Trabajo Práctico

Fundamentos de Algoritmos

Profesor: Hernan Czemerinski

Integrantes: Victoria Sosa, Juan Spadaro y Nicolas Viñolo

Mayo 2022

arboles_de_la_especie

Dicho método cumple con el orden de complejidad **O(N)** debido a que tenemos N cantidad de árboles en el dataset. Una forma intuitiva de ver esto es observando que estamos en presencia de un for con N iteraciones. Sin embargo, es importante resaltar que la complejidad temporal de un programa no se expresa en función de las instrucciones de un lenguaje de programación, sino en función del tamaño de la entrada.

```
1   def arboles_de_la_especie(self, especie):
2       arboles_especie = []
3       for arbol in self._dataset:
4           if arbol.especie() == especie:
5               arboles_especie.append(arbol)
6       return arboles_especie
7
```

- fila 2, 4, 5 y 6 ---> **O(1)** debido a que corresponden a operaciones simples y la ejecución individual de cada paso no depende de N (ej. creación de listas, cumplimiento o no de una condición, agregar un elemento a una lista, devolución de una lista)
- fila 3 ---> **O(N) corresponde al peor de los casos, donde recorremos todos los árboles del dataset, es decir, iteramos N veces**

cantidad_por_especie

```
1   def cantidad_por_especie(self, minimo):
2       diccionario = dict()
3       for arbol in self._dataset:
4           cantidad_de_la_especie = len(self.arboles_de_la_especie(arbol.especie()))
5           if cantidad_de_la_especie >= minimo:
6               diccionario[arbol.especie()] = cantidad_de_la_especie
7       return diccionario
```

- fila 2, 5, 6 y 7 ---> **O(1)** debido a que corresponden a operaciones simples y la ejecución individual de cada paso no depende de N (ej. creación de un diccionario, cumplimiento o no de una condición, asignación de clave y valor para el diccionario, devolución de un diccionario)
- fila 3 y 4 ---> **cada una por separado tienen O(N) donde iteramos N veces recorriendo todos los árboles del dataset.**
  Sin embargo, dado que la fila 4 se encuentra anidada a la fila 3 el peor de los casos va a ser de complejidad $O(N^2)$. Esto sucede porque para

cada árbol del dataset se itera N veces y al contar con N árboles, la complejidad algorítmica termina siendo O(N^2).

arbol_mas_cercano

```
1   def distancia_euclidiana(x1, x2, y1, y2):
2           x1 = float(x1)
3           x2 = float(x2)
4           y1 = float(y1)
5           y2 = float(y2)
6           distancia_euclidiana = ((x1 - x2)**2 + (y1 - y2)**2)**(1/2)
7           return distancia_euclidiana
8
9   def arbol_mas_cercano(self, especie, lat, lng):
10      distancia_minima= None
11      arbol_distancia_minima = []
12      for arbol in self._dataset:
13          if arbol.especie() == especie:
14              arbol_distancia = distancia_euclidiana(lat, arbol.latitud(), lng, arbol.longitud())
15              if distancia_minima == None:
16                  distancia_minima = arbol_distancia
17                  arbol_distancia_minima = arbol
18              if arbol_distancia < distancia_minima:
19                  distancia_minima = arbol_distancia
20                  arbol_distancia_minima = arbol
21          return arbol_distancia_minima
```

- fila 2 a 7 y 10, 11, y de 13 a 21 ---> **O(1)** debido a que corresponden a operaciones simples y la ejecución individual de cada paso no depende de N (ej. operaciones matemáticas, asignación de una variable, cumplimiento o no de una condición, devolución de un resultado u objeto)
- fila 12 ---> **O(N) corresponde al peor de los casos, donde recorremos todos los árboles del dataset, es decir, iteramos N veces.**