

# Tarea 1 de estructura de datos

## Introducción:

Para recopilar estos datos se ocupó las librerías faker y times, en los cuales en un ciclo se agregaron 10,20,100 y 1000 datos y luego en otro ciclo se llamaban las funciones buscar y eliminar para los 10,20,100 y 1000 datos recopilando así los tiempos en cada caso tal como en la siguiente imagen.

```
132 fake=Faker()
133 lista=Lista_s()
134 array=list()
135 for i in range(1000):
136     nombre=fake.first_name()
137     apellido=fake.last_name()
138     telefono=fake.phone_number()
139     mail=fake.email()
140     lista._insertar(nombre.lower(),apellido.lower(),telefono,mail)
141     array.append(apellido.lower())
142 inicio=time()
143 for p in range(10):
144     apellido=array[p]
145     lista.eliminar(apellido)
146 final=time()
```

## Datos recopilados:

Insertar:	Lista doblemente enlazada	Árbol binario de búsqueda	Árbol AVL	Árbol 2-3	Hash
10 datos	0.005	0.004	0.005	0.001	0.005
20 datos	0.011	0.011	0.010	0.002	0.011
100 datos	0.059	0.056	0.062	0.013	0.055
1000 datos	0.688	0.573	0.685	0.132	0.657

Buscar:	Lista doblemente enlazada	Árbol binario de búsqueda	Árbol AVL	Árbol 2-3	Hash
10 datos	0.001	0.001	0.001	0.001	0.001
20 datos	0.001	0.001	0.001	0.001	0.001
100 datos	0.002	0.001	0.001	0.001	0.002
1000 datos	0.074	0.008	0.004	0.005	0.004

Eliminar:	Lista doblemente enlazada	Árbol binario de búsqueda	Árbol AVL	Árbol 2-3	Hash
10 datos	0.002	0.001	0.001	0.001	0.001
20 datos	0.003	0.001	0.001	0.001	0.001
100 datos	0.013	0.003	0.002	0.001	0.002
1000 datos	0.061	0.024	0.026	0.017	0.010

## Complejidades teóricas:

	Access	Search	Insertion	Deletion
<u>Doubly-Linked List</u>	$\theta(n)$	$\theta(n)$	$\theta(1)$	$\theta(1)$
<u>Binary Search Tree</u>	$\theta(\log(n))$	$\theta(\log(n))$	$\theta(\log(n))$	$\theta(\log(n))$
<u>AVL Tree</u>	$\theta(\log(n))$	$\theta(\log(n))$	$\theta(\log(n))$	$\theta(\log(n))$
<u>B-Tree</u>	$\theta(\log(n))$	$\theta(\log(n))$	$\theta(\log(n))$	$\theta(\log(n))$
<u>Hash Table</u>	N/A	$\theta(1)$	$\theta(1)$	$\theta(1)$

## Análisis de resultados:

### Poca cantidad de datos:

Al ingresar, buscar y eliminar una cantidad reducida de datos no hay diferencias significativas en las estructuras.

### Gran cantidad de datos:

Insertión: cómo se puede apreciar en la tabla de datos la estructura más eficaz para insertar gran cantidad de datos son el árbol 2-3 y al contrario la menos eficaz sería la lista enlazada doble

Busqueda: en cuanto a búsqueda la estructura mas eficaz sería el Hash seguido por poco del árbol avl y el menos eficaz la lista enlazada doble quedando por debajo de las otras por una diferencia significativa

Eliminación: A la hora de eliminar muchos datos la mas eficaz seria Hash seguido del árbol 2-3, y la menos eficaz la lista doblemente enlazada

GitHub: <https://github.com/dani0f/Tareaedd>