Introducción a la Programación

Augusto González Omahen

Primer Semestre de 2022

Clase práctica 11: Ordenamiento y Búsqueda

Ejercicio 1: entrando en calor

Escribir en Python una función `esta_en` la cual recibe una lista no vacía de strings ordenados de menor a mayor y un **string**, devuelve si el **string** se encuentra en la lista o no. Debe tener una complejidad algorítmica **logarítmica** o menor.

```
1  lista_abcdef: List[str] = [ 'alta', 'altura', 'bandida',
       'coqueta', 'dinamita', 'expensiva']
2
3  Test que debe pasar:
4  assertTrue(esta_en('coqueta', lista_abcdef))
5  assertFalse(esta_en('guapa', lista_abcdef))
6  assertFalse(esta_en('racineta', lista_abcdef))
```

Ejercicio 2: hora de votar

a) Nos encargan la tarea de buscar en un padrón de elecciones, datos estadísticos curiosos. En este caso, nos piden que encontremos personas con el mismo nombre que votan en distintos distritos.

Debemos implementar una función que recibe un padrón separado por distritos (una lista de listas de strings ordenados alfabéticamente) y un nombre (string) y devuelve los índices de aquellos distritos en donde hay alguna persona con ese nombre. Además debemos determinar la complejidad algorítmica de esta función.

Algunas aclaraciones

- Los nombres tienen a lo sumo una longitud de 30 caracteres
- La cantidad de personas en cada distrito es similar
- No se puede usar `in` para saber si una lista contiene un elemento
- Pueden usar la función auxiliar `esta_en`, que tiene complejidad logarítmica, para saber si un elemento está en la lista

Ejercicio 2 - Ejemplo

Por ejemplo:

```
padron: List[List[str]] = [
    ['Smith Alice', 'Smith Bob'],  # padron del distrito 0
    ['Smith Carl', 'Smith Lee'],   # padron del distrito 1
    ['Smith Alice', 'Smith Carl'], # padron del distrito 2
]


Test que debe pasar:
nombre1: str = 'Smith Alice'
nombre2: str = 'Smith Lee'
assertEqual(distritos_donde_vota(padron, nombre1), [0, 2])
assertEqual(distritos_donde_vota(padron, nombre2), [1])
```

Ejercicio 2 - Ejemplo

Por ejemplo:

```
padron: List[List[str]] = [
    ['Smith Alice', 'Smith Bob'],  # padron del distrito 0
    ['Smith Carl', 'Smith Lee'],   # padron del distrito 1
    ['Smith Alice', 'Smith Carl'], # padron del distrito 2
]


Test que debe pasar:
nombre1: str = 'Smith Alice'
nombre2: str = 'Smith Lee'
assertEqual(distritos_donde_vota(padron, nombre1), [0, 2])
assertEqual(distritos_donde_vota(padron, nombre2), [1])
```

b) Calcular la complejidad algorítmica.

Qué nos llevamos hoy

- Vimos más ejemplos de cómo calcular complejidades
- Funciones con órdenes $\mathrm{O(N)}$, $\mathrm{O(N^2)}$ y $\mathrm{O(N * log(P))}$.
- Cómo calcular el orden de complejidad cuando llamamos a otra función.
- Un problema con 2 soluciones diferentes con órdenes de complejidad algorítmica diferentes.

Con lo visto, ya pueden resolver la Guía de Ejercicios 7 completa!