**Análisis de Complejidad**

**Nombre:** Nicolas Merchan Cuestas

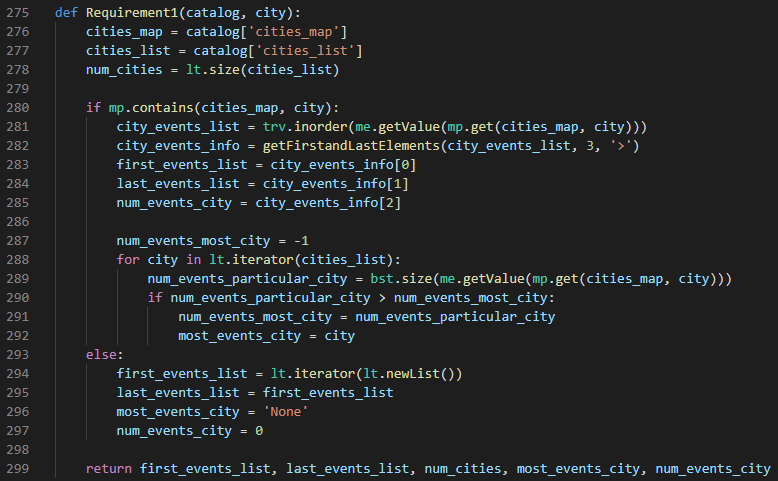
**Código:** 202112109

**Correo:** [n.merchan@uniandes.edu.co](mailto:n.merchan@uniandes.edu.co)

**Notas:**

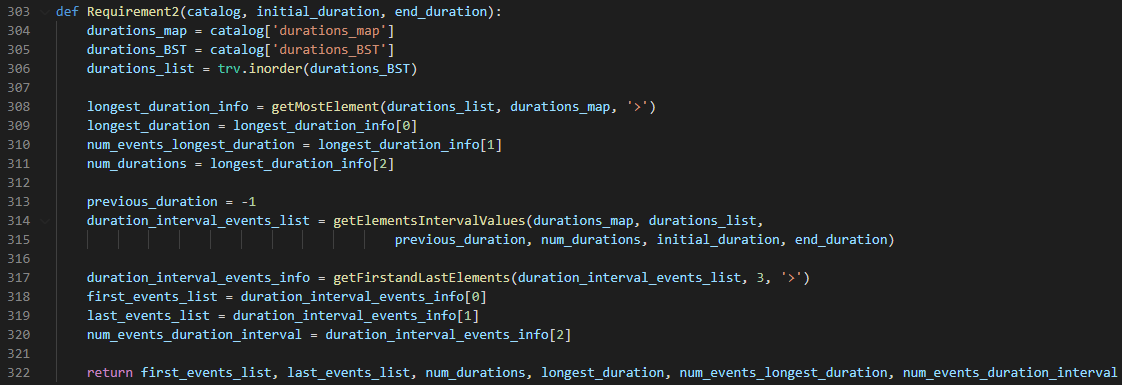
* Los resultados de las pruebas de tiempos de ejecución y las correspondientes gráficas de los mismos se encuentran en ‘*Datos - Análisis de Complejidad.xslx*’.
* El programa ‘Test\_Function.py’ realiza las pruebas de tiempos de ejecución de manera automática e ingresa los resultados en un archivo EXCEL llamado ‘Test\_Data.xlsx’.

