями.

5.2. На экспериментальные точки не нанесены доверительные интервалы.

5.3. Аппроксимация экспериментальных результатов не выполнена.

Написание отчета

6.1. В отчет включена постановка задачи.

6.2. В отчёт включены параметры вычислительного узла.

6.3. В отчёт включена часть результатов измерений.

6.4. В отчёте есть частичное обоснование экспериментальных результатов.

6.5. Приложение к отчёту не содержит готовую среду для воспроизведения тестов.

1.1	В алгоритме нет ненужных вычислений	
1.2	На вычислительном узле завершены все некритичные для тестирования и ресурсоёмкие задачи	
1.3	Выполнен тестовый запуск, сделаны предварительные замеры времени	
1.4	Составлен план тестирования	
1.5	План тестирования содержит разнообразные размеры данных	

1.6	Оценено время тестирования, если слишком велико или слишком мало — п.1.4	
2.1a	Охарактеризован лучший случай данных для алгоритма	
2.1b	Сгенерированы лучшие данные	
2.2a	Охарактеризован худший случай данных для алгоритма	
2.2b	Сгенерированы худшие данные	
2.3	Сгенерированы случайные данные	
3.1	Подготовлено окружение для корректного, с учётом погрешности, измерения времени работы алгоритма	
3.2	Для каждого измерения выполняется несколько холостых запусков	
3.3	Написаны автоскрипты запуска тестирования	
4.1	Запущено тестирование	
5.1	В отчёте на графиках экспериментальные точки не соединены линиями	
5.2	На экспериментальные точки нанесены доверительные интервалы	
5.3	Выполнена аппроксимация экспериментальных результатов функциями	
6.1	В отчёт включена постановка задачи	
6.2	В отчёт включены параметры вычислительного узла	
6.3	В отчёт включены все результаты измерений	
6.4	В отчёте есть обоснование экспериментальных результатов	

6.5	Приложение к отчёту содержит готовую среду для воспроизведения тестов	