**Лабораторная работа по теме «Генетические алгоритмы и информационная безопасность»**

**Цель лабораторно работы**: научиться применять механизмы генетических алгоритмов на практике, реализовать программу, позволяющую обратимо скрыть содержимое изображения с помощью механизмов скрещивания и мутации.

**Требования**

* ОС Windows (либо ОС Linux) с установленным Python-интерпретатором и IDE (IDE опциональна, если вы собираетесь использовать Google Colab или Jupyter Notebook);
* Предварительно загруженное тестовое изображение «Лена»;
* Подключение к сети Интернет;
* Библиотеки для Python: numpy, PIL и matplotlib.

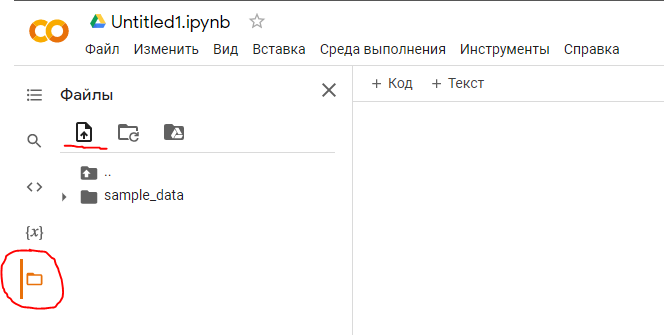
**Постановка задачи**

Требуется изучить и реализовать алгоритм, описанный в статье <https://docsdrive.com/pdfs/ansinet/itj/2006/516-519.pdf>. Предполагается использование языка программирования Python для решения поставленной задачи (возможно использование и других языков).

**Рекомендации по использованию инструментов для решения задачи**

Основное предложение – использовать Google Colab.

Необходимо загрузить тестовое изображение «Лена» (поставляется вместе с лабораторной работой) в сессионное хранилище, как показано на рисунке ниже.



Для простоты и скорости вычислений используется квадратное черно-белое изображение 128 на 128 пикселей. Усложнение работы возможно за счет обработки цветных изображений с различным соотношением сторон. В качестве различных вариантов работы студентам могут быть предложены различные изображения.

Для наглядности изображение предлагается делить на вектора, состоящие из 128 байт, то есть фактически из матрицы изображения требуется взять каждую ее строку. Это и будут искомые вектора. Обозначим множество эти векторов . Это множество содержит векторов длины (в данном случае совпадает с размером изображения).

Формулы для вычисления чисел R1 и R2 приведены ниже (в статье некоторые символы не пропечатаны).

В случае цветного изображения числа R1 и R2 нужно вычислить для каждого слоя. Операции скрытия также выполняются для каждого слоя отдельно. Константа 256 используется, так как значение пикселя в монохромном изображении лежит в диапазоне [0, 255].

Для каждого вектора вычисляются значения CrossoverIndex, CrossoverIteration, MutationIndex, MutationIteration. CrossoverIteration и MutationIteration определяют какое количество необходимо выполнить операции скрещивания и мутации для соответствующих векторов. CrossoverIndex и MutationIndex определяют начальное состояние генератора псевдослучайных чисел для соответствующих операций мутации и скрещивания.

Рассмотрим алгоритм для вычисления этих значений. Введем переменные и . Тогда для каждого сформированного ранее вектора :

Для каждого вектора значения и инкрементируеются: и . Если в какой-то момент или превосходят , то их значения одновременно обнуляются.

Перед этапом программирования нужно реализовать блок схему, описывающую алгоритм. Ее примерный и упрощенный вид для процедуры сокрытия изображения представлен ниже.



Рассмотрим, как выполняется скрещивание. Для каждого вектора генератор псевдослучайных чисел инициализируется значением CrossoverIndex. Затем операция перемены мест выполняется CrossoverIteration раз, где и сгенерированные случайные числа.

Операция мутации выполняется аналогичным образом. Для каждого вектора генератор псевдослучайных чисел инициализируется значением MutationIndex. Затем операция инвертирования пикселя выполняется MutationIteration раз, где сгенерированное случайное число.

Операция восстановления изображения выполняется в обратном порядке. Важно учесть, что при восстановлении требуется сначала выполнить операцию мутации, а затем уже скрещивания.

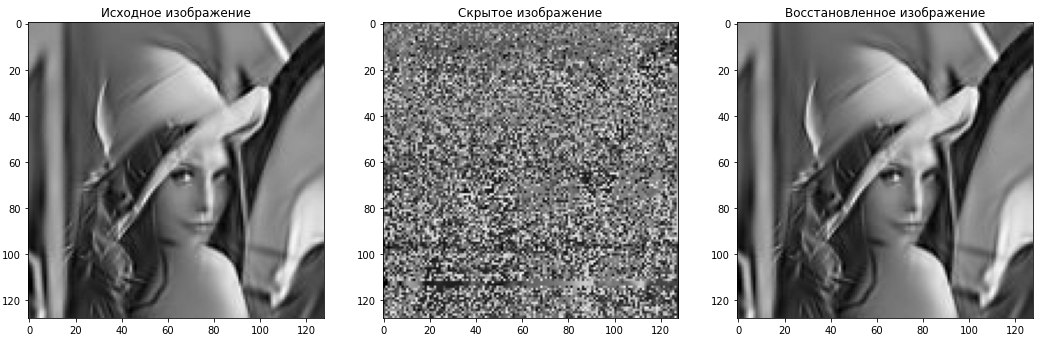
При этом операция скрещивания при восстановлении изображения отличается от операции скрещивания в случае сокрытия. Перестановки для каждого вектора нужно выполнять в обратном порядке.

В рамках лабораторной работы допустимо не добавлять числа R1 и R2, необходимые для декодирования, к скрытому изображению.

В качестве генератора псевдослучайных числе предлагается использовать numpy.random.

**Ожидаемый результат**

1. Алгоритм позволяет скрыть содержимое изображения, работает корректно и без ошибок.
2. Реализована возможность восстановления изображения без появления дополнительных «артефактов» (размытости, шумов и т.д.)



Примерный минимальный код для решения лабораторной работы приведен в Jupyter Notebook, поставляемым вместе с данным руководством.

**Рекомендованная литература**

1. Mohammed A.F. Al- Husainy. Image Encryption Using Genetic Algorithm. <https://docsdrive.com/pdfs/ansinet/itj/2006/516-519.pdf>
2. Материалы лекции по генетическим алгоритмам.
3. Справочник по командам Python.