Presentació

En aquesta pràctica ens familiaritzarem amb el llenguatge de programació Python tot implementant un criptosistema històric, concretament una variant de la xifra en ZigZag o *rail fence*, una xifra de transposició senzilla en què les lletres es reordenen seguint un patró similar a una tanca.

Objectius

Els objectius d'aquesta pràctica són:

1. Familiaritzar-se amb l'entorn de treball en Python i el *framework unittest*.
2. Implementar un criptosistema històric.

Descripció de la Pràctica

La xifra *rail fence* bàsica consisteix a reordenar les lletres del missatge en clar seguint un patró descrit per una tanca. El número de carrils o files de la tanca ve determinat per la clau del criptosistema, que serà compartida per l'emissor i el destinatari del missatge.

Així doncs, per tal de xifrar un missatge, l'escriurem sobre la tanca, situant una lletra a cada carril, dibuixant una forma d'ona o zigzag. Començarem a escriure a la fila superior, i seguirem escrivint una lletra a cada fila, fins arribar a la fila inferior. Quan arribem a la fila inferior, repetirem el procés en sentit invers, escrivint una lletra a cada fila fins arribar a la fila superior. Anirem repetint aquest procediment fins a escriure la totalitat del missatge en clar sobre la tanca.

Una vegada s'ha escrit el missatge a la tanca, procedirem a llegir el missatge per files, començant també per la fila superior i llegint totes les files, fins a arribar a la fila inferior. D'aquesta manera, s'obté el text xifrat.

Així, per exemple, si volem xifrar el missatge `CRYPTOGRAPHY` utilitzant la clau $k = 3$, procedirem a crear una tanca de tres carrils, i hi escriurem el missatge en zigzag, començant per la primera fila (fila superior) i dibuixant un zigzag.

```
C---T---A---
-R-P-O-R-P-Y
--Y---G---H-
```

Per obtenir el missatge xifrat, procedirem a llegir les lletres que hi ha a la tanca per carrils, començant per la primera fila, després la segona, etc. El missatge xifrat serà doncs `CTARPORPYYGH`.

Una vegada rebi el missatge xifrat, el receptor procedirà a realitzar el procés invers per tal d'obtenir el missatge en clar: situarà les lletres a la tanca per files i llegirà el missatge en zigzag. Noteu que per tal que el procés de desxifrat sigui satisfactori, el receptor haurà de saber quants carrils té la tanca.

En aquesta pràctica implementarem una petita variant d'aquest esquema de xifratge, en la qual la tanca podrà tenir forats. Els forats seran posicions on no s'hi podrà escriure cap lletra. A l'hora d'escriure el text sobre la tanca, si ens trobem amb algun forat, aleshores s'ometrà aquella posició, i la lletra passarà a escriure's en la propera posició disponible, seguint la mateixa forma de zigzag que descriu la variant bàsica.

En aquesta variant doncs, la clau de xifrat passarà a contenir tant el número de carrils de la tanca, com les posicions dels forats que té.

Vegem un exemple de xifrat amb la variant del *rail fence* que implementarem en aquesta pràctica. Suposem que volem enviar de nou el missatge `CRYPTOGRAPHY` utilitzant la clau $k = (3, [0, 4])$, és a dir, la clau serà ara una tanca de tres carrils també, però hi haurà un forat en el primer carril en la cinquena posició (noteu que comencem a contar amb índex 0):

```
C---*---R---Y
-R-P-T-G-A-H-
--Y---O---P--
```

Fixeu-vos com l'aparició del forat ha fet desplaçar les posicions de les lletres a partir de la `T`. El text xifrat serà ara `CRYRPTGAHYOP`, que correspon a la lectura per files del text que hi ha a la tanca. Noteu que pot ser que els forats es trobin també en posicions que no afectin l'escriptura del text sobre la tanca, com seria per exemple la posició $[1, 0]$

En aquesta pràctica implementarem la versió modificada del xifrat de *rail fence*. Per fer-ho, dividirem la feina en tres funcions:

1. la generació de la clau,
2. el xifrat d'un text en clar,
3. i el desxifrat d'un text xifrat.

Cadascun dels exercicis correspon a la programació d'una d'aquestes funcions. A continuació se'n descriuen els detalls.

