document: 2019092801
subject: spsi
name: spsi_tareas_20182019.tex

Universidad de Granada
Grados Universitarios Oficiales

Grado en Ingeniería Informática

Cuaderno de tareas de SPSI

Autor:
fco. m. garcía olmedo

17 de octubre de 2019

Índice

0. Normas 3

1. Tarea 1 4

2. Tarea 2 5

3. Tarea 3 5

4. Tarea 4 6

0. Normas

El alumnado deberá observar lo siguiente:

1. Para su realización, el alumno usará en cada ejercicio el lenguaje sugerido por considerar que es el idóneo para su ejecución.
2. Los ejercicios propuestos no son obligatorios, excepto los requeridos como parte del primer criterio de evaluación, apartado "Evaluación" de la guía docente. Sin embargo, el alumnos es vivamente animado a completarlos todos para alcanzar una evolución óptima en la asignatura y unos conocimientos apropiados.
3. La entrega de las partes requeridas para la evaluación de cada práctica serán entregadas telemáticamente vía la plataforma elegida. Cada contenido que el alumno desee entregar penderá en su sistema de un directorio del tipo `spsi_1920_prac_nn`, donde `nn` serán dos dígitos entre 0 y 9; por ejemplo,

   `spsi_1920_prac_03` o `spsi_1920_prac_11`

   La entrega será de lo que resulte de comprimir con `zip` o `tar.gz` la carpeta `spsi_1920_prac_nn` en cuestión. El fichero a entregar llevará el nombre

   `spsi_1920_prac_nn_iniciales_dni.ext`

   Por ejemplo: Antonio de la Merced Benegas, con nif 12345678Z entregará

   `spsi_1920_prac_13_amb_12345678.tar.gz`

o

`spsi_1920_prac_13_amb_12345678.zip`

4. Los ejercicios propuesto más abajo tendrán ocasionalmente carácter ampliatorio de la asignatura, debiendo buscar el alumno la documentación complementaria en la bibliografía o fuera de ella.

5. La copia de código supondrá la invalidación de la práctica, con sus correspondientes consecuencias en la media.

6. Cuando se pide un "guión en `Python`" (resp. `Sagemath`) deberá estar contenido en un fichero de extensión `py` o `ipynb` (resp. `ipynb`).

1. Tarea 1

1. Elabore un guión en `Python` que contenga funcionalidades sobre `string` capaces de hacer por separado lo siguiente:

   a) Determinar los caractéres que ocurren en el `string` y la frecuencia de cada uno de ellos.

   b) Eliminar del `string` los caractéres en blanco.

   c) Adaptar los caractéres del `string` a un alfabeto dado.

   d) Partir en bloques de un determinado tamaño, pasado como argumento, el `string`.

   e) Traducir el `string` a una lista de enteros según una biyección prefijada.

   f) Obtenga del `string` una representación binaria suya.

2. Elabore un guión en `Python` con una función, digamos `iSqrt(n)` que ejecutada calcule la raíz cuadrada entera para el número n arbitrariamente elevado. Use `iSqrt(n)` en otra función, digamos `iSqrt(n,m)`, para poder obtener bajo demanda m dígitos decimales de la raíz cuadrada de n.

3. Elabore un guión en `Python` que culmine con una función `iCrt` capaz de decidir si determinado sistema de congruencia enteras tiene solución y la proporcione caso de existir.

4. Elabore un guión en `Sagemath` que culmine en una función `iCRT` capaz de decidir si determinado sistema de congruencias, enteras o polinomiales, tiene solución y la proporcione caso de existir.

5. Elabore un guión en `Python` que culmine en una función `qExp(a,b,n)`, donde a y b sean números enteros cualesquiera y n cualquier número natural no nulo, que calcule a^b mód n.

2. Tarea 2

1. Elabore un script en `Python` que culmine en una función —digamos `afin(a,b,x)`— que, previamente normalizado en un alfabeto de 26 caracteres según las funcionalidades del apartado 1 de la Sección 1, cifre un string de acuerdo con la clave afín $\langle a, b\rangle$, cuando x tome el valor 0 y descifre con la clave afín de descifrado derivada de $\langle a, b\rangle$, cuando x tome cualquier otro valor.

2. Implemente el ataque al criptosistema afín según la prueba *chi-cuadrado*.

3. Implemente el ataque al criptosistema afín según la prueba de las *diferencias cuadradas*.

4. Ponga en claro, con una explicación razonada del procedimiento seguido, el siguiente texto cifrado mediante el criptosistema afín:

```
BZBTJAVGQGVOTBVGTKFNGQGVMTTNWVYNBFZBATDTBZYNYTBRANBMTDTBTDEVFZ
BGTYTBNKNGNBWGNXKNBGTWZGVBZTFQGTNQRAWVYTGVFQTGBTXVWGNBFTMVXYTD
NSNQVGTBWTTKTFTAWVLRTQNGTDTBTGZANDDTBZIKTNKEVFIGTTAIRBDNYTKNBQ
ZTSNBLRTMZMTATAFZBIVBLRTBBRIFNGZAVBFZBGTINJAVBDVFVKVBYTKMZTUVQ
NBWVGYTATQWRAVQNDTABZAWTFVGTAKNBZAFTABNBQGNYTGNBYTKVDTNAVWTAJV
XVNEZRANMNBWNQGVQZTYNYLRTTCQKVWVXVFZBFVXLRTTBWNBTFIGNYNQVGKNFN
AVYTKDGTNYVGYTWVYNBKNBDVBNB
```

3. Tarea 3

Haga lo siguiente:

1. Elabore un guión en `Python` con las herramientas necesarias para el formateo de cualquier texto en español y poder obtener el `string` idóneo para serle aplicado el cifrado de Vigenère.

2. Elabore un guión de `Python` que culmine en funciones o función de cifrado y descifrado para el *cifrado de Vigenère*.

3. Idee razonadamente al menos una extensión del criptosistema de Vigenère inspirada en el cifrado afín y valore su fortaleza.

4. Implemente en `Python` las herramientas necesarias para poder llevar a cabo el *examen de Kasiski*. Póngalo en práctica sobre algún texto cifrado.

5. Implemente en `Python` las herramientas necesarias para poder llevar a cabo la *prueba de Friedman*. Póngala en práctica sobre algún texto cifrado.

4. Tarea 4

Haga lo siguiente:

1. Elabore un guión en `Python` con las herramientas necesarias para el formateo de cualquier texto en español y poder obtener el `string` idóneo para serle aplicado el cifrado de Hill.
2. Elabore un guión de `Sagemath` que culmine en funciones o función que implemente el criptosistema de Hill.
3. Imagine al menos una extensión del criptosistema de Hill inspirada en el cifrado afín y eventualmente en el de Vigenère. Aporte la implementación que estime oportuna.