**Tecnología Digital 1: Introducción a la Programación - TP2**

**Autores:**

Deserti Gotta, Josefina

Estrada, Lucas

Galvan Orozco, Miranda

**Aclaraciones preliminares**

**Es necesario aclarar que para calcular la complejidad algorítmica de cada método tenemos en cuenta los peores casos.**

**Justificación de por qué los programas tienen la complejidad requerida**

**Método tamano():**

return len(self.campanas) **O(1)**

len(self.campanas) tiene una complejidad **constante** porque len() es siempre de orden constante. Acá está trabajando sobre una lista.

**Método barrios():**

barrios:set[str] = set() **O(1)**

for campana in self.campanas: **O(N)**

barrios.add(campana.barrio) **O(B)**

return barrios **O(1)**

Si hacemos la ecuación: nos queda orden N (por la cantidad de campanas en el dataset) \* B (por la cantidad de barrios únicos en el dataset).

barrios.add(), en su peor caso, tiene complejidad O(B).

**Método campanas\_del\_barrio():**

vr:list[CampanaVerde] = [] **O(1)**

for campana in self.campanas: **O(N)**

if campana.barrio == barrio: **O(1)**

vr.append(campana) **O(1)**

return vr **O(1)**

Crear una lista vacía tiene orden constante, al igual que añadir un valor a una lista con append. Los condicionales también son constantes. Si hacemos la ecuación: nos queda orden N (por la cantidad de campanas en el dataset).

**Método cantidad\_por\_barrio():**

vr:dict[str,int] = {} **O(1)**

barrios = self.barrios() **O(N \* B)**

for barrio in barrios: **O(B)**

vr[barrio] = 0 **O(1)**

campanas = self.campanas\_del\_barrio(barrio) **O(N)**

for campana in campanas: **O(N)**

if material in campana.materiales: **O(1)**

vr[barrio] += 1 **O(1)**

return vr **O(1)**

La creación de vr es O(1). Ya vimos antes que el método barrios() tiene orden O(N \* B).

For barrio in barrios tiene O(B), ya que depende de la cantidad de barrios en el dataset. campanas = self.campanas\_del\_barrio tiene O(N).

for campana in campanas tendrá O(N) ya que la cantidad de veces que se ejecuta el ciclo depende de la cantidad de campanas en el dataset. El condicional dentro de este ciclo tiene O(1).

return tiene orden constante.

Si hacemos la ecuación: Nos queda que el tiempo dependerá de la cantidad de campanas y barrios.

**Método ordenar\_tres\_campanas():**

if (lista[0].distancia(lat,lon) > lista[1].distancia(lat,lon)): **O(1)**

lista[0], lista[1] = lista[1], lista[0] **O(1)**

if (lista[0].distancia(lat,lon) > lista[2].distancia(lat,lon)): **O(1)**

lista[0], lista[2] = lista[2], lista[0] **O(1)**

if (lista[1].distancia(lat,lon) > lista[2].distancia(lat,lon)): **O(1)**

lista[1], lista[2] = lista[2], lista[1] **O(1)**

Los condicionales y reasignaciones de variables son todas de orden constante. Si hacemos la ecuación nos queda .

**Método tres\_campanas\_cercanas():**

camps\_cerca:list[CampanaVerde] = []

if len(self.campanas) < 3:

for i in range(len(self.campanas)):

camps\_cerca.append(self.campanas[i])

if len(self.campanas) == 2:

if (camps\_cerca[0].distancia(lat,lon)> camps\_cerca[1].distancia(lat,lon)):

camps\_cerca[0], camps\_cerca[1] = camps\_cerca[1], camps\_cerca[0]

for i in range(3):

camps\_cerca.append(self.campanas[i])

if (camps\_cerca[0].distancia(lat,lon) > camps\_cerca[1].distancia(lat,lon)):

camps\_cerca[0], camps\_cerca[1] = camps\_cerca[1], camps\_cerca[0]

if (camps\_cerca[0].distancia(lat,lon) > camps\_cerca[2].distancia(lat,lon)):

camps\_cerca[0], camps\_cerca[2] = camps\_cerca[2], camps\_cerca[0]

if (camps\_cerca[1].distancia(lat,lon) > camps\_cerca[2].distancia(lat,lon)):

camps\_cerca[1], camps\_cerca[2] = camps\_cerca[2], camps\_cerca[1]

for campana in self.campanas:

self.ordenar\_lista\_de\_tres(campana, camps\_cerca, lat, lon)

return camps\_cerca

La creación de una lista vacía tiene orden constante.

El for i in range(len(self.campanas) tiene orden len(self.campanas), que por comodidad lo llamaremos N.

El for i in range(3) tiene orden **constante** porque el ciclo se realiza una cantidad fija de veces.

camps\_cerca.append() tiene orden **constante** porque sólo se añade un valor a la lista.

Todos los condicionales y sus operaciones en este código tienen orden constante.

for campana in self.campanas tiene orden N porque la cantidad de ejecuciones del ciclo depende de la cantidad de campanas en el dataset. Su operación es constante.

**return** es una operación de orden constante.

Si hacemos la ecuación: Podemos ver que la complejidad de éste método es lineal.