1 Implementació de la xifra rail fence (10 punts)

1. Funció que implementa la generació de la clau de la variant de la xifra *rail fence*. (**3 punts**)

   La funció generarà una clau aleatòria de la variant de la xifra de *rail fence*, tenint en compte les restriccions especificades als paràmetres de la crida. La funció prendrà com a variables d'entrada tres paràmetres opcionals: `max_rails`, `num_holes` i `max_hole_pos`; i retornarà la clau generada.

- La variable `max_rails` contindrà el número màxim de carrils que podrà tenir la tanca de la clau a generar. El valor per omissió d'aquest paràmetre serà 10.
- La variable `num_holes` contindrà el número de forats de la tanca a generar. El valor per omissió d'aquest paràmetre serà 0.
- La variable `max_hole_pos` contindrà la posició horitzontal màxima dels forats de la tanca. El valor per omissió d'aquest paràmetre serà 100.
- La funció retornarà una tupla de dos elements amb la clau generada. El primer element serà un enter amb el número de carrils de la tanca. El segon element serà una llista de tants elements com forats hi hagi a la tanca (`num_holes`). Cada forat estarà especificat per una tupla, amb el primer element indicant el carril i el segon la posició horitzontal (columna) del forat.

Exemple:

- `max_rails`: `10`
- `num_holes`: `4`
- `max_hole_pos`: `12`
- `sortida`: `(3, [(1, 10), (0, 6), (0, 8), (2, 2)])`

2. Funció que implementa el xifrat amb la variant de la xifra de *rail fence*. (**3 punts**)

   La funció prendrà com a variables d'entrada dos paràmetres: `message` i `key`; i retornarà el missatge xifrat.

   - La variable `message` contindrà una cadena de caràcteres amb el text en clar a xifrar. La cadena de caràcteres podrà contenir lletres en majúscula i minúscula, espais i símbols bàsics de puntuació ('!','-', i '_').
   - La variable `key` contindrà la clau de xifrat, tal com la retorna la funció de generació de claus.
   - La funció retornarà una cadena de caràcters amb el text xifrat corresponent al text en clar.

   Exemple:

   - `key`: `(3, [(1, 10), (0, 6), (0, 8), (2, 2)])`
   - `message`: `CRYPTOGRAPHY`
   - `sortida`: `CPHRYTGRPYOA`

3. Funció que implementa el procés de desxifrat amb la variant de la xifra de *rail fence*. (**4 punts**)

   La funció prendrà com a variables d'entrada dos paràmetres: `ciphertext` i `key`; i retornarà el missatge en clar.

   - La variable `ciphertext` contindrà una cadena de caràcteres amb el text xifrat. La cadena de caràcteres podrà contenir lletres en majúscula i minúscula, espais i símbols bàsics de puntuació ('!','-', i '_').
   - La variable `key` contindrà la clau de xifrat, tal com la retorna la funció de generació de claus.

- La funció retornarà una cadena de caràcters amb el text en clar corresponent al text xifrat.

Exemple:

- `key`: `(3, [(1, 10), (0, 6), (0, 8), (2, 2)])`
- `ciphertext`: `CPHRYTGRPYOA`
- `sortida`: `CRYPTOGRAPHY`

Criteris d'avaluació

La puntuació de cada exercici es troba detallada a l'enunciat.

Format i data de lliurament

La data màxima de lliurament de la pràctica és el **2/10/2018** (a les 24 hores).

Juntament amb l'enunciat de la pràctica hi trobareu l'esquelet de la mateixa (fitxer amb extensió .py). Aquest fitxer conté les capçaleres de les funcions que cal que implementeu per a resoldre la pràctica. Aquest mateix fitxer és el que heu de lliurar un cop hi codifiqueu totes les funcions.

Addicionalment, també us proporcionarem un fitxer amb testos unitaris per a cadascuna de les funcions que cal que implementeu. Podeu fer servir aquests testos per comprovar que la vostra implementació gestiona correctament els casos principals, així com per obtenir més exemples concrets del que s'espera que retornin les funcions (més enllà dels que ja es proporcionen en aquest enunciat). Noteu, però, que els testos no són exhaustius (no es proven totes les entrades possibles de les funcions). Recordeu que no es pot modificar cap part del fitxer de testos de la pràctica.

El lliurament de la pràctica constarà d'un únic fitxer Python (extensió .py) on hagueu inclòs la vostra implementació.