Fundamentos de Algoritmos – Trabajo Práctico 1

Master in Management + Analytics, Escuela de Negocios, UTDT

Primer semestre 2023

Observaciones generales:

- El trabajo se debe realizar en grupos de **dos personas**.
- El código fuente debe estar bien comentado. Cualquier aclaración adicional que se considere necesaria debe ser incluida como comentarios en el código fuente.
- La resolución del TP debe subirse al formulario *TP1: Entrega* en la página de la materia en el campus virtual.
- El programa entregado debe correr correctamente en Python 3.
- La fecha límite de entrega es el domingo 16 de abril a las 23:55.
- Los TPs serán recibidos hasta 48hs pasada la fecha estipulada, pero en ese caso se re-escalará el rango [60,100] al rango [60,80].

Un número $n \in \mathbb{N}_0$ (donde $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$; es decir, los enteros mayor o iguales que 0) se considera *peculiar* si y sólo si n es múltiplo de 22 y además la representación en base decimal de n alterna dígitos pares e impares. Por ejemplo, el número 21890 es peculiar porque $21890 = 22 * 995$ y además alterna 2 (par), 1 (impar), 8 (par), 9 (impar), 0 (par). Otros ejemplos de números peculiares son 616 y 1034.[1]

El objetivo de este trabajo es crear un programa en Python que ofrezca las siguientes funcionalidades:

- dados $n, m \in \mathbb{N}_0$, determinar si n y m son ambos pares, o bien ambos impares;
- dado $n \in \mathbb{N}_0$, determinar si los dígitos de n alternan su paridad;
- dado $n \in \mathbb{N}_0$, determinar si n es peculiar (en particular, el 0 es un número peculiar);
- dado $n \in \mathbb{N}_0$, obtener el n-ésimo número peculiar (definimos al 0 como el 0-ésimo número peculiar);
- dados $n, m \in \mathbb{N}_0$, obtener la cantidad de números peculiares entre n y m (inclusive en ambos casos).

[1] Es posible probar que los números peculiares son infinitos.

Al ejecutar el programa se deberá desplegar un menú desde el cual seleccionar la funcionalidad a la que se desea acceder. Dependiendo de la opción elegida y los argumentos ingresados, el programa tendrá que mostrar la respuesta que corresponde. A continuación puede observarse parte de una ejecución del programa en la cual el usuario consulta si el número 21890 es o no peculiar:

```
Funciones disponibles
---------------------
A. Misma paridad [n,m]
B. Dígitos alternan paridad [n]
C. Es peculiar [n]
D. N-ésimo peculiar [n]
E. Cantidad de peculiares entre [n,m]
F. Finalizar

Seleccione una opción: C
Ingrese n: 21890
El 21890 es peculiar.

Presione ENTER para continuar.
```

La ejecución debe concluir únicamente si se selecciona la opción `Finalizar`; caso contrario, tiene que desplegar nuevamente el menú de opciones.

En la siguiente tabla se ilustran los mensajes que debe imprimir el programa para algunas funciones y valores ingresados:

Función seleccionada	Argumento(s)	Mensaje por pantalla
`Misma paridad [n,m]`	2, 4	`El 2 y el 4 tienen la misma paridad.`
`Misma paridad [n,m]`	2, 5	`El 2 y el 5 no tienen la misma paridad.`
`Dígitos alternan paridad [n]`	147	`Los dígitos de 147 alternan paridad.`
`Dígitos alternan paridad [n]`	17	`Los dígitos de 17 no alternan paridad.`
`Es peculiar [n]`	616	`El 616 es peculiar.`
`Es peculiar [n]`	57	`El 57 no es peculiar.`
`N-ésimo peculiar [n]`	0	`El 0-ésimo peculiar es el 0.`
`N-ésimo peculiar [n]`	15	`El 15-ésimo peculiar es el 1298.`
`Cantidad de peculiares entre [n,m]`	100, 200	`Entre 100 y 200 hay 0 peculiares.`
`Cantidad de peculiares entre [n,m]`	100, 1000	`Entre 100 y 1000 hay 6 peculiares.`

Se deben entregar los siguientes dos archivos:

- **`peculiar.py`**, con el código de las funciones cuyos encabezados se muestran a continuación:

```
def misma_paridad(n,m):
def alterna_paridad(n):
def es_peculiar(n):
def n_esimo_peculiar(n):
def cant_peculiares_entre(n,m):
```

  Además, este archivo puede incluir funciones auxiliares en caso de considerarlo necesario.

- **`tp1.py`**, con el código principal para ejecutar el programa. Este código es el encargado de imprimir el menú, invocar a las funciones definidas en `peculiar.py` y mostrar los mensajes por la pantalla.

La carpeta adjunta `templates` contiene archivos de ejemplo para usar como referencia.

Observaciones:

- Definir todas las funciones auxiliares que se consideren necesarias.
- La función `print` solamente puede invocarse en el cuerpo principal del código, en el archivo `tp1.py`. Es decir, no puede usarse en las funciones requeridas ni en las funciones auxiliares.
- No se puede importar ninguna biblioteca de Python para resolver el trabajo práctico.
- Solo pueden usarse las instrucciones vistas en clase.
- Puede suponerse que el usuario siempre invocará a las funciones de manera correcta. Es decir, si hay errores de tipo o valor en los argumentos provistos, no se espera comportamiento alguno del programa (podría colgarse o terminar en un error, por ejemplo).