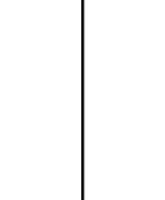
**Documento de análisis | Reto 3**

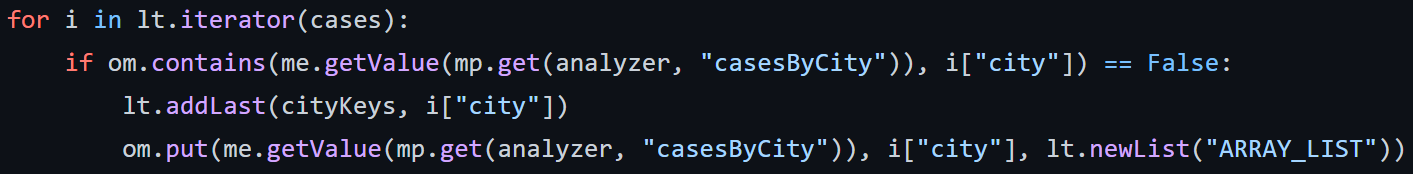
Integrantes:

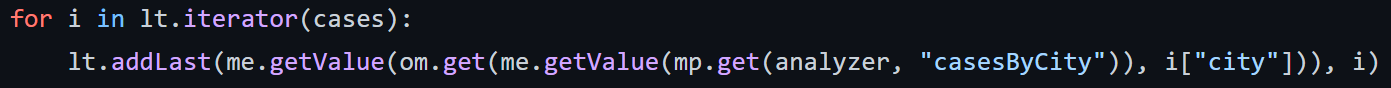
Moisés Agudelo, 202113485, [m.agudeloo@uniandes.edu.co](mailto:m.agudeloo@uniandes.edu.co)

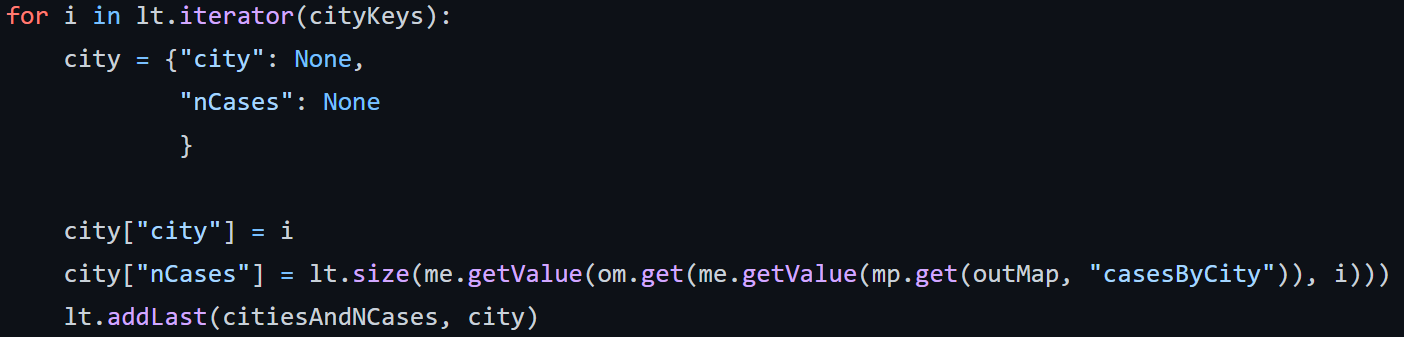
Sergio Franco, 202116614, [s.francop@uniandes.edu.co](mailto:s.francop@uniandes.edu.co)

**Requerimiento 1**

**Análisis de complejidad**

****O(n)



O(n)

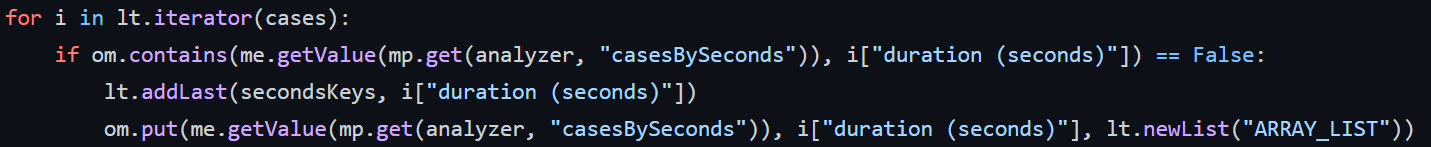
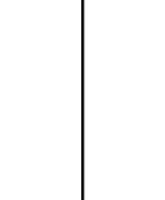
O(n)

