PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE
INSTITUTO DE INGENIERIA MATEMATICA Y COMPUTACIONAL

Álgebra abstrácta y aplicaciones, Semestre II 2025
Tarea 1
Fecha de entrega: Domingo, Octubre 12, 2025 (23:59 hora Santiago de Chile)

1 RSA [3 puntos]

En este ejercicio tienen qué construir un script de Python que recibe cómo su input dos cosas:

- La llave pública del RSA en el formato (E, N)
- Un mensaje cifrado con esta llave pública.

Basado en esta información su script tiene qué recuperar el mensaje original. (Recureden que E, N y el mensaje son simplemente números.) El esqueleto de su srcript está en `rsa.py` y deben completar la funcción `crackRSA(E,N,ciphertext)`.

Para validar su implementación, el mensaje original para los siguientes parámetros:

- $(E, N) = (10025, 3000001358000041547)$
- `ciphertext` $= 17916678860318220$.

Es:

- `message` $= 2310$.

(En mi computador esto se demorá 5 minutos para ejecutar.)

Si necesitan un repaso de RSA, una buena fuente es:

- `https://marceloarenas.cl/iic3242-14/clases/comp-prop-imp.pdf`

Para testear su implementación pueden ocupar el ejemplo del libro del curso (ver ejemplo 7.5):

- `http://abstract.ups.edu/aata-es/section-public-key-crypt.html`

2 Erasure codes [3 puntos]

Para este ejercicio deben implementar el proceso de correción de errores para erasure codes explicados en clases. Recuerden que en un erasure code, tenemos un mensaje protegido con una cantidad de parity bits, dónde cada parity bit cubre ciertos conjuntos de bits del mensaje original. En un erasure code, los errores resultan en un bit desconocido, no en un flip entre 0 y 1.

Para este ejercicio se les entrega un archivo `erasure.py` con dos clases:

- `GenTests` con cual pueden generar cualquier cantidad de tests.
- `CodeSpec` cual especifica nuestro erasure code con todos sus parámetros y permite corregir los erasures.

Su tarea es completar el método `correctErasures` de la clase `CodeSpec`. Para especificar un código usamos los siguientes parámetros:

- `messageBits` – el número de bits del mensaje original
- `parityBits` – número de parity bits, cada uno protegiendo una parte del mensaje
- `paritySets` – un arreglo del largo `parityBits`, representando las posiciones protegidas por un parity bit específico.

Para dar un ejemplo, podemos imaginar que nuestros mensajes tienen 4 bits y que utilizaremos 2 parity bits para proteger el mensaje. Por lo tanto, nuestro código tendrá 6 bits; posiciones $0-3$ son reservadas para el mensaje, y las posiciones 4 y 5 serán los parity bits. También, tendremos dos parity sets. Por ejemplo, estos pueden ser $\{0,1,4\}$ y $\{1,2,3,5\}$. *Noten que el parity bit está incluido en su parity set.* Quiere decir, que el primer parity bit, en la posición 4 del código, corresponde a parity set $\{0,1,4\}$. En el código cada parity bit cuenta la paridad de lo elementos demás en su parity set. Abajo damos un ejemplo:

- `messageBits = 4, parityBits = 2`
- `parity sets =` $[\{0,1,4\},\{1,2,3,5\}]$
- `message = [1,0,1,1]`
- `code = [1,0,1,1,1,0]`
- Aquí `code[4]=1` dodo que `message[0]=1` y `message[1]=0`.

Al transmitir un código, algunas posiciones se pierden y nos llega un código con ciertas posiciones marcadas con `'?'`. Los parity sets y bits nos permiten reconstruir bit en esta posición si solo si hay un parity set dónde aparece esta posición *y todas otras posiciones son marcadas con 0 o 1*. Por ejemplo, si recibimos el código `['?','?',1,1,1,0]` en el ejemplo arriba, las dos posiciones faltantes podemos corregir usando los parity sets de la siguiente manera:

- Primero usamos parity set $\{1,2,3,5\}$ para ver que en la posición 1 debe estar un 0, transformando el código a `['?',0,1,1,1,0]`.
- Segundo, usamos el parity set $\{0,1,4\}$ para reconstruir bit en la posición yero cómo 0, pero sabiendo que el mensaje es `['?',0,1,1,1,0]` reconstruido en paso 1.

Este ejemplo demuestra que el orden en cual usamos parity sets puede variar y que hay que ir iterando siempre y cuando se puede reconstruir un bit cual es desconocido en algún parity set.
En el archivo `erasure.py` deben completar el método `correctErasures(code)` cual, al recibir un código con ciertas posiciones desconocidas. Noten que estas posiciones pueden ser del mensaje original y de parity bits. El código incluye varios ejemplos para testear su solución.

Entrega

Para entregar su tarea, deben subir los dos archivos .py cómo un .zip al buzón de canvas.