Higher National School of Mathematics

List Of Projects
(Classical cryptography)

Team Projects

Project 1

Sliding Image Puzzle

You have been assigned to develop a program that will aid a **sliding image puzzle** manufacturer in partitioning images for their puzzles.

Figure 1: Sliding Puzzle

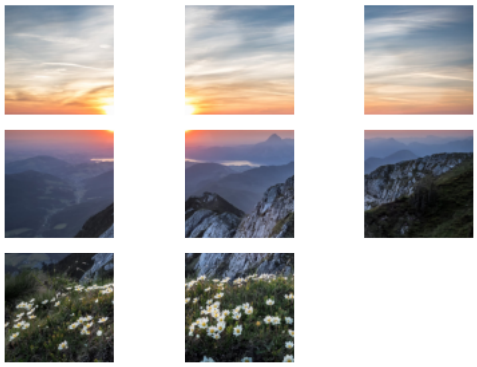
The program should be designed to accomplish the following tasks:

- **Task 1** Read `images.txt`, which contains the links to the images.

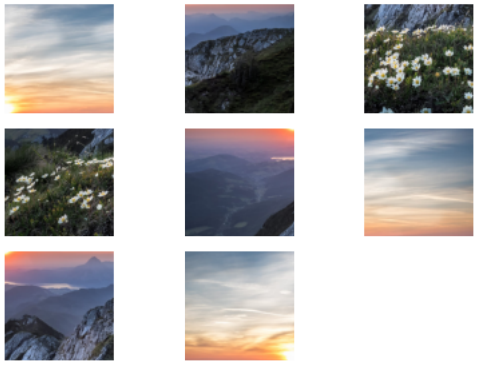
Click here to get images.txt

- **Task 2** For images that are not square-shaped, crop them to obtain the maximum-sized square image possible located in the middle of the original image.

- **Task 3** Partition each image into eight equal parts, as illustrated in the the follwing figure.



- **Task 4** Generate a permuted grid of the eight small parts of the cropped image.



- **Task 5** Store each cropped image and its corresponding eight small parts in a directory named `puzzle_i`, where `i` represents the image's order in the file.

Your program should be programmed to perform these tasks effectively and efficiently. It should be thoroughly documented, and clear instructions should be provided to your users.

Project 2

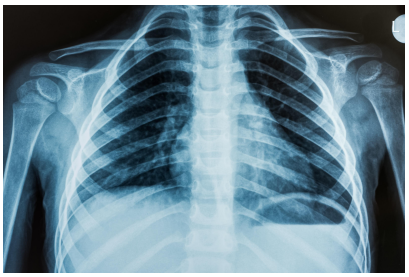
Medical Image Encryption and Decryption using XOR and PRNGs

A clinic has to send medical images to another institution, but they are concerned about the safety of the images in transit. They have requested your help to develop a solution to secure the images so that they cannot be intercepted or accessed by unauthorized individuals.

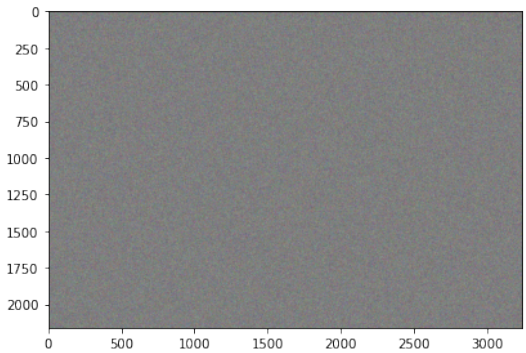
Project Requirements Your task is to create a program that will encrypt the medical images using XOR and pseudo-random number generator (PRNG) and then save the encrypted images to a safe location. You will then decrypt the encrypted images using the same function to ensure that the original images are restored.

To achieve this, you will need to complete the following steps:

- **Task 1** Read the list of medical images from the given directory. Download the set of images from here



- **Task 2** Implement with python three PRNGs using any of the PRNG algorithms available (e.g., Lehmer generator). List of prngs
- **Task 3** For each medical image, encrypt the image using XOR with a list of random numbers generated with the three PRNGs: xor each byte with a byte from the prng output.



- **Task 4** Save the encrypted image in a folder named `encrypted_data`.
- **Task 5** For each encrypted image, decrypt the image using the same XOR function and PRNGs (use the same keys as encryption step).
- **Task 6** Compare the decrypted image with the original image to ensure that the image was decrypted successfully.
- **Task 7** Write a report on the project, including details on the implementation, the results, and any challenges encountered.

Project 3

The 0-1 Knapsack problem

The 0-1 Knapsack problem is a classic optimization problem in computer science and mathematics. The problem is defined as follows:

Given a set of items, each with a weight and a value, and a knapsack with a maximum weight capacity, determine the maximum value that can be put into the knapsack without exceeding its weight capacity.