**Requerimiento 1**



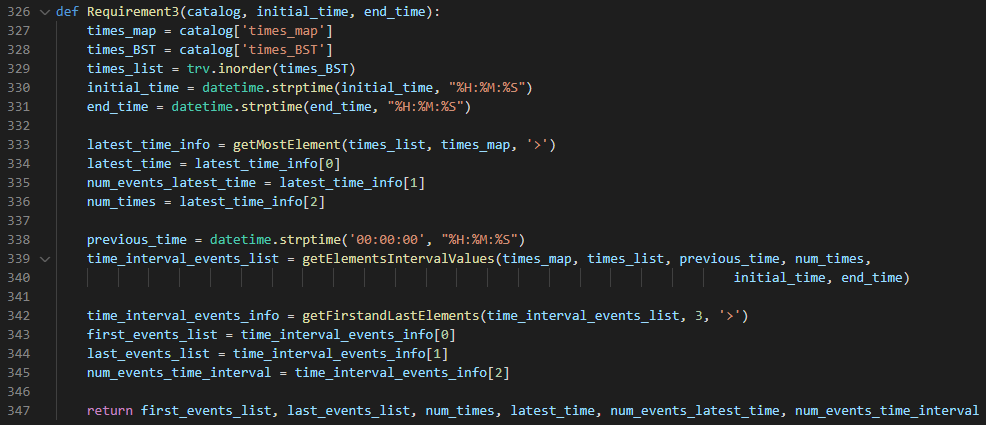
La complejidad del requerimiento 1 está dada por la función **Requirement1()**. El proceso de búsqueda de los avistamientos de la ciudad ingresada por parámetro se lleva cabo por medio de una búsqueda en el la tabla de Hash **catalog[‘cities\_map’]**. La tabla de Hash **catalog[‘cities\_map’]** tiene como llaves los nombres de las ciudades y como valores asociados a las llaves, BST’s de los avistamientos ocurridos en las ciudades respectivas ordenados cronológicamente por sus fechas y horas. La complejidad asociada al proceso anteriormente descrito es **O(1)**. Paralelamente, la búsqueda de la ciudad con más avistamientos se basa en un recorrido por los BST’s referentes a las ciudades haciendo uso de la lista ordenada **catalog[‘cities\_list’]**, la cual contiene como valores los nombres de las ciudades. Así, es posible comparar todas las ciudades ingresando los valores en las posiciones individuales de **catalog[‘cities\_list’]** en **catalog[‘cities\_map’]** como llaves. La complejidad asociada al proceso anteriormente descrito es **O(log n)**. Es propicio resaltar que el recorrido anteriormente descrito se realiza sobre las ciudades registradas, no sobre la totalidad de avistamientos registrados. La cantidad de ciudades crece, aproximadamente, de manera logarítmica con el incremento de los avistamientos registrados. Finalmente, la complejidad del requerimiento 1 es **O(log n)**¸ dado que el proceso más complejo tiene dicha complejidad asociada.

**Requerimiento 2**



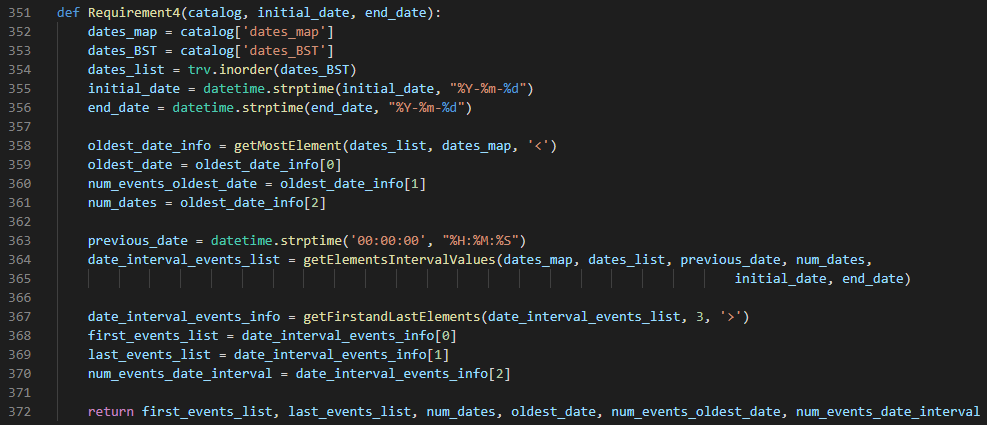
La complejidad del requerimiento 2 está dada por la función **Requirement2()**. La tabla de Hash **catalog[‘durations\_map’]** tiene como llaves las duraciones y como valores asociados a las llaves, BST’s de los avistamientos ocurridos con las duraciones respectivas ordenados ascendentemente por sus fechas y horas. El BST **catalog[‘durations\_BST’]** tiene como llaves y valores asociados a las llaves las duraciones de los avistamientos. La lista ordenada **durations\_list** contiene todas las duraciones de los avistamientos ordenados de menor a mayor en función de la duración. El orden de **durations\_list** se da porque es producto del recorrido inorder del BST **catalog[‘durations\_BST’]**. El proceso de búsqueda de la mayor duración se da por medio de la función **getMostElement()**, la cual busca el último elemento de **durations\_list**. La complejidad asociada al proceso anteriormente descrito es **O(1)**. El proceso de búsqueda de los avistamientos con una duración dentro del intervalo ingresado por parámetro se lleva cabo por medio de la función **getElementsIntervalValues()**, la cual realiza una búsqueda lineal en **durations\_list** que se detiene cuando encuentra la duración superior del intervalo o una duración mayor a la misma. La complejidad asociada al proceso anteriormente descrito es **O(lon n)**. Es propicio resaltar que el recorrido anteriormente descrito se realiza sobre las duraciones registradas, no sobre la totalidad de avistamientos registrados. La cantidad de duraciones crece, aproximadamente, de manera logarítmica con el incremento de los avistamientos registrados. Finalmente, la complejidad del requerimiento 2 es **O(log n)**¸ dado que el proceso más complejo tiene dicha complejidad asociada.

