## Входные данные и вывод алгоритмов

Оба алгоритма получают на вход одинаковые данные, и на выход выдают также одинаковый результат.

Входные данные в алгоритмы:

* Строка test, в которой мы будем переставлять сегменты. Для работы с строками, будет использоваться библиотека string. Говоря об элементах строки, начинаем их нумерацию с 0.
* Целые числа m, n, p, такие что 0 <= m < n < p. Эти числа обозначают срезы строки test[m, n] и test[n+1,p], которые мы будем менять местами

Вывод алгоритмов:

* Строка, которая идентична строке test, за исключением того, что подстроки test[m, n] и test[n+1,p] поменяны местами.

## Организация данных о тесте

Данные о тесте, который будут использовать два алгоритма храняться в файлах. Так оба алгоритма смогут работать на одних данных, и данные для тестов можно будет сохранять между запусками программы. Размер теста лежит отдельно, чтобы было удобно генерировать разные тесты одной длины не вводя каждый раз длину заново, а также для того, чтобы не приходилось каждый раз для функций, где нужна только длина теста, считывать весь тест и заново просчитывать длину.

Чтение файлом и запись информации в них будет осуществляться при помощи библиотеки fstream.

| Имя файла | Что в нем храниться | Функции для работы с файлом |
| --- | --- | --- |
| len\_test.txt | Тут храниться число, обозначающие длину тестовой строки | set\_len\_test() - запись  get\_len\_test() - чтение |
| test.txt | Тут храниться тестовая строка | set\_test() - запись  get\_test() - чтение  generate()\_random\_test |
| parameters.txt | Тут хранится 3 целых положительных числа через пробел, обозначающие границы сегментов, которые алгоритмы будут переставлять | set\_parameters() - запись  get\_parameters() - чтение  generate\_random\_parameters() |

### Описание функции generate\_random\_test

Первым шагом функция берет требуемый размер теста, используя функцию get\_len\_test. После инициализируется пустая string переменная test. Затем генерируем с помощью функции rand() число от 33 до 126 включительно, это число - ascii код символа, который мы добавим к test в конец. Будем добавлять символы в test, пока он не достигнет требуемой длины. Так мы получим рандомную тестовую строку для дальнейших тестов программы. В конце функции полученный тест записывается в файл test.txt с помощью функции set\_test().

Ascii коды для добавляемых символов мы берем в диапазоне от 33 до 126, так как именно в этом промежутке находятся латиница и арифметические и пунктуационные знаки, наподобие “+” или “!”. К примеру, на диапазоне от 0 до 31 включительно лежат специальные символы, вроде “☺ ”, которые не желательно использовать в тесте, так как не все компьютеры его правильно интерпретируют.

## Описание функции generate\_random\_parameters

Первым шагом функция берет требуемый размер теста, используя функцию get\_len\_test(). Далее инициализируем три переменные типа int: m, n, p. С помощью функции rand() генерируем три числа, такие что:

* m >= 0 и m < длины тестовой строки - 2 (это нужно, чтобы для переменных n и p, которые должны быть больше m, осталось место);
* n > m и n < длины тестовой строки - 1 (это нужно, чтобы для переменной p, которая должны быть больше n, осталось место);
* p > n и p < длина тестовой строки.

Далее полученные значения записываются в файл parameters.txt при помощи функции set\_parameters().

## Функция, которая меняет местами фрагменты одной длины

Данная функция названа swap\_eq.

Алгоритм последовательным перебором потребует замену местами срезов одинаковой длины в трех местах. Поэтому данную функцию лучше вынести в отдельную, чтобы не пришлось каждый раз перебирать элементы по-очереди в цикле.

На вход функции будут подаваться указатели на начало первого сегмента и начало второго сегмента, а также длина обоих сегментов.

Далее в цикле с помощью функции swap по-очереди элементы обоих сегментов будут меняться местами. То есть, если i - это целое число, i ∈ [0, длина сегментов-1], то для всех i поменяем местами элементы таким образом: swap(\*(a+i), \*(b+i))

Таким образом мы поменяем равные по длине подстроки, не передавая в саму функцию всю строку, а только указатели на некоторые ее элементы и длину изменяемых подстрок.