In the 0-1 version of the problem, each item can only be taken once (either included in the knapsack or not), hence the name (0-1). The goal is to maximize the total value of the items in the knapsack while keeping the weight of the knapsack within the maximum capacity.

The 0-1 Knapsack problem has important applications in various fields, such as resource allocation, financial portfolio optimization, and logistics planning.

- **Task 1**: Problem Description Write a brief description of the 0-1 Knapsack problem, including its goal, and how it can be represented as a graph.
- **Task 2**: Research the dynamic programming Method Research the dynamic programming Method and write a brief explanation of how it works, including the steps involved in solving the 0-1 Knapsack problem.
- **Task 3**: Implement the dynamic programming method Implement the dynamic programming in Python, using NumPy to perform the necessary calculations.
- **Task 4**: Test the Implementation Test the implementation using a small example problem and verify that it produces the correct solution.
- **Task 5**: Write a Report Write a report summarizing the project, including the problem description, the solution approach, the implementation details, and the test results.

References:

1. https://en.wikipedia.org/wiki/Knapsack_problem

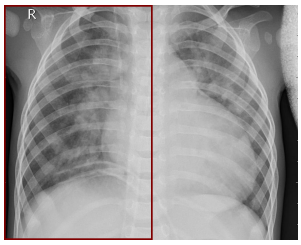
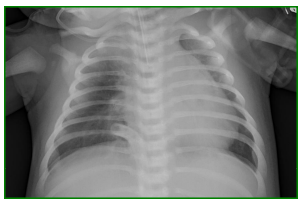
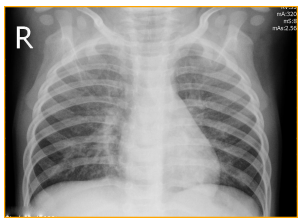
Project 4

COVID-19 Detector

A hospital has reached out to us for help in classifying a large dataset of X-ray images to determine whether a person has COVID-19 or not. The hospital has been overwhelmed with patients and needs an automated system to assist their medical staff in diagnosing COVID-19 accurately and efficiently.

The project will consist of the following tasks:

- **Task 1**: Read the gray images in the data set image by image, Download the data set from here.
- **Task 2**: Calculate the average of pixels having a value ≥ 128 (near to white color) for each image
- **Task 3**: Classify the image based on the calculated average value:
  - if the average ≤ 50, the person is negative for Covid
  - if the average > 50 and ≤ 60, the person is suspected to have Covid
  - if the average > 60, the person has Covid
- **Task 4**: Process the images depending on their Covid status:
  - Convert the gray images to RGB format
  - For negative Covid images, add a green border to the image
  - For suspected Covid images, add an orange border to the image
  - For confirmed Covid images:
    - Split the image vertically into two parts
    - Calculate the average pixel value near to white for each part
    - Determine which part of the image has a higher average value
    - Add a red border to the original image on the side of the sick lung
- **Task 5**: Save the final result image to a folder named `results`

You can further break down each task into subtasks and write code to accomplish each subtask.

Project 5

Solving the Assignment Problem using the Hungarian Method

The assignment problem is a classic optimization problem in which a set of agents must be assigned to a set of tasks, with each agent being assigned to exactly one task and each task being assigned to exactly one agent. The goal is to minimize the total cost or time required for the assignments, which can be represented as a square cost matrix, where the element in row i and column j represents the cost of assigning agent i to task j.

- **Task 1**: Problem Description Write a brief description of the Assignment Problem, including its goal, and how it can be represented as a square cost matrix.
- **Task 2**: Research the Hungarian Method Research the Hungarian Method and write a brief explanation of how it works, including the steps involved in solving the Assignment Problem.
- **Task 3**: Implement the Hungarian Method Implement the Hungarian Method in Python, using NumPy to represent the cost matrix and perform the necessary calculations.
- **Task 4**: Test the Implementation Test the implementation using a small example problem and verify that it produces the correct solution.
- **Task 5**: Write a Report Write a report summarizing the project, including the problem description, the solution approach using the Hungarian Method, the implementation details, and the test results.

References:

1. http://biblio.univ-antananarivo.mg/pdfs/randrianatoandroFenoarisoaT_MP_MAST_21.pdf
2. https://en.wikipedia.org/wiki/Hungarian_algorithm

Project 6

Random generators' tests

Find the project details here: https://ydjemmada.github.io/projects/Project_6.pdf

Project 7

Will be here soon

Individual Projects

You should choose one of the project:

- **RSA encryption scheme**,
- **3DES encryption scheme**,
- **EL-GAMAL encryption scheme**,
- **Diffie–Hellman key exchange.**

or 4 project of the 6th teams' project or join a team.