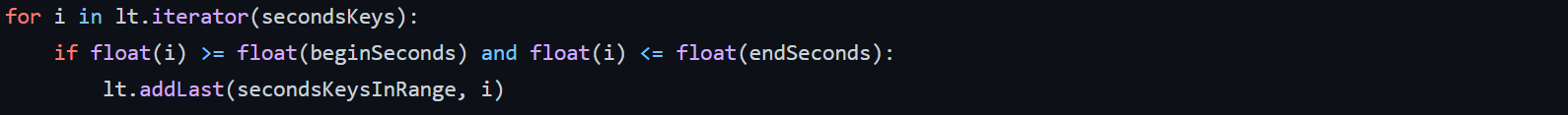
O(nlogn)

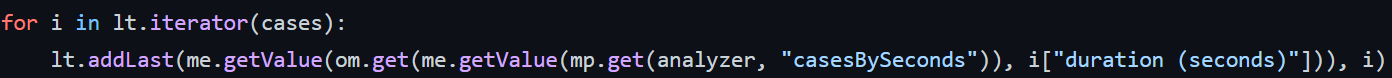
Complejidad temporal: O(nlogn)

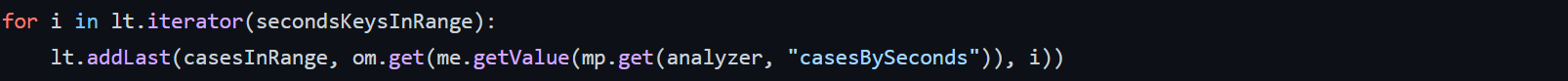
Al analizar los fragmentos de código que no son de tiempo constante tenemos una complejidad en total de O(n) + O(n) + O(n) + O(nlogn), la cual se puede reescribir como O(n + n +n +nlogn), operándose, O(3n + nlogn), las constantes se omiten, O(n+nlogn) y en la suma de complejidades se deja el termino mayor, concluyendo que la complejidad es O(nlogn), debida a lo ordenamientos de listas de diccionarios.

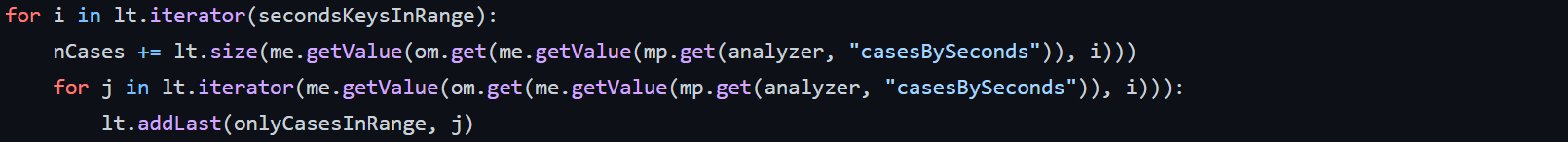
**Requerimiento 2 | Sergio Franco**

**Análisis de complejidad**

O(n)

O(n)

O(n)

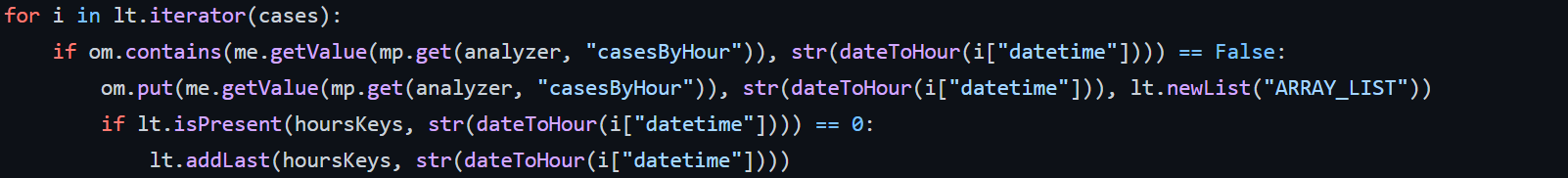
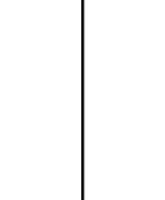
O(n)

