Tarea 1

En esta tarea deberá implementar una libreta de contactos que permita guardar, buscar, imprimir y eliminar contactos utilizando las diversas estructuras de datos que hemos visto a lo largo del curso.

Para esto, debe implementar las siguientes estructuras de datos:

- Lista enlazada (puede ser del tipo que estime conveniente)
- Árbol binario de búsqueda
- Árbol AVL
- Árbol 2-3
- Hash

Cada una de las estructuras debe poseer métodos para insertar, buscar y eliminar un contacto, además de un método que permita imprimir la lista de contactos en orden alfabético.

Cada contacto tiene nombre, apellido, teléfono y email.

El criterio para guardar cada contacto es por orden alfabético basándose en el apellido de cada persona.

Se debe, además de la implementación de cada una de las estructuras, crear un script que permita escoger al usuario qué tipo de estructura de datos desea utilizar para su libreta de contactos y además permita ejecutar las distintas opciones que la libreta debe poseer.

Junto al código, deberá realizar un pequeño análisis comparativo sobre el tiempo de ejecución de las distintas estructuras al insertar, buscar y eliminar un contacto. Para esto puede hacer uso de la librería `Time` (https://docs.python.org/3.6/library/time.html). Haga sus pruebas insertando en una estructura que posea 10, 20, 100 y 1000 contactos. Puede incluir en su análisis los ordenes de complejidad teóricos de cada una de las funciones que tienen las estructuras para así demostrar sus resultados.

Entrega

Fecha límite: Martes 26 de Junio a las 23:59 hrs.

Para la entrega de la tarea deberá crear un repositorio en el sitio GitHub (github.com) y enviar el link del repositorio al mail `john.bidwell@mail.udp.cl` con el asunto `Tarea EDD xxx` (donde `xxx` corresponde al nombre del alumno). Dicho repositorio deberá contener cada uno de los archivos de código en formato `.py` y el informe de análisis en formato `.pdf`.

Tendrá puntos extra si hace un buen uso de git, es decir, hace commits para identificar los cambios que ha hecho.

Recomendaciones

Haga la tarea con tiempo.

Utilice las pautas vistas en la primera ayudantía para escribir bien su código.

Utilice un archivo `.py` para cada estructura de manera que se mantenga el código organizado.

Para el set de datos de las pruebas puede utilizar la librería `Faker` (https://faker.readthedocs.io/en/latest/).

Git

Puede utilizar GitHub a través de:

- GitHub Desktop
- Terminal
- Página web

Guía sencilla para aprender Git http://rogerdudler.github.io/git-guide/index.es.html

Se recomienda realizar un commit para identificar cada uno de los cambios que hace en su código.

Este sería un ejemplo del flujo de trabajo que deben hacer con Git:

```
git init # Iniciliza un repositorio en la carpeta que nos encontremos
git add . # Añade todos los archivos, pueden utilizar git add xxx para
añadir un archivo en específico
git commit -m "Mi primer commit"
git push origin master
```

Luego al hacer un cambio:

```
git add "AVL.py"
git commit -m "Implementado AVL"
```

```
git add "hash.py"
git commit -m "Implementado hash"
```

```
git add "AVL.py"
git commit -m "Fix en función insertar para AVL"
```

...