	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor:	Jesus Cruz Navarro
Asignatura:	Estructura de Datos y Algoritmos II
Grupo:	Grupo 1
No de Práctica(s):	Práctica 4 – Algoritmos de búsqueda parte 1
Integrante(s):	Edwin Jaret Santiago Díaz
No. de Equipo de cómputo empleado:	22
No. de Lista o Brigada:	22
Semestre:	2022 - 2
Fecha de entrega:	13 marzo 2022
Observaciones:	

CALIFICACIÓN: _____

Algoritmos de búsqueda parte 1
Objetivos

1. Implementar los algoritmos de Búsqueda Lineal y Búsqueda Binaria para buscar llaves en colecciones de números enteros positivos.
2. Comparar los tiempos de ejecución de los algoritmos antes mencionados.
3. Implementar un programa que utilice tanto Búsqueda Binaria como Búsqueda Lineal para la resolución de problemas.

Desarrollo

Programa 1

En el programa "algoritmos_busqueda.py" se desarrolló e implementó los algoritmos Búsqueda Lineal con centinela, Búsqueda Lineal optimizada y Búsqueda Binaria en los cuales, dado un arreglo de un tamaño **n** se tenía que encontrar un elemento aleatorio (puede ser desde 0 hasta **n/2**) y se debía de retornar el índice en el que se encontraba, sino se encontraba se le notifica al usuario.

Además, se calculó el tiempo que tarda cada algoritmo y se ejecutó 3 veces por cada tamaño **n** para que se pueda obtener un promedio y después se pueda graficar el comportamiento de los estos.

El programa imprime:

- El tamaño del arreglo.
- El elemento por buscar.
- El índice donde se encuentra el elemento y el tiempo que tardó en encontrarlo para cada algoritmo en segundos.

Así se ve la salida del programa:

n = 200,000	Tiempo Promedio
Practica 4 Tamaño del arreglo: 2000000 Elemento a buscar: 504866 Busqueda Lineal con Centinela Se encontró el dato en el índice: 1694580 y tardó 0.084719 Busqueda Lineal Optimizada: Se encontró el dato en el índice: 1694580 y tardó 0.072978 Busqueda Binaria: Se encontró el dato en el índice: 1009429 y tardó 0.000025	**Búsqueda Lineal con Centinela**: 0.050424 [s]
Practica 4 Tamaño del arreglo: 2000000 Elemento a buscar: 975729 Busqueda Lineal con Centinela Se encontró el dato en el índice: 1096745 y tardó 0.050722 Busqueda Lineal Optimizada: Se encontró el dato en el índice: 1096745 y tardó 0.047798 Busqueda Binaria: Se encontró el dato en el índice: 1951499 y tardó 0.000070	**Búsqueda Lineal Optimizada:** 0.044589 [s]
Practica 4 Tamaño del arreglo: 2000000 Elemento a buscar: 787248 Busqueda Lineal con Centinela Se encontró el dato en el índice: 309730 y tardó 0.015833 Busqueda Lineal Optimizada: Se encontró el dato en el índice: 309730 y tardó 0.012992 Busqueda Binaria: Se encontró el dato en el índice: 1573426 y tardó 0.000068	**Búsqueda Binaria:** 0.000054 [s]

