Riddles Documentation

Dell Technologies

March 2024

Contents

List of Figures

1 Introduction

The following is a detailed description of all the riddles that the fox can attempt.
Please note that you should ensure that the answer you provide is of the **correct data type** as specified for each riddle.
Also, you need to be aware that each riddle has a different **timeout** period that if exceeded, even if your answer is correct, you will not get awarded with any fake messages.

2 Security

2.1 Security - Medium

- **Riddle Id:**
  sec_medium_stegano

- **Problem Statement:**
  Legends speak of a hidden message encoded within the fabric of reality itself, waiting to be unveiled by those who possess the knowledge of SteganoGAN, a magical tool that merges artistry with encryption

  Armed with your skills, and a keen eye for hidden patterns, you must decode the message woven into images defy conventional perception. Each layer peeled back reveals a fragment of the encrypted truth, bringing you closer to qualifying to the next phase of the hackathon and hopefully winning

  Your task is: given an image containing a hidden message, your task is to uncover this secret message. But remember, things are not always as they seem. The key to solving this riddle lies in the hidden layers of the image, where noise is not just noise.

  Here's a hint: The tool you need to decode this mystery is closer than you think. It's a package developed by us, used for creating steganographic images using adversarial training.

- **Input Description:**
  Your input consists of a single image.

- **Expected Output:**
  The secret message hidden in the image in the form of a **string**.

- **Acceptance Criteria:**
  Exact matching between your output and the truth value of the hidden message.

- **Riddle Award:**
  A budget of **2** fake messages.

2.2 Security - Hard

- **Riddle Id:**
  sec_hard

- **Problem Statement:**
  In the kingdom of Cypheria, a message of great importance must be encoded to safeguard it from prying eyes. The message, containing the location of an ancient treasure, must be encrypted using the Data Encryption Standard (DES) algorithm. Your task is: given a pair of key and plain text, you should encrypt the plain text using the DES algorithm.

- **Input Description:**
  Your input consists of two strings in that order:
  - Key: represented in hexadecimal format in a string of length 16 characters
  - Plain text: represented in hexadecimal format in a string of length 16 characters

  All hexadecimal letters (A-F) are written in capital letters.

- **Expected Output:**
  Cipher text in the form of a **string** of length 16, and all the letters should be capital.

- **Example:**
  Input: ("266200199BBCDFF1","0123456789ABCDEF")

  Your **output** should be : "4E0E6864B5E1CA52"

- **Acceptance Criteria:**
  Exact match between your output and the truth value. Please note that you are NOT allowed to use a python library to solve this, your implementation must be from scratch.

- **Riddle Award:**
  A budget of **3** fake messages.

3 Computer Vision

3.1 Computer Vision - Easy

- **Riddle Id:**
  cv_easy

Figure 1: Input example.

Figure 2: Expected output example.

- **Problem Statement:**
  A very well-known phrase used by schoolchildren to explain their failure to turn in an assignment is "the dog ate my homework". But unfortunately, your dog really did shred an important image that you need to turn in, and the dog shredded that image vertically.

  Luckily, you were holding the first shred, so you know the first left most shred is in place. Also, you were extremely lucky to have the shreds of equal width which is **64** pixels.

  Your task is: you should write an algorithm to reassemble the paper and solve this puzzle.

- **Input Description:**
  Your input consists of a NumPy array representing a shredded image (see example for more clarity).

- **Expected Output:**
  A **list of integers** representing the shreds' indices after ordering them (zero-based).

- **Example:**
  Taking the image in figure 1, as input, the output should be this: [0, 11, 7, 1, 8, 9, 3, 5, 6, 4, 10, 2] so that if you followed this array's order you can reassemble the image back. In this case, you should put the 0 shred in first place, the 11th shred in second place, the 7th shred in the third place and so on. Beware the shreds count start from 0 not 1. If you did do you should find the puzzle solved as shown in figure 2.