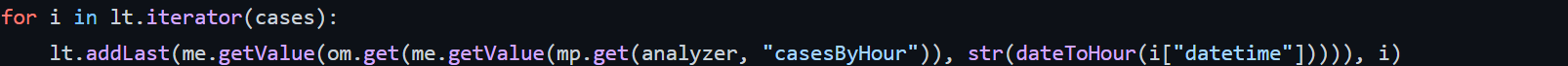
O(n^2)

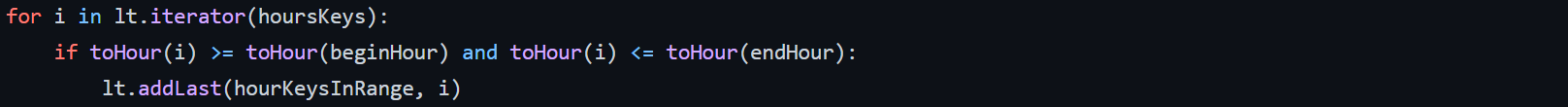
O(nlogn)

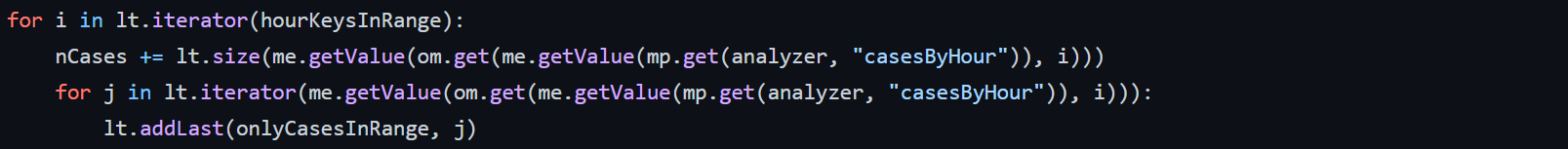
Al analizar los fragmentos de código que no son de tiempo constante tenemos una complejidad en total de O(n) + O(n) + O(n) + O(n) + O(n^2) + O(nlogn), la cual se puede reescribir como O(n + n + n + n +nlogn), operándose, O(4n + n^2 + nlogn), las constantes se omiten, O(n+ n^2 + nlogn) y en la suma de complejidades se deja el termino mayor, concluyendo que la complejidad es O(n^2), debida a un ciclo O(n) dentro de otro ciclo O(n).

**Requerimiento 3 | Moisés Agudelo**

**Análisis de complejidad**

O(n)

O(n)

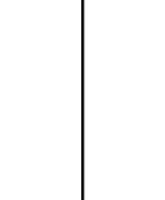
O(n)

O(n^2)

O(nlogn)

Al analizar los fragmentos de código que no son de tiempo constante tenemos una complejidad en total de O(n) + O(n) + O(n) + O(n^2) + O(nlogn), la cual se puede reescribir como O(n + n + n +nlogn), operándose, O(3n + n^2 + nlogn), las constantes se omiten, O(n+ n^2 + nlogn) y en la suma de complejidades se deja el termino mayor, concluyendo que la complejidad es O(n^2), debida a un ciclo O(n) dentro de otro ciclo O(n).

**Requerimiento 4**

**Análisis de complejidad**

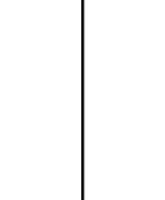
Texto

Descripción generada automáticamente

O(n)

Al analizar los fragmentos de código con solo un for que itera la lista, concluyendo así la complejidad O(n).

**Requerimiento 5**

**Análisis de complejidad**

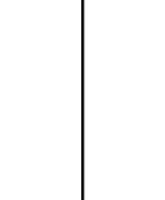
**Texto

Descripción generada automáticamente**

O(n)

Al analizar los fragmentos de código con solo un for que en el mismo se hace uso de if que iteran la lista, se puede concluir una complejidad de O(n).

**Requerimiento 6**

**Análisis de complejidad**