Facultad de Ingeniería
Escuela de Informática y Telecomunicaciones

Tarea Estructura de Datos: Lista de contactos

Integrante: Nicolás Felipe Román Cartes
Profesor: Nicolas Rosso Chamorro
Ayudante: John Bidwell Boitano

Fecha de realización: 1 de julio de 2018
Fecha de entrega: 1 de julio de 2018

0.1. Introducción

En este informe se planea tanto la comparación de los tiempos de complejidad de las estructuras implementadas en la tarea, y de cómo estas fueron implementadas para realizar las listas de contactos solicitadas. La carpeta "Script" Es la contenedora del menú, el cual puede inicializar cualquiera de las 4 estructuras y realizar las acciones solicitadas hasta que el usuario finalice la aplicación, en cambio la carpeta "Time" fue la realizada para realizar los tiempos de complejidad gracias a la librería "Faker" , para así obtener los resultados mostrados a continuación.

0.2. Comparación

A continuacion se mostraran los tiempos de complejidad obtenidos en la actividad:
Insercion:
Tiempo Lista:10 contactos >1 seg – 20 contactos >1 seg –100 contactos >1 seg – 1000 contactos 5>x>4 segs
Tiempo ABB: 10 contactos >1 seg – 20 contactos >1 seg –100 contactos >1 seg – 1000 contactos >4 segs
Tiempo AVL:10 contactos >1 seg – 20 contactos >1 seg – 100 contactos >1 seg – 1000 contactos >4 segs
Tiempo Hash:10 contactos >1 seg – 20 contactos >1 seg – 100 contactos >1 seg – 1000 contactos 6>x>5
Busqueda:
Tiempo Lista:10 contactos >1 seg – 20 contactos >1 seg – 100 contactos 3>x>2 seg – 1000 contactos 22 segs
Tiempo ABB: 10 contactos >1 seg – 20 contactos >1 seg – 100 contactos >2 seg – 1000 contactos 20 segs
Tiempo AVL:10 contactos >1 seg – 20 contactos >1 seg – 100 contactos >2 seg – 1000 contactos 16 segs
Tiempo Hash:10 contactos >1 seg – 20 contactos >1 seg – 100 contactos >3 seg – 1000 contactos 18 segs
Eliminar:
Tiempo Lista:10 contactos >1 seg – 20 contactos >1 seg – 100 contactos 3>x>2 seg – 1000 contactos 10 segs
Tiempo ABB:10 contactos >1 seg – 20 contactos >1 seg – 100 contactos >1 seg – 1000 contactos 3 segs
Tiempo AVL:10 contactos >1 seg – 20 contactos >1 seg – 100 contactos >1 seg – 1000 contactos 2 segs
Tiempo Hash:10 contactos >1 seg – 20 contactos >1 seg – 100 contactos >1 seg – 1000 contactos 3 segs
Y ahora se muestran los tiempos estimados:

Common Data Structure Operations

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Singly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Hash Table	N/A	Θ(1)	Θ(1)	Θ(1)	N/A	O(n)	O(n)	O(n)	O(n)
AVL Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Binary Search Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)

0.3. Conclusión

En el caso de los tiempos de complejidad de las estructuras con 100 o menos contactos la diferencia varia en milisegundos, siendo muy poco significativa. Mientras en el caso de los 1000 contactos el más óptimo para la inserción serían los árboles, siendo el más óptimo el 2-3 (El cual no implemente), ya que este puede almacenar mayor cantidad de datos por Nodo. En el caso de la búsqueda el más óptimo sería el AVL, estimo que el código de Hash, ya que el tamaño de este es de tamaño 26 no logra optimizar tanto la búsqueda, ya que cada nodo tendrá un ABB y la división no será muy grande. De esto podemos concluir que la mejor estructura para búsqueda, eliminación y inserción seria la tabla de Hash, seguido del AVL (el cual tiene ventaja del ABB ya que se equilibra automáticamente) y por último la lista.

0.4. Bibliografía

https://github.com/JohnBidwellB/EDD-2018-1