**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Российский химико-технологический университет имени Д.И. Менделеева»**

**Кафедра информационных компьютерных технологий**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1**

Выполнил студент группы КС-36 Перминова П.А.

Приняли: Пысин Максим Дмитриевич

Краснов Дмитрий Олегович

Дата сдачи:

**Оглавление**

[**Описание задачи**](#_5wyzu7t61d7b) **3**

[**Описание метода/модели**](#_k3f55hfgbc2p) **4**

[**Выполнение задачи**](#_45uzn5swogwx) **5**

[**Заключение**](#_irj83k67q3up) **6**

# Описание задачи

В рамках лабораторной работы необходимо изучить и реализовать метод сортировки перемешиванием.

Для реализованного метода сортировки необходимо провести серию тестов для всех значений N из списка (1000, 2000, 4000, 8000, 16000, 32000, 64000, 128000), при этом:

* в каждом тесте необходимо по 20 раз генерировать вектор, состоящий из N элементов
* каждый элемент массива заполняется случайным числом с плавающей запятой от -1 до 1.
* каждый массив после генерации необходимо отсортировать и замерить время, требуемое на сортировку
* Результат замера для каждой попытки каждого теста записать в файл общий файл.

По окончанию всех тестов необходимо нанести все точки, полученные в результате замеров времени на график где на ось абсцисс(Х) нанести N, а на ось ординат(Y) нанести значения времени на сортировку. По полученным точкам построить график лучшего (минимальное время для каждого N), худшего (максимальное время для каждого N) и среднего (среднее время для каждого N) случая.

# Описание метода/модели

Тип сортировки перемешиванием представляет собой улучшенный вариант пузырькового способа упорядочивания данных. Он является двунаправленным и потому еще называется метод сортировки коктейлем. Принцип данного способа состоит в том, что мы делаем проход сначала в одну сторону массива, сравнивая и сортируя по очереди соседние два числа массива. Когда мы дошли до конца массива, наиболее крупное число оказывается в правой его части.

В пузырьковом методе мы возвращались в начало массива и повторяли все заново. В этом методе мы не возвращаемся, а идем влево, попарно сравнивая соседние элементы. Результатом прохода в левую сторону станет минимальное число, которое окажется в начале массива. Затем мы снова начинаем движение вправо, сортируя и «выталкивая» предпоследнее число результирующего массива. Дойдя до конца, снова меняем направление сортировки и продолжаем выполнять действия до тех пор, пока массив не будет обработан.

# Выполнение задачи

Программа алгоритма написана на языке Python.

В ходе выполнения программы выводятся для каждого количества элементов (N = 1000, 2000, 4000, 8000, 16000, 32000, 64000, 128000) минимальное, максимальное и среднее затраченное на сортировку время из 20 проведенных тестов.

В программе используются функции, которые:   
1) генерируют массив из передаваемого в функцию количества элементов N

