Pablo Rocha

A01028638

Reflexión de Actividad Integradora

Importancia de las Listas Ligada:

- Debido a su estructura las listas ligadas borran y agregan elementos de cualquier lugar de la estructura en tiempo constante, a diferencia de arrayas dinámicos.
- Complejidad temporal de algunos métodos:

Indexado- O(n),

Inserción- O (1),

Búsqueda - O(n).

Borrado- O (1),

- Complejidad de espacio:

O(n)

- Pueden crecer y disminuir su tamaño en tiempo de compilación por lo que no se tiene que decidir un tamaño de la estructura desde el inicio.
- Insertando y borrando elementos no tenemos que mover todo, si no solo sus apuntadores a los siguientes elementos.
- No se gasta memoria porque siempre se usa el espacio de la lista.
-

Desventajas de las Listas Ligada:

Si no sabemos lo largo de la lista es imposible hacer búsqueda binaria u otros métodos donde se necesita saber el largo.

Tenemos que pasar por la lista para encontrar el elemento.

Usan un poco de más memoria por almacenar los apuntadores.

No se puede recorrer en sentido contario (esta se elimina en el caso de Listas Doblemente Ligados)

Importancia de las Listas Doblemente Ligadas:

Estas también señalan al elemento anterior.

Son mejores porque buscando cualquier elemento por posición podemos buscarlo sabiendo si se encuentra más cerca del final o del inicio y así disminuyendo el tiempo, como se tarda la mitad del tiempo seria complejidad O(n/2).

Importancia de usar Listas Doblemente Ligadas en este caso:

Primero es claro que usar listas ligadas en un ejercicio como este es muy útil, debido a que no conocemos su tamaño, y se podrían agregar elementos en cualquier lado. Como es una lista de intentos de entrada es muy posible que la lista siga creciendo y tengamos que seguir añadiendo elementos. También sería posible que un log fuera equivocado y se quisiera borrar. En una lista ligada solo borramos el elemento y cambiamos los apuntadores. Si fuera un vector o algo del tiempo, todos los elementos después del borrado se tienen que recorrer.

Algunas ventajas por la cual se usa Lista Doblemente Ligada:

	Listas Ligadas	Listas Doblemente Ligadas
Acceso	Solo se puede recorrer en una dirección	Se puede recorrer en ambas direcciones. Si quisiéramos agarrar los elementos más recientes esto lo haría mucho más rápido.
Apuntador	Solo requiere 1; siguiente	Requiere 2: siguiente y anterior
Memoria	Menos espacio	Mas espacio es muy poco debido a que solo es un apuntador mas
Eficiencia	Menos eficiente	Mas eficiente por sus dos apuntadores
Implementación	En stack	En stack, árbol binario, heap, etc.
Complejidad	Insertar y borrar elementos O(n)	Insertar y borrar elementos O (1)

Ejemplo De Mejora de Doble Lista Ligada:

Para usar el método de quickSort se tiene que continuamente comparar la data de los elementos, en una lista ligada esto tardaría O(n) pero en mi método de getdata checo si el índice es mayor que la mitad, si lo es, me voy por ser lado. De esta forma la complejidad baja a 0(n/2) en el peor caso.

Considerando que la entrada es de 16 mil datos y que cada búsqueda tarda 1 milisegundo, esta optimización me ahorra 8 mil milisegundos.

Poniendo un contador note que este método se usó 553450 veces en el algoritmo de quickSort.

Si multiplicamos las veces que se usó por 8 mil milisegundos, el resultado es:

4,427,600 milisegundos o 4427.6 segundos o 73 minutos. Claro que todo esto es tomando a consideración que cada búsqueda tarda 1 milisegundo lo cual no es el caso.