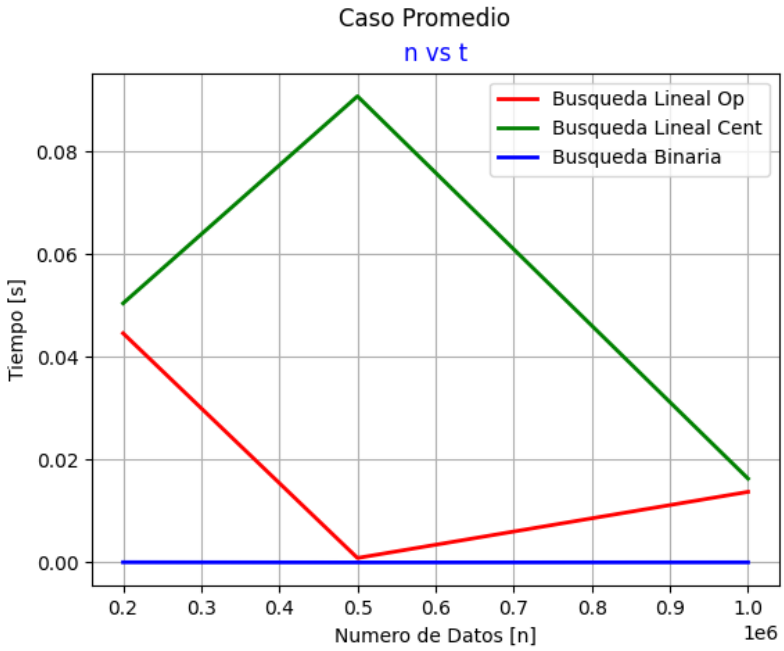
n = 500,000	
Tamaño del arreglo: 500000 Elemento a buscar: 133732 Busqueda Lineal con Centinela Se encontró el dato en el índice: 63693 y tardó 0.003113 Busqueda Lineal Optimizada: Se encontró el dato en el índice: 63693 y tardó 0.003216 Busqueda Binaria: Se encontró el dato en el índice: 267038 y tardó 0.000019 Tamaño del arreglo: 500000 Elemento a buscar: 198666 Busqueda Lineal con Centinela Se encontró el dato en el índice: 477770 y tardó 0.020894 Busqueda Lineal Optimizada: Se encontró el dato en el índice: 477770 y tardó 0.020549 Busqueda Binaria: Se encontró el dato en el índice: 397346 y tardó 0.000023 Tamaño del arreglo: 500000 Elemento a buscar: 39730 Busqueda Lineal con Centinela Se encontró el dato en el índice: 61475 y tardó 0.003205 Busqueda Lineal Optimizada: Se encontró el dato en el índice: 61475 y tardó 0.002711 Busqueda Binaria: Se encontró el dato en el índice: 79580 y tardó 0.000016	**Búsqueda Lineal con Centinela**: 0.09070 [s] **Búsqueda Lineal Optimizada:** 0.0008825 [s] **Búsqueda Binaria:** 0.000019 [s]
n = 1,000,000	
Tamaño del arreglo: 1000000 Elemento a buscar: 466027 Busqueda Lineal con Centinela Se encontró el dato en el índice: 265728 y tardó 0.011634 Busqueda Lineal Optimizada: Se encontró el dato en el índice: 265728 y tardó 0.010603 Busqueda Binaria: Se encontró el dato en el índice: 931720 y tardó 0.000018 Tamaño del arreglo: 1000000 Elemento a buscar: 290232 Busqueda Lineal con Centinela Se encontró el dato en el índice: 275246 y tardó 0.012894 Busqueda Lineal Optimizada: Se encontró el dato en el índice: 275246 y tardó 0.010803 Busqueda Binaria: Se encontró el dato en el índice: 580610 y tardó 0.000048 Tamaño del arreglo: 1000000 Elemento a buscar: 215054 Busqueda Lineal con Centinela Se encontró el dato en el índice: 454557 y tardó 0.024441 Busqueda Lineal Optimizada: Se encontró el dato en el índice: 454557 y tardó 0.019830 Busqueda Binaria: Se encontró el dato en el índice: 431052 y tardó 0.000052	**Búsqueda Lineal con Centinela:** 0.016323 [s] **Búsqueda Lineal Optimizada:** 0.013745 [s] **Búsqueda Binaria:** 0.000039 [s]

Y cuando no se encuentra el elemento en el arreglo, se le notifica al usuario de esta manera:

```
        Practica 4
Tamaño del arreglo:  2000000
Elemento a buscar:  1706502
Busqueda Lineal Con Centinela:  No se encontró el dato
Busqueda Lineal Optimizada:  No se encontró el dato
Busqueda Binaria:  No se encontró el dato
```

Para realizar la gráfica con los datos obtenidos se utiliza la librería **matplotlib** a través de un entorno virtual, esta no está incluida en la carpeta comprimida, sin embargo, el algoritmo se encuentra en el programa “algoritmos_busqueda_graficas.py”.

La grafica realizada es la siguiente.



Se puede demostrar que la Búsqueda Binaria es el algoritmo más eficiente de los 3 y la Búsqueda Lineal con Centinela es la menos eficiente.

Programa 2

En el programa "valores_repetidos.py" se hizo la combinación de los algoritmos Búsqueda Lineal y Búsqueda Binaria para encontrar el número de valores repetidos de un valor aleatorio (de 0 al 10) en un arreglo de 30 elementos.

El programa imprime:

- El tamaño de la lista.
- El elemento por buscar.
- La lista (debe de estar ordenada).
- La cantidad de veces que se encontró el elemento en el arreglo.
- El tiempo que le tomó al algoritmo.
- Desde qué índice hasta qué otro índice se encuentran los números repetidos.

Así se ve la salida del programa.

```
      Practica 4
Tamaño del arreglo:  30
Elemento a buscar:  1
Lista:  [0, 0, 0, 1, 3, 3, 4, 4, 4, 4, 5, 5, 6, 7, 7, 8, 8, 8, 8, 9, 9, 10, 10, 10, 11, 11, 11, 12, 12, 13]
Busqueda Binaria: Se encontró:  1  veces y se tardó 0.000004  segundos
Indice incial:  3 Indice final 3
```

Conclusiones

El algoritmo de Búsqueda Binaria es la mejor opción si se desea buscar un elemento en listas estáticas y de gran tamaño, sin embargo la lista debe de estar ordenada para que sea útil el algoritmo (aunque este no es un problema ya que se puede utilizar la función *sort()*). La complejidad algorítmica es de O(log(n)) mientras que la búsqueda lineal tiene una complejidad de O(n).

Mientras realizaba el segundo programa encontré una manera de realizar el conteo de los elementos de un arreglo y esto es a través de las funciones de Python en donde se guarda en un diccionario el elemento y el número de veces que se encuentra en dicho arreglo.