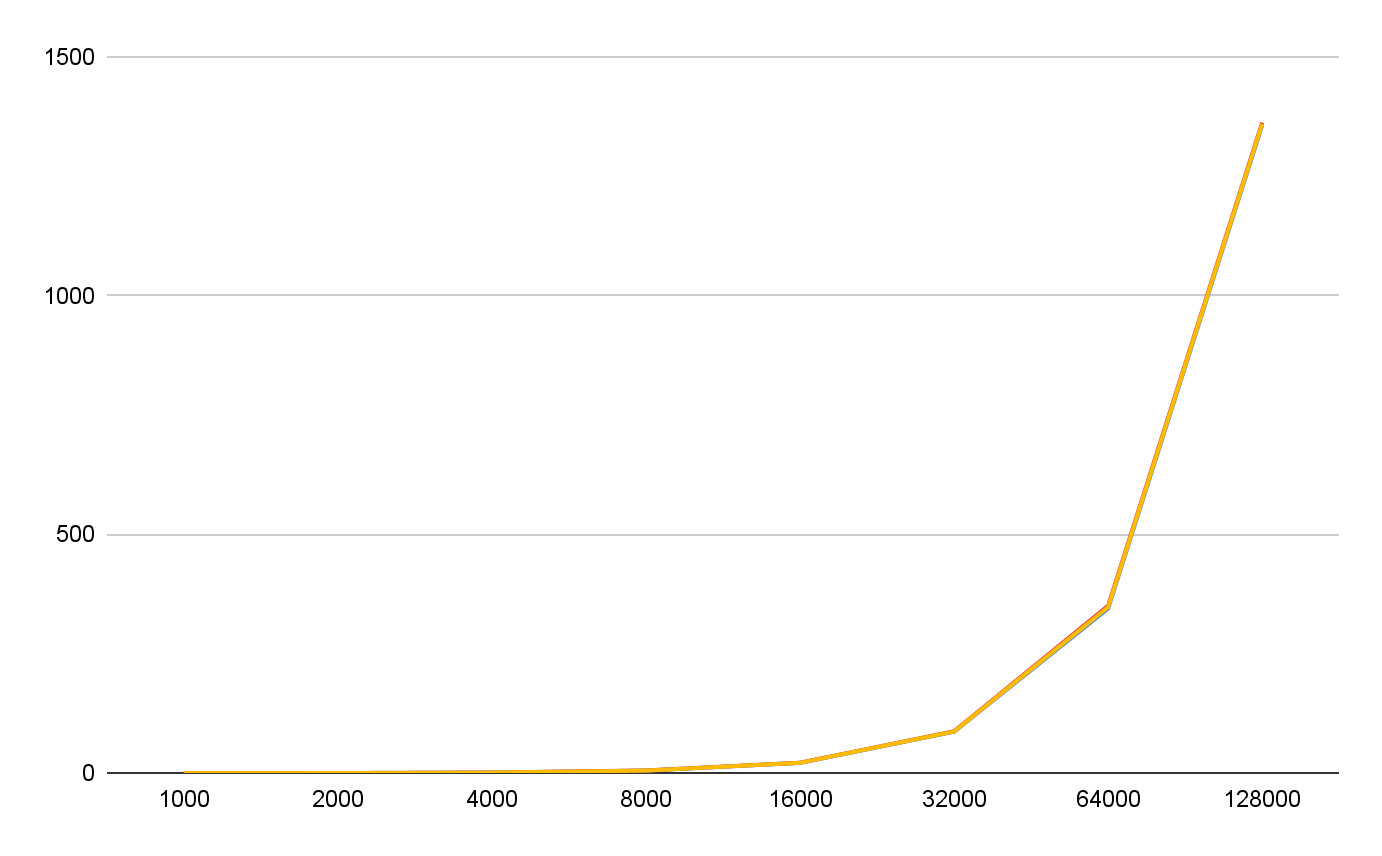
2) функция сортировки массива методом перемешивания

3) главная функция main, в которой вызываются функции создания и сортировки массива и выводится затраченное на сортировку время.

Вывод программы:

|  |  |  |  |
| --- | --- | --- | --- |
| количество элементов | min | max | среднее |
| 1000 | 0,06882333755 | 0,09518241882 | 0,08359624147 |
| 2000 | 0,3281495571 | 0,3519668579 | 0,3409191251 |
| 4000 | 1,330279589 | 1,38166666 | 1,362456846 |
| 8000 | 5,378294706 | 5,788332701 | 5,495492005 |
| 16000 | 21,59679651 | 21,95827913 | 21,75093305 |
| 32000 | 86,76159215 | 87,87134647 | 87,18271818 |
| 64000 | 344,6793313 | 350,7518556 | 347,6773887 |
| 128000 | 1356,564094 | 1362,352426 | 1359,45826 |

График:



# Заключение

Методы пузырьковой сортировки считаются простейшими, однако, время выполнения при большом количестве элементов время, затраченное на нее, в реалиях использования языка Python не считаю оптимальным. Скорость работы сортировки увеличивается из-за того, что в процессе сортировки программа каждый раз пробегает по всему массиву и делает большое количество сравнений. Данный метод сортировки лучше использовать на небольших количествах элементов.