**Requerimiento 3**



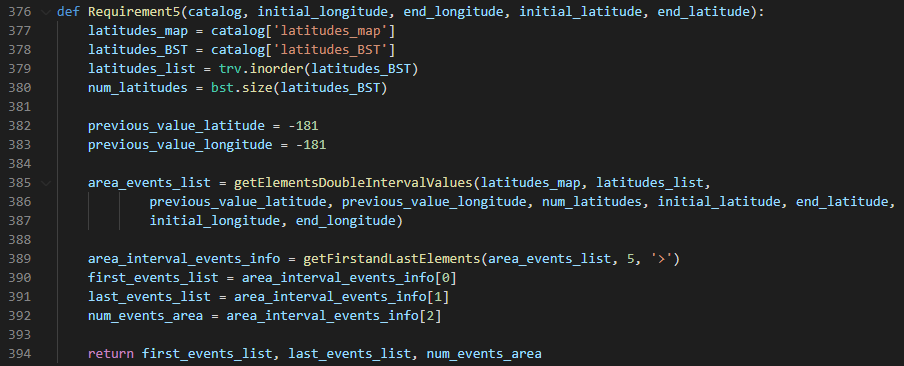
La complejidad del requerimiento 3 está dada por la función **Requirement3()**. La tabla de Hash **catalog[‘times\_map’]** tiene como llaves los tiempos y como valores asociados a las llaves, BST’s de los avistamientos ocurridos en los tiempos respectivos ordenados ascendentemente por sus fechas. El BST **catalog[‘times\_BST’]** tiene como llaves y valores asociados a las llaves los tiempos de los avistamientos. La lista ordenada **times\_list** contiene todos los tiempos de los avistamientos ordenados de menor a mayor en función del tiempo. El orden de **times\_list** se da porque es producto del recorrido inorder del BST **catalog[‘times\_BST’]**. El proceso de búsqueda del mayor tiempo se da por medio de la función **getMostElement()**, la cual busca el último elemento de **times\_list**. La complejidad asociada al proceso anteriormente descrito es **O (1)**. El proceso de búsqueda de los avistamientos ocurridos en un tiempo dentro del intervalo ingresado por parámetro se lleva cabo por medio de la función **getElementsIntervalValues()**, la cual realiza una búsqueda lineal en **times\_list** que se detiene cuando encuentra el tiempo superior del intervalo o un tiempo mayor al mismo. La complejidad asociada a el proceso anteriormente descrito es **O(lon n)**. Es propicio resaltar que el recorrido anteriormente descrito se realiza sobre los tiempos registrados, no sobre la totalidad de avistamientos registrados. La cantidad de tiempos crece, aproximadamente, de manera logarítmica con el incremento de los avistamientos registrados. Finalmente, la complejidad del requerimiento 3 es **O(log n)**¸ dado que el proceso más complejo tiene dicha complejidad asociada.

**Requerimiento 4**



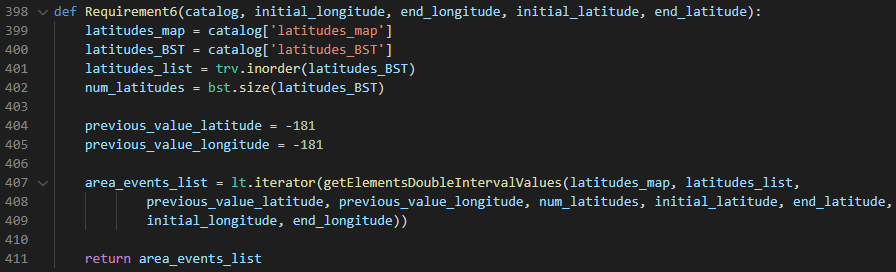
La complejidad del requerimiento 4 está dada por la función **Requirement4()**. La tabla de Hash **catalog[‘dates\_map’]** tiene como llaves las fechas y como valores asociados a las llaves, BST’s de los avistamientos ocurridos en las fechas respectivas ordenadas ascendentemente por sus tiempos. El BST **catalog[‘dates\_BST’]** tiene como llaves y valores asociados a las llaves las fechas de los avistamientos. La lista ordenada **dates\_list** contiene todas las fechas de los avistamientos ordenados de menor a mayor en función de la fecha. El orden de **dates\_list** se da porque es producto del recorrido inorder del BST **catalog[‘dates\_BST’]**. El proceso de búsqueda de la fecha más antigua se da por medio de la función **getMostElement()**, la cual busca el primer elemento de **dates\_list**. La complejidad asociada al proceso anteriormente descrito es **O (1)**. El proceso de búsqueda de los avistamientos ocurridos en una fecha dentro del intervalo ingresado por parámetro se lleva cabo por medio de la función **getElementsIntervalValues()**, la cual realiza una búsqueda lineal en **dates\_list** que se detiene cuando encuentra la fecha superior del intervalo o una fecha mayor al mismo. La complejidad asociada a el proceso anteriormente descrito es **O(lon n)**. Es propicio resaltar que el recorrido anteriormente descrito se realiza sobre las fechas registradas, no sobre la totalidad de avistamientos registrados. La cantidad de fechas crece, aproximadamente, de manera logarítmica con el incremento de los avistamientos registrados. Finalmente, la complejidad del requerimiento 4 es **O(log n)**¸ dado que el proceso más complejo tiene dicha complejidad asociada.

