# *Análisis de complejidad de los algoritmos*

## *Módulo Carpeta*

***agregar_mensaje***:El análisis de complejidad en agregar_mensaje sería O(log n), porque lo que hace el método es insertar el mensaje en un heap, y la inserción en heap depende del número de mensajes (n) que ya contiene la carpeta.

***eliminar_mensaje:***El análisis de complejidad en eliminar_mensaje sería O(n), porque debe recorrer toda la lista hasta encontrar el mensaje  a eliminar, lo que depende de la cantidad de mensajes almacenados en la carpeta.

***lista_mensajes***:La complejidad sería O(n log n), donde n es la cantidad de mensajes en la carpeta, porque se hace una copia de la lista de mensajes y luego se extraen todos los elementos del heap uno por uno utilizando heappop(), que es O(log n) por extracción. Al repetir esto para los n mensajes, la complejidad total seria O(n log n).

***buscar_mensajes:***El análisis de complejidad en buscar_mensajes es O(n) porque al ser una búsqueda recursiva en un árbol con carpetas y subcarpetas la eficiencia de la operación depende directamente de la cantidad de subcarpetas que existan.

***mover_mensaje:***El análisis de complejidad en mover_mensajes sería O(n) porque depende de la cantidad de carpetas a recorrer para encontrar la carpeta de origen del mensaje.

***Hijos directos (carpetas principales):***
El análisis de complejidad de los hijos de la raíz en este caso serían Bandeja de entrada y Bandeja de salida. Sería O(1) ya que la búsqueda se hace directamente en el diccionario de la raíz.

***obtener_carpeta_padre:***La complejidad es O(1) en el mejor de los casos y O(n) en el peor, siendo n la cantidad de subcarpetas. En el peor caso, debe recorrer todo el árbol sin encontrar la carpeta buscada.Utilizamos este metodo para poder implementar el metodo de crear una subcarpeta anidada en el modulo usuario con el metodo:
crear_subcarpeta_anidada()
En este metodo lo que se hace es primero verificar si la carpeta padre es la raiz y luego si no lo es, utilizar el metodo obtener_carpeta_padre()
para encontrar la carpeta padre en el arbol y ahi usar el metodo agregar_subcarpeta() e insertar la carpeta nueva en el lugar deseado.

***agregar_subcarpeta:***El análisis de complejidad en agregar_subcarpeta es O(1), ya que las operaciones de verificación e inserción en el diccionario de subcarpetas se realizan en tiempo constante, sin depender de la cantidad de carpetas existentes.

***obtener_subcarpeta:*** En el mejor de los casos el tiempo de ejecución sería de O(1) porque buscaría directamente en el nivel de la raíz y ahí la búsqueda la hace en el diccionario de la raíz. Y en el peor de los casos el coste de obtener la subcarpeta sería de O(n) ya que depende de la cantidad de subcarpetas anidadas que haya y la cantidad de veces que tenga que aplicar la recursividad el algoritmo hasta encontrar la subcarpeta.

***obtener_carpeta:*** Su complejidad sería O(n), porque el método busca de manera recursiva entre todas las subcarpetas hasta encontrar la carpeta que contiene el mensaje, y en el peor caso debe recorrer todas las carpetas del árbol.

## *Módulo usuario*

***guardar_mensaje_recibido:*** La complejidad es **O(log n)**, porque el método primero obtiene la subcarpeta "Bandeja de entrada" realizando una búsqueda directa en un diccionario O(1) y luego agrega el mensaje al heap de mensajes de dicha carpeta, donde la inserción tiene un costo de O(log n), siendo *n* la cantidad de mensajes almacenados en la bandeja.

***guardar_mensaje_enviado:*** La complejidad sería **O(log n)**, porque el método primero obtiene la subcarpeta “Bandeja de salida” mediante una búsqueda directa en el diccionario de subcarpetas para luego agregar el mensaje al heap de esa carpeta, con el coste de **O(log n)** donde n es la cantidad de mensajes almacenados en la bandeja.

***get_bandeja_entrada:*** Su complejidad sería **O(1)**, porque obtiene la subcarpeta “Bandeja de entrada” directamente del diccionario de subcarpetas que se encuentra en la raíz.

**get_bandeja_salida:** Su complejidad sería **O(1)**, porque el método obtiene directamente la subcarpeta “Bandeja de salida” del diccionario de subcarpetas que se encuentra en la raíz.

***crear_subcarpeta_anidada:*** La complejidad es **O(n)**, ya que para crear una subcarpeta anidada primero debe buscar recursivamente la carpeta padre dentro del árbol de carpetas. En el peor de los casos, debe recorrer todas las subcarpetas hasta encontrar la indicada o confirmar que esta no existe.

**creacion_regla_de_filtro:** La complejidad es **O(1)**, ya que solo realiza operaciones de tiempo constante como conversiones a minúsculas y agregar un elemento a una lista,

**aplicar_filtro**: El análisis de complejidad es de **O(n)** ya que para obtener la carpeta de destino dependiendo de la palabra clave en el asunto depende de la cantidad de tuplas a recorrer que haya en la lista reglas_filtrado.

**filtrar_mensaje:** Su complejidad es **O(n)**, porque debe recorrer todas las reglas de filtrado para determinar si el mensaje coincide con alguna de ellas donde *n* es la cantidad de reglas de filtrado definidas por el propio usuario.

## *Módulo Servidor*

**registrar_usuario:** Registra un nuevo usuario en el servidor de correo. Primero verifica si el correo ya existe en el diccionario de usuarios; si no, crea un nuevo objeto Usuario y lo almacena usando su correo como clave.
Su complejidad es **O(1)**, porque la verificación de existencia y la inserción se realizan en tiempo constante promedio en un diccionario de Python.

**get_usuarios:** Devuelve una copia del diccionario de usuarios para garantizar la seguridad de los datos internos del servidor para que no haya modificaciones accidentales en el diccionario original.La complejidad es O(n), porque necesita recorrer todos los elementos del diccionario al crear la copia.

**listar_mensajes:** Muestra todos los mensajes de la bandeja de entrada o salida de un usuario registrado. La complejidad es O(n), donde n es el número de mensajes en la bandeja que se seleccionó.

**`recibir_mensaje:`** Permite a un usuario consultar cuántos mensajes tiene en su bandeja de entrada.La complejidad es **O(1)**, ya que las operaciones que realiza son todas de tiempo constante promedio.

## *Módulo Mensaje*

**`set_mensaje:`** Actualiza el contenido del mensaje en el objeto `Mensaje`, convierte el texto a string y lo asigna al atributo privado `__mensaje`.
Su complejidad es **O(1)**, ya que sus operaciones son simples

**`set_destinatario:`** Actualiza el correo electrónico del destinatario del mensaje y valida que tenga el carácter `'@'` .para ver que sea un correo electronico valido,su complejidad es **O(n)**, donde **n** es la longitud del string ingresado del nuevo destinatario.

***set_remitente***: Al igual que set_destinatario su complejidad es de **O(n)** donde el valor de **n** es la longitud del string ingresado del nuevo remitente.

**set_asunto**: Actualiza el asunto del mensaje convirtiendo mediante un str() el valor recibido a texto, y su complejidad es **O(n)** por la longitud del string por que la conversión a cadena requiere recorrer todo el contenido del valor.

**get_destinatario, get_remitente, get_asunto:**
Estos métodos *getters* tienen una complejidad de **O(1)**, por que solo acceden a un valor ya almacenado sin la necesidad de recorrer estructuras ni procesar datos adicionales.