

ANALISIS DE EFICIENCIA TRABAJO PRACTICO FINAL

En el siguiente cuadro se muestra el orden de eficiencia de los métodos y funciones utilizadas en nuestro código. Se plantea de la siguiente manera para resumir la complejidad del código y que sea más clara su visualización.

METODOS Y FUNCIONES UTILIZADAS	DESCRIPCION DE LA OPERACIÓN	CASO PEOR/MEJOR/PROMEDIO
CLASE	SERVIDOR	
registrar_registrar	La inserción (append) al final de una lista de Python es de tiempo constante (amortizado).	Todos: $O(1)$
ingresar_usuario	Búsqueda lineal en la lista de N usuarios. Debe recorrer la lista hasta encontrar o no al usuario.	Mejor: $O(1)$ (Usuario es el primero). Peor/Promedio: $O(N)$
enviar_msj	Búsqueda lineal del usuario receptor en la lista de N usuarios. El anexo de mensajes a las listas es $O(1)$.	Mejor: $O(1)$ (Receptor es el primero). Peor/Promedio: $O(N)$
CLASE	USUARIO	
desencolar_mensajes_urgentes	El tiempo está dominado por las P extracciones de la cola ($O(1)$ cada una). La operación de concatenación final de listas también es lineal al tamaño de los mensajes movidos (P) y el buzón existente.	Todos: $O(P)$
agregar_regla_filtro	Se debe realizar un recorrido (búsqueda) sobre las MC carpetas para encontrar la carpeta de destino de la regla.	Todos: $O(MC)$
aplicar_filtros_a_buzon	Eficiencia Multiplicativa (No Normal): Esta operación es muy costosa ya que su tiempo es el producto del número de mensajes (L), el número de reglas (R) y el coste de la búsqueda de la carpeta (MC).	Todos: $O(L \cdot R \cdot MC)$
CLASE	COLA DE PRIORIDADES	
encolar	Se debe recorrer linealmente la lista enlazada (de P elementos) para encontrar la posición donde insertar el mensaje manteniendo el orden de prioridad.	Mejor: $O(1)$ (Máxima urgencia). Peor/Promedio: $O(P)$
desencolar	Acceso directo al primer elemento de la lista enlazada (el más urgente). Operación de tiempo constante.	Todos: $O(1)$
CLASE	CARPETA(ESTRUCTURA DEL ARBOL)	
encontrar_carpeta	Búsqueda recursiva (DFS) que visita las MC carpetas del árbol en el peor caso.	Mejor: $O(1)$ (En la carpeta actual). Peor/Promedio: $O(MC)$
eliminar_mensaje_por_asunto	Recorrido lineal de los L mensajes en esta carpeta específica.	Mejor: $O(1)$ (Es el primer mensaje). Peor/Promedio: $O(L)$
buscar_y_extraer_mensaje	Recorrido recursivo de todas las carpetas y mensajes. En el peor caso, visita la totalidad de los M mensajes del usuario.	Mejor: $O(1)$ (En carpeta actual). Peor/Promedio: $O(M)$
CLASE	RED SERVIDORES(GRAFO)	
agregar_nodo / agregar_conexione	Inserción en estructuras de datos básicas (diccionarios y listas) en tiempo constante.	Todos: $O(1)$
encontrar_ruta_dfs	Algoritmo estándar de Búsqueda en Profundidad. La eficiencia es óptima, ya que visita cada servidor (V) y cada conexión (E) una sola vez.	Todos: $O(V+E)$
encontrar_ruta_bfs	Algoritmo estándar de Búsqueda en Amplitud. Su eficiencia es idéntica al DFS y también es óptima para grafos no ponderados.	Todos: $O(V+E)$
enviar_mensaje_por_red	El tiempo es la suma de la búsqueda de ruta ($O(V+E)$) y la búsqueda final del usuario en el servidor de destino ($O(N)$). Se compone de múltiples operaciones lineales.	Todos: $O(V+E+N)$

ANALISIS DE EFICIENCIA TRABAJO PRACTICO FINAL

Justificación de Eficiencia (Normales)

1. Eficiencia Constante $O(1)$ (Excelente)

Esta es la mejor eficiencia. Su tiempo de ejecución no cambia, sin importar cuántos usuarios o mensajes haya en el sistema.

- **Aplicaciones clave:** Inicializaciones, registro de usuarios en el servidor y la extracción del mensaje más urgente (desencolar).

2. Eficiencia Lineal $O(N)$, $O(L)$, $O(M_C)$, $O(V+E)$ (Aceptable)

El tiempo de ejecución es directamente proporcional al tamaño de la entrada. Si la entrada se duplica, el tiempo se duplica.

- **Búsquedas $O(N)$:** Buscar un usuario o servidor en una lista es lineal.
Optimización posible: Cambiar las listas de usuarios por estructuras de **Tabla Hash** (Diccionarios), lo que llevaría la búsqueda a $O(1)$.
- **Grafo $O(V+E)$:** La búsqueda de la ruta más corta/profunda en la red es la eficiencia óptima que se puede lograr en un grafo.
- **Manejo de Carpetas $O(M)$:** Las operaciones de mover, extraer o buscar un mensaje requieren recorrer el árbol de carpetas, lo cual es lineal al número de elementos.

3. Eficiencia Multiplicativa $O(L.R.MC)$ (Crítico / No Deseado)

La operación de aplicar_filtros_a_buzon tiene una complejidad muy alta que crece rápidamente. **No es una eficiencia normal** y representa un cuello de botella. Se justifica porque se debe:

1. Iterar sobre cada mensaje (L).
2. Para cada mensaje, iterar sobre cada regla (R).
3. Para cada regla, buscar la carpeta de destino (M_C).

Esta interdependencia causa un tiempo de respuesta muy largo en sistemas con muchas reglas y mensajes.

Diagrama de clases del Servidor de correo Grupo N°21

