Introducción a la Programación

Augusto González Omahen

Primer Semestre de 2022

Clase práctica 9: Complejidad algorítmica

Ejercicio 1: Cuáles podrían ser O(1)

a. Una función que recibe una lista no vacía de enteros y devuelve si los enteros están ordenadas de menor a mayor o no.

b. Una función que recibe una lista no vacía de enteros ordenada de menor a mayor y devuelve el elemento más grande que contiene.

c. Una función que recibe una lista no vacía de enteros ordenada de menor a mayor y devuelve si está el numero 13 o no.

d. Una función que recibe un string y devuelve si es palíndromo o no.

e. Una función que recibe un número natural y devuelve si es o no primo.

Ejercicio 2 and the philosopher's stone

La primera vez que leí Harry Potter, allá cuando une era feliz en la primaria, hubo una cosa que me había llamado la atención, el nombre de **Dirgah Hagrid**

Ejercicio 2 and the philosopher's stone

La primera vez que leí Harry Potter, allá cuando une era feliz en la primaria, hubo una cosa que me había llamado la atención, el nombre de **Dirgah Hagrid**

Claro, *DigrahHagrid* es un palíndromo!

Ejercicio 2 and the philosopher's stone

La primera vez que leí Harry Potter, allá cuando une era feliz en la primaria, hubo una cosa que me había llamado la atención, el nombre de **Dirgah Hagrid**

Claro, *DigrahHagrid* es un palíndromo!

En ese momento me pregunté si había más palíndromos escondidos en el libro, fue mucho tiempo después que pensé el siguiente pseudocódigo que toma un texto (en forma de lista de strings) y me devuelve la cantidad de palabras que son palíndromos:

Ejercicio 2 and the philosopher's stone

Cuidado que algo que NO sabemos es la cantidad de letras que pueda tener una palabra cualquiera y TAMPOCO sabemos la cantidad de palabras que tiene el libro.

```
1  cant_palindromos(libro:List[string])-> int:
2     i = 1
3     contador = 0
4
5     mientras i < len(libro):
6        iesima_palabra = libro[i]
7        if es_palindromo(iesima_palabra)
8           contador = contador + 1
9        i = i + 1
10
11    devolver contador
```

a. ¿Cuál es su complejidad algorítmica?

Ejercicio 2 and the philosopher's stone

Cuidado que algo que NO sabemos es la cantidad de letras que pueda tener una palabra cualquiera y TAMPOCO sabemos la cantidad de palabras que tiene el libro.

```
1  cant_palindromos(libro:List[string])-> int:
2     i = 1
3     contador = 0
4
5     mientras i < len(libro):
6        iesima_palabra = libro[i]
7        if es_palindromo(iesima_palabra)
8           contador = contador + 1
9        i = i + 1
10
11    devolver contador
```

a. ¿Cuál es su complejidad algorítmica?

b. ¿Qué pasaría si ahora sabemos que ninguna palabra del libro *Harry Potter y la Pierda Filosofal* supera las 20 letras?

Ejercicio 2 and the philosopher's stone

Cuidado que algo que NO sabemos es la cantidad de letras que pueda tener una palabra cualquiera y TAMPOCO sabemos la cantidad de palabras que tiene el libro.

```
1  cant_palindromos(libro:List[string])-> int:
2     i = 1
3     contador = 0
4
5     mientras i < len(libro):
6        iesima_palabra = libro[i]
7        if es_palindromo(iesima_palabra)
8           contador = contador + 1
9        i = i + 1
10
11    devolver contador
```

a. ¿Cuál es su complejidad algorítmica?

b. ¿Qué pasaría si ahora sabemos que ninguna palabra del libro *Harry Potter y la Pierda Filosofal* supera las 20 letras?

c. ¿Y si ahora además sabemos que el libro completo tiene solo 76.944 palabras?

A modo de cierre

Hoy vimos ejemplos de:

- Cómo calcular la complejidad algorítmica de un programa.
- Funciones con órdenes $\mathrm{O}(1)$, $\mathrm{O}(\mathrm{N})$ y $\mathrm{O}(\mathrm{N}^2)$.
- Cómo calcular el orden de complejidad cuando llamamos a otra función.
- Un problema con diferentes órdenes de complejidad dependiendo de los parámetros de entrada.