## Функция, которая замеряет время выполнения алгоритма

Данная функция будет называться measure.

Замерять время программа будет с помощью библиотеки ctime и функции clock() из нее, которая позволяет находить текущее время в тактах процессора.

Сначала программа инициализирует переменную unsigned int start\_time = текущему времени. Затем выполняется алгоритм, время которого мы замеряем. А после этого программа инициализирует еще одну переменную unsigned int end\_time = новому текущему времени. Если длина тестовой строки не слишком большая для вывода (<= 1000 символов), то программа также выводит результат перестановки, полученный алгоритмом. Независимо от этого, в конце программа выведет затраченное алгоритмом время, которое равно end\_time - start\_time.

## Функция hello()

Функция hello() нужна только для вывода информации пользователю об устройстве функции main и тому, как с ней работать.

## Устройство функции main

В функции main сначала запускается функция hello() и выводит вспомогательную информацию. Затем создается переменная int swt, через нее пользователь будет выбирать, какой сценарий программы хочет запустить. Далее идет цикл, который не завершиться, пока swt не станет равно 6, то есть пользователь не решит завершить программу. В начале цикла у пользователя запрашивается число, к которому приравнивается swt. Далее в цикле находится конструкция switch, который выбирает, что будет делать программа дальше, в зависимости от значения swt. Сценарии могут быть такими:

| Значение swt | Краткое описание | Сценарий программы |
| --- | --- | --- |
| 0 | Сгенерировать тест | Программы запрашивает у пользователя размер для теста и записывает его в файл len\_test.txt, используя функцию set\_len\_test(). Дальше генерирует тест, используя функцию generate\_random\_test(). Если длина теста меньше или равна 1000, то выводит полученный тест. Иначе просто выводит сообщение о том, что тест был сгенерирован. |
| 1 | Сгенерировать срезы | Программа генерирует границы срезов, используя функцию generate\_random\_parameters(). После выводит полученные индексы границ срезов. |
| 2 | Ввести свой тест | Программа запрашивает у пользователя строку. Далее записывает размер этой строки в len\_test.txt и сам тест в test.txt с помощью функций set\_len\_test() и set\_test() соответственно. |
| 3 | Ввести свои срезы | Инициирует три переменные типа int: m, n и p. Затем по очереди запрашивает у пользователя их значения и проверяет их корректность. В конце, если все проверки были пройдены, записывает m, n и p в файл parameters.txt, используя функцию set\_parameters() |
| 4 | Получить данные о тесте | Получает из файлов информацию о тесте, используя функции get\_len\_test(), get\_test() и get\_parameteres(), а затем выводит пользователю. Если get\_len\_test() выдал число большее 1000, то функция get\_test() не будет использоваться и сам тест показан не будет |
| 5 | Провести замеры | Измеряет сначала время выполнения для первого алгоритма, а потом для второго, используя функцию measure() |
| 6 | Завершить программу | Позволяет выйти из цикла и завершить программу |
| Другое |  | Вызывает функцию hello() и выводит список корректных значений swt и сценариев того, что программа будет делать при каждом из них |

## 

## Замеры времени выполнения алгоритмов

Время измерено в тактах процессора.

| Длина строки | Номер попытки | Алгоритм последовательным перебором | Среднее | Алгоритм на основе двойной инверсии | Среднее |
| --- | --- | --- | --- | --- | --- |
| 100 | 1 | 44 | 51,2 | 10 | 19 |
| 2 | 50 | 36 |
| 3 | 57 | 19 |
| 4 | 50 | 12 |
| 5 | 55 | 18 |
| 300000 | 1 | 2067 | 2291,8 | 2622 | 3322,8 |
| 2 | 2382 | 3729 |
| 3 | 2090 | 3472 |
| 4 | 3043 | 3949 |
| 5 | 1877 | 2842 |
| 900000 | 1 | 7796 | 7488,4 | 13525 | 13969,8 |
| 2 | 6897 | 13112 |
| 3 | 9542 | 19142 |
| 4 | 6615 | 12028 |
| 5 | 6592 | 12042 |

## Ссылка на код проекта

<https://github.com/DillFennel/Switch-of-slices.-Test-of-computing-power>