**Requerimiento 5**



La complejidad del requerimiento 5 está dada por la función **Requirement5()**. La tabla de Hash **catalog[‘latitudes\_map’]** tiene como llaves las latitudes registradas y como valores asociados a las llaves, BST’s de los avistamientos ocurridos en las latitudes respectivas ordenadas ascendentemente por sus longitudes. El BST **catalog[‘latitudes\_BST’]** tiene como llaves y valores asociados a las llaves las latitudes de los avistamientos. La lista ordenada **latitudes\_list** contiene todas las latitudes de los avistamientos ordenados de menor a mayor en función de la latitud. El orden de **latitudes\_list** se da porque es producto del recorrido inorder del BST **catalog[‘latitudes\_BST’]**. El proceso de búsqueda de los avistamientos ocurridos en una latitud y longitud dentro de los intervalos ingresados por parámetro se lleva cabo por medio de la función **getElementsDoubleIntervalValues()**, la cual realiza una búsqueda lineal en **latitudes\_list** y las listas producto de los recorrido inorder de los BST’s referentes a cada latitud en **catalog[‘latitudes\_map’]**. Dicho recorrido se detiene cuando encuentra la latitud y longitud superior de los intervalos o una latitud y longitud mayores a los mismos. La complejidad asociada a el proceso anteriormente descrito es **O(lon n)**. Es propicio resaltar que el recorrido anteriormente descrito se realiza sobre la latitudes y longitudes registradas, no sobre la totalidad de avistamientos registrados. La cantidad de latitudes y longitudes crece, aproximadamente, de manera logarítmica con el incremento de los avistamientos registrados. Finalmente, la complejidad del requerimiento 5 es **O(log n)**¸ dado que el proceso más complejo tiene dicha complejidad asociada.

**Requerimiento 6**



La complejidad del requerimiento 6 está dada por la función **Requirement6()**. La tabla de Hash **catalog[‘latitudes\_map’]** tiene como llaves las latitudes registradas y como valores asociados a las llaves, BST’s de los avistamientos ocurridos en las latitudes respectivas ordenadas ascendentemente por sus longitudes. El BST **catalog[‘latitudes\_BST’]** tiene como llaves y valores asociados a las llaves las latitudes de los avistamientos. La lista ordenada **latitudes\_list** contiene todas las latitudes de los avistamientos ordenados de menor a mayor en función de la latitud. El orden de **latitudes\_list** se da porque es producto del recorrido inorder del BST **catalog[‘latitudes\_BST’]**. El proceso de búsqueda de los avistamientos ocurridos en una latitud y longitud dentro de los intervalos ingresados por parámetro se lleva cabo por medio de la función **getElementsDoubleIntervalValues()**, la cual realiza una búsqueda lineal en **latitudes\_list** y las listas producto de los recorrido inorder de los BST’s referentes a cada latitud en **catalog[‘latitudes\_map’]**. Dicho recorrido se detiene cuando encuentra la latitud y longitud superior de los intervalos o una latitud y longitud mayores a los mismos. La complejidad asociada a el proceso anteriormente descrito es **O(lon n)**. Es propicio resaltar que el recorrido anteriormente descrito se realiza sobre la latitudes y longitudes registradas, no sobre la totalidad de avistamientos registrados. La cantidad de latitudes y longitudes crece, aproximadamente, de manera logarítmica con el incremento de los avistamientos registrados. Finalmente, la complejidad del requerimiento 6 es **O(log n)**¸ dado que el proceso más complejo tiene dicha complejidad asociada.