- **Acceptance Criteria:**
  You will only be rewarded if you are able to give the correct order of the shreds.

- **Riddle Award:**
  A budget of **1** fake message.

3.2 Computer Vision - Medium

- **Riddle Id:**
  cv_medium

- **Problem Statement:**
  Having an image with high quality is important for multiple reasons, including better analysis and interpretation, effective image processing, improved machine learning models, and accurate diagnosis and decision making.

  Some images may contain patches, and the removal of patches is important when they contain unwanted defects or noise that can affect the analysis or visual quality of an image.

  Your task is: given a patched image, you should develop an algorithm that can identify and remove a small, patched image from a larger image, and then interpolate the missing pixels.

- **Input Description:**
  The input consists of two images: - A large image (referred to as the 'base image') in the form of - A small image (referred to as the 'patch') in the form of

  The patch is an image that has been superimposed onto the base image at an arbitrary location, size. The orientation of the patch is multiples of 90 degrees (0, 90, 180, 270 degrees).

- **Expected Output:**
  The output should be the base image with the patch removed and the resulting gap filled in, and the size of the resulting image must be the same as the large image in the form of a **list**.

- **Example:**
  The following is a visualization of an example input and output.

Figure 3: Base image with patch example.

- **Evaluation:**
  The algorithm's performance can be evaluated based on its ability to accurately identify and remove the patch, as well as the quality of the interpolated pixels. This could be quantified using a suitable image similarity metric, such as Structural Similarity Index (SSIM), when comparing the output image to the original base image (before

Figure 4: Patch example.

Figure 5: Expected output example.

the patch was added). It is worth noting that this riddle takes a longer time to run in comparison with the other riddles.

- **Acceptance Criteria:**
  The acceptance criteria could be that the SSIM between the output image and the original base image is above a certain threshold (SSIM greater than or equal to 85%). This threshold would represent the acceptable margin of error for the task.

- **Riddle Award:**
  You get 2 fake messages if you exceed the allowed SSIM.

3.3 Computer Vision - Hard

- **Riddle Id:**
  cv_hard

- **Problem Statement:**
  This CV riddle is about using AI to answer questions about a given image. You would be given a question and an image then you are asked to find the answer for the question from the provided image (Note you are bounded by a timeout so human-aided solutions are not possible).

- **Input Description:**
  You are provided with 2 inputs, the first being a **string** representing the question that needs to be answered and the second being an **RGB image object** that is loaded using **Pillow** library.

- **Expected Output:**
  An **integer** representing your answer to the question.

- **Example:**

For example, for the image below, figure 6, given that it is an image of **2 dogs** and **1 cat**, if the question is:
"How many dogs are in the image?", you would be expected to return the integer number **2**.

Figure 6: Example of input image for the hard computer vision riddle.

- **Acceptance Criteria:**
  The returned **integer** is equal to the correct answer.

- **Riddle Award:**
  A budget of **3** fake messages.

4 Machine Learning

4.1 Machine Learning - Easy

- **Riddle Id:**
  ml_easy

- **Problem Statement:**

  Time series forecasting is the process of analyzing time-stamped data using statistical methods and modeling to predict future values and trends, aiding in strategic decision-making across various industries, it involves building models based on historical data to make informed predictions about future outcomes. One use of time series forecasting can be used to predict the number of security attacks on a specific platform.

  Your task is: developing a machine/deep learning model that can accurately forecast the number of attacks on a specific website based on historical time series data that is given in the received materials. This involves preprocessing the data and designing a solution that can handle the inherent unpredictability and seasonality of attacks.

Given a dataset of attacks data with daily intervals for 500 days, the task is to forecast the attacks for the next 50 days.

- **Input Description:**
  The input data is a time series dataset of security attacks. Each data point represents the daily number of attacks for over a year. The dataset may contain trends and patterns that are typical, such as daily and weekly cycles, sudden spikes, and possibly missing or outlier data points.

- **Expected Output:**
  The output should be a forecast of security attacks for a future time period, as a **list of floats**. This forecast should be in the same format as the input data (i.e., the number of attacks for each time interval).

- **Evaluation:**
  The model's performance will be evaluated using the Root Mean Square Error (RMSE), which measures the average magnitude of the errors in a set of forecasts, considering their direction. It's expressed in the same units as the original data, so lower values are better.

- **Acceptance Criteria:**
  The model will be considered acceptable if the RMSE is less than or equal to 35.

- **Riddle Award:**
  A budget of **1** fake message.

4.2 Machine Learning - Medium

- **Riddle Id:**
  ml_medium

- **Problem Statement:**
  In machine learning, connected components refer to sets of vertices in a graph that are connected to each other through edges but not connected to vertices outside the group. They are used to identify clusters or sub-graphs of nodes that are interconnected within a graph. In the field of network security, identifying connected components can help in understanding the structure of a network, detecting isolated or vulnerable components.

  Your task is: detecting connected components using machine learning algorithms, you have a complex connected shape surrounded by groups of points. Machine Learning algorithms such as classification, clustering is helpful to map points based on their features to either belonging to the shape or not. You need to use your skills to map the point to the correct group. Your training data can be found in the received materials,

and consists of two features representing the x and y coordinates, and the label that has values either 0 for connected shape, and -1 for a point that doesn't belong to the shape.

- **Input Description:**
  The input is a NumPy array for the (x, y) coordinates of the point.

- **Expected Output:**
  The expected output is an **integer** either 0 for the shape or -1 for any point that doesn't belong to the connected component.

- **Example:**
  Input = (0,0)

  Predicted label = -1

- **Evaluation:**
  We will compare your classified label with the actual label from our end.

- **Acceptance Criteria:**
  You will be rewarded if you get the same group as expected, since this is a classification problem, there is no margin of error.

- **Riddle Award:**
  A budget of **2** fake messages.

5 Problem Solving

5.1 Problem Solving - Easy

- **Riddle Id:**
  problem_solving_easy

- **Problem Statement:**
  In the heart of Cairo, a young linguist named Fatima is working on a unique project. She has a vast collection of ancient Egyptian texts, and she wants to understand the most frequently used words in these texts.

  She has a **list** of **words** from these texts and a **number X**. Her task is to find the **X most recurring words** in the list. The **result** should be a **list** of these words, arranged in order of **decreasing** frequency. If there's a **tie** in frequency, the words should be arranged in **lexicographical** order.

  As Fatima delves into the task, she uncovers fascinating patterns and insights about the language and culture of ancient Egypt. You, as a fellow linguist, step in to assist

her. Your combined efforts not only enhance Fatima's understanding of the ancient texts but also contribute significantly to the field of linguistics. Your assistance proves invaluable in helping her navigate the complexities of the task, making the journey of discovery even more rewarding.

Notes:

- Number of words is in the range [1, 500].
- Length of each word in list is in the range [1, 10].
- All words are composed of lowercase letters from the English alphabet.
- X is in the range [1, The total count of **distinct** words[i]].

- **Input Description:**
  A **list of strings** and **an integer**.

- **Expected Output:**
  A **List of strings**.

- **Example:**
  Given the following input, the following output would be expected:

  Input: words=["pharaoh","sphinx","pharaoh","pharaoh","nile", "sphinx","pyramid","pharaoh", "sphinx","sphinx"], X=3

  Output: ["pharaoh","sphinx","nile"]

- **Acceptance Criteria:**
  The returned list of strings matches the correct answer (same elements and same order of appearance of elements)

- **Riddle Award:**
  A budget of **1** fake message.

5.2 Problem Solving - Medium

- **Riddle Id:**
  problem_solving_medium

- **Problem Statement:**

  In a world where communication is key, there exists a secret language known only to a select few. This language, used to encode messages, has a unique characteristic: it

uses a specific **rule** to transform ordinary strings into **encoded** sequences.

The rule is simple yet effective: for a given integer **'x'** and an encoded string **'w'**, the rule is **x[w]**. This means that **'w'** is duplicated **'x'** times, creating a sequence that is both intriguing and complex.

You, as a skilled cryptographer, have been tasked with a mission of utmost importance: to **decode** these encoded messages and reveal their true meaning. Your job is to take an **encoded string** and **return** it in its original, **decoded** form. With your expertise and knowledge, you are the only one capable of understanding the intricacies of this secret language and unveiling the hidden messages within.

So, brace yourself, for you are about to embark on a thrilling journey of codes, ciphers, and secrets. Good luck!

Notes:

- **x** is a **positive** integer.
- Encoded string length falls in the range of 1 to 30.
- Encoded string only contains English alphabet in **lowercase**, numerical **digits**, and **square bracket** symbols.
- Encoded string is a **legitimate** input.
- Every **integer** in the encoded string falls in the range of 1 to 300.
- Decoded string does not contain any digits.
- Decoded string's length is maximum 10^5.

- **Input Description:**
  An encoded **string**.

- **Expected Output:**
  The decoded **string**.

- **Example:**
  Given the following encoded string as input, the following output would be expected:
  Input: "3[d1[e2[l]]]"
  Output: "delldelldell"

- **Acceptance Criteria:**
  The returned string is an exact match of the correct answer

- **Riddle Award:**
  A budget of **2** fake messages.

5.3 Problem Solving - Hard

- **Riddle Id:**
  problem_solving_hard

- **Problem Statement:**

  In the heart of a bustling city, there lived a renowned detective known for his sharp intellect and keen observation skills. His world was a vast grid of dimensions **x by y**, with his office located at the **top-left** corner (**grid[0][0]**).

  One day, a mysterious letter arrived, challenging the detective to solve a complex case located at the **bottom-right** corner of the city (**grid[x-1][y-1]**). However, the detective had a peculiar habit - he would only move **south** or **east** while navigating the city streets.

  Your role in this intriguing tale is to calculate the number of **unique routes** the detective could take to reach the scene of the case, given the city's dimensions **x** and **y**. So, brace yourself and join the detective on this thrilling journey of mystery and discovery!

  Notes:

  - **x** is in the range [1, 100].
  - **y** is in the range [1, 100].
  - Number of unique paths will not exceed $2 * 10^9$.

- **Input Description:**
  2 **integers**, the first representing **x** and the second representing **y**.

- **Expected Output:**
  An **integer** representing the number of unique paths to destination.

- **Example:**
  Given the following input, that translates to figure 7, for the detective to reach "Hacktrick" in the bottom-right corner, the following output would be expected:

  Input: x = 3, y = 2.
  Output: 3.
  Explanation: Starting from the top-left corner, there exist 3 distinct paths to reach the bottom-right corner:

  1. Move East, then South twice.
  2. Move South twice, then East.

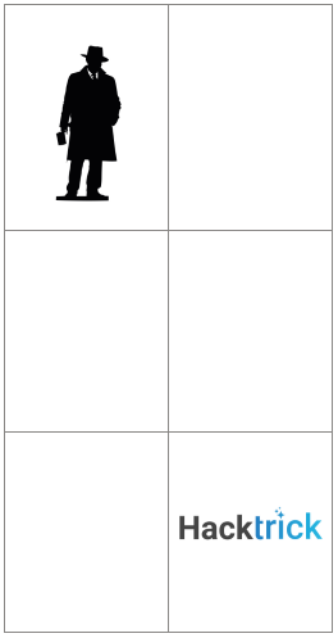

Figure 7: Example of a 3x2 grid.

3. Move South, then East, then South again.

- **Acceptance Criteria:**
  Returned integer matches the correct answer.

- **Riddle Award:**
  A budget of